University of Glasgow

DEPARTMENT OF

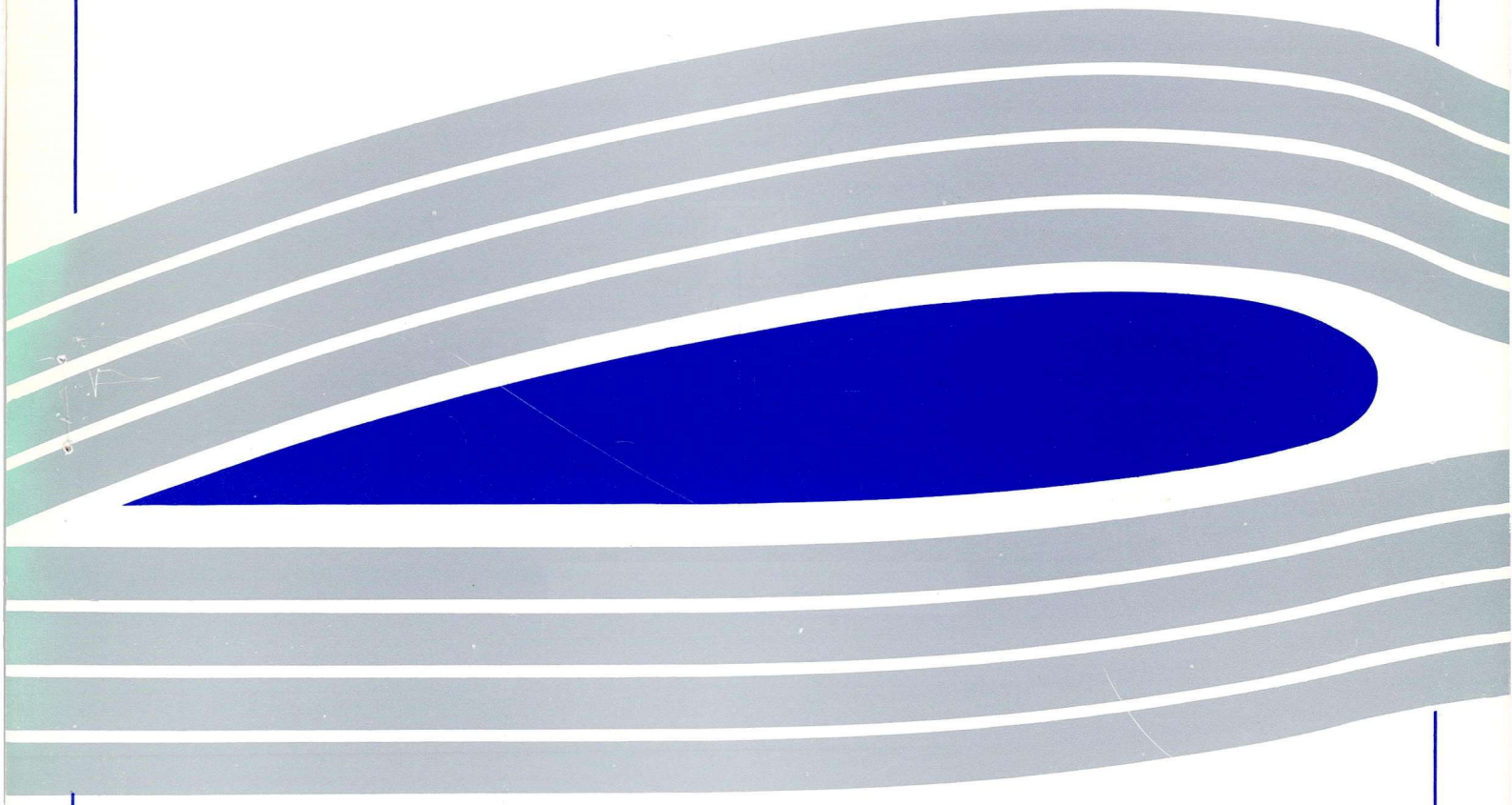
AEROSPACE ENGINEERING

UNIVERSITY OF GLASGOW
- 5 NOV 1996
LIBRARY

Engineering
PERIODICALS

Domain/Mesh Decomposition of Unstructured Grids with Pre-Ordering and Smoothing

Yufeng Yao



Engineering
PERIODICALS

Domain/Mesh Decomposition of Unstructured Grids with Pre-Ordering and Smoothing

Yufeng Yao

Dept. of Aerospace Engineering
Aero. Report 9506
University of Glasgow

May 19,1995

Domain/Mesh Decomposition of Unstructured Grids with Pre-Ordering and Smoothing

Yufeng Yao
Dept. of Aerospace Engineering
University of Glasgow
Glasgow G12 8QQ
e-mail:yufeng@aero.gla.ac.uk

Abstracts

Increasingly large scale computations are using unstructured discrete computational grids. A typical example is unstructured grid calculations based on finite volume methods (FVM) in computational fluid dynamics (CFD). One of the efficient ways to deal with such large scale problems is parallelization. The present paper will focus on domain/mesh decomposition. This is the first step for distributing unstructured computational domains on a MIMD-type parallel computer system. A graph theory framework for this problem will be constructed. Based on the framework three domain decomposition algorithms: recursive coordinate bisection (RCB), recursive angular bisection (RAB) and recursive graph bisection (RGB), will be introduced, tested and discussed. A pre-ordering and smoothing technique is proposed. It is necessary in the procedure for obtaining a 'good' domain partitioning result. Another interesting method, called the domain decomposition technique (DDT), is also investigated, which is driven in an inverse way, i.e. domain decomposition followed by mesh construction. Finally a simple and direct strategy called the mesh tailor technique (MTT) is discussed. Numerical comparisons using 2D CFD problems will be given. The further research work required to carry out a parallel implementation of a flow problem will be mentioned.

Keywords: domain/mesh decomposition, unstructured grid, pre-ordering and smoothing

Section 1: Introduction

Many large scale computations are emerging using unstructured discrete computational grids because of their ability for dealing with complex geometries and allowing simple grid adaption. A typical example is the unstructured grid calculation based on finite volume methods (FVM) in computational fluid dynamics (CFD). With advances in parallel computer systems, one of the efficient ways of calculating large scale computations is to implement a parallel computation procedure. For those domains discretised with a structured mesh the partitioning is simple and straightforward. However for those domains discretised with an unstructured mesh the partitioning is not easy and direct. The reason is that generally for unstructured meshes, the elements are ranked in a random order. It is difficult to find a cutting point or line with a natural ordering. Hence a major problem when implementing such large scale unstructured grid problems with parallel computing is the efficient decomposition (or partition) of the underlying computational domain. This is the focus of the discussion in this paper.

First we will investigate three widely used algorithms for partitioning unstructured domains. The three algorithms considered are recursive, i.e., the computational domain is subdivided by some strategy into two subdomains, and then the same strategy is applied to the subdomains in a similar way. In this way a partition into p=2**k subdomains is obtained after carrying out k of these recursive partitioning steps. These three algorithms

thus only differ by the partition strategy of a single domain into two subdomains. These three algorithms are:

1) recursive coordinate bisection (RCB);
2) recursive angular bisection (RAB);
3) recursive graph bisection (RGB).

The RCB and RGB algorithms have been used by a number of researchers. In particular RCB is a very direct approach, which comes immediately to mind, when considering the partitioning problem. In this paper another version of RCB, here named RAB, is also discussed when considering the angle ordering instead of the coordinate ordering. Another very recent method, named recursive spectral bisection (RSB) developed by Pothen, Simon and Liou[1], will not be discussed here. The RSB method is based on the computation of an eigenvector of the Laplacian matrix associated with the graph. As the cost of the spectral partitioning is very high (even using a Lanczos algorithm to compute the eigenvalue problem), it has yet to be determined if the spectral partitioning will have practical merit.

In section 2, we will formulate a general framework for the partitioning problem based on some graph theory notation. In section 3, the three partitioning algorithms, RCB, RAB and RGB, will be introduced and discussed. Section 4 will discuss in particular a pre-ordering and smoothing technique which is necessary in the procedure for obtaining a 'good' partitioning result. Also another interesting method, called the domain decomposition technique (DDT), is discussed in section 5, which firstly divides the domain into several sub-domains and secondly constructs an unstructured mesh within sub-domains. Section 6 will discuss a simple and direct partitioning strategy, named the mesh tailor technique (MTT). Section 7 will give some quantitative comparisons of algorithms directed at 2D CFD problems. Section 8 will offer conclusions and indicate further work required to refine these methods.

