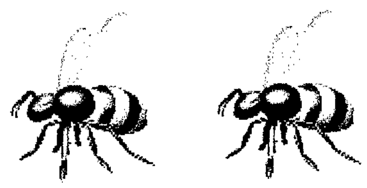
Parallel CFD Workshop

Experiences In Implementation



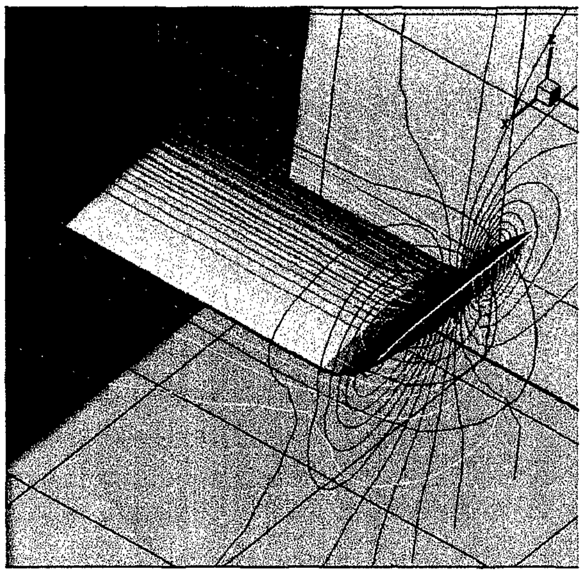
June 16-18, 1999

İ.T.Ü. Maçka Sosyal Tesisleri

Istanbul Technical University

Faculty of Aeronautics and Astronautics

http://www3.itu.edu.tr/~parcfdws



ABSTRACTS

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE 22 June 1999	3. REPORT TYPE AND DATES COVERED Conference Proceedings

4. TITLE AND SUBTITLE Parallel Computational Fluid Dynamics (CFD) Workshop	5. FUNDING NUMBERS F61775-99-WF009
6. AUTHOR(S) Conference Committee	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Istanbul Technical University Maslak- Istanbul 80626 Turkey	8. PERFORMING ORGANIZATION REPORT NUMBER N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) EOARD PSC 802 BOX 14 FPO 09499-0200	10. SPONSORING/MONITORING AGENCY REPORT NUMBER CSP 99-5009
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE A

13. ABSTRACT (Maximum 200 words)

The Final Proceedings for Parallel Computational Fluid Dynamics (CFD) Workshop, 16 June 1999 - 18 June 1999

This is an interdisciplinary conference. Topics to include: Turbulent Flows; Multiphysics Flows; Aerodynamic Flows; Multidisciplinary Applications; Parallel Computing in Aerospace and Mechanical Engineering; Parallel and Sequential Grid Partitioning Techniques; Parallel Grid Generation; Load Balancing; Domain Decomposition, etc.

14. SUBJECT TERMS EOARD, Aerodynamics, CFD, Parallel Computing, Turbulent Flows	15. NUMBER OF PAGES 61
	16. PRICE CODE N/A

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19, SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102

ACKNOWLEDGEMENT

We wish to thank the following for their contribution to the success of this conference:

Istanbul Technical University

Indiana University-Purdue University Indianapolis,
School of Engineering and Technology

Istanbul Technical University Foundation

United States Air Force
European Office of Aerospace Research and Development

European Cooperation in the field of
Scientific and Technical Research

ECCOMAS European Community on Computational Methods in Applied Sciences

SiliconGraphics Silicon Graphics Turkey

roketsan

TUMTMK

This workshop is organized in accordance with the *Collaborative Agreement between Istanbul Technical University and the Indiana University Purdue University Indianapolis.* In addition, as Organizers we would like to thank to all who contributed for realization of this Workshop.

On behalf of the Organizing Committee,
Ulgen Gulcat

CONTENTS
(in alphabetical order)

'Parallel Computation of Thermocapillary Bubble and Drop Migration in Zero Gravity', S.Nas, İTÜ, Turkey

'High-Order Accurate, Three-Dimensional Euler Solver on Distributed Computers', Y.Özyörük, METU, Turkey

'Parallel Computation of Multi-Passage Cascade Flows with Overset Grids', İ.H.Tuncer, METU, Turkey

'What About Load Imbalance?', R.L.Verweij, A.Twerda, T.W.J. Peeters, Delft, Netherlands

'Parallel Adaptive Flow Solutions on Unstructred Mesh', E.Yılmaz, M.Ş.Kavsaoğlu, METU, Turkey

'Usage of the New OpenMP Standard in Parallelization', Igor Zacharov, SGI, Special lecture

Submitted to Parallel CFD '99 Workshop, June 16-18, 1999
Istanbul Technical University, Istanbul, Turkey

A PARALLEL 3D EULER SOLVER FOR UNSTEADY AERODYNAMICS

H. U. Akay, A. Uzun, and C. E. Bronnenberg

CFD Laboratory
Department of Mechanical Engineering
Indian University-Purdue University Indianapolis
Indianapolis, Indiana USA

ABSTRACT

1. Introduction

Calculations of flows around moving bodies immersed in fluids are of major interest to aerodynamicists for determining lift and drag characteristics of flight vehicles such as aircrafts, projectiles, etc. These calculations can also be coupled with structural dynamic models of moving bodies for determining aeroelastic forces resulting from combined effects of structural deformations and fluid pressures.

