4th GRACM Congress on Computational Mechanics
GRACM 2002
Patra, 27-29 June, 2002
GRACM

AGGLOMERATION MULTIGRID AND PARALLEL PROCESSING FOR THE NUMERICAL SOLUTION OF THE EULER EQUATIONS ON 2D AND 3D UNSTRUCTURED GRIDS

N.K. Lambropoulos, D.G. Koubogiannis, K.C. Giannakoglou

Lab. Of Thermal Turbomachines,
National Technical University of Athens,
P.O. Box 64069, 15710 Athens, Greece.
e-mail: kgianna@central.ntua.gr

Key words: Euler equations, multigrid, unstructured 2D and 3D grids, parallel processing.

Abstract. *This paper aims at coupling two known CFD techniques, namely multigrid and parallelization, in an existing Euler equations solver for 2D and 3D unstructured grids. The solver is based on a time-marching formulation for the high-subsonic/transonic flow equations and a pointwise implicit solution algorithm. The gain from the combined use of multigrid and parallelization is the reduction of both CPU cost and elapsed computational time associated with the use of the aforementioned software. Multigrid is employed using a finite-volume agglomeration algorithm without explicitly defining coarser grids, the latter being a common practice in structured grids' multigrid. Parallelization relies on the concurrent processing of grid subsets on a cluster of networked processors. The unstructured grid is partitioned through a genetic algorithm based tool which yields equal load per processor and minimal inter-processor communication of data during the iterative scheme. The computational gain is demonstrated in two problems, for which the quality of the numerical solutions has been assessed in previous publications. These problems are: the study of the quasi-3D flow in a controlled diffusion compressor cascade, meshed using triangular elements, and that of the 3D flow around an aircraft wing, meshed using tetrahedral elements.*

1 INTRODUCTION

Nowadays, Euler and Navier-Stokes equations solvers are standard tools which support CFD computations for the analysis or design of engine components. They are in widespread use in the R&D departments of industrial outfits with activities in the field of aeronautics, fluid mechanics, turbomachinery etc. As a consequence, there is need for enhancing any CFD tool used for industrial purposes with techniques capable of reducing the CPU cost of a single computation. Multigrid [1] is likely to be the most effective technique to achieve this goal. Starting from the initial or fine grid, a sequence of gradually coarser grids is defined. Through repetitive computations on each one of them and the application of the necessary operators, low- and high-frequency error components are eliminated, leading thus to faster convergence. Multigrid techniques on structured grids, where the coarser grids are defined by merely eliminating grid lines from the initial grid, have reached a certain maturity and are actually in widespread use. However, this is not the case for unstructured grids, where coarsening techniques and even discretization schemes are still under development. For hyperbolic problems, only a few papers on multigrid acceleration for unstructured grids can be found in the literature, for instance [2].

From a different point of view, the use of multiple processors for a single-computation contributes to the reduction of the elapsed computational time. Either on the expensive parallel computers of the past decade(s) or on cheap and easily-accessed clusters of networked PC-processors, the multidomain technique offers a useful means for the parallelization of existing flow solvers. Grids and, consequently, data partitions need to be created which are then associated to an equal number of intercommunicating processors, one by one. For the sake of parallel efficiency, during the parallel computation, these partitions should yield evenly loaded processors and minimize the inter-processor exchange of data along their interfaces. The parallel efficiency of a multidomain computation depends on various aspects (quality of grid partition, numerical schemes, inter-processor communication protocol, etc). In the past, a variant of the flow solver that will be used herein was ported on two multiprocessor platforms, namely a 48-processor Intel Paragon and a cluster of PCs [3].

In this paper, multigrid and parallel processing is combined in a finite-volume method for the time-marching solution of the Euler equations for compressible flows in 2D and 3D domains with unstructured grids. Multigrid is employed using the agglomeration technique for vertex-centered discretizations, where finite-volume cells (rather than grid cells) are fused to form coarser grid cells. By doing so, graph- rather than grid-based operations are defined, giving rise to a unique formulation for either 2D or 3D grids with triangular or tetrahedral elements respectively. Software parallelization is carried out on a cluster of networked PC-processors, communicating through the PVM message-exchange protocol. As it will be explained below, the agglomeration technique is applied on the initial grid without affecting the definition of subdomains. Only the initial (viz. fine) grid is decomposed into subdomains using a genetic algorithm based, low-cost partitioner, described in [4]. A quasi-3D case (the flow in a compressor cascade) and a 3D one (the flow around a wing) are used to demonstrate the gain achieved by the proposed method. It is evident that periodicity conditions in the turbomachinery cases create additional difficulties and should be taken into account during the application of multigrid and/or parallelization.