Section 2: The Partitioning Problem

So-called "efficient" partitions are both dependent on the problem and the computer considered. Given NP, the number of processors, one generally would like to partition the given problem into NP sub-problems of about equal size (this process called load balancing), and at the same time minimize the amount of communication information needed between processors in a parallel computation. Minimizing the communication is a function of both the length of the boundary of the subdomains, as well as of the number of neighbouing subdomains. For an explicit algorithm, the achievement of good load balancing is probably more important than minimizing communication costs, whereas for an implicit algorithm with higher communication requirements the situation might be just the reverse.

In this work the main target is a implicit implementation on a MIMD-type computer with a moderate number of parallel processors, for example the Intel iPSC/860 or a Workstation Cluster --- a parallel computer system based on a series of workstations in which each one represents one working processor. The target application finally to test the approach is a point implicit two dimensional Euler/NS solver for unstructured grids [2,3]. With this computer/application combination in mind the partitioning problem can be defined more precisely.

The partitioning problem can be considered as a generalization of the graph bisection problem, which is defined as followings: Given an undirected graph G, with the set of vertices V (either nodes or center points of each element) and the set of edges E,

$$G=(V,E),\ \text{partition}\ V=V_1\cup V_2,\ V_1\cap V_2=0,\ \text{such that}$$

$$|Ee|=\left\{e\middle|e\in E;e=(v_1,v_2);v_1\in V_1;v_2\in V_2\right\}$$

is minimized, subject to some constraint on the partition. Here we choose $|V1| = |V2|$, if $n = |V|$ is even and $|V1| = |V2| - 1$, if n is odd.

The assumption that the underlying problem can be expressed as an undirected graph is in no way restrictive. For example, for our target application, the upwind cell-centered finite-volume flow solver for the Euler/NS equations, the solution variables are associated with each element and flux computation is performed at the edges of each non-overlapping control volume. Each edge connects a pair of control volumes. In the partitioning which we are planning to use, mesh triangles are assigned to a particular processor. Fluxes are computed by the individual processors associated with the triangles. Communication is required along the inner edges, which are shared between the adjacent triangles residing in different processors. Hence for the purposes of establishing the partitioning of the problem, i.e., the assignment of triangles to different processors, we consider the dual graph. The triangles of the original mesh are vertices of the dual graph, and two triangles are considered to be adjacent vertices of the graph, if and only if they share an edge in the original mesh. A graph partitioning of this dual graph will thus yield an assignment of triangles to processors. In a similar way most general partitioning problems can be transformed to a graph partitioning problem. The approach used here is thus quite generally applicable.

The relationship between the unstructured mesh and its dual graph is shown in Figure 1 and Figure 2.

Section 3: Partitioning Algorithms

The general idea behind the three partitioning algorithms is to used an optimal strategy to partition a domain into two subdomains, and then to apply the same algorithms recursively for k steps until p=2**k subdomains have been obtained. The three algorithms thus only differ in the partitioning strategy for a single domain into two subdomains.

1) Recursive Coordinate Bisection (RCB)

This is probably the easier algorithm conceptually. It is based on the assumption that along with the set of vertices V=(v1,v2,...,vn), there are also two or three dimensional coordinates available for the vertices. For each $v_i \in V$ we thus have an associated tuple vi=(xi,yi) or triple vi=(xi,yi,zi), depending on whether we have a two or three dimensional model. A simple bisection strategy for the domain is then to determine the coordinate direction of longest expansion of the domain. Without loss of generality, assume that this is the x-direction. Then all vertices are sorted according to their x-coordinate. Half of the vertices with small x-coordinates are assigned to one domain, the other half with the large x-coordinates are assigned to the second subdomain. The algorithm for RCB is summarized in table 1:

Table 1: Recursive Coordinate Bisection (RCB)

---

1) Determine the longest expansion of the domain (x,y,or z direction)
2) Sort the vertices according to coordinates in the selected direction
3) Assign half of the vertices to each subdomain
4) Repeat recursively (divide and conquer)

Figure 3 gives an example of application of the RCB algorithm on an unstructured mesh used for the calculation of the flow over a NACA 0012 airfoil resulting in 8 partitioning.

2) Recursive Angular Bisection (RAB)

Another method similar to RCB, here named RAB, is also considered. The only difference is that it ranks in the angle ordering instead of the coordinate ordering. The algorithm of RAB is the same as that of RCB except for the ordering procedure. Figure 4 gives the results of RAB applied to a NACA 0012 airfoil shape resulting in 8 sub-domains.

3) Recursive Graph Bisection (RGB)

The weakness of both RCB or RAB is that the algorithm does not take full advantage of the connectivity information given by the graph. For efficiency, the main goal is to minimize the number of graph edges, which are connecting different subdomains. Thus instead of using the Euclidean distance between the vertex coordinates, a better way is to consider the graph distance between vertex given by d(vi,vj)=|shortest path connecting vi and vj|. With this change in metric one can define a new partitioning algorithm, which here is called recursive graph bisection (RGB).

First two vertices of maximal or near maximal distance in the graph are determined. Then all other vertices are sorted in the order of increasing distance from one of the extremal vertices. Finally vertices are assigned to two sub-domains according to the graph distance. The only difficulty is the determination of the diameter (or at least of a pseudo-diameter) of the graph. However there exist some very good heuristic algorithms for that purpose. These algorithms are also quite well-known in the engineering structures community, since they can also be used for reducing the storage requirements of sparse matrices in envelope or skyline storage format. Here the reverse Cuthill-McKee (RCM) algorithm of SPARSPAK[4] is used.

The RCM algorithm first finds two pseudo-peripheral vertices in the graph (i.e. vertices which have a very large distance, but which are not necessarily the pair of vertices with maximum distance). Then starting from one of the vertices, the root vertex, a so-called level structure is constructed. The level structure is a convenient way of orginizing the vertices in the graph in sets of increasing distance from the root. Hence the level structure delivered by the RCM algorithm forms the basis for the recursive graph bisection algorithm. Half of the vertices, the ones which lie closer to the root are assigned to one subdomain, the remaining vertices to the other subdomain. If we start out with a connected graph then by construction it is guaranteed that at least one of the two subdomains (the one including the root) is connected. The algorithm of RCM is summarized in table 2.

Table 2. Algorithm of Reverse Cuthill-Mckee

---

1) Find vertex with lowest degree. This is the ROOT vertex.
2) Find all neighbouring vertices connecting to the ROOT by incident edges. Order them by increasing vertex degree. This forms level 1.
3) Form level k by finding all neighboring vertices of level k-1 which have not been previously ordered. Order these new vertices by increasing vertex degree.
4) If vertices remain, go to 3).

Figures 5 and 6 illustrate nonzero entries of Laplacian matrix produced from natural ordering and RCM ordering. It shows that the band of the matrix is significantly reduced using RCM ordering.

By using RCM the algorithm of RGB can be summarized in table 3.

Table 3: Recursive Graph Bisection (RGB)

---

1) Use the RCM algorithm to compute a level structure
2) Sort vertices according to the RCM level structure
3) Assign half of the vertices to each sub-domain
4) Repeat recursively (divide and occupy)

Figure 7 gives the results of the RGB algorithm for a NACA 0012 airfoil resulting in 2 sub-domains.

Section 4: Pre-ordering and Smoothing Technique

Although in the above three strategies the load balance can be guaranteed strictly, it is difficult to visualize the partitioning results, i.e. the connectivity of the sub-domain and the smoothness of the cutting line. From the above result some serious problem still remains. One is that elements sometimes become seperated when using the RGB algorithm (see fig.7) This is obviously not a 'good' partitioning. The reason is that theoretically the RGB partitioning can only guarantee ONE sub-domain in which a root vertex is connected. Another problem is that the cutting line will normally take the saw-tooth shape (see fig.3,4,7) resulting an increase in the cost of communications between sub-domains. These two pheonomena are also repeated in other publications on partitioning[5]. To date however the method of overcoming these shortcomings appears not to have been addressed. In this report two algorithms are proposed to solve the problem.

