ELSEVIER

Comput. Methods Appl. Mech. Engrg. 184 (2000) 467–483

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

Completely parallel compressible flow simulations using adaptive unstructured meshes

P. Leyland [a,*], R. Richter [b,1]

[a] *Departement de Mécanique, Ecole Polytechnique Federale de Lausanne, LMF-IMHEF, CH-1015 Lausanne, Switzerland*
[b] *Silicon Graphics, Inc., CRAY Research, Route des Avouillons 30, CH-1196 Gland, Switzerland*

Abstract

New and efficient completely parallel strategies for unstructured mesh solvers with dynamically adaptative meshes are developed. From highly complex academic research problems of unsteady compressible flow simulation to full scale aircraft problems, all are simulated with high performance results. Dynamic load balancing is obtained for simple but robust domain partitioners, who work directly on the network of processors. The code is hybrid as it allows elements of different types to co-exist in the same mesh.

1. Introduction

Unstructured grid CFD has become popular in the past years due to its flexibility for mesh generation and adaptation. Most parallelisation strategies for solving CFD problems are based implicitly on a so-called ''Master–Slave'' paradigm. The Master is responsible for the data management related to the in- and out-put, the domain decomposition and the subsequent data distribution to the slaves. Furthermore, if grid optimisation techniques are involved, they are in general also performed on the Master. The slaves then work on nearly independent entities and communicate with their neighbours mainly for domain boundary updates. Such a strategy has been used with reasonable efficiency for steady-state calculations, where static adaptation during calculation requires 5–6 passes to obtain an optimised final computational mesh [13]. In a recent paper, Mavriplis [9] has discussed the limitations and demonstrated the relative performance of unstructured calculations using heavily the master concept.

For the efficient handling of dynamic auto-adaptive grids required by unsteady flow simulations where a large number of different meshes may be created in a complete computational cycle, this method becomes completely inefficient due to memory requirements and computational efficiency. A completely parallel Slaves only paradigm using dynamically adaptive unstructured meshes is presented for compressible flow simulations. The complete algorithm, from the data transfer through the successive mesh adaptation, their optimisation up to the solution process is executed on the network of the processors, without communicating with an external ''Master'' or ''host node''. Data are first distributed arbitrarily to a certain number of processors depending upon memory requirements and careful and consistent renumbering procedures maintain transparency. Domain partitioning is then performed through the network, redistributing the

* Corresponding author. Tel.: +41-21-693-38-57; fax: +41-21-693-36-46.
E-mail addresses: penelope.leyland@epfl.ch (P. Leyland), richter@sgi.com (R. Richter).
[1] Tel.: +41-22-999-95-65; fax: +41-22-999-95-99.

0045-7825/00/$ - see front matter
PII: S0045-7825(99)00240-6

different partitions to the pre-requisite number of processors. This approach proves to be extremely efficient and also is the only possible way for high memory problems where the host concept becomes too costly.

The use of a simple but robust domain partitioning algorithm allows for a truly parallel dynamic load balancing algorithm. The difference between the global speedup and that of the solver remains less than 1% in the case of static mesh adaptation with up to 5–6 passes of adaptation. For dynamic adaptation where up to 500 different meshes can be generated, the time spent adapting and redistributing the mesh remains small, representing approximately 10–15% of the total time. The parallel adaptation is a true parallel algorithm as proven by the performance results for a variety of 2D and 3D test cases. The mesh adaptation algorithm is a non-hierarchical one [11,13]. This method allows more freedom for mesh optimisation, but renders more algorithmic complexity than other "background" mesh techniques [4–8]. In particular, the authors in [4,5] adopt a background octree structure for hybrid prismatic-tetrahedral meshes, and base their parallel load balancer on the subsequent subdivision of the octree. The adaptation follows the distribution of the octants, and hence "remembers" the master octant. The authors also suggest another kind of dynamic load balancing using a homogenisation of the run time per processor. In the case of non-hierarchical mesh adaptation, where no record of the successive filiation during the coarsening-refinement is kept, it is mandatory to associate these techniques to a careful layout of memory access. Frequent internal local reordering and renumbering of the mesh entities and their connectivities allows for efficient global distribution of both the mesh adaptation criteria and the physical resolution. The method is developed using the shmem programming model on the CRAY T3D, as this method avoids multiple buffers for inter-processor communication; data can be read/written from any PE by its memory reference address, or its local PE address (pointer), rather than physically be copied (and received). However, its extension to other message passing or shared memory models is possible. The code has been ported directly on the T3E without modification, and shmem is now available on other platforms (Origin 2000, for example).

2. Slaves only parallel CFD solvers with dynamic mesh adaptation

Implementation of unstructured mesh solvers on parallel networks requires a domain decomposition tool for partitioning the grid. The inter-processor communication requires copies of data on the boundaries of these partitions. A single buffer is used for all communication between the processors. The general methodology is independent of the element type: triangles, quadrilaterals, prisms, pyramids, tetrahedra and hexahedra, and also of the type of numerical scheme employed.

For mesh adaptation algorithms based upon the Master–Slave concept the initial domain decomposition, the subsequent local refinement, optimisation, and de-refinement processes are performed in serial on a host node and then redistributed back to the slaves. Such a strategy avoids what could be thought of as complicated network search and distribution allocation for the different entities, but cannot avoid the inevitable tasks of reordering and renumbering the geometrical and mesh definitions: nodes, edges, elements. This is necessary after each adaptation and the intermediate optimisation stages. These overheads become extremely high with an increasing number of processors, even for steady-state adaptation, as illustrated within the comparison below (Fig. 12). For unsteady and high memory 3D situations, these overheads become real bottlenecks.

Three-dimensional parallel dynamic mesh adaptation requires a coherent and optimal memory layout of data. A complete cycle: generation–resolution–adaptation–resolution requires also reprojection to the defining boundary surfaces. This is only feasible within an integrated concept; whereby the cycle *CAD definition* $\rightarrow$ *Surface Mesh* $\rightarrow$ *Initial Volume Mesh* $\rightarrow$ *Solve* $\rightarrow$ *Adapt* $\rightarrow$ *Re-project* $\rightarrow$ *Mesh* $\rightarrow$ *Solve* is written within a common environment. The present work treats the internal cycle *resolution–adaptation*; nevertheless data coherence has been conceived to be part of a complete cycle.

In order to minimise buffers, the layout of data throughout the network is optimised. A renumbering procedure is introduced within the partitions in such a way that each mesh entity has a relative record of its position within each submesh (processor). This means that the boundary values for one processor can be copied to neighbouring ones via a single buffer which maintains the same order as from the reordering procedure. In order to remain completely parallel, the internal nodes of each partition are considered *globally* (global pointer), and for the interface the contributions belong to **one** single domain (processor)

(local pointer). In general, there is no exchange of fluxes across the interfaces as these are automatically known when the fluxes are summed up globally.

The balance between dynamic load balancing and a higher quality of the partitions and their distribution is not evident, especially for dynamic grid adaptation, where the partitions can change rapidly and the time for the communication of data layout information becomes a challenge. For these reasons, a simple partitioning method is employed here, based on RCB.

3. Schemes on unstructured meshes

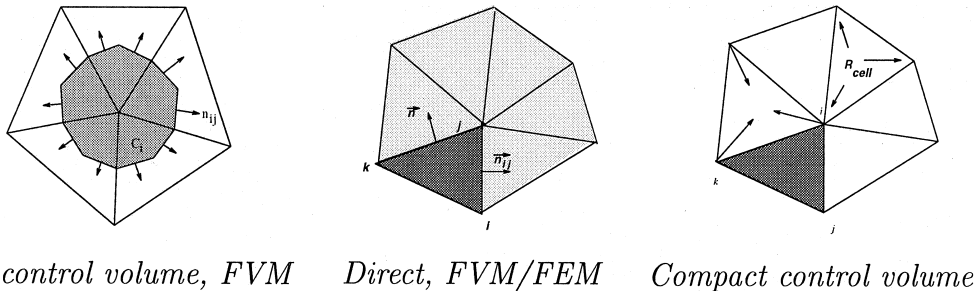
Numerical schemes for compressible flow on unstructured meshes are basically of two types. The first is based on finite volume schemes on equivalent control volumes, that can be the computational element (or cell) itself, a collection of neighbouring cells around the discretisation node or the so-called dual cell, see Fig. 1, [2]. These "dual" control volumes are the dual topology of the linear finite element simplex. They are constructed by taking the barycentres of the contributing elements to a discretisation node i. The second type are finite element type discretisations, which can be continuous approximations, stabilised by an equivalent upwinding and added discontinuity capturing terms, such as SUPG methods [1,3], discontinuous approximations, or new multi-dimensional schemes using wave decomposition arguments [16]. As for structured mesh formulations, the schemes can be considered as cell centred in the first case, whereby the numerical flux function defining the scheme is evaluated across the faces of the control volume, or cell vertex in the second. Cell vertex schemes involve weighted combinations of the cell residuals, as in standard finite element methods. In both cases the scheme is defined using the influence of the surrounding cells to a node. For truly compact schemes, all information is derived within the cell taking into account the flux balance across the edges/faces.