2 EULER EQUATIONS : THE FINITE-VOLUME SCHEME

A vertex-centered finite-volume formulation of the Euler equations is used. Closed polylines (2D) (Fig. 1) or closed polyhedra (3D) constitute the integration areas/volumes where mass, momentum and energy fluxes should be balanced. These fluxes are computed by considering pairs of vertices with adjacent finite-volumes where a local 1D Riemann problem is solved through the Roe approximation, [5]. If $\vec{W}$ is the conservation variables array, PQ is the interface between edge-linked nodes P and Q, $\vec{n}_{PQ}$ is the normal unit vector to PQ and $\vec{H}$ stands for the numerical flux computed at the left (L) and right (R) sides of the interface, then the flux crossing the interface is given by

$$\vec{\Phi}_{PQ} = \frac{1}{2}\left(\vec{H}^{inv}(W^L_{PQ}, \vec{n}_{PQ}) + \vec{H}^{inv}(W^R_{PQ}, \vec{n}_{PQ}) - \frac{1}{2}\left|A_{PQ}\right|(W^L_{PQ}, \vec{n}_{PQ}) \right) \tag{1}$$

where $A = \frac{\partial \vec{H}}{\partial \vec{W}}$ is to be computed using Roe-averaged quantities. Equation (1) is a second-order monotone scheme for convective fluxes. From the numerical point of view, this discretization yields a linearized system of equations which is solved through the pointwise implicit Jacobi method. Written in conservative delta form, the system of equations for node P, fig.1, reads

$$\underline{D}^n_P \cdot \delta\vec{W}^{n+1}_P + \sum_{Q \in K_N} \underline{O}^n_{PQ} \cdot \delta\vec{W}^{n+1}_Q = \vec{R}^n_P \tag{2}$$

where (n) stands for the Jacobi subiteration counter, $\vec{R}_P$ for the residual array at node P, $\underline{D}^n_P, \underline{O}^n_{PQ}$ for left-hand-side coefficient matrices whereas K_N denotes the set of neighboring nodes to P. For convenience, eq. (2) can also be cast in the following matrix form

$$L(\delta\vec{W}) = \vec{R} \tag{3}$$

At each node, the pseudo-time step is computed through stability considerations. In the multigrid scheme which is analyzed below, time-steps are multiplied by empirical factors (multiplied by 2 in 2D and by 5 in 3D) whenever switching to the next coarser grid.

The parallelization of this solver on a cluster of eight personal computers with two processors each (Linux operating system, PVM communication protocol) has been described in [3] and will not be repeated here. Multidomains with equally loaded partitions and minimal interfaces, perfectly suited for this kind of computations, are carried out by an in-house tool based on genetic algorithms [4].

3 AGGLOMERATION MULTIGRID AND MULTIPROCESSING

Given the initial (finest) unstructured grid, an identical agglomeration algorithm is used either for 2D or 3D grids. Regardless their dimension, 2D or 3D grids are handled through their equivalent graph, where graph nodes stand for finite-volumes and graph links denote that the corresponding volumes share a common boundary. In order to form the coarser grid or graph, current graph nodes are swept one by one and groups of maximum k neighboring finite-volumes (k=4 in 2D and k=8 in 3D), depending on “the availability”, are allowed to fuse to a single coarser graph node. Upon completion of the coarsening algorithm, all coarser nodes formed by only one fine grid node are eliminated by merging them with their less populated neighboring coarser grid node. Despite the simplicity of the coarsening procedure, care should be taken in grids with periodic boundaries; in cascade flows, for instance, all the periodic nodes are swept in priority, ensuring that pairs of periodic nodes are kept together during coarsening.

With the agglomeration scheme described above, a sequence of coarser grids or graphs can be defined; it is evident that different coarser graphs with different computational performance may result if the fine graph nodes are swept in a different way. On the sequence of grids or graphs, Full Multigrid combined with the Full Approximation Scheme (FMG-FAS [1]) is employed. On the coarser graphs, the governing equations are rediscretized using the finite-volumes defined during the agglomeration algorithm. All numerical fluxes are computed as in eq. (1). FMG-FAS can be considered as a two-stage scheme which is illustrated in fig.3 and explained below:

For the initialization of the solution variables on the coarser grids, a linear restriction operator $\hat{I}_h^H$ (where h and H will denote any two consecutive grids, fine and coarse respectively) is employed on the sequence of grids, starting from the initial one. In the first part, the algorithm starts by performing a small number of iterations on the coarsest grid where the computation of residuals and the Jacobi subiterations are relatively cheap in CPU time. Then, the coarsest level field is projected to the next finer grid according to the so-called prolongation I_H^h operator. In fig.3, downward-upward routes AB represent V-cycles based on FAS whereas, in upward routes BA, values computed over a coarse grid are projected to the finer one. In the second part, V-cycles where information travels from the finest to the coarsest level and backwards, until convergence, are carried out.