One technique attempted is pre-ordering. Generally the mesh generated by advancing front technique(AFT) or delauney triangulation(DT) is in a random order. Hence the graph of the mesh is also ordered randomly. This state has resulted in the phemonemon of separated elements (see fig.7). This problem can be overcome by carrying out a pre-ordering step before implementing the partitioning. The algorithm of pre-ordering is as follows:

Table 4: Pre-ordering algorithm

---

1) read in the mesh file
2) list and number elements in 1-D natural ordering
3) select an ordering rule, i.e. the barycenter of element, the minimum coordinate value of its node, etc.
4) pre-order all elements according the rule selected.

The results of RGB are found to be different whether or not pre-ordering is used. This can be seen in Figure 8 with pre-ordering when compared to Figure 7.

Another technique is called smoothing. After partitioning the domain under the rule of load balancing using a particular strategy, there will normally be produced a saw-tooth shape boundary between sub-domains. The smoothing algorithm is used to adjust some vertices and change their identity, which will result in a cutting line with relative smoothness. The algorithm of smoothing is as follows:

Table 5: Smoothing Algorithm

---

1) do the partitioning using either RCB, RAB or RGB giving the initial sub-domain (without smoothing)
2) Flag each vertex with the number the same as that of the sub-domain it belongs to
3) do loop over each sub-domain { for each vertex counting the identity of its neighbouring vertex. If the identity of all its neighbours did not belong to this sub-domain then it means this vertex is separated from its sub-domain. Find out where it belongs and change its identity. If more neighbouring vertices belong to same neighbouring sub-domain found then this vertex is better belonging to that sub-domain. Change their identity. } end the do loop
4) re-counting the number of vertices in each sub-domain (this may result in a small lack of balance.)
5) output the sub-domain depending on its flag number

Figs 8-11 give the result with smoothing. Compared with Figs 3,4,7 it can be seen that using the smoothing technique, partitioning results are improved. Figs 13-16 illustrate the unstructured mesh over the NACA 0012 airfoil shape with 8 partitioning using the RGB method with or without pre-ordering and smoothing. It can be seen that by using both pre-ordering and smoothing improved results are obtained.

Section 5: Domain Decomposition Technique (DDT)

All the above discussions are based on certain pre-conditions, i.e. given a domain of interest, firstly construct the unstructured mesh, then partition the mesh using a particular strategy. The advantage of using this approach is that the load balance can be retained, although pre-ordering and smoothing need be considered as discussed above. The shortcoming is a lack of knowledge of the quality of the partitioning results, i.e. the relationship between each sub-domain and the shape of cutting lines. Hence another approach is to consider the inverse. For the given domain, divide it first into several sub-domains. In this way we can organise the relationships of the sub-domains and make the connections as simple as possible as well as the shape of cutting lines. Secondly we can construct the unstructured mesh in each sub-domain. Unfortunately one cannot guarantee strictly the load balancing in this way. This strategy is called the domain decomposition technique (DTT). The algorithm of the DDT is as follows:

Table 6: DDT algorithm

---

1) Define the interested domain
2) Divide the domain into several sub-domain
3) for each sub-domain use Delaunay Triangulation (DT) to construct the unstructured mesh. As the DT method always takes the given boundary points as its triangular node, hence there will be no over-lapping points occuring in the common line between neighbouring sub-domains
4) Construct the relationship between the sub-domains and the data structure of the communication information

---

Figure 12 gives an example for the NACA 0012 airfoil shape with 4 sub-domains using the DDT method. Ref.[6] applies the same strategy in parallel CFD simulation.

Section 6: Mesh Tailor Technique (MTT)

Although another suitable bisection strategy, recursive spectral bisection(RSB), is not discussed here, results show that the subdomains obtained from RSB approach are connected (although there is no theoretical guarantee for it), and nicely rounded and compact. The cost of the RSB method has yet to be determined and thus whether it will have practical merit. The impression of the result of domain partitioning is that the domain is simply cut into several parts. This suggests the use of some basic simple geometric shapes such as squares, triangles, circles, curved lines etc. to tailor the given mesh graph. Based on this idea the following algorithm of the mesh tailor technique (MTT) is proposed.

Table 7: mesh tailor technique (MTT)

---

1) Unstructured mesh construction
2) Define the number of partitions and set the tailor strategy
3) Use the basic tailor elements to cut the domain
4) Flag and divide the sub-domain

Figs 17-20 show some typical results by using the above algorithm, which use either one type of basic tailor element or combination of them.

Section 7 : Comparisons

In this section we will give some quantitative comparisons between the algorithms. Before unbarking on this we will make some general observations of the algorithms: Both RCB and RAB produce long and narrow sub-domains. RGB creates more compact sub-domains, but sometimes they will have disconnected sub-domains. Hence in order to get a better partitioning, a smoothing technique is needed for the RCB and RAB algorithms, while both pre-ordering and smoothing techniques are necessary for the RGB algorithm. For the DDT method, the size and shape of sub-domains depend on the domain decomposition strategy chosen. More compact, connected sub-domains can be obtained after careful investigation. The problem is how to achieve load balance between sub-domains. As for the MTT method, the shape of sub-domains need be selected before the method is implemented. It is hoped in the near future that more general curves can be used to cut and partition the domain with the help of CAD.

To obtain a more quantitative comparison , the number of cutting edges, named Ec, in each sub-domains will be presented. For the condition of balanced load the value of Ec represents the information exchange cost. The great the value Ec the more CPU time will be wasted on communications.

Table 8 gives the number of elements in each sub-domain(Elem_sub) and the number of cutting edges(Ec) for three algorithms RCB,RAB and RGB with or without pre-ordering and smoothing. The unstructured mesh around the NACA 0012 airfoil shape considered here is generated by the AFT method. It results in 4854 elements and 2504 nodes.

Table 8 The number of sub-domain elements and cutting edges (Elem_sub and Ec) --- nelem=4854 node=2504

Method	O	S	elements in sub-domain	... Ec
RCB	No	No	606 607 607 607 606 607 607 607	... 232
RCB	No	Yes	605 606 606 608 607 608 607 607	... 227
RAB	No	No	606 607 607 607 606 607 607 607	... 262
RAB	No	Yes	596 611 604 605 601 619 599 619	... 206
RGB	No	No	606 607 607 607 606 607 607 607	... 330
RGB	Yes	No	606 607 607 607 606 607 607 607	... 366
RGB	No	Yes	591 603 597 626 586 612 616 623	... 255
RGB	Yes	Yes	591 603 607 616 593 605 613 626	... 253

Table 9 illustrates the results on fine mesh which includes 21152 elements.

Table 9 The number of sub-domain elements and cutting edges (Elem_sub and Ec) --- nelem = 21152 node = 10666

Method	O	S	elements in sub-domain	... Ec
RCB	No	No	8 * 2644	... 504
RCB	No	Yes	2643 2644 2646 2643 2643 2644 2643 2646	...499
RAB	No	No	8 * 2644	... 594
RAB	No	Yes	2617 2644 2644 2641 2645 2655 2629 2677	...501
RGB (8)	No	No	8 * 2644	... 805
RGB (8)	Yes	Yes	2598 2629 2620 2661 2603 2682 2657 2702	...595
RGB (16)	No	No	16 * 1322 = 21152	... 1210
RGB (16)	Yes	Yes	1274 1324 1297 1333 1309 1311 1308 1352 1296 1307 1331 1351 1311 1346 1336 1366	...928

* Notes: O represents pre-ordering. S represents smoothing.

Table 8 shows that RGB will normally creates more cutting edges than RCB and RAB. By using a smoothing technique the number of Ec will reduce in each of the RCB, RAB and RGB methods. For the RGB algorithm in particular, by using both smoothing and pre-ordering, Ec reduces by 25% compared to that without its use. From Table 9 we obtain the same conclusions.

Table 10 illustrates the 4-subdomain partitioning results using the DDT method. Also we are interested in comparing the number of elements in each sub-domain and the number of cutting edges. Here we use Delauney Triangulation to generate the unstructured mesh in each sub-domain.