Flows around oscillating bodies may be solved either by using relative coordinates attached to the bodies [1], or an Arbitrary Lagrangian-Eulerian (ALE) approach [2] on grids deforming continuously with the movements of the bodies. The use of a relative coordinate system eliminates the need for grid movements for single-body problems. For multi-bodies moving independently, however, the relative coordinate approach requires the use of a separate grid patch attached to each body. A sliding mesh technique is typically utilized in this case for maintaining the balance of fluxes between the interfaces of patches moving independently. Once the patches of grids are created properly in advance, there is no need for mesh updates for up to very large movements of bodies. With the ALE/deforming grid approach, on the other hand, both single-and multi-body problems can be solved by using the same computational grid and coordinate system. Although this approach is restricted to moderate movements of the bodies, large movements can also be treated by introducing new grids at selected intervals of the movements.

Because of the large-grid size and high CPU time requirements for accurate solution of unsteady external flows, there has been a considerable interest in CFD community for parallel solution of such problems. Our earlier work for parallel solution of these problems was based on a domain-decomposition approach using the relative coordinate approach and explicit time integrations [3]. More recently, we have developed an ALE and deforming grid based domain-decomposition approach for the solution of similar problems with implicit time integrations [4]. In this paper, we will summarize our recent results obtained and experiences gained with this approach.

2. Present Study

Present study is based on modification of a flow code, USM3D, originally developed at the NASA Langley Research Center [5]. The modifications are of three folds:

1. Development of a dynamically deforming grid algorithm.
2. Arbitrary partitioning of the computational grid for parallel computations.
3. Parallelization of the solver to run on network of workstations and PCs.

Conservation variable form of the compressible Euler equations is cast into a cell-centered finite-volume formulation using tetrahedral elements on unstructured grids. Conservation variables consisting of $\rho, \rho u_i$, and ρE are solved at the center of each finite volume. Transient Euler equations are integrated using a backward-Euler implicit time integration scheme.

For movements of the immersed bodies, the computational grid is deformed continuously to conform to the instantaneous position of the boundaries at each time step using an ALE formulation. Boundary movements are distributed to interior grid points by solving a set of equilibrium equations of a structural network consisting of elastic springs attached to the edges of the finite volumes.

A domain-decomposition approach consisting of subdomains with overlapped interfaces is used for parallelization. The overlaps at the interfaces maintain time accuracy of implicit calculations. A general divider algorithm, based on a publicly available computer program, METIS, is developed to prepare block and overlapped-interface data for the parallel solver [6]. The interface communication between blocks is achieved by means of the message-passing library PVM (Parallel Virtual Machine).

3. Test Cases

Flows around both single- and multi-body problems will be presented as test cases. More specifically, an oscillating aircraft configuration will be presented as an example of a single-body problem and a flapped wing configuration will be presented as an example of a multi-body problem. Shown in Figures 1 through 4 are the sample grids and partitions used for these configurations. Results including speedups and timings on UNIX and LINUX operating systems will be presented at the time of the *Parallel CFD Workshop.*

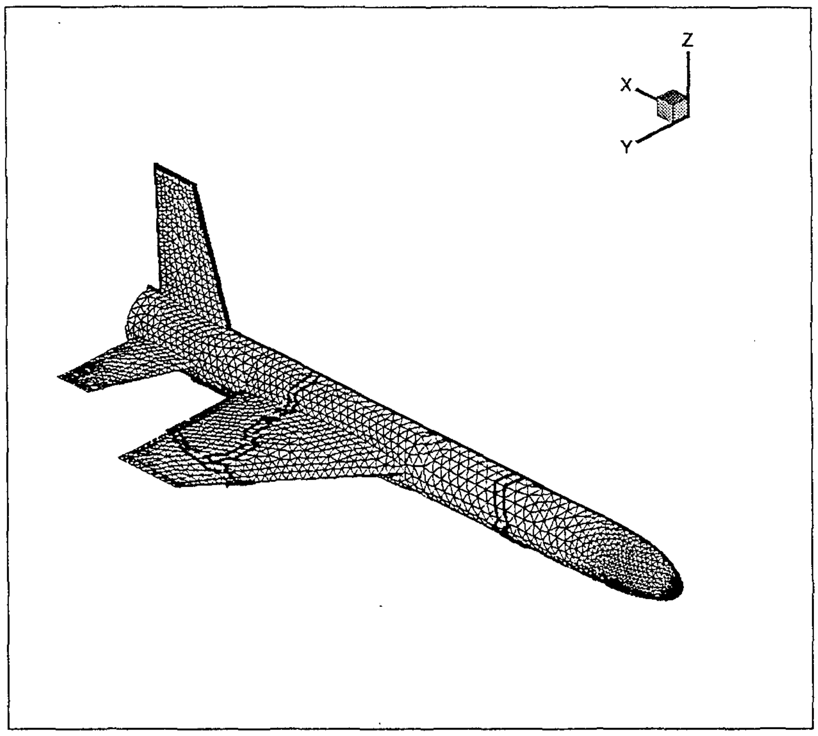


Figure 1. Partitioned surface grid on the aircraft configuration.