Considering the conservation form of the Navier–Stokes system:

$$\begin{cases} \partial W/\partial t + \operatorname{div} \cdot \mathscr{F}(W(\vec{x},t)) = S & \text{for } (\vec{x},t) \in \Omega \times \mathbb{R}^+, \\ W(\vec{x},0) = W_0(\vec{x}) & \text{for } (\vec{x}) \in \Omega, \\ +\text{boundary conditions} & \text{for } (\vec{x},t) \in \Gamma \times \mathbb{R}^+, \end{cases} \tag{1}$$

where W denotes the state vector, $\mathscr{F}(W)$ denotes the flux terms, and S the source terms.

The calculation domain $\Omega \times \mathbb{R}^+$ is discretised such that $\Omega_h = \mathscr{T}_h \approx \bigcup_i C_i$, where the C_i represent the computational control volumes. The notion of unstructured grid means that the number of neighbouring nodes for a given cell is not constant. The grid can however be triangular, quadrilateral (2D), tetrahedral, hexahedral, prismatic, pentahedral (3D), or be made up of other geometrical shapes such as diamonds. The same constraints of regularity and admissibility as for structured grids are necessary to maintain precision within the numerical discretisation. The overall grid, Ω_h can be *hybrid*, i.e. different element types can coexist within the same mesh (subdomain), as theoretically the solver can handle a mixture of several simplicial types. Structured hexahedral meshes present higher precision for precise tracking of forces and



Dual control volume, FVM *Direct, FVM/FEM* *Compact control volume*

Fig. 1. Different control volume types for unstructured meshes, illustrated here for triangular grids; the concepts remain identical for other polyhedra.

viscous effects, and unstructured tetrahedral solvers allow enhancement of accuracy of discontinuity capturing by adaptation.

The variational form of Eq. (1) for linearly varying test functions gives:

Let $\mathscr{V}_h = \{v \in C^0(\Omega), v|_e \text{ linear } \forall e \in \mathscr{T}_h\}$. Find $W \in \mathscr{V}_h^5$ such that $\forall \varphi \in \mathscr{V}_h$,

$$\int_{\Omega_h} \partial W/\partial t \varphi \, \mathrm{d}\vec{x} = -\sum_{j\in N(i)} \int_{\partial \mathscr{C}_{ij}} F(W)\cdot \vec{n}_{ij}\,\mathrm{d}\sigma - \int_{\partial \mathscr{C}_i \cap \Gamma} F(W)\cdot \vec{n}_\Gamma \,\mathrm{d}\sigma + \sum_{e\in \mathscr{T}_h} \int_e F^V(W,\nabla W)\cdot \nabla\varphi \,\mathrm{d}\vec{x}, \tag{2}$$

where $\partial\mathscr{C}_{ij}$ denotes the interface boundary of the dual cell around the node i with respect to a neighbouring node $j \in N(i)$. F denotes the convection fluxes and F^V denotes the viscous ones, which are always evaluated in a standard finite element way directly on the simplex.

The conservation equations are integrated over the control volumes.

The numerical scheme is defined by the numerical flux function which, for equivalent finite volume techniques, is the approximation of the flux integral in the normal direction $\vec{n}_{ij}$ across the interface between cells (elements or dual control volumes)

$$\int_{\partial\mathscr{C}_{ij}} F(W)\cdot\vec{n}_{ij}\,\mathrm{d}\sigma = \Phi_{F_{ij}}(W_i, W_j, \vec{n}_{ij}),$$

and can be expressed as a redistribution of the the flux integral to the nodes in a finite element/compact formulation

$$\int_e F(W)_i \nabla\varphi_i \,\mathrm{d}\vec{x} \equiv \sum_{i\in\{e\}} \Phi_e^i$$

with

$$\Phi_e = \oint_{\partial e} F(W(t,x))\vec{n}_e \cdot \mathrm{d}\vec{x},$$

where ∂e defines the surface of the cell e with inward normal $\vec{n}_e$. The flux or residual $\Phi_e \equiv R_e$ is then distributed to the nodes following a certain residual distribution scheme [1]. This can be represented by a distribution matrix D_e:

$$R_i = \frac{1}{|C_i|}\sum_{j,k\in C_i} D_e^k Vol_e R_e.$$

Conservation implies $\sum_{i,j\in V_e} D_e^k = Id$. A centred scheme is obtained with $D^k = \frac{1}{3}$ (thus requiring artificial viscosity for stability), whereas Lax–Wendroff, SUPG and multi-dimensional schemes correspond to more complex developments for this matrix.

Three typical schemes on unstructured meshes are considered here: two based upon a spatial approximation using the Galerkin finite volume method on the "dual" control volumes of the underlying P1 Galerkin finite element approximation, and one compact scheme of Lax–Wendroff type scheme. The dual topology concept is valid for any type of finite element simplex, and the variational formulation is exact for the convective fluxes over this dual topology for linear approximations. For the dual topology, a finite volume Jameson type scheme and an Osher approximate Riemann solver are considered. Second order accuracy and discontinuity capturing is assured with dissipation terms coming from appropriate limiters calculated on the upwind (downwind) elements for the Osher flux, and an accurate dissipation construction for the Jameson scheme. In these cases, information required to compute the numerical dissipation $\Phi^*_{F_{ij}}(W_i, W_j, \vec{n}_{ij})$ is obtained via virtual nodes i^*, j^* or upwinded and downwinded elements to the interface $\partial\mathscr{C}_{ij}$ (see Fig. 2). For the compact Lax–Wendroff type scheme, information is limited to the elements connected to a given node i. Numerical disspation is included within the distribution matrix which is a

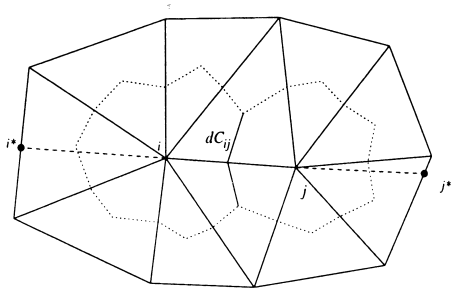


Fig. 2. The information needed to compute the dissipative terms and second-order accuracy is obtained via virtual nodes i^* and j^* or up- and down-wind elements.

function of an approximation of the jacobians [15]. An edge-based artificial viscosity term is added for more accurate shock capturing [10].

4. Parallel implementation of the numerical schemes

For completely parallel implementation of CFD solvers on unstructured grids, the computational domain is partitioned into subdomains directly on the network of processors. This is mandatory for dynamic mesh adaptation algorithms where the mesh is being adapted continuously during calculation. Although each processor may in theory have more than one submesh, for the algorithm presented here there is a one to one correspondence. Common nodes between adjacent submeshes are repeated within each submesh, each partition is therefore an *independent* structure. Data locality is hence assured.

Data are exchanged within a boundary layer of information on each side of the inter-domain partition's boundary. There are several ways of defining these layers, and the reconstruction of global information from each partition depends to some extent on the underlying numerical algorithm. Essentially two ways of defining this boundary exchange can be considered: the first uses a layer corresponding to the actual interface elements on each side, *Direct*, the second by considering the *Dual* decomposition. For the Direct partition, each element lies in one partition, only interface nodes need to be duplicated. For the Dual partition, the element layer is duplicated and each node lies within a distinct partition. One or two layers of *ghost* information are necessary in both cases for precise reconstruction of the fluxes, depending on the schemes, Fig. 3. The compact schemes require information of the common interface elements to be exchanged, whereas the finite volume type schemes require a second layer in order to reconstruct the nodes i^* in Fig. 2. The relative efficiency of each method is not so dependent on the type of scheme – compact cell vertex or equivalent finite volume but rather a balance between precision and the number of interface nodes. For the non-overlapping Direct partition, the computational fluxes are not exchanged between

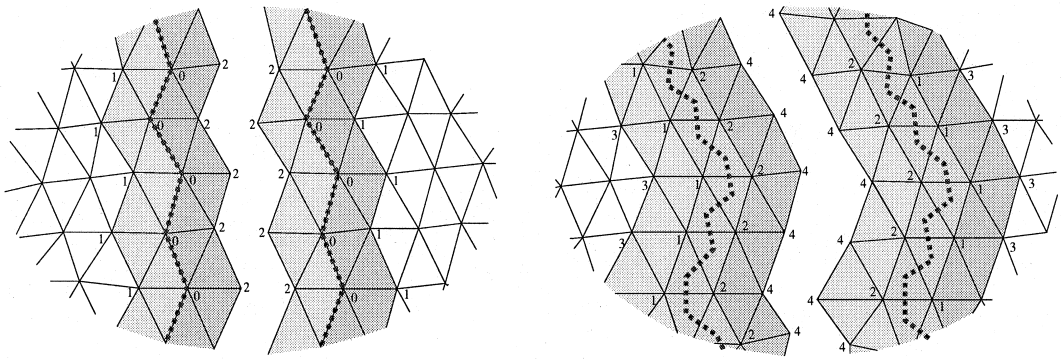


Fig. 3. Direct (left) and Dual (right) partitioning.

processors, as they are exchanged automatically globally. For the Dual type of partition, the physical nodal entities on the shared layer are exchanged.