Table 10 The number of sub-domain elements and cutting edges (Elem_sub and Ec)

Method	elements in sub-domain	... Ec
DDT	1040 1178 1184 1090	... 200

This table shows that the number of elements in each sub-domain is different (min=1040 Max=1184). By using the DDT method one can define the position of the boundary points(edges) resulting in the same number and position on the boundary. However it is not possible to know how many elements will be generated within the defined domain. Thus how to retain a reasonable load balance when using DDT needs still to be investigated.

Finally in table 11 we illustrate the results of the MTT method. The four tailor strategies considered here are square, triangle, circular and curved line tailoring.

Table 11 The number of sub-domain elements and cutting edges (Elem_sub and Ec) --- nelem=4854 node=2504

Method	S	elements in sub-domain	... Ec
MTT_square	No	606 606 606 606 606 606 616 602	... 254
MTT_square	Yes	605 604 610 605 606 607 614 603	... 247
MTT_tri	No	606 607 607 607 606 607 607 607	... 276
MTT_tri	Yes	604 603 607 602 614 606 606 612	... 249
MTT_circle	No	606 607 607 607 606 607 607 607	... 217
MTT_circle	Yes	601 608 605 607 607 607 611 608	... 202
MTT_curve	No	606 606 606 609 606 606 606 609	... 268
MTT_curve	Yes	601 602 608 615 603 598 614 613	... 236

Section 8: Conclusions

From the above investigating the following conclusions are made:
1) RCB, RAB and RGB are three efficient methods of partitioning an unstructured mesh. They can be developed to deal with 3D meshes in theory.
2) With a pre-ordering and smoothing technique the quality of partitioning can be considerably improved.
3) The DDT is a potential method of partitioning. But the means of keeping load balance still needs further investigation.
4) The MTT is a simple and direct method. Although it has been successfully used in 2D it still needs more theoretical development. It is speculated that CAD method could provide a way forward.

Following partitioning, the next step is to implement the parallel procedure using an Euler/NS solver on an unstructured mesh. The major challenge is efficient message communication between sub-domains. An improved data structure must be constructed to make this.

References

[1] Pothen,H.Simon,and K.P-.Liou "Partitioning Sparse Matrices with Eigenvectors of Graphs" SIAM J. Mat. Anal. Appl., 11(3):430-452,1990
[2] Y.F.Yao "Simulation of Compressible Inviscid Flow on Adaptive Remeshing Unstructured Meshes" Dept. of Aerospace Engineering Aero Report 9424, University of Glasgow, 1994
[3] Y.F.Yao "A Navier-Stokes Solver for Laminar Viscous Flow on General Grid Topologies" Dept. of Aerospace Engineering Aero. report 9503, University of Glasgow, 1995
[4] Alan George & J.W.Liu "Computer Solution of Large Sparse Positive Definite Systems" Prentice Hall, Englewood Cliffs, 1981
[5] T.J.Barth "Aspects of Unstructured Grids and Finite-Volume Solvers for the Euler and Navier-Stokes Equations" AGARD R-787, 1992
[6] N.A.Verhoeven et al. "Interim Stage in the Development of a Parallel Mesh Generator" ACME UK'95 Jan.4-5 Oxford