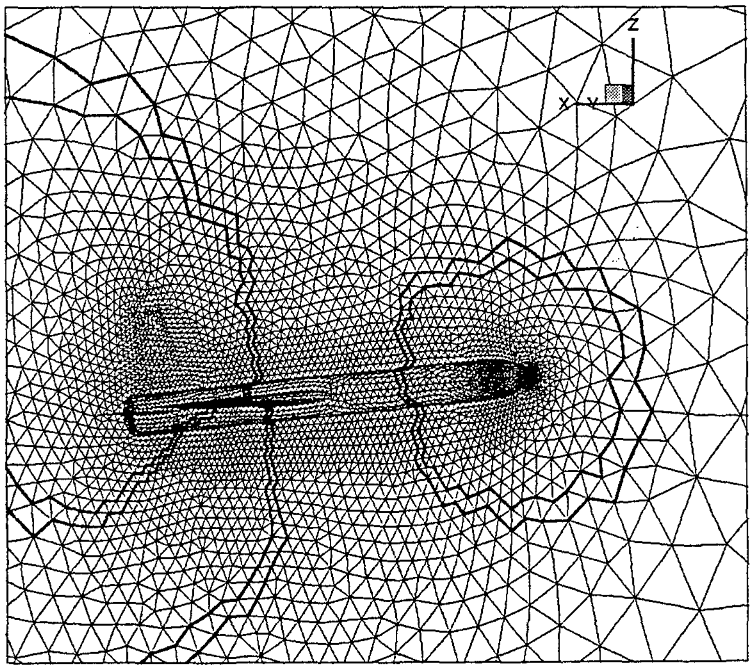


Figure 2. Deformed grid on the symmetry plane at 5 degrees angle of attack.

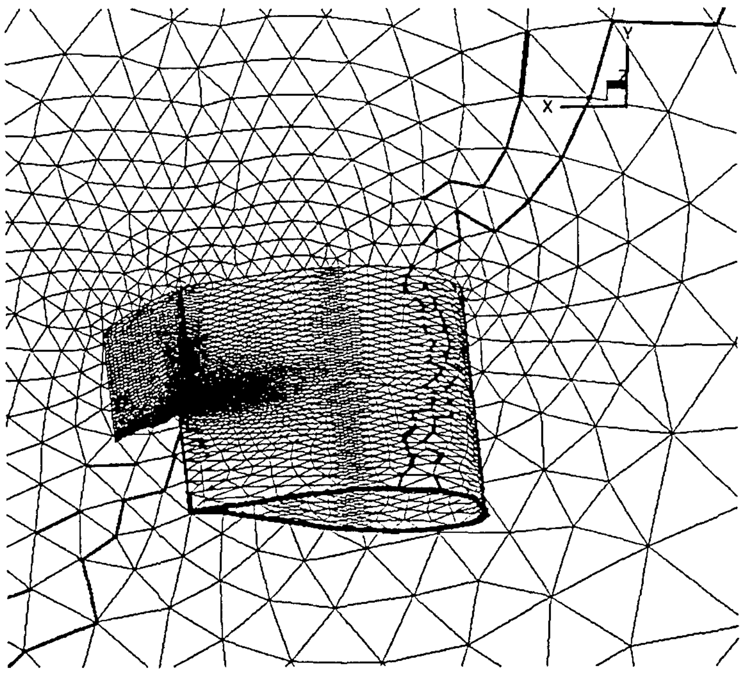


Figure 3. Surface grid of a partially flapped wing with partitions.

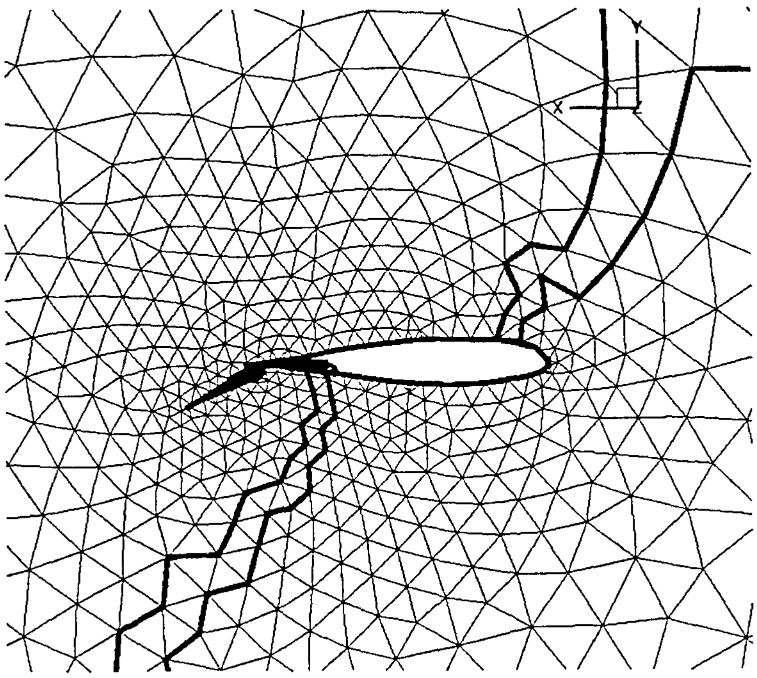


Figure 4. Grid on the root plane of the wing with partitions.