FAS requires that eq. (3) be first relaxed over the current grid (h) in order to compute $\vec{W}^h$ and R^h; then, on the next coarser grid H, equation

$$L_H(\vec{W}^H) = L_H(\hat{I}_h^H \vec{W}^h) + I_h^H R^h \tag{4}$$

is relaxed. The solution on grid h is finally updated as follows

$$\vec{W}_{NEW}^h = \vec{W}_{OLD}^h + I_H^h(\vec{W}^H - \hat{I}_h^H \vec{W}^h) \tag{5}$$

If more than two grids are employed, eqs. (3) and (4) are to be solved repetitively starting from the finest down to the coarsest grids and eq. (5) is used when moving backwards in the V-cycle.

In the linear restriction operator used herein, coarse grid values are computed using weighted average (Fig. 2)

$$\hat{I}_h^H \ : \ \Phi_I = \left(\sum_{i=1}^{4} A_i \Phi_i\right) \Big/ \left(\sum_{i=1}^{4} A_i\right) \tag{6}$$

where A_i stands for the area of each constituent control volume. Prolongation is based on the point injection scheme

$$I_H^h \ : \ \Phi_I = \Phi_i \tag{7}$$

where i stands for the constituents of coarse node I. Finally, the RHS of equations written on the current grid (h) are transferred to the next coarser grid (H) through the summation operator

$$I_h^H \ : R_I = \sum_{i=1}^{4} R_i \tag{8}$$

The parallelization of the multigrid-enhanced flow solver is based on a couple of interventions to the algorithm described in [3]. The master processor, i.e. the one reading the grid, data related to the definition of subdomains and the flow problem data, proceeds with the definition of the coarser grids through agglomeration. This is the starting, sequential part of the algorithm. Then, computations are carried out by concurrently running and regularly communicating processors. Since a number of finite-volumes defined on the coarser grids are split between subdomains and shared by more than one processors (this occurs also at the finest grid level) communication is required at each and every level of the multigrid sheme.

4 METHOD APPLICATIONS

In the first test case the flow through a compressor cascade is investigated. The isentropic exit Mach number and the inlet flow angle are equal to 0.457 and 47^o, respectively, where the latter corresponds to zero angle of attack. This is a quasi-3D case where a varying streamtube thickness was used. A fully unstructured grid was generated using an in-house grid generator, which consists of 5,473 nodes and 10,470 triangles. For multigrid, five coarsening steps were carried out. The number of nodes in each one of these grids is tabulated below:

Level 1	Level 2	Level 3	Level 4	Level 5
5,473 nodes	1,583 nodes	532 nodes	248 nodes	169 nodes

The initial grid is shown in fig. 9, where four subdomains (aiming at the parallel execution on four processors) are indicated in colour. The local time steps at the finest grid nodes are computed using CFL=50, where on the coarser grids CFL was multiplied by a coefficient, as described in a previous section. The CFL=50 value was fixed after the first ten V-cycles, during which CFL increased linearly, starting from a very small value. 15 Jacobi subiterations have been used. In a different way, figs. 4 and 5 demonstrate quantitatively the gain achieved by (a) using multigrid on a single processor where CPU time was divided by ~4.5, (b) using the multidomain concept on four processors and a single grid, where the parallel speed-up was ~3.5 and (c) using parallel multigrid through which the elapsed time was divided by ~11.8 compared with that required in a single-grid, single-processor computation. In all these runs, the same results have been computed which match accurately the measured isentropic Mach distribution, as shown in fig. 10. In this case, we refrained from using more processors than four, due to the relatively small size of the grid.

In the second problem, the inviscid flow around a 3D wing at infinite Mach number 0.6 and zero angle of attack was computed. The initial (finest) mesh consists of 71,665 nodes and 397,984 tetrahedra. The three-level multigrid used created the sequence of grids as in the following table:

Level 1	Level 2	Level 3
71,665 nodes	11,401 nodes	1,763 nodes

The CFL number for the finest grid was equal to CFL=100 and increased linearly to that value within the first ten cycles. Fig. 11 gives a partial view of the computational grid, its partition into 16 subdomains and the computed isobar contours on a part of the surface grid. Working with a single grid (i.e. the finest one), multigrid is capable of dividing the CPU time by ~2.8. For the same grid, multiprocessing on four CPUs yields a parallel speed-up of ~2.7 whereas for 16 CPUs this speed-up value approaches ~7.6 (parallel efficiency about 47%). When parallel multigrid is used, the elapsed time for 16 CPUs is divided by ~19, which is an important gain for CFD applications. Parallel speed-up is expected to increase in Navier-Stokes computations, since the computational load per processor will be much higher, a fact that renders the communication overhead almost negligible.