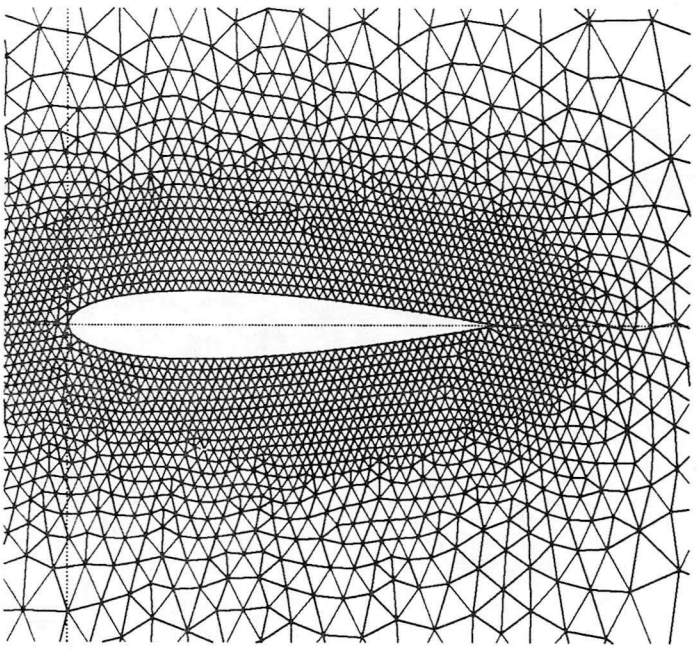


Figure 1 Unstructured Mesh around NACA 0012 generated by AFT method

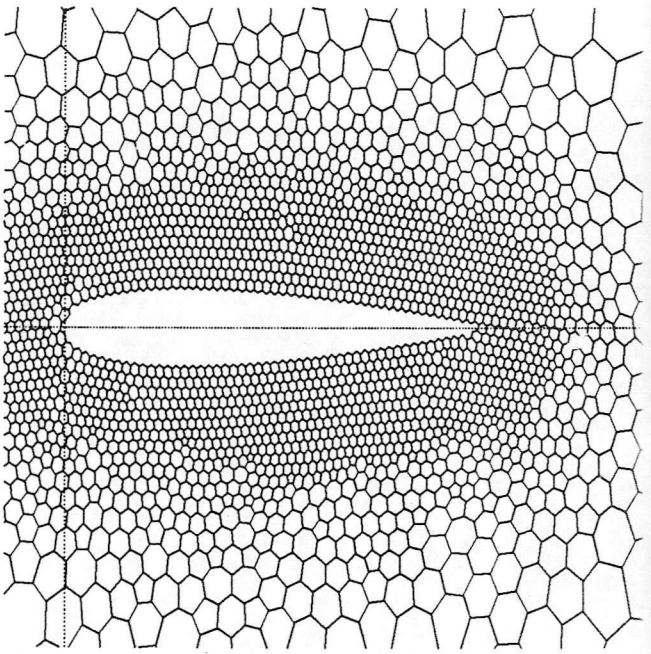


Figure 2 Dual Graph around NACA 0012 in Compariso to Figure 1

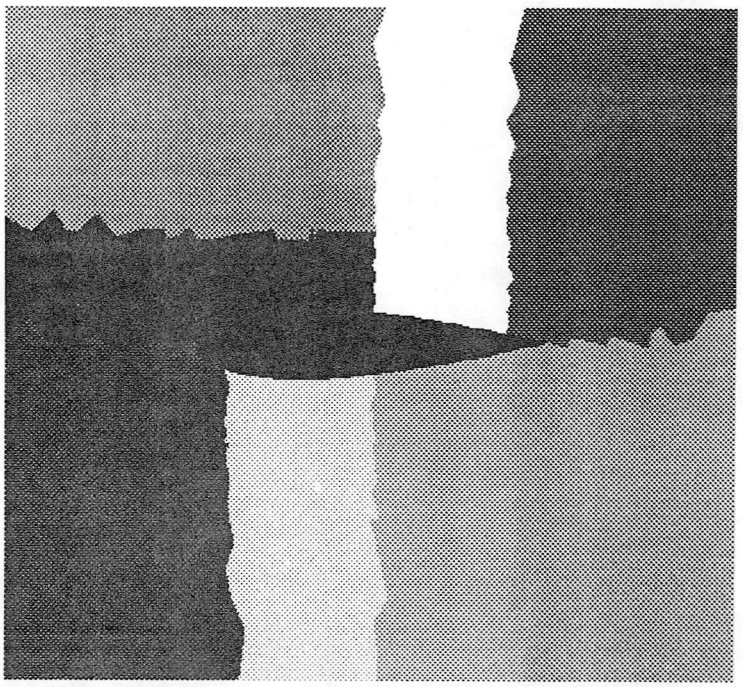


Figure 3 RCB 8 sub-domain partitioning without smoothing

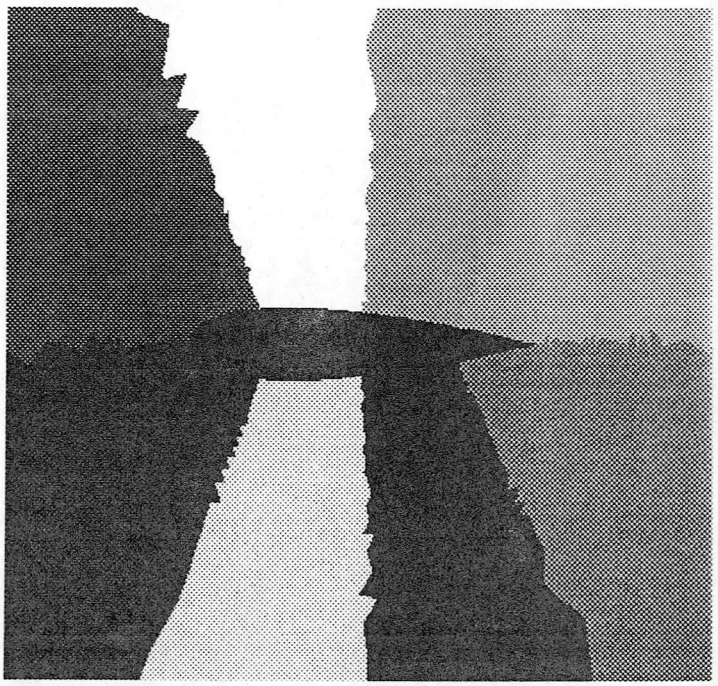


Figure 4 RAB 8 sub-domain partitioning without smoothing

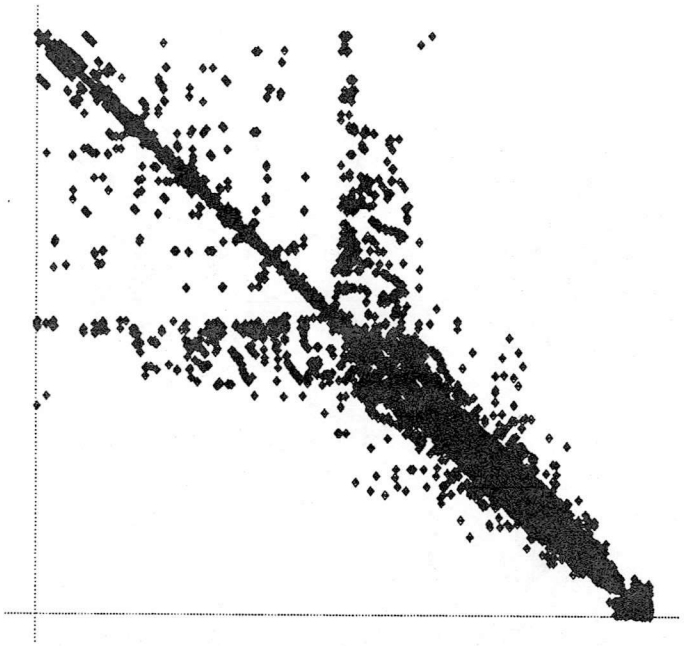


Figure 5 Nonzero entries of Laplacian Matrix from natural ordering

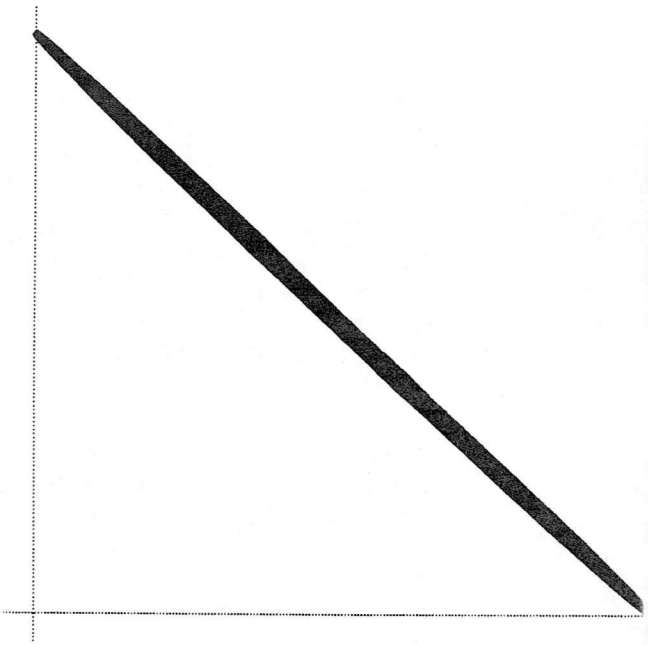


Figure 6 Nonzero entries of Laplacian Matrix after reverse Cuthill–Mckee ordering