The two kinds of interfaces are illustrated for a 2D supersonic shock reflection problem on a highly adapted grid of 25 168 nodes and 49 932 elements. Three schemes of varying computational complexity are tested, for two equivalent finite volume schemes on dual control volumes of opposite complexity – the centred Jameson type scheme and a second order Osher's approximate Riemann solver, and an element compact Lax–Wendroff scheme, Fig. 4. A balance between inter-domain communication update cost and computational cost becomes less important for the more complex schemes. The analysis given for a 2D test case gives the most unfavourable situation as possible when the number of processing elements (PEs) increases the ratio, interface nodes/interior nodes, becomes smaller, communication overheads are thus high. For 3D cases, the analysis is valid with lower communication overheads for high number of processors.

Different communication libraries are also compared on the same figures for the CRAY T3D. The standard PVM, the optimised P-PVM and MPI message passing models allow portable programming, and are particularly adapted to Master–Slave concepts. The shmem – *Explicit Shared Memory* model – communicates via the shared memory of the network. This method allows read/write operations from any processor by the memory reference address or explicitly by referencing a processor. Processors can access one another individually or collectively throughout the network. Single stream buffers provide fast communication which helps the programming of the completely parallel approach. Such libraries are becoming available on most multi-processor platforms. Again, the choice of a programming model becomes less important for increasing computational complexity, as shown in the same Fig. 4, at least for steady-state calculations.

The partitioning algorithms considered have thus been restricted to the most simple ones: essentially RCB and RGB – recursive coordinate and geometric bissection, as their fast execution directly on the network is feasible allowing dynamic redistribution of the new submeshes during the adaptation.

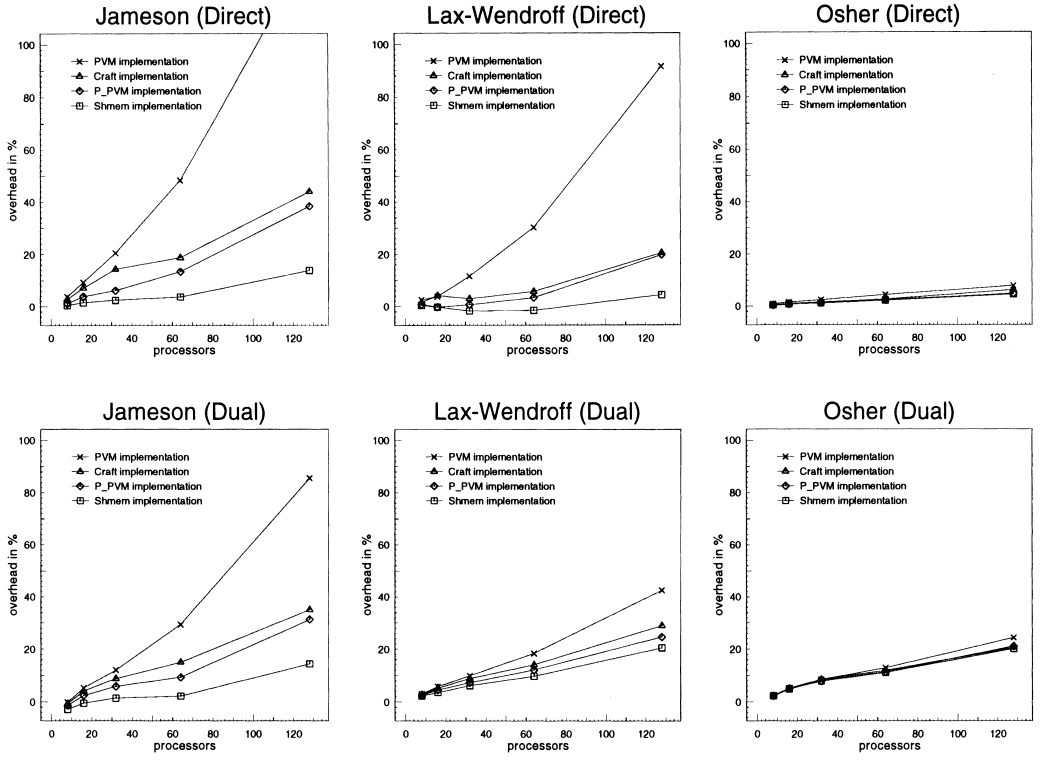


Fig. 4. Communication comparisons for two finite volume schemes on the dual topology and one cell vertex compact Lax–Wendroff scheme. The cost is represented by the percentage of communication overhead as a function of the number of processors.

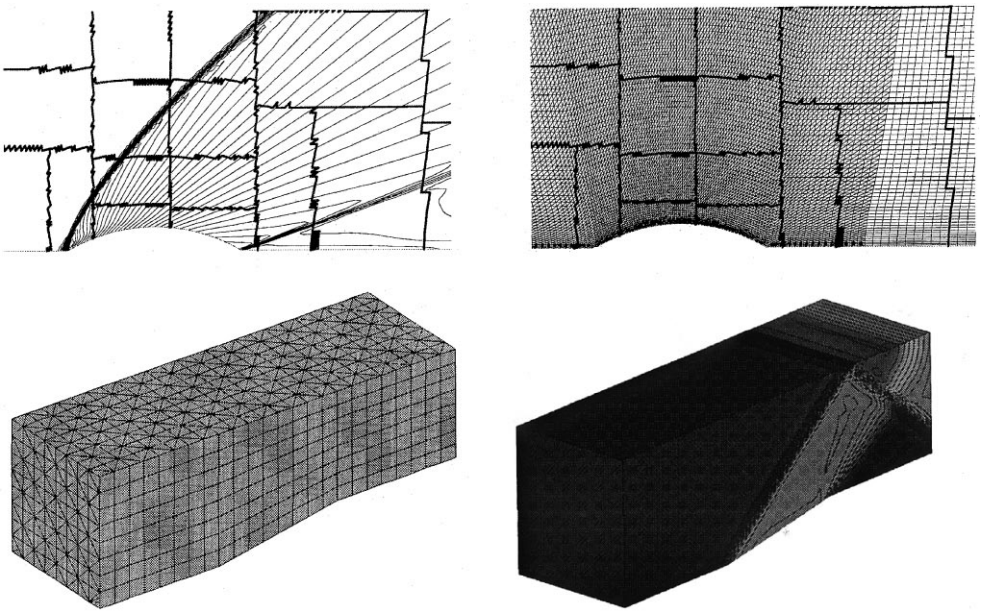


Fig. 5. Supersonic flow in channels, using a hybrid mesh, and with prismatic elements. Iso-density lines are shown. The calculation took 10 min on 16 processors of a T3D for a mesh of 99000 nodes/184320 cells.

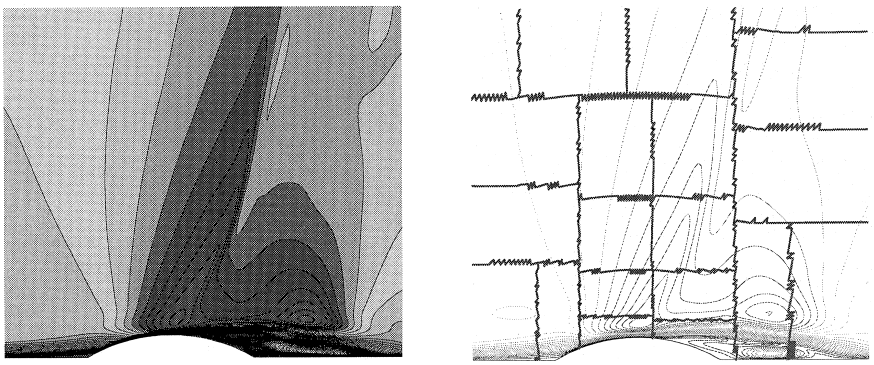


Fig. 6. Iso-Mach number distributions for unsteady transonic $\mathrm{Mach}_\infty = 0.8$ viscous flow in over a bump in a channel, using a hybrid mesh of 36120 nodes and 50176 cells. The shock–boundary layer interaction is well captured, and the recirculation bubble grows in accordance with experiments.