ACKNOWLEDGEMENTS

Permission granted by the NASA Langley Research Center for modifications of the flow code, USM3D, used in this research is gratefully acknowledged.

REFERENCES

1. O.A. Kandil and H.A. Chuang, "Computation of Steady and Unsteady Vortex-Dominated Flows with Shock Waves," *AIAA Journal*, Vol. 26, 1988, pp. 524-531.

2. J. Y. Trepanier, M. Reggio, H. Zhang, and R. Camarero, "A Finite Volume Method for the Euler Equations on Arbitrary Lagrangian-Eulerian Grids," *Computers and Fluids*, Vol. 20, No. 4, pp. 399-409, 1991.

3. H.U. Akay, A. Ecer, and A. Acikmese, "Variable Time-Stepping Strategies for Explicit and Parallel Solution of Unsteady Viscous and Inviscid Compressible Flows," *Parallel Computational Fluid Dynamics,* Edited by P. Sciano, et al., Elsevier Science, The Netherlands, 1997, pp. 328-335.

4. A. Uzun, H.U. Akay, and C. Bronnenberg, "Parallel Computations of Unsteady Euler Equations on Dynamically Deforming Unstructured Grids," *Parallel CFD '99*, Williamsburg, VA, May 23-26, 1999.

5. N. T. Frink, P. Parikh, and S. Pirzadeh, "A Fast Upwind Solver for the Euler Equations on Three-Dimensional Unstructured Meshes," *AIAA Paper 91-0102*, 1991.

6. C. E. Bronnenberg, "GD: A General Divider Program for Unstructured Grids," *User's Manual*, CFD Laboratory, IUPUI, 1999.

FAST AND ACCURATE PARALLEL N-S SOLUTIONS WITH PSEUDO-SECOND-ORDER FINITE ELEMENTS

A.R.Aslan, F.O.Edis, Ü.Gülçat
İstanbul Technical University
Faculty of Aeronautics and Astronautics
Maslak, İstanbul 80626 Turkey
gulcat@itu.edu.tr

ABSTRACT

Unsteady ıncompressıble vıscous flow studıes inevitably require the iterative solution for the pressure equation at each tıme step. This constitutes the major difficulty in terms of parallel implementation To overcome this burden, the iterative domain decomposition techniques are extensively used. Two and three dimensional implementations of such study and the associated efficiency of such methods are given in [1 and 2] where, second order accurate time discritizations is used wıth equal order velocity and pressure elements.

Using equal order interpolations create spurious oscillations in the presure field while consuming most of the computer time for one time step calculations. In order to remedy these two drawbacks, pseudo second order interpolation is implemented by subdividing a parent pressure element into sub-elements and defining velocity approximations on these sub-elements. With this, for hexaheadral elements for example, the number of pressure points are reduced approximately to one eights of its original value while the velocity points remain the same. The detailed comparison concerning the various aspects of the numerical results obtained with and without using the pseudo-second-order elements, is given in [3].

In this study, pseudo second order finite elements are used in solving the Navier-Stokes eqautions. This means, the momentum equation is solved with a fine mesh while the pressure eqaution, which is a Poisson's equation, is solved using the coarse mesh. Since solution to the pressure equation involves much less points, the overall computational effort is reduced drasticaly if an iterative technique is employed for the solution of the Poisson's equation. For an iterative method, the computational complexity is proportional with square of the number of unknowns, i.e. if the number of unknowns is reduced to one half, the number of computations are reuced to one quarters!

A cluster of DEC Alpha XL266 work stations running Linux operating system, interconnected with a 100 Mbps TCP/IP network is used for computataions. Public version of the PVM 3.3 is used as the communication library. As the test case, Lid-driven cubic cavity flow with a Reynolds number of 1000 is studied with the mesh shown

in Figure1. The pressure solution obtained after 1000 time steps at dimensionless time 30 is presented in Figure 2. Solutions obtained with more domains will be presented in the full paper which will also show the efficiency and the speed-up obtained via the proposed method.

[1] Aslan, A.R., Edis, F.O., Gülçat, Ü., 'Accurate incompressible N-S solution on cluster of work stations', Parallel Cfd '98 May 11-14, 1998, Hsinchu, Taiwan

[2] Aslan, A.R., Gülçat, Ü., Edis, F.O., 'Accurate solutions of Navier-Sokes Equations with parallel computations', Fourth ECCOMAS Computational Fluid Dynamics Conference, 7-11 September, 1998, Athens, Greece

[3] Edis, F.O., Aslan, A.R., 'Efficient incompressible flow calculations using pq2q1 elements', Communications in Numerical Methods in Engineering, Vol 14, pp 161-178, (1998).