5 CONCLUSIONS

Multigrid acceleration and parallelization on a cluster of PCs were added to an existing inviscid flow solver for compressible fluids, handling unstructured grids. A common formulation for both 2D (with triangular elements) and 3D (with tetrahedral elements) grids was developed. This was possible due to agglomeration multigrid, which allows finite-volumes to be put together to yield the finite-volumes of the coarser grid. The parallelization was based on the PVM protocol, using up to 16 processors. The gain achieved through the combined use of these techniques was noticeable. The elapsed time of a run on a 2D or 3D grid is divided by more than one order of magnitude, using parallel multigrid on 16 CPUs. An important conclusion drawn is that the extra multigrid overhead during parallelization is very low and fully affordable. These encouraging results will be pronounced in in Navier-Stokes computations with parallel multigrid, where the computational burden per processor is much higher compared to the communication overhead.

ACKNOWLEDGEMENTS

The authors would like to thank Mr. A. Giotis, NTUA post graduate student for his support on issues related to the grid partitioner or the use of the PVM protocol. This research was funded by the Institute of Communication and Computer Systems (ICCS/NTUA). Finally, the first author has been supported through a Hellenic State Scholarship's Foundation grant.

REFERENCES

[1] Brandt, A., (1984), *Multigrid Techniques: Guide with Applications to Fluid Dynamics*, Von Karman Lecture Series 1984-4.

[2] Venkatakrishnan, V. and Mavriplis, D.J. (1995), "Agglomeration Multigrid for the Three-Dimensional Euler Equations", AIAA Journal, 33, No.4.

[3] Giotis, A.P., Koubogiannis D.G. and Giannakoglou K.C. (2000) "A Parallel CFD Method for Adaptive Unstructured Grids with Optimum Static Grid Repartitioning", Parallel CFD 2000, Trondheim, Norway.

[4] Giotis, A.P., Giannakoglou K.C., (1998), "An Unstructured Grid Partitioning Method Based on Genetic Algorithms", Advances in Engineering Software, Vol. 29, No. 2, pp. 129-138.

[5] Roe, P., (1981), "Approximate Riemann Solvers, Parameter Vectors and Difference Schemes", J. of Comp. Phys., 43:357-371.

[6] Steinert, W., Eisenberg B. and Starken H. (1990), "Design and Testing of a Controlled Diffusion Airfoil Cascade for Industrial Axial Flow Compressor Applications", ASME paper 90-GT-140.

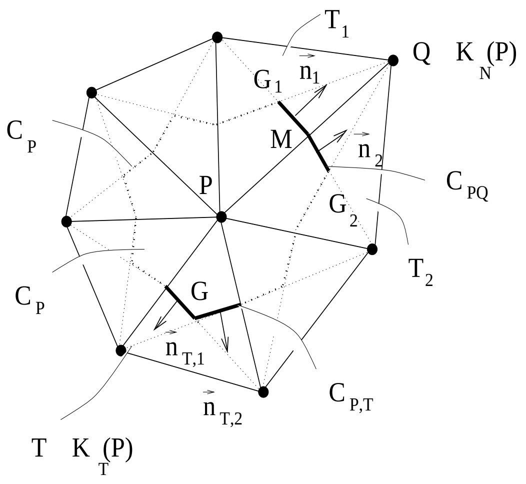


Figure 1. Integration of equations in the finite volume of Node P

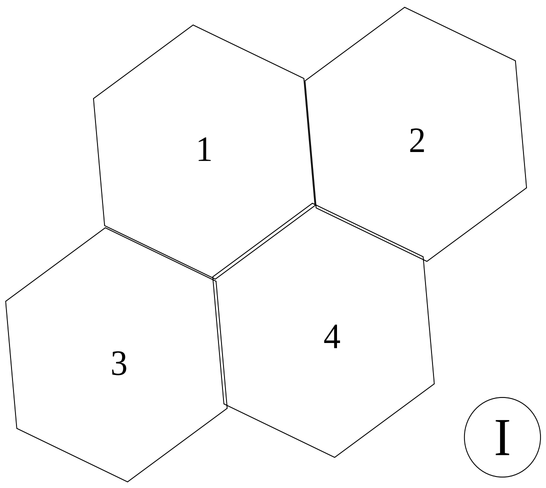


Figure 2. Control volumes agglomeration procedure. Four nodes (1,2,3,4) are fused together to form coarse node I

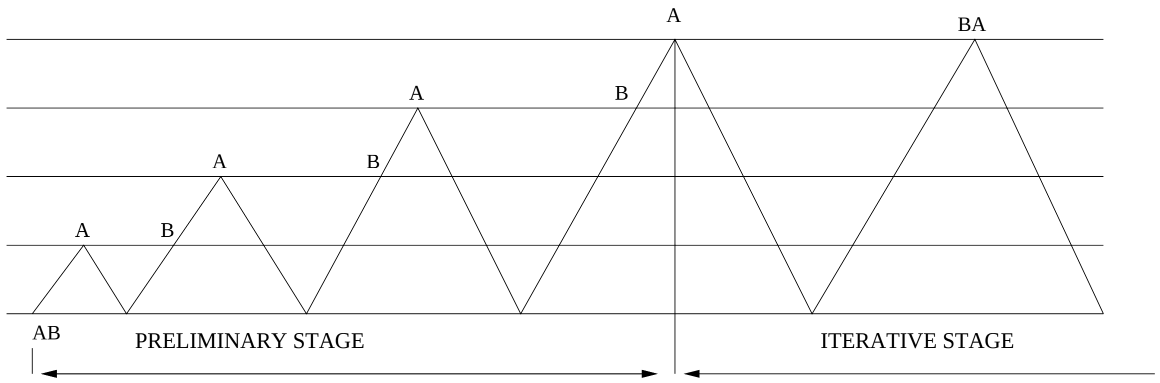


Figure 3: Schematical representation of FMG-FAS scheme.