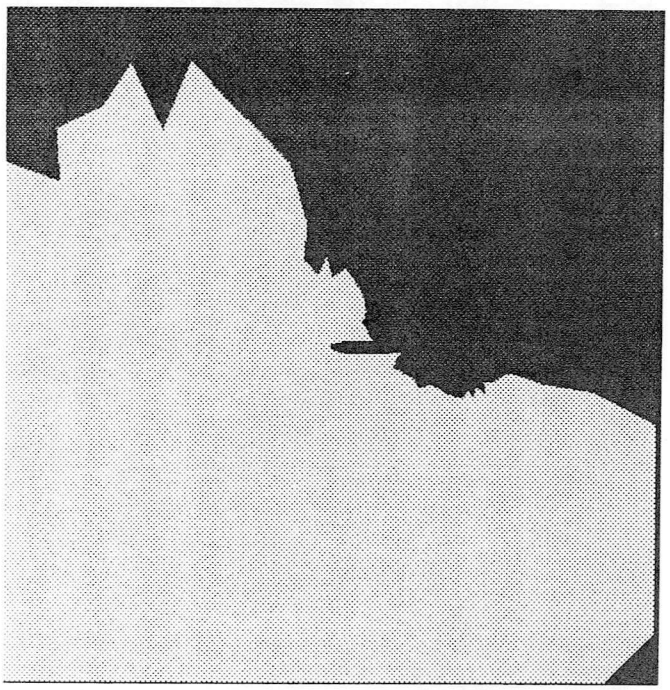


Figure 7 RGB 2 sub–domain partitioning without pre–ordering and smoothing

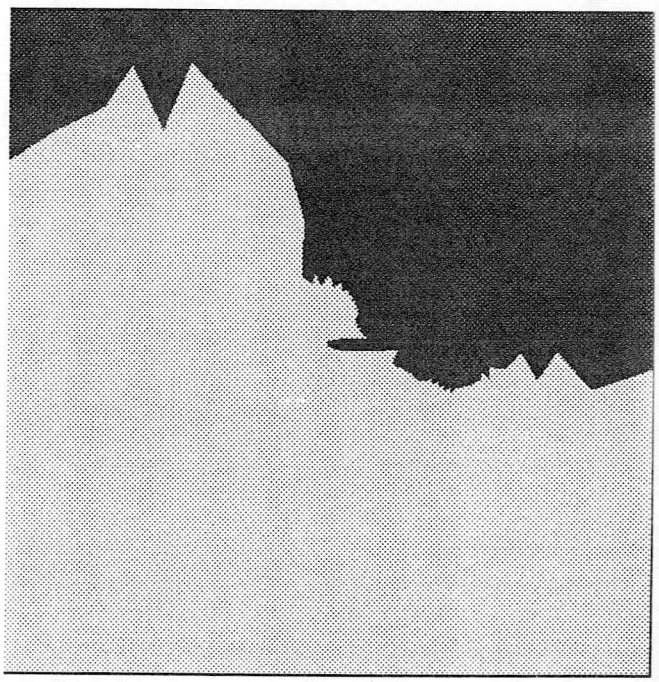


Figure 8 RGB 2 sub–domain partitioning with pre–ordering but no smoothing

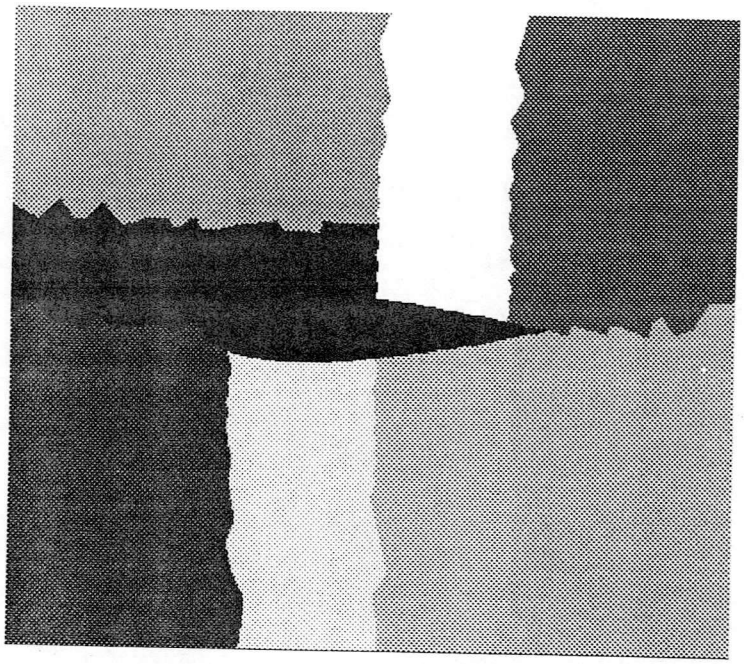


Figure 9 RCB 8 sub-domain partitioning with smoothing

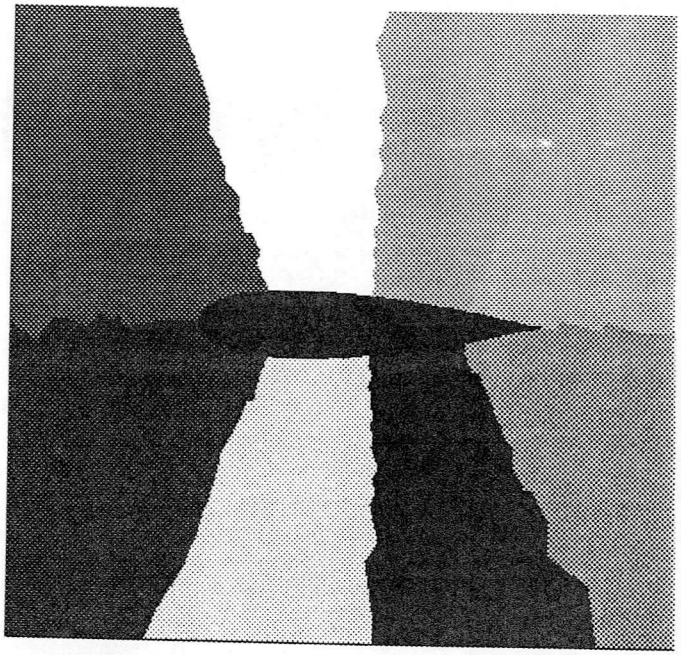


Figure 10 RAB 8 sub-domain partitioning with smoothi

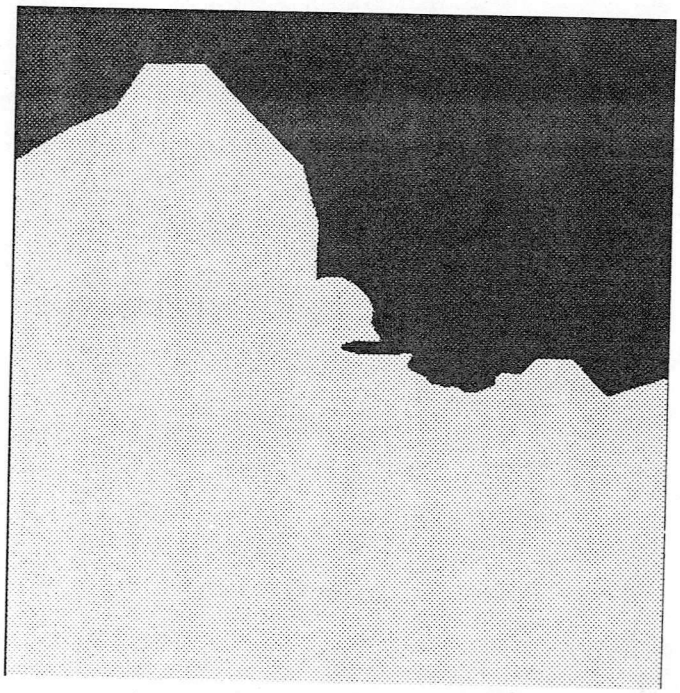


Figure 11 RGB 2 sub-domain partitioning with pre-ordering and smoothing

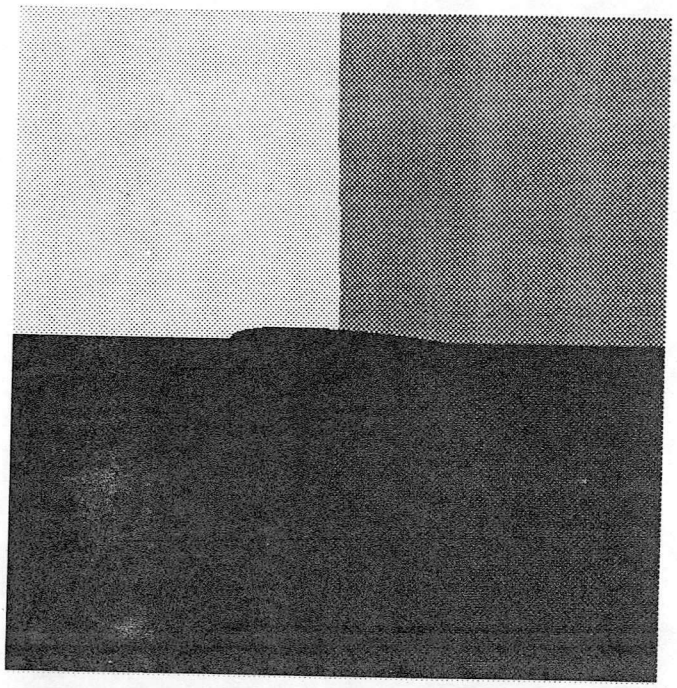


Figure 12 DDT 4 sub-domain partitioning by Delauney Triangulation Method

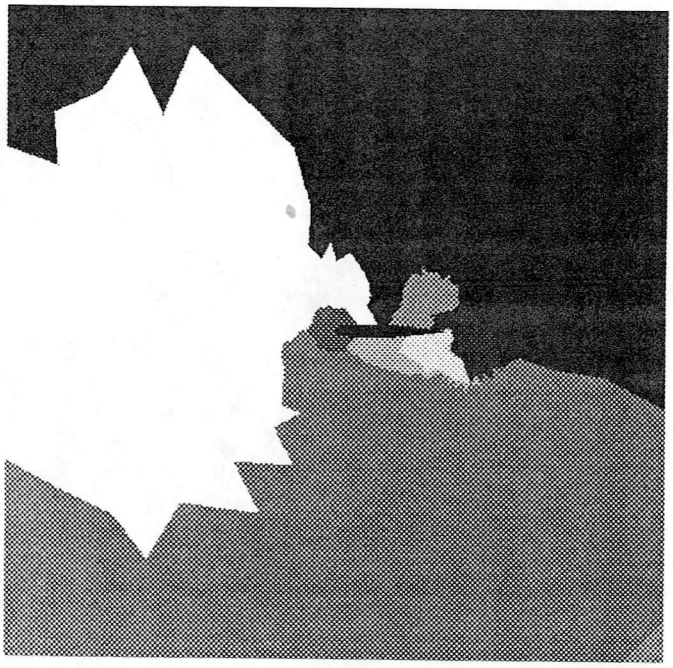