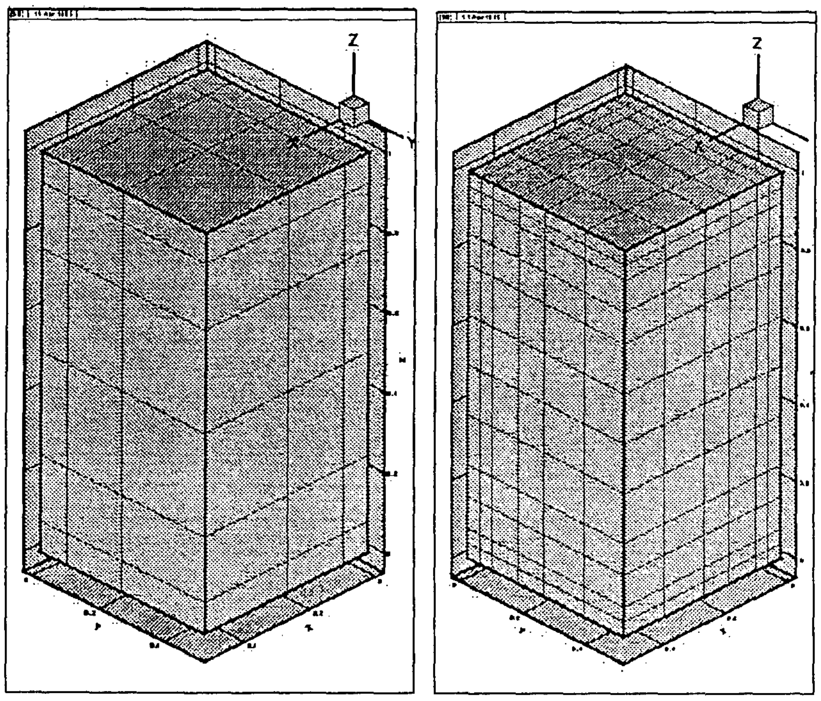


Figure 1. left: grid used for pressure calculations(7x4x4, 1of 2 domains) , right: grid used for velocity calculations(13X7X7, 1 of 2 domains)

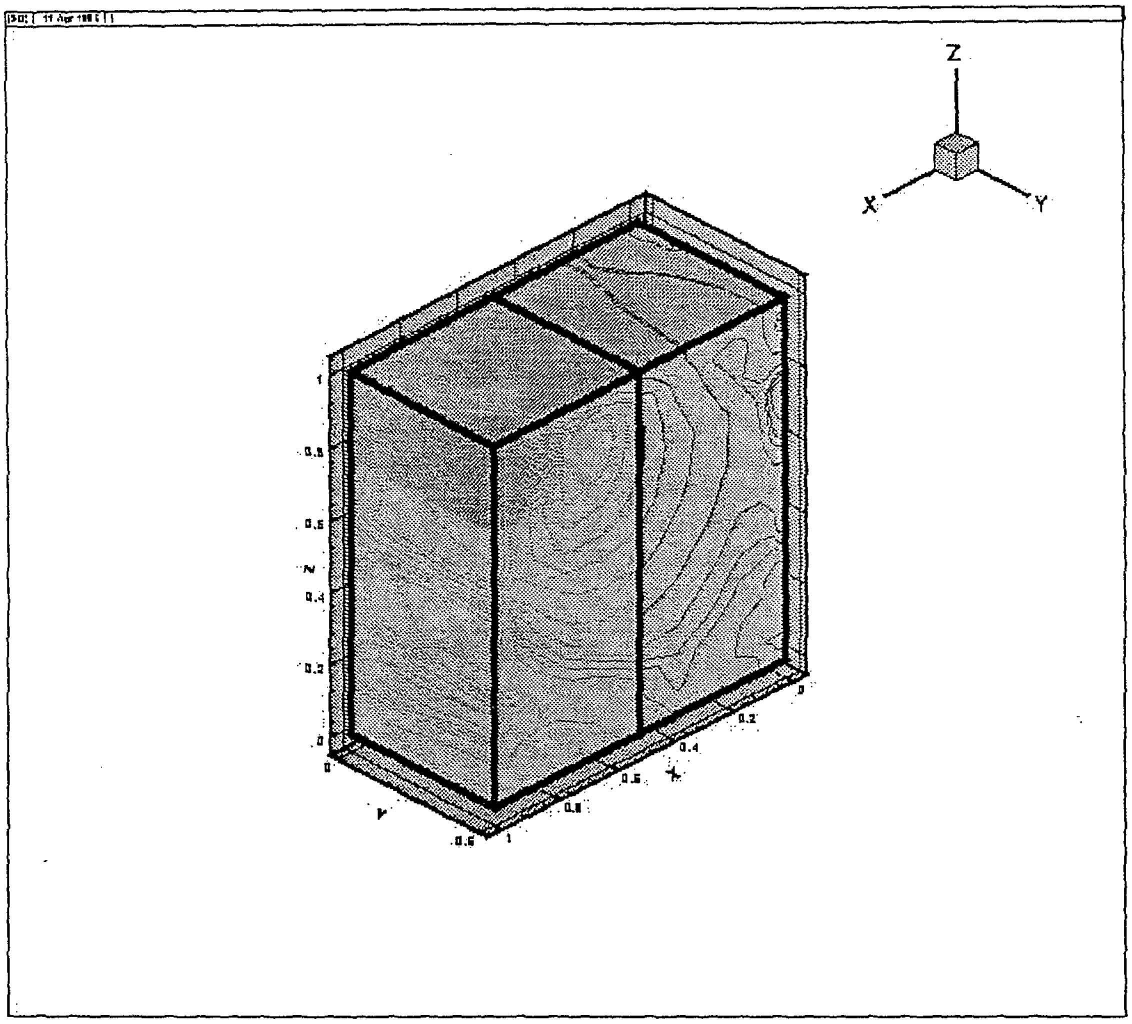


Figure 2. Pressure countours obtained (lid is driven in positive X direction).