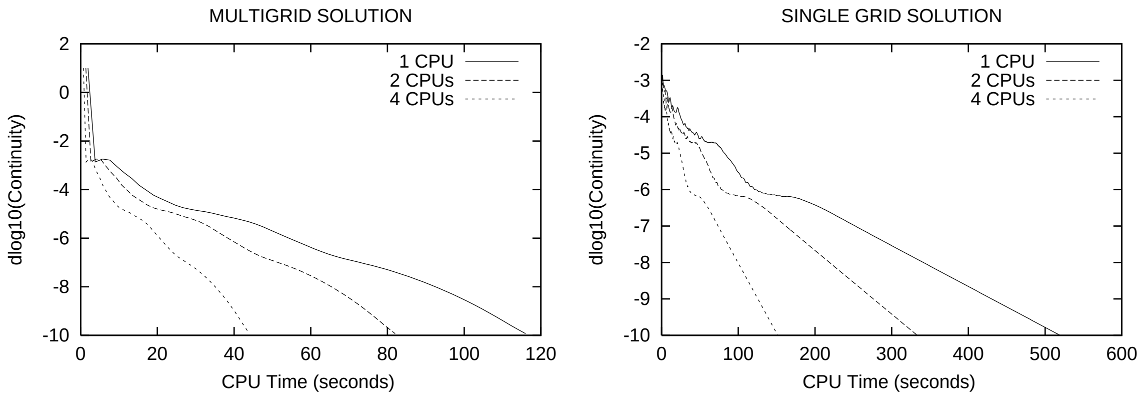
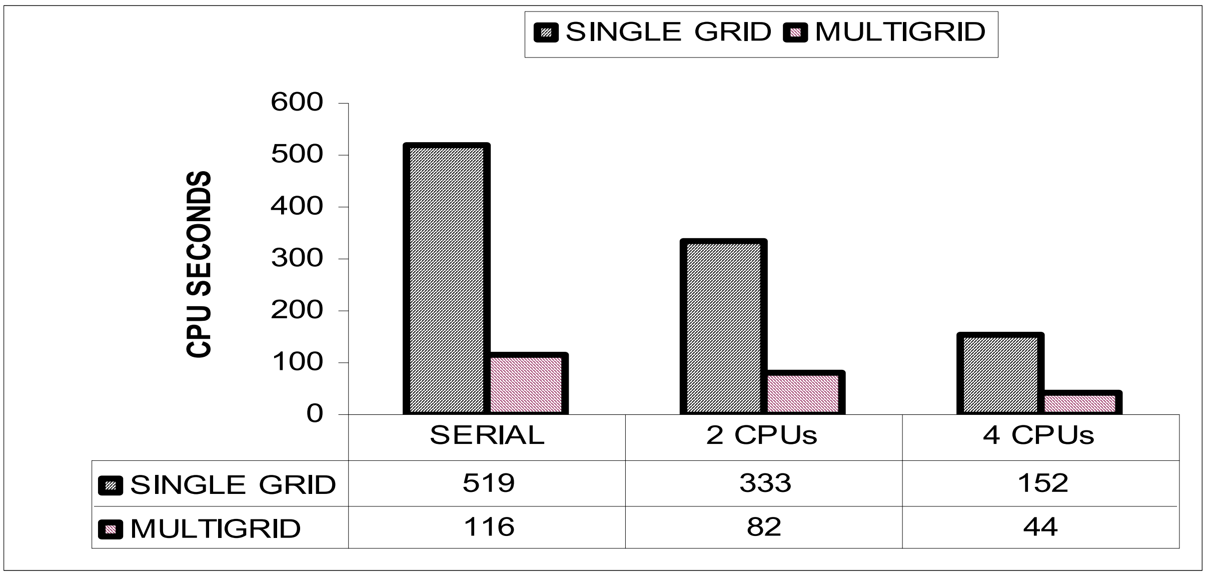


Figure 4. Inviscid flow through a compressor cascade. The RMS of the residual of the continuity equation is plotted against elapsed computing time, for one, two and four processors (with-left and without-right the use of multigrid).



	SERIAL	2 CPUs	4 CPUs
▨ SINGLE GRID	519	333	152
▨ MULTIGRID	116	82	44

Figure 5. Inviscid flow through a compressor cascade. Ellapsed computing time for one, two and four processors (without-left and with-right the use of multigrid).