Figure 13 RGB 8 sub-domain partitioning without pre-ordering and smoothing

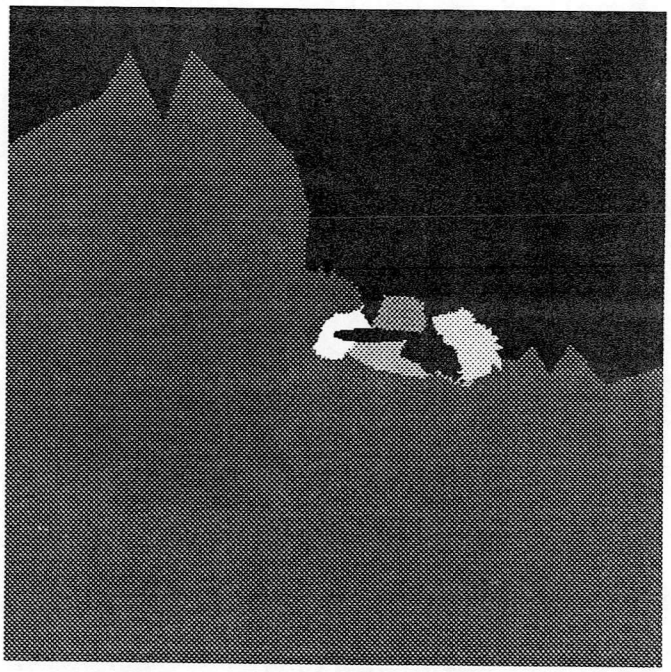


Figure 14 RGB 8 sub-domain partitioning with pre-ordering but without smoothing

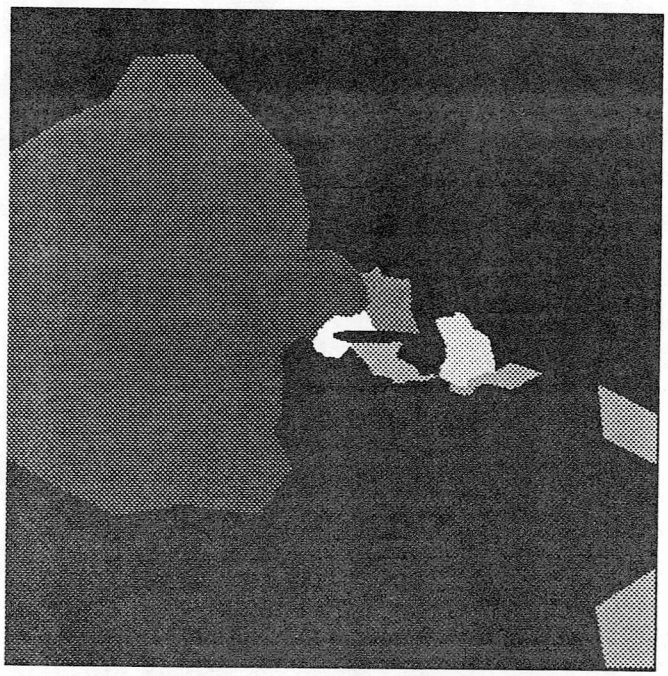


Figure 15 RGB 8 sub-domain partitioning without pre-ordering but with smoothing

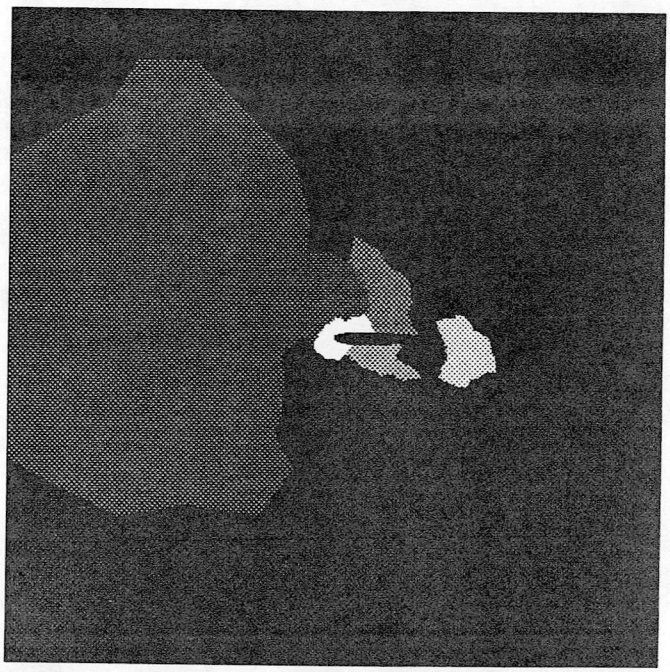


Figure 16 RGB 8 sub-domain partitioning with both pre-ordering and smoothing

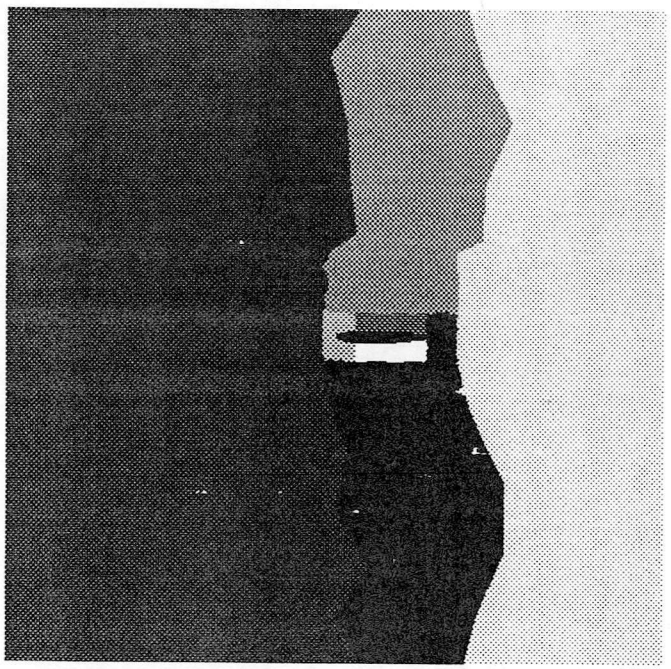


Figure 17 NACA 0012 8 sub-domain partitioning with MTT_square strategy

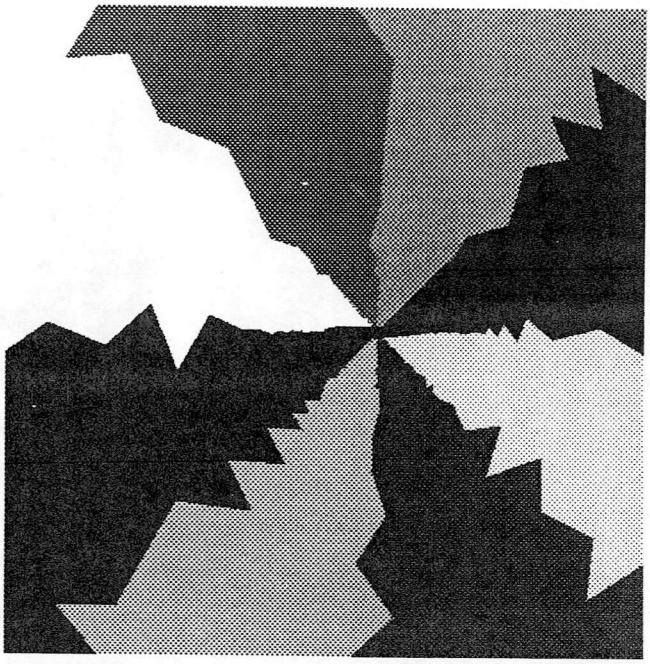


Figure 18 NACA 0012 8 sub-domain partitioning with MTT_triangle strategy

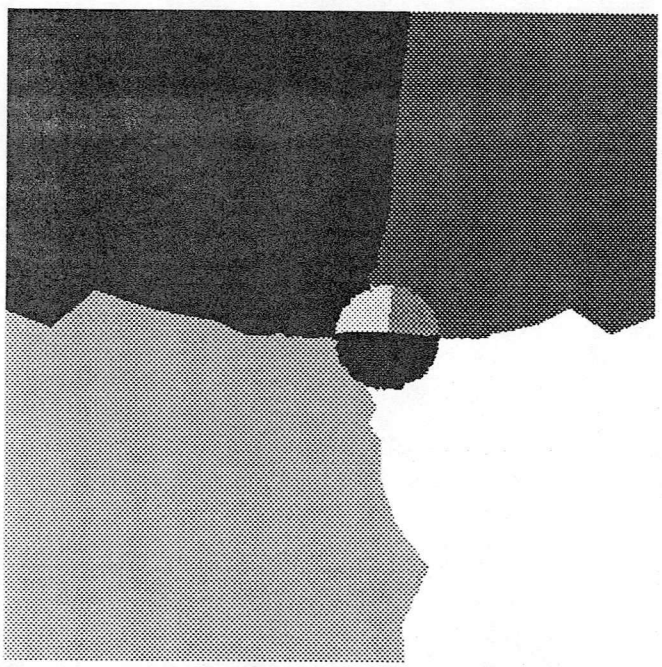