Abstract for a Talk at the

Parallel CFD Workshop '99

June 16 – 18 1999, Istanbul, Turkey

Parallelization of a 3-D Flow Solver with special respect to the parallel equation solver

Thomas Bönisch, Roland Rühle
Parallel Computing, High Performance Computing Center Stuttgart, Allmandring 30.
D-70550 Stuttgart, Germany
boenisch@hlrs.de

Short Abstract

This paper will show the techniques used to parallelize a 3-D flow solver to compute Euler- and Navier-Stokes supersonic flows around reentring space vehicles in a wide altitude-velocity range. The target architectures for the parallelization are massively parallel computers with distributed memory. Therefore MPI was used as message passing library. One main focus of the parallelization was to find an efficient parallel equation solver for the Jacobi line relaxation method used in the sequential case. The obtained performance of the parallel program is good, the scaleup is very good.

Abstract

In the URANUS (Upwind Relaxation Algorithm for Nonequilibrium Flows of the University Stuttgart) [1,2] flow simulator the unsteady, compressible Navier-Stokes equations in the integral form are discretized in space using the cell-centred finite volume approach. The inviscid fluxes are formulated in the physical coordinate system and are calculated with Roe/Abgrall`s approximate Riemann solver. Second order accuracy is achieved by a linear extrapolation of the characteristic variables from the cell-centres to the cell faces. TVD limiter functions applied on forward, backward and central differences for non-equidistant meshes are used to determine the corresponding slopes inside the cells, thus characterising the direction of information propagation and preventing oscillation at discontinuities. The viscous fluxes are discretized in the computational domain using classical central and one-sided difference formulas of second order accuracy.
Time integration is accomplished by the Euler backward scheme and the resulting implicit system of equations is solved iteratively by Newton`s Method, which theoretically provides the possibility of quadratic convergence for initial guesses close to the final solution. The time step is computed locally in each cell from a given CFL number. To gain full advantage of the behaviour of Newton`s method, the exact Jacobians of the flux terms and the source term have to be determined.
The resulting linear system of equations is iteratively solved by the Jacobi line relaxation method with subiterations to minimize the inversion error. A simple preconditioning technique is used to improve the condition of the linear system and to simplify the LU-decomposition of the block-tridiagonal matrices to be solved in every line relaxation step.

The boundary conditions are formulated in a fully implicit manner to preserve the convergence behaviour of Newton's method [3].

The usage of the sequential URANUS on high-end workstations and even on today's vector computers shows that the computation time (turnaround time) and the memory requirements especially when changing to real gas methods and/or using fine meshes for real world problems are too high to use these platforms. The solution to this problem is to parallelize this code.

The requirements for the parallel code have been:

Portability

The parallel program should run on nearly all available especially parallel platforms. Therefore only a parallel programming model which is available also on distributed memory machines could be used. We decided to use message passing with MPI [6] to assure best possible portability, because MPI is available on nearly all parallel platforms from workstation clusters across shared memory parallel vector platforms to massively parallel systems with distributed memory, maybe consisting of shared memory multiprocessor nodes.

Usability

There should be nearly no difference in the code handling between the parallel and the sequential code. This means every engineer with some experience in using the sequential code should be able to handle the parallel one.

Maintainability and Extensibility

The code structure should only be changed as much as absolutely necessary to develop an efficient parallel program. Because it should be possible for the development engineers to maintain the parallel code. For this reason a domain decomposition method with a two-cell overlapping region at the subdomain boundaries was used. The overlapping region guarantees that there is no communication necessary during the set up of the equation system, even if you use the limiters for second order accuracy.
This additionally guarantees the easy extensibility of the program, because there is nearly no change in the program structure. You only have take care of the correct handling of the overlapping regions and the (sub)domain boundaries. This is very important since in the future there should be only one version of the program, the parallel, because it is nearly impossible to do a parallel development of two codes and in addition this is really not necessary because the parallel code also runs on a single processor with one process.

An important step was the parallelization of the Jacobi line relaxation solver. The idea we found in [7] was the usage of an additional splitting method to reduce the coupling between the matrix parts located on different processors. This leads to the possibility to solve these different matrix parts in parallel with a following communication step to exchange the results between the neighbours in order to update the right hand side for the next subiteration or iteration. Based on this idea we developed two solvers with different communication patterns. One communicates only two times but needs about 20% more computation time, the other one communicates after every line relaxation step but needs no additional computational effort. The second solver is used by default and shows good performance especially on platforms with an accordingly fast network normally provided on today's massively parallel computers. The first solver is used if the network is not powerful enough for example in the case of metacomputing [8]. A third method especially used for block structured meshes is currently under development.

The resulting parallel program was tested on different hardware platforms including Cray T3E, IBM RS/6000 SP, Hitachi SR2201, NEC SX-4 and SGI Origin. We also do tests on a workstation cluster.