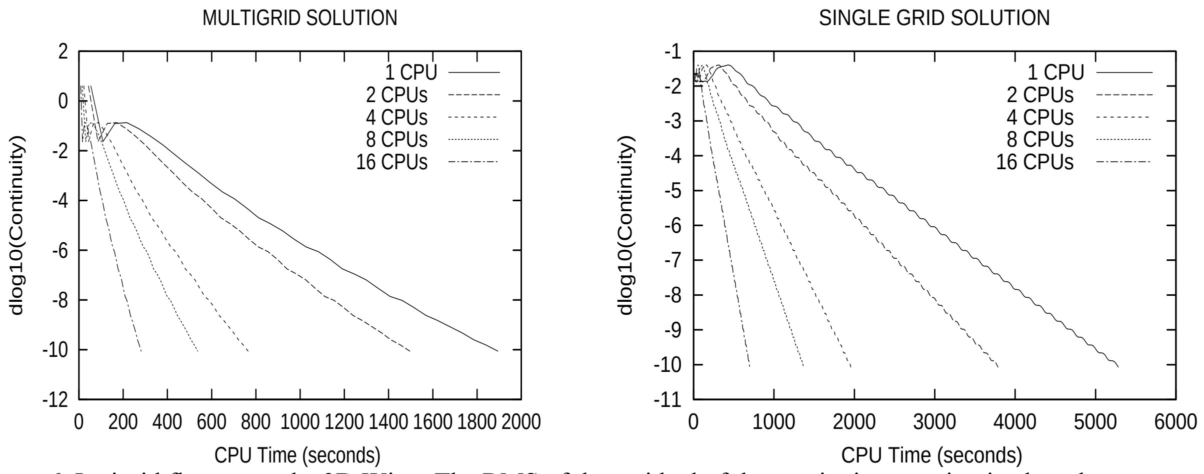
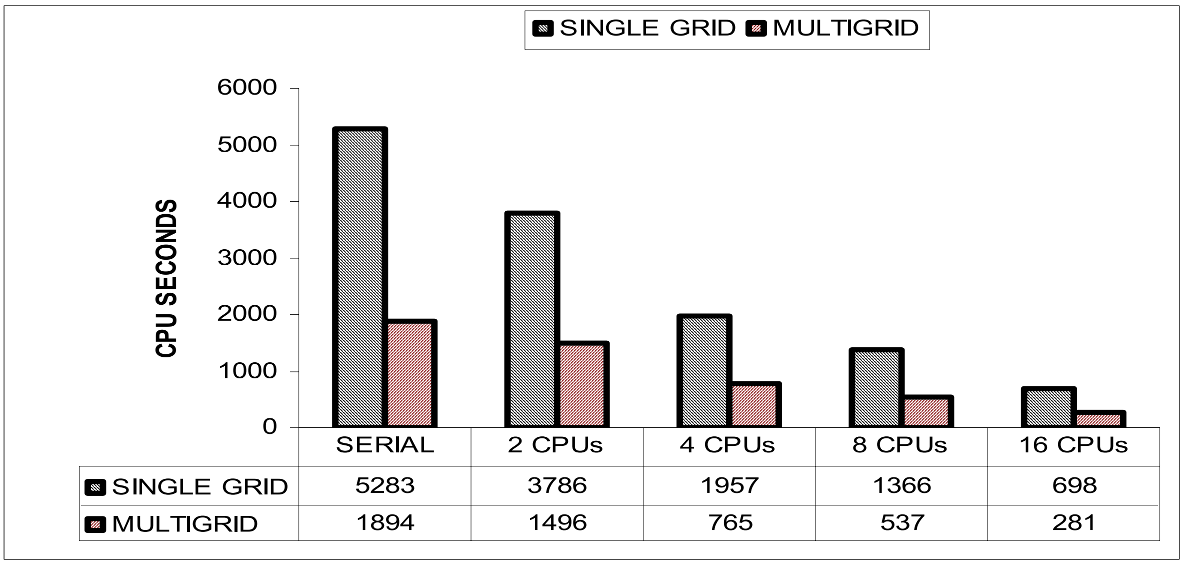


Figure 6. Inviscid flow around a 3D Wing. The RMS of the residual of the continuity equation is plotted against elapsed computing time, for up to sixteen processors (with-left and without-right the use of multigrid).



	SERIAL	2 CPUs	4 CPUs	8 CPUs	16 CPUs
▨ SINGLE GRID	5283	3786	1957	1366	698
▨ MULTIGRID	1894	1496	765	537	281

Figure 7. Inviscid flow around a 3D Wing. Ellapsed computing time for up to sixteen processors (without-left and with-right the use of multigrid).