Figure 19 NACA 0012 8 sub-domain partitioning with MTT_circular strategy

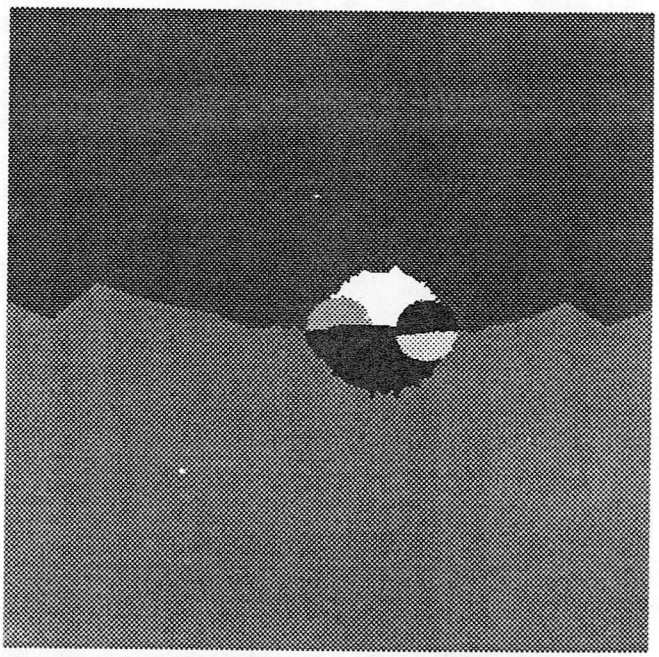


Figure 20 NACA 0012 8 sub-domain partitioning with MTT_curved line strategy

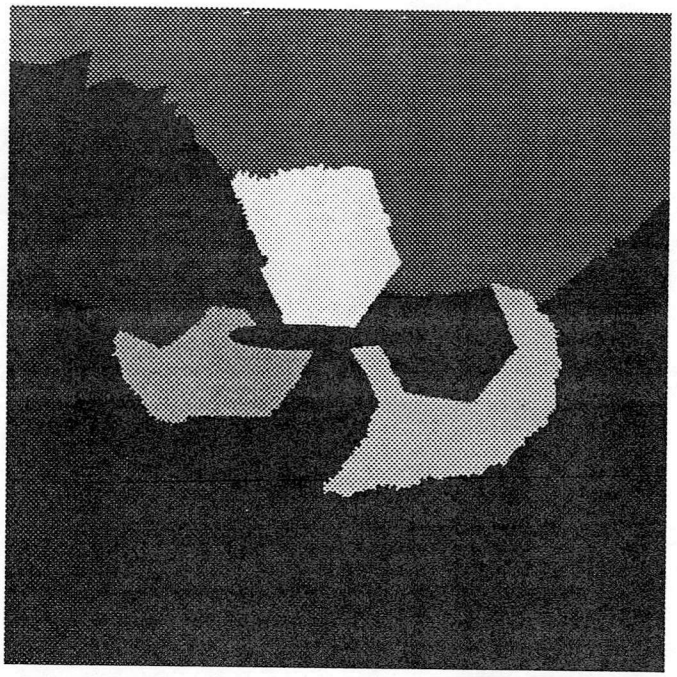


Figure 21 NACA 0012 finemesh 8 sub–domain partitioning by RGB without S & O

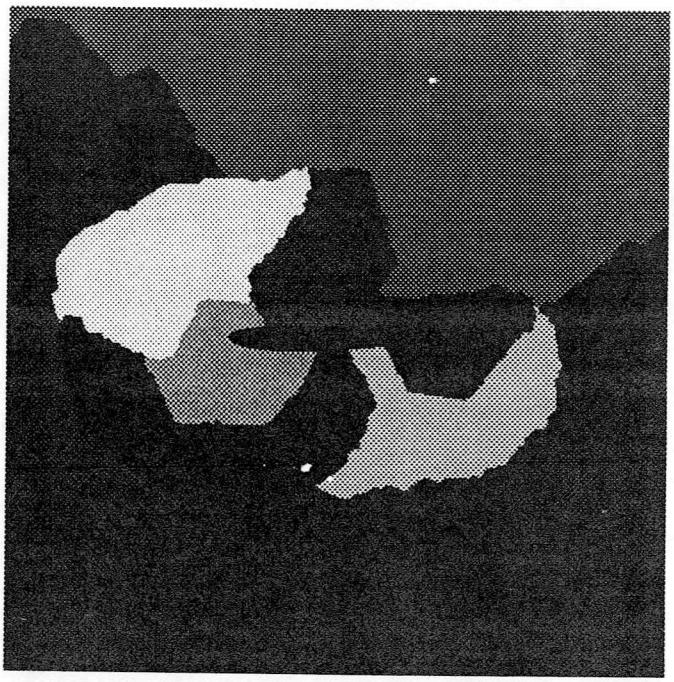


Figure 22 NACA 0012 finemesh 8 sub–domain partitioning by RGB with S & O

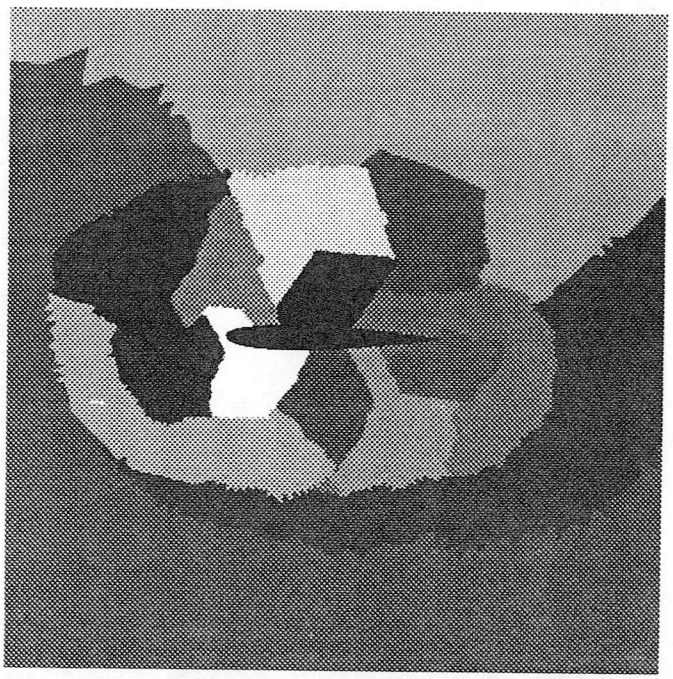


Figure 23 NACA 0012 finemesh 16 sub–domain partitioning by RGB without S & O

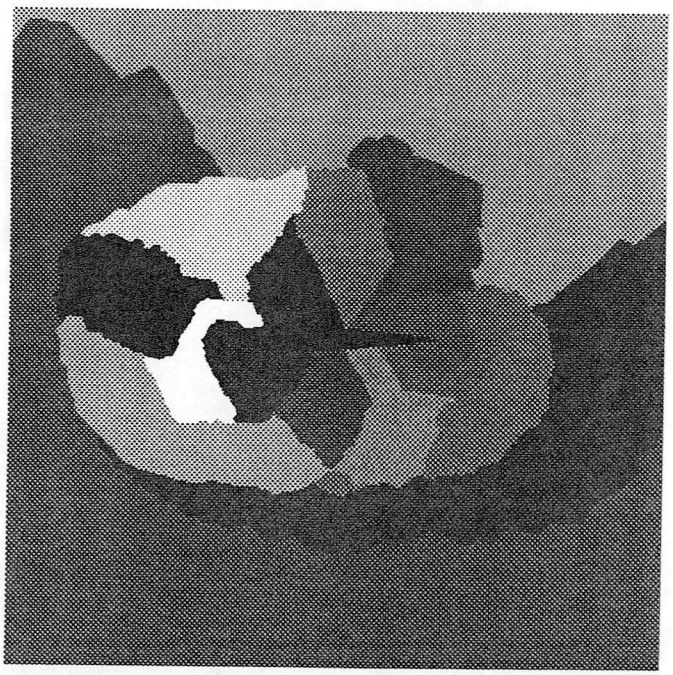


Figure 24 NACA 0012 finemesh 16 sub–domain partitioning by RGB with S & O

NOTE: S --- smoothing
O --- pre–ordering