The same concepts remain valid for hybrid grids, where several element types co-exist. This is illustrated by flow over a bump in a channel; first for a supersonic inviscid solution, Fig. 5, second by the unsteady viscous flow for a transonic regime, where the shock wave – boundary layer interaction, and a growing recirculation bubble is well captured, Fig. 6.

The memory layout of the mesh entities, nodes, edges, cells is designed to facilitate the communication and to minimise the number of intermediate information buffers for the inter-processor copies. Boundary values for one processor are copied to the neighbouring processors directly in memory. The totally parallel approach requires systematic reordering and renumbering of the *local* data to maintain data locality and performance issues for the inter-processor updates. This becomes of prime importance for the dynamic parallel mesh adaptation paradigm. In Section 8, the results show that the time spent in repartitioning diminishes per processor, giving a dynamically load balanced operation.

5. Coherent memory/data layout

The completely parallel environment discussed in the previous section requires a parallel I/O system and a globally coherent data structure. The dynamic mesh adaptation demands frequent intermediate writings to the I/O, due to data alteration, reordering and renumbering during calculation. This is taken care of by using a data file archival/retrieval system, DF3, which has an internal dynamic memory manager.

The layout of the geometrical entities such as the edges, NSEG, the connectivities, NCELL, and so on respect the node, edge and cell ordering of Fig. 6; for example the array NSEG(−3:1) corresponds to the layout of Fig. 7. At each stage of modification of the local PE entities, the global address and the local flagging must be re-established to maintain consistency according to these numberings. For example, during the parallel partitioning phase, a colouring arguement is used to identify the nodes and the cells with their new processor (PE), Fig. 8.

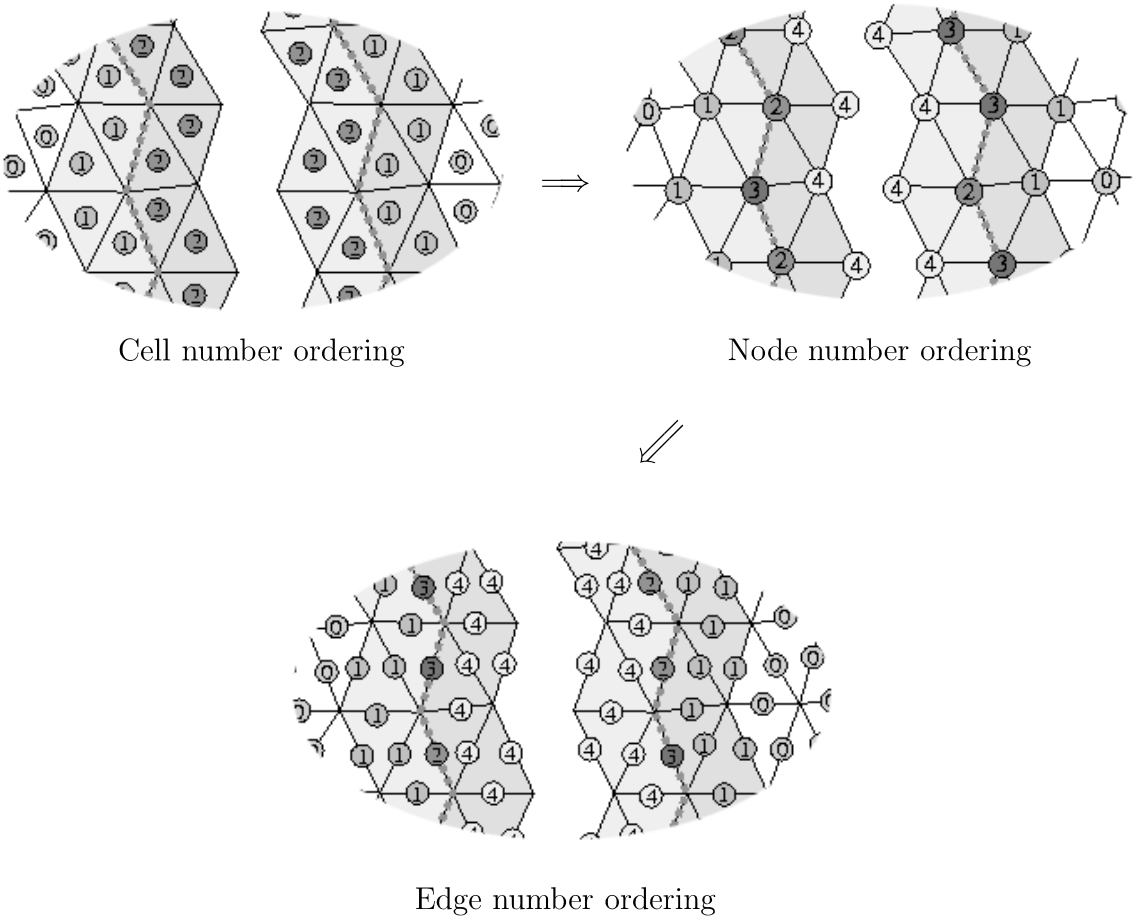


Fig. 7. Coherent reordering and renumbering phase of the mesh boundary entities after each mesh manipulation.

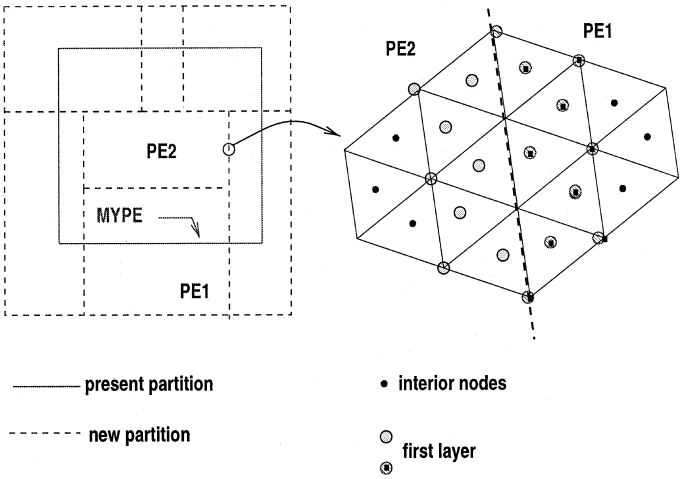


Fig. 8. Parallel colouring after dynamic domain partitioning according to the coherent numbering flags.

6. Completely parallel mesh adaptation

One of the main advantages of unstructured grids are their flexibility for the generation of grids over singular complex geometries, and their possibilities to adapt dynamically the mesh by localised refinement/de-refinement during the calculation in a conformal manner with no hanging nodes. A new adaptation strategy using a non-hierarchical data structure has been developed [11], which allows greater flexibility for optimisation of the quality and control of the different levels of adaptation. The adaptation algorithm considers local mesh refinement and coarsening, and performs smoothing, stretching and mesh alignment without the use of a background mesh. The criteria and constraints within this algorithm are based not only on physical properties for tracking the discontinuities and other preponderant phenomena, but also upon new structural properties of the mesh.