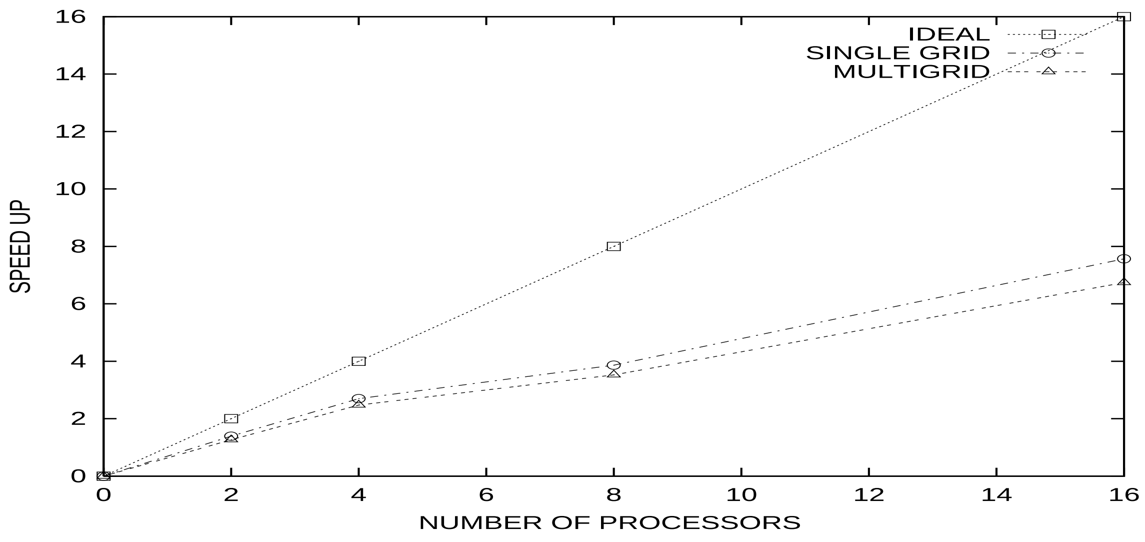


Figure 8. Inviscid flow around 3D Wing. Parallel speed-up curves, with and without multigrid.

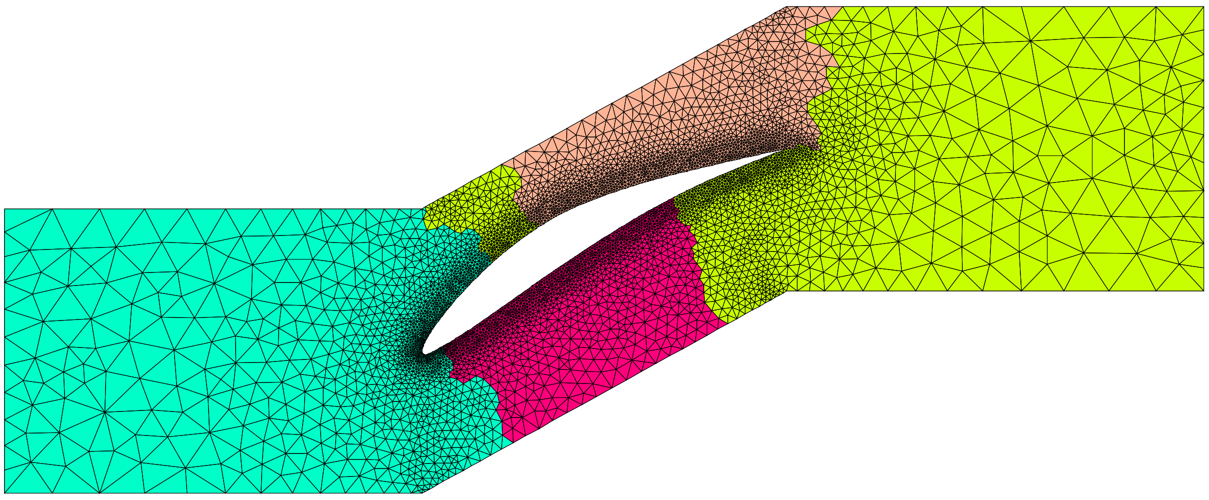


Figure 9. Controlled diffusion compressor cascade (5473 nodes / 10470 triangles). View of the 2D grid used in the computation that has been partitioned into four subdomains