The difference between the solutions computed sequential and in parallel is negligible compared with the small differences between computation and experiment. The numerical efficiency is nearly the

Figures

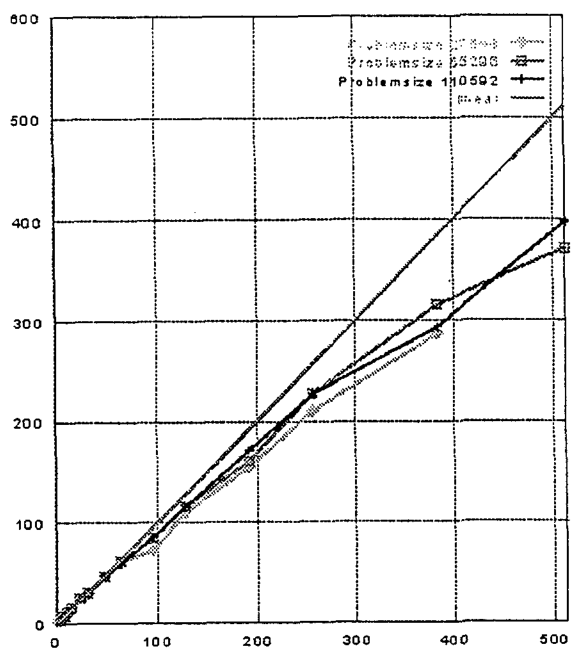


Figure 1: Speedup of the 3-D URANUS code for three different problem sizes on a Cray T3E

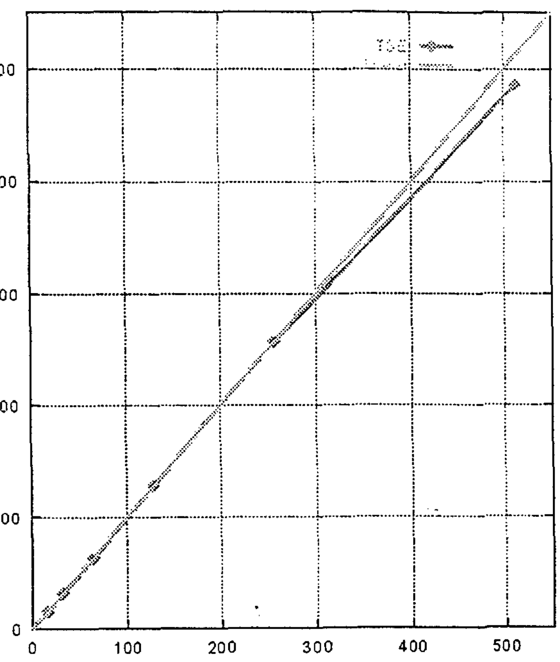


Figure 2: Scaleup of the 3-D URANUS code from 16 to 512 Processors on a Cray T3E

same as in the sequential program. The speedup measured at the Cray T3E in Stuttgart was found to be nearly 400 for 512 processors [Figure 1] and the scaleup is with more than 95% [Figure 2] very good. So we expect to be able to use several thousand processors efficiently.

References

1. Schöll, E., Frühauf, H.-H.: An Accurate and Efficient Implicit Upwind Solver for the Navier-Stokes Equations in Notes on Numerical Fluid Mechanics, Numerical Methods for the Navier-Stokes Equations, Hebecker F.-K., Ranacher R., Wittum G. (Eds.), Proceedings of the International Workshop on Numerical Methods for the Navier-Stokes Equations, Heidelberg, Germany, October 1993, Vieweg, 1994.
2. Frühauf, H.-H., Daiß, A., Gerlinger, U., Knab, O., Schöll, E.: Computation of Reentry Nonequilibrium Flows in a Wide Altitude and Velocity Regime, AIAA Paper 94-1961, June 1994.
3. Gerlinger, U., Frühauf, H.-H., Bönisch, T. : Implicit Upwind Navier-Stokes Solver for Reentry Nonequilibrium Flows, AIAA Paper 97.
4. Bönisch, T., Geiger, A.: Implementierung eines 3-D Strömungscodes auf Parallelrechnern, Forschungs- und Entwicklungsberichte RUS-35, Computing Center University of Stuttgart, Germany, 1996.
5. Bönisch, T., Rühle, R.: Portable Parallelization of a 3-D Flow-Solver, in D.R. Emerson et. al. (Eds.) ‚Parallel Computational Fluid Dynamics', Recent Developments and Advances Using Parallel Computers, Elsevier Science B. V., 1998, 457-464.
6. MPI Forum: MPI: A Message-Passing Interface Standard, University of Tennessee, Knoxville, Tennessee, USA, 1994.
7. Hofhaus, J.: Entwicklung einer effizienten, parallelisierten und impliziten Lösung der Navier-Stokes Gleichungen für dreidimensionale, inkompressible und instationäre Strömungen, Master Thesis, RWTH Aachen, Germany, 1993.
8. Bönisch, T., Rühle, R.: Adaptation of a 3-D Flow-Solver for use in a Metacomputing Environment, Proceedings of Parallel CFD '98, to appear.