The completely parallel dynamic mesh adaptation algorithm takes place entirely on the network of processors. No one processor is pre-defined as a "Master" on which partitioning of the new meshes and the mesh adaptation modules are performed, all stages from the initialisation, through the adaptation and the

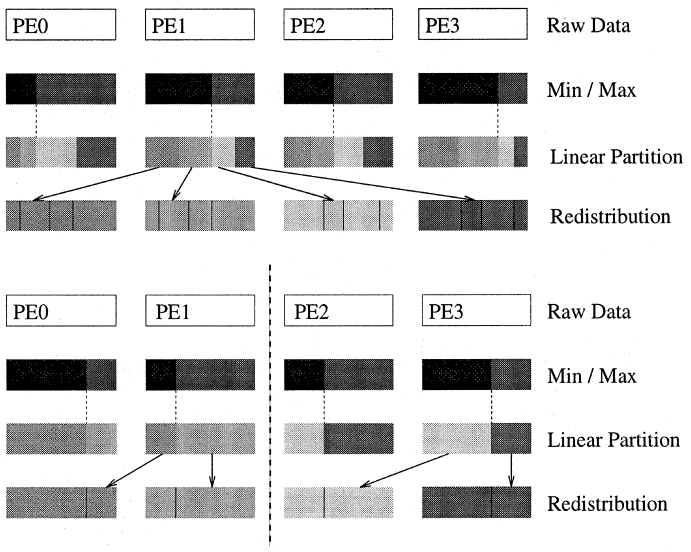


Fig. 9. Parallel sorting algorithm for dynamic domain partitioning with RCB.

associated repartitioning are executed on the network. The stages encountered with completely parallel mesh adaptation are summarised as follows:

1. *Initial input file*. This file is transferred continuously onto the parallel network, filling up a first layer of neighbouring processors with the mesh entities – nodes, element connectivities, boundary patch definitions – until completion. In the case where the memory requirements are limited, these entities are first split by 4, or 8 and so on. This number of PEs is independent of the number of PEs that will be used for the calculations. The data are then dispatched to the working number of PEs.

2. *Domain decomposition*, *reordering*, *renumbering*. Once the raw data have been distributed, the grid is then partitioned using a straightforward partitioning algorithm (RCB, RGB or RIB) *on the network*. The mesh entities are marked out together with the appropriate boundary interface entities which will be copies from neighbouring partition ones. Then all entities are reordered and renumbered according to the new internal structure. This coherent numbering allows a direct copy via a single buffer during the communication phase, Fig. 7.

3. *Initialisation and solution*. Once the initial partition is set up, a first solution is calculated throughout the processors. The different criteria for mesh adaptation (refinement or de-refinement) mark out the selected edges for adaptation independently of their locality.

4. *Mesh adaptation phase*. During the phase of mesh adaptation using non-hierarchical local refinement, and de-refinement, no history of the filiation of the mesh entities is kept. This requires consistent renumbering and reordering after *every* structural or geometrical modification within a single adaptation cycle (criteria marking, de-refinement, refinement, smoothing and stretching). However, using a suitable communication channel such as the shmem programming model, the overheads of these repeated operations

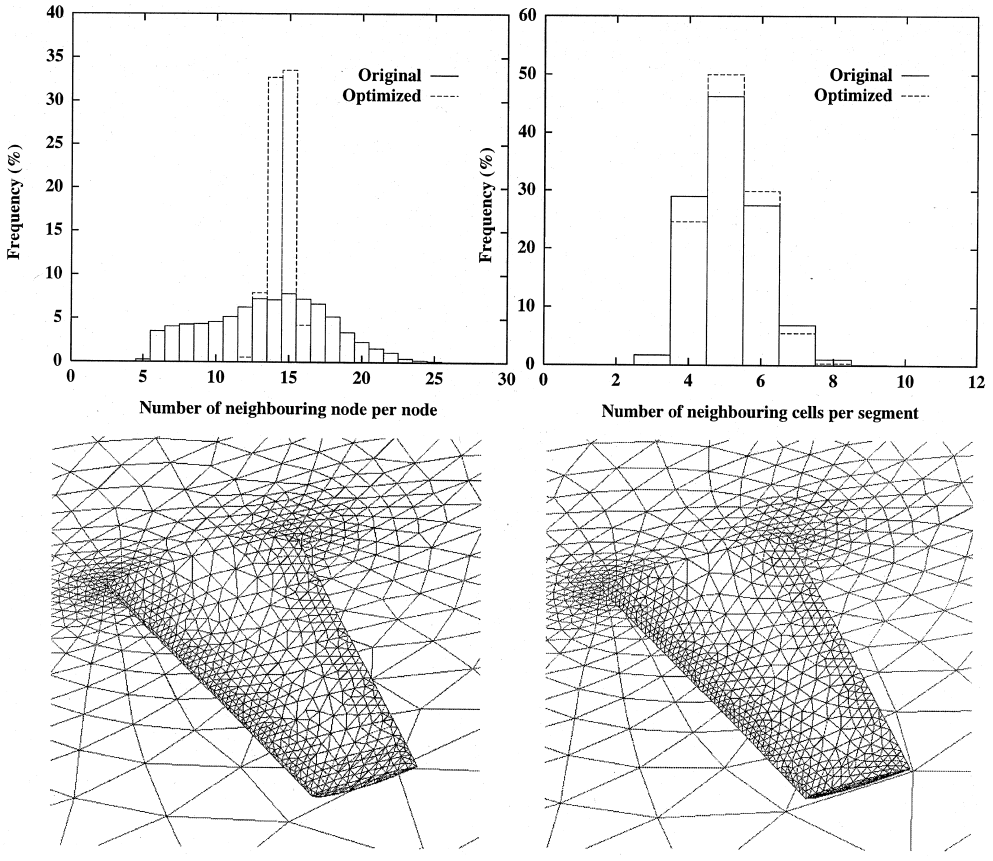
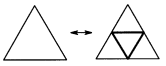


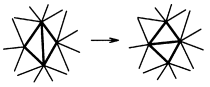
Fig. 10. Quality control optimisation for the ONERA M6 wing, the original mesh is on the left (below).

within one complete adaptation phase remain minimal for this "Slaves only" completely parallel paradigm (see Figs. 12 and 16). The different stages are:

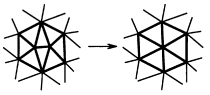
- de-refinement respecting boundary patches
- local refinement



- reorder and renumber
- structural optimisation – diagonal swapping



- reorder and renumber
- structural optimisation – edge collapsing



- reorder and renumber
- stretching, smoothing, regularisation

(For details of the these mesh adaptation methods, see [11,12,14].)

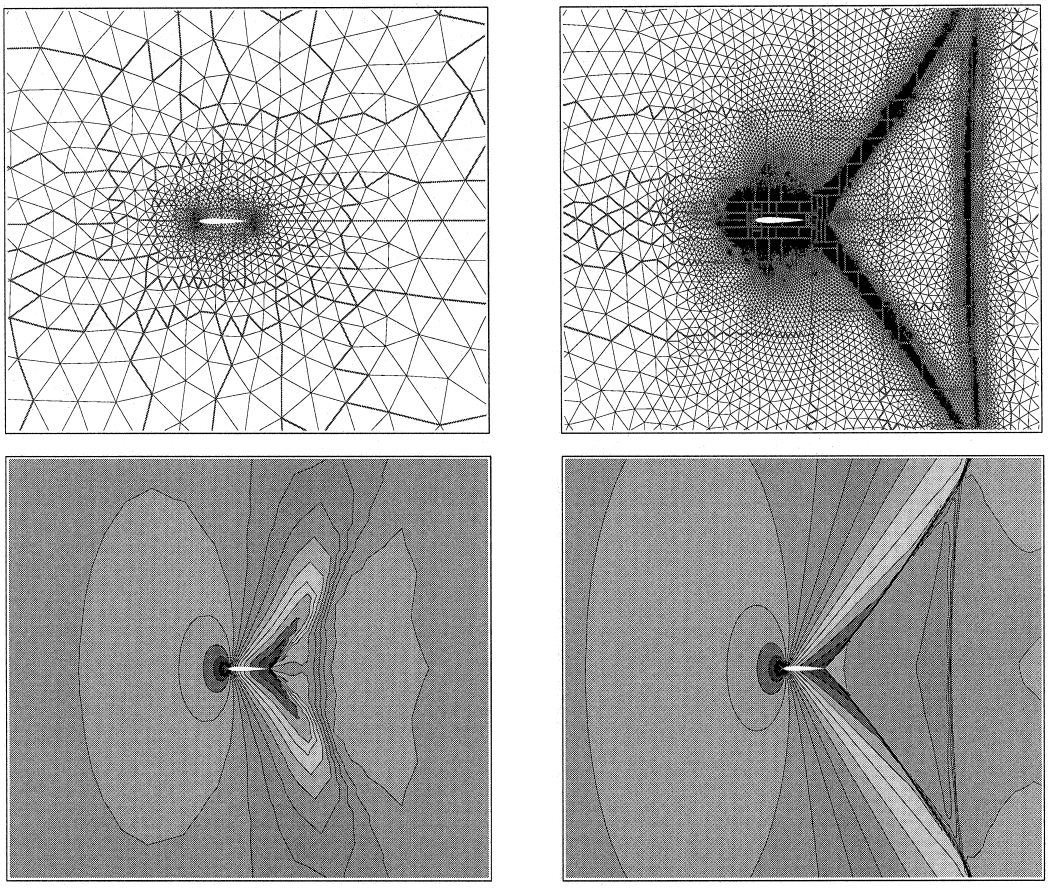


Fig. 11. Initial mesh and solution with 1704 nodes, 3332 elements and final mesh and solution (iso-Mach) after 5 cycles of parallel adaptation for the flow over a Naca 0012 airfoil at $M_\infty = 0.95$, $\alpha = 0.0°$.