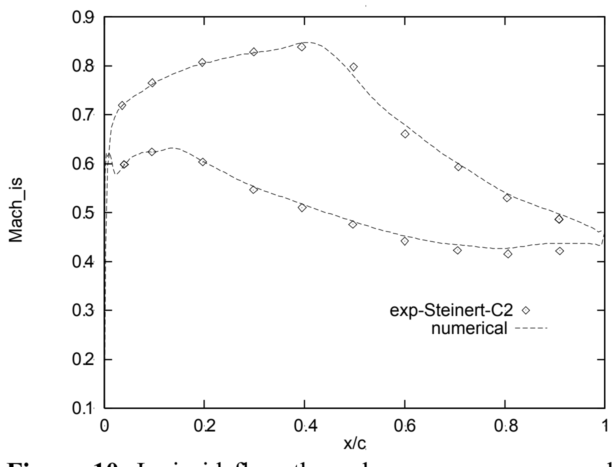
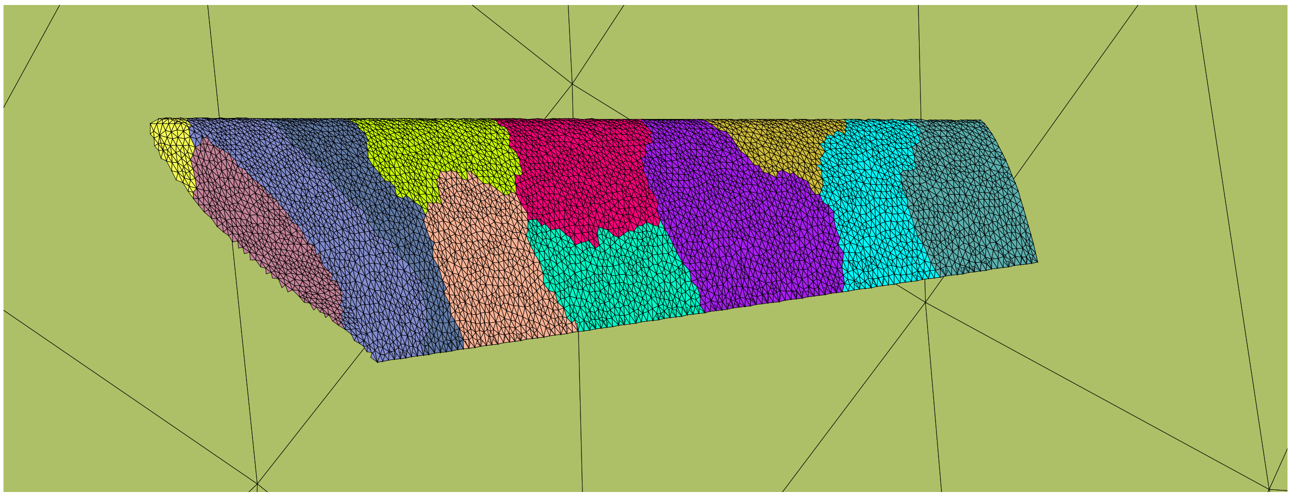


Figure 10. Inviscid flow through a compressor cascade. Isentropic Mach number distribution over the blade surface and comparison with measurements (from [6]).



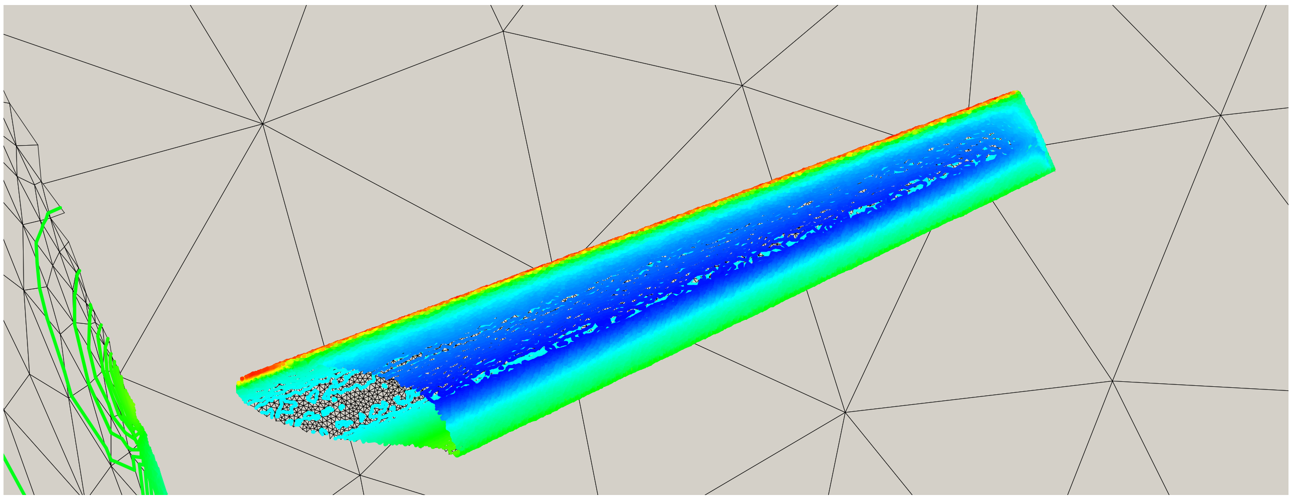


Figure 11. Inviscid fow around a 3D wing (71665 nodes / 397984 tetrahedra). View of the surface grid used in the computation, the partitioned domains and the computed isobar contours.