Submitted to Parallel CFD' 99 Workshop, June 16-18, 1999
Istanbul Technical University, Istanbul, Turkey

SOFTWARE, HARDWARE AND ALGORITHM CONSIDERATIONS IN REDUCING COMMUNICATION COST FOR PARALLEL COMPUTING

A.Ecer, I.Tarkan, and E. Lemoine

Department of Mechanical Engineering
School Of Engineering and Technology, IUPUI
Indianapolis Indiana, USA

ABSTRACT

One of the objectives of parallel computing is to solve larger and larger problems without being penalized by the cost of communicating. This cost depends on the available hardware and software tools as well as the computational algorithm to be utilized. In this paper, we demostrate on several test cases, the performance of two clusters of workstations: one with Unix workstations and the other one of NT workstations which are connected. The communication network is Ethernet with different types of switches. The test cases involve a blocked solution of the three-dimensional heat equation by an explicit integration scheme and a three-dimensional Navier-Stokes code. The problems are solved for different size problems and communication cost vs. computation cost is evaluated. The evaluation of the communication cost is studied for switched ethernet networks. The network efficiency is studied. The second part of the study involves improvement of the algorithm for reducing the communication cost. Filtering of the solution in time is considered for improving the efficiency of the algorithm. The convergence rate vs. elapsed time is compared for different algorithms operating on different networks.

REDUCING PARALLELIZATION OVERHEADS FOR INCOMPRESSIBLE FLOWS USING PSEUDO-SECOND-ORDER VELOCITY INTERPOLATION

F. O. Edis, Ü. Gülçat, A. R. Aslan
Istanbul Technical University,
Faculty of Aeronautics and Astronautics
Maslak, Istanbul 80626 Turkey
gulcat@itu.edu.tr

ABSTRACT

Most solution schemes for unsteady incompressible viscous flows require implicit solution of the Poisson's equation for pressure, which constitutes the major difficulty in terms of parallel implementation. Iterative domain decomposition techniques are widely used to overcome this difficulty. Efficiency of such a method is presented in [1] for an implementation on a workstation cluster.

In Finite Element computation of incompressible flows, first-order interpolation element both for velocity and pressure are preferred over second order velocity interpolation, due to its computational efficiency. However equal order interpolation elements do not satisfy the so-called Ladyzhenskaya-Babuska-Brezzi condition and often produce spurious oscillations in the pressure field. Elements combining first-order velocity interpolation with first-order pressure interpolation and still satisfying the LBB condition are proposed in [2]. These elements, which can be considered as pseudo-second-order velocity interpolation elements, satisfy the LBB condition while offering lower computational requirements compared to equal-order interpolation elements. Pseudo-second-order interpolation is achieved by subdividing a parent pressure element into sub-elements and defining first-order velocity interpolations on these sub-elements. Detailed comparison of results, obtained with quadrilateral/hexahedral (pQ2Q1) and triangular/tetrahedral (pP2P1) shaped pseudo-second-order velocity interpolation elements with those, obtained with corresponding equal-order interpolation elements, can be found in [3-5].

Parallel implementation of a fractional step time discretization Finite Element scheme using pseudo-quadratic velocity/linear pressure interpolation tetrahedral elements is presented in this paper. An iterative non-overlapping domain decomposition technique [1,6] is utilized for the distributed solution of the Poisson's equation for the pressure. pP2P1 elements are especially attractive within this scheme as the number of pressure elements are far less than the number of velocity elements, yielding a better efficiency for the parallelization of the Poisson solver.

A cluster of DEC Alpha XL266 workstations running with Linux operating system, interconnected with a 100 Mbps TCP/IP network is used for computations. Public version of the Parallel Virtual Machine 3.3 is used as the communication library.
Lid-driven flow in a square cavity with a Reynolds number of 1000 is selected as a test case to demonstrate the efficiency and accuracy of the methods used. Two domain computational mesh is presented Figure 1. Solution obtained after 3000 time steps at dimensionless time 30 are presented as pressure iso-surfaces in Figure 2.

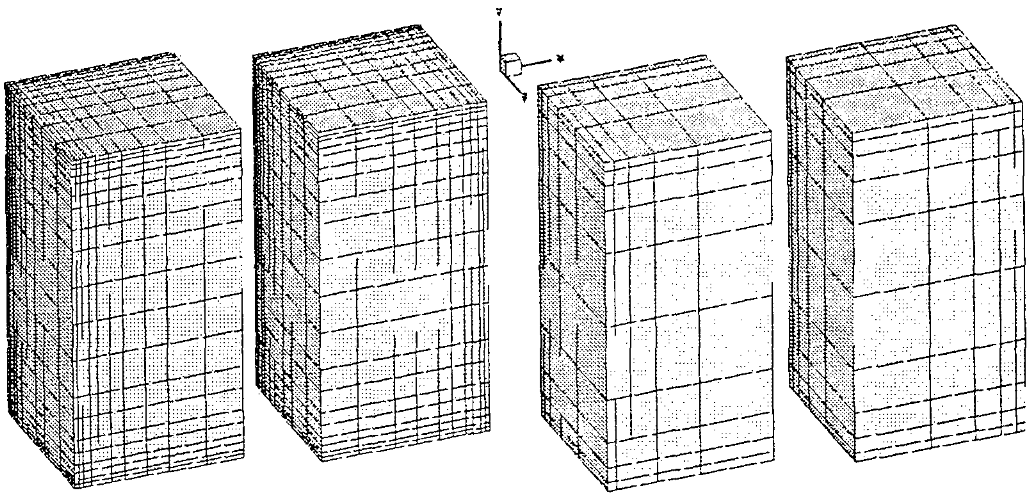


Figure 1. Computational mesh used for the two subdomain parallel solution of the lid-driven flow in a cubic cavity (left: 21x21x11 velocity, right: 11x11x6 pressure mesh).