5. *Partitioning – return to* [2]. It is now necessary to proceed to a new partition of the resultant mesh for redistribution and further calculation. The partitioning with RCB or RGB on a parallel network comes down to a parallel sorting algorithm. This is resumed in Fig. 9. Each PE keeps part of the original data and redistributes the remainder to its neighbours. The graph of the co-ordinates of each cell centre in each physical dimension is split around a pivot which is determined in the following way:

- Find the direction that is the most spread out.
- In this direction, the co-ordinate which has the same number of cells on all sides is marked out. On each PE the cells are organised on one side or another of this provisional pivoting value. The most populated side is split succcessively until the value of the pivot is attained. A global summing operation using shmem is required to ensure that this local value splits the cell numbers equally.
- On each processor, the cell centres are organised with respect to their global address and redistributed as shown in Fig. 9. When partitioning, the left and right parts are divided successively by $\{\frac{1}{2} * \mathrm{NPE}\}$, and this mean value is changed until the left and right parts are equilibrated.
- Once this operation is completed, the nodes and cells are marked (coloured) according to their "new" PE. The new boundary nodes are retrieved by looking if there is a difference between the colour of the cell and its nodes, if this is the case the node (and hence the cell) is a partition boundary one, and their addresses are augmented accordingly.
- The cells are renumbered taking into account their new position, then the nodes are renumbered accordingly.
- The domain interface nodes at level "0" are now renumbered, and finally those at level "1", according to the convention introduced in Fig. 7. This requires boundary information to be spread across neighbour-

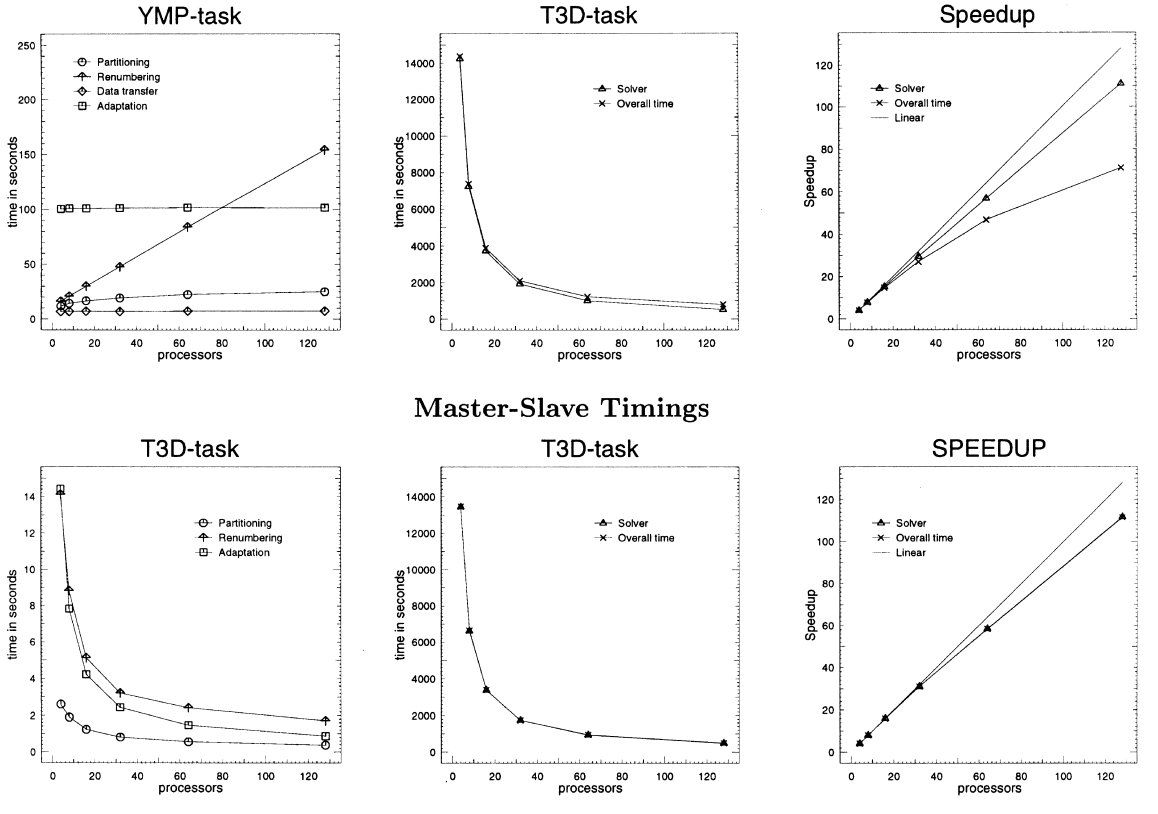


Fig. 12. Comparison of the performance of the Master–Slave and Slaves-only paradigms for the fish-tail shock flow over a NACA 0012, calculated with five successive adaptations from an initial mesh of 1704 nodes, 3332 elements to a final mesh of 54472 nodes and 108628 elements. Such a mesh is necessary to capture the normal wake shock in the correct position; $M_\infty = 0.95$, $\alpha = 0°$.

ing cells. Particular care is taken to distinguish different kinds of adjacent physical boundary patch cells with the PE boundary levels.
- The cell segments are created and renumbered.
- Finally the values of the domain interface nodes are ordered consistently, and an internal dimensioning for the PEs entities re-established.

7. Mesh optimisation

A highly important task within unstructured mesh adaptation is the optimisation of the meshes to eliminate degenerate configurations, and to obtain more acceptable quality. This procedure has been performed successfully within the parallel paradigm. The optimisation procedure enables a reduction of the number of nodes and cells of up to 20%, increasing the regularity of the global mesh. These procedures regulate the number of neighbours per nodes and edges, which removes a large number of flat elements. The optimisation of a coarse Euler mesh for the M6 ONERA wing is given in Fig. 10.

8. Results and comparisons of the adaptation paradigms

Efficiency of the completely parallel ''Slaves only'' mesh adaptation paradigm can be illustrated for a 2D steady-state computation of a transonic flow over a NACA 0012 airfoil which presents a ''fish-tail'' shock formation at its trailing edge, for which a normal shock is stabilised within the wake, only once an appropriate level of adaptation has been achieved, Fig. 11, [13].

From Fig. 12 it can be noticed that for the Master–Slave paradigm, the renumbering process becomes increasingly inefficient with higher number of processors: up to 80% overhead. For the Parallel adaptation

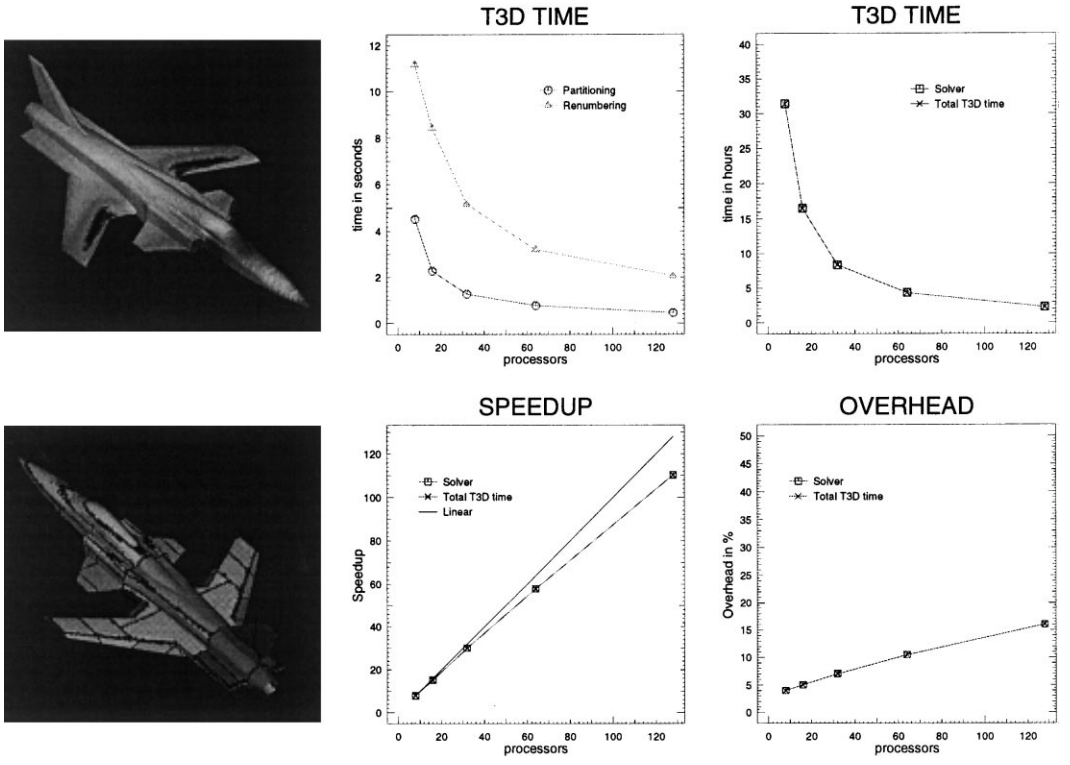


Fig. 13. Left: The partitions the pressure field on the surface for a laminar Navier–Stokes solution for the X29 aircraft, at $M_\infty = 0.75,\ \alpha = 5^\circ \rightarrow 37^\circ,\ \mathrm{Re}_\infty = 10^6$, on an optimised mesh of 136787 nodes and 726713 elements; Centre and right: Performance timings on the CRAY T3D.

paradigm, the global overhead for adaptation remains less than 1% and the speedup obtained becomes a speedup of *computation* instead of a measure of data manipulation as in the Master–Slave case.

Exactly the same conclusions can be drawn for the same kind of calculation in 3D. This can be illustrated by the timings of the decomposition and renumbering process of a totally parallel calculation of a transonic flow over the experimental complex unstable manoeuvrable X29 aircraft, Fig. 13. Indeed, the time for renumbering/partitioning is now inverted compared to the 2D case, due to the better balance between boundary to interior nodes for 3D calculations. The mesh has almost 800 000 tetrahedral cells, and is relatively structured near the wall to allow for viscous laminar flow to be simulated. The size of the mesh gave no difficulty in being initialised on even a few number of PEs (2–4) on the parallel network; then the subsequent redistribution, decomposition and calculation has a low memory overhead, whereas for the Master–Slave paradigm a high memory charge was required. A computational polar varying the angle of attack from 5° to 40° was accomplished, each individual solution taking 2–4 h on 64 processors of the CRAY T3D, for a computational mesh of 137 103 nodes and 723 520 elements. The scalability is optimal, and the complex physical phenomena are captured despite the coarseness of the grid for viscous flows, Figs. 13–15.

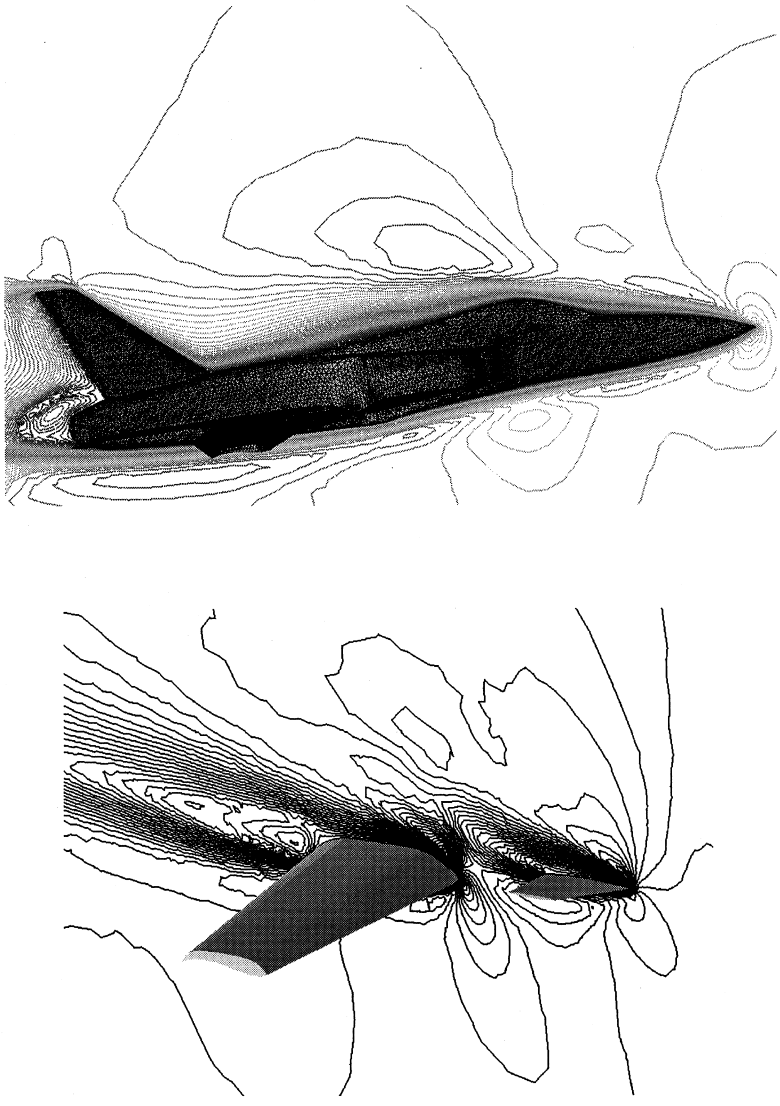


Fig. 14. Iso-Mach lines in the symmetry plane and across the forward swept wings with thin supercritical airfoil shapes of the X29 with laminar transonic conditions.

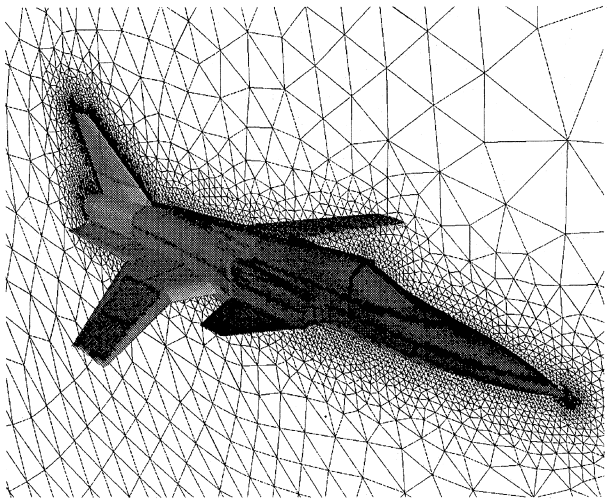


Fig. 15. The surface partitions and the mesh in the symmetry plane for the experimental X29 aircraft.

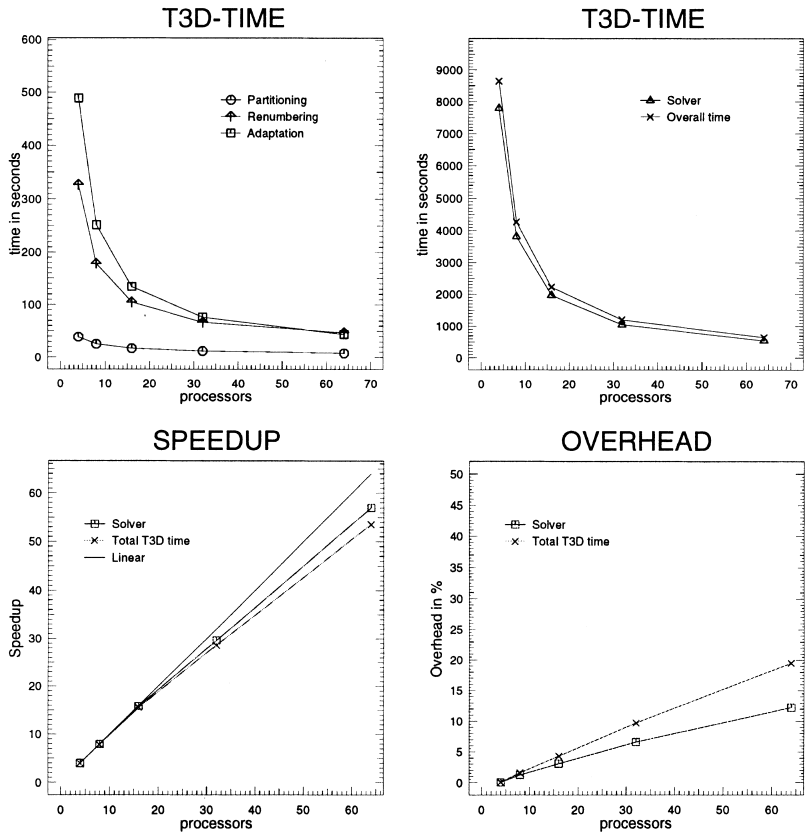


Fig. 16. Completely parallel timings for the unsteady flow over a forward step at $M_\infty = 3$.

9. Unsteady flow

As an example of the efficiency of the completely parallel adaptation algorithm for unsteady simulations, an academic test case of a supersonic flow over a forward step was calculated. This problem has extremely fast moving shocks, and their subsequent reflections and interactions give rise to complex formations of

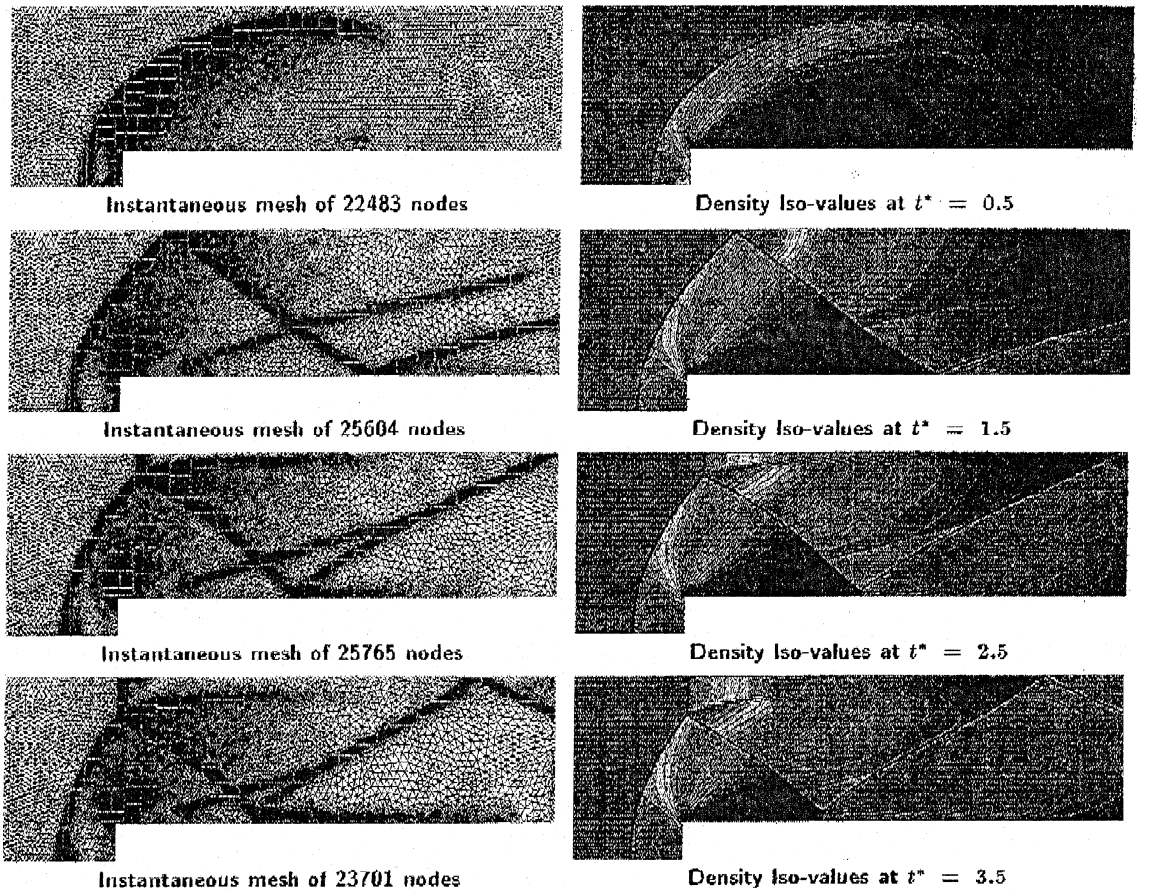


Fig. 17. An example of parallel adaptation: Unsteady supersonic flow and dynamic adaptative mesh evolution over a forward step calculated entirely in parallel on the network.

weaker shocks and contact discontinuities. Fig. 17 shows several different stages of the solution with their corresponding partitions. The mesh is modified every 100 iterations of the flow solver and over 400 cycles of mesh adaptation are performed. At each intermediate adaptation or optimisation step the internal entities are renumbered and reordered in memory, for a cost less than 20% of the total time needed for a complete unsteady simulation. Moreover, the parallel execution on 64 PEs of the CRAY T3D for this forward step calculation runs at an overall efficiency of over 80%. The time spent in adaptation and renumbering is reversed with respect to the steady-state case, Figs. 12 and 16. The domain decomposition algorithm becomes naturally a parallel load balancing one.

10. Conclusions

In this paper, a completely parallel dynamically auto-adaptive mesh strategy is presented. The privileged host node approach is inefficient in performance and memory and becomes heavy for unsteady auto-adaptive simulation. The completely parallel approach presented here is extremely efficient and speedups become now speedups of work rather than a measure of data manipulation. All internal procedures of refining and de-refining, optimisation and domain partitioning are executed directly on the network. The paradigm is illustrated with a variety of test cases for multiple element types. Complete scalability is obtained for complex 3D calculations, as shown by the simulation of the stalling polar for the X29 aircraft. Performance analysis is conclusive as the cost of adaptation and redistribution remains globally less than 15% giving in all an 80% efficiency. A large effort has been made on coherent computational environment.

Acknowledgements

This work was part of a CRAY-EPFL PATP Agreement. The authors thank also Dr. J. Peiro and Prof. J. Peraire for the initial mesh of the X29.

References

[1] F. Chalot, M. Mallet, M. Ravachol, A comprehensive finite element solver for low and high speed aircraft design, AIAA Paper. 94-0814.
[2] A. Dervieux, Steady Euler simulations using unstructured meshes, in: G. Geymonat (Ed.), Partial Differential Equations of Hyperbolic Type and Applications, World Scientific Publishing Co. Pte. Ltd., Singapore, 1987, pp. 34–105.
[3] T.J.R. Hughes, L.P. Franca, M. Mallet, A new finite element formulation for computational fluid dynamics, Comput. Meth. Appl. Mech. Eng. 54 (1986) 223–234 and 63 (1987) 97–112.
[4] T. Minyard, Y. Kallinderis, K. Schulz, Parallel load balancing for dynamic execution environments, AIAA Paper 96-0295, Reno, NV, January 1996.
[5] T. Minyard, Y. Kallinderis, Applications of grid partitioning and parallel dynamic load balancing, AIAA Paper 97-0879, Reno, NV, January 1997.
[6] P. Leyland, N. Maman, F. Benkhaldoun, B. Larrouturou, Dynamical mesh adaptation criteria for accurate capturing of stiff phenomena in combustion, Int. J. Numer. Meth. Heat and Mass Transfer (1993).
[7] R. Löhner, Three-dimensional fluid-structure interaction using a finite element solver and adaptive remeshing, Comput. Syst. Eng. (1990).
[8] D. Mavriplis, Adaptive unstructured meshes, Lecture Notes VKI, 25–28 March 1995.
[9] D. Mavriplis, Three-dimensional high-lift analysis using a parallel unstructured multigrid solver, NACA/CR-1998-207682, ICASE Report No. 98-20, May 1998.
[10] J. Peraire, Unstructured grid methods for compressible flows, Lecture notes CFD COSMASE, Lausanne, Switzerland, 1993.
[11] R. Richter, Schémas de capture de discontinuités en maillage non-structuré avec adaptation dynamique, Ph.D. Thesis, EPFL, 1993.
[12] R. Richter, P. Leyland, Precise pitching airfoil computations by use of dynamic unstructured meshes, AIAA Paper 93-2971, AIAA 24th Fluid Dynamics Conference, Orlando, FL, 1993.
[13] R. Richter, P. Leyland, Distributed CFD using auto-adaptive finite elements, ICASE/LaRC Workshop on Adaptive Grid Methods, Hampton, VA, 1994.
[14] R. Richter, P. Leyland, Entropy correcting schemes and non-hierarchical auto-adaptive dynamic finite flement type meshes, Int. J. Numer. Meth. Fluids 20 (1995).
[15] M. Rudgyard, T. Schoenfeld, G. Audemar, R. Struijs, P. Leyland, A modular approach for computational fluid dynamics, in: Proceedings ECCOMAS94, Stuttgart, 1994.
[16] H. Paillere, E. van der Weide, H. Deconinck, Multidimensional upwind methods for inviscid and viscous compressible flow, 26th VKI lecture series on Computational Fluid Dynamics, VKI LS 1995-02, March 1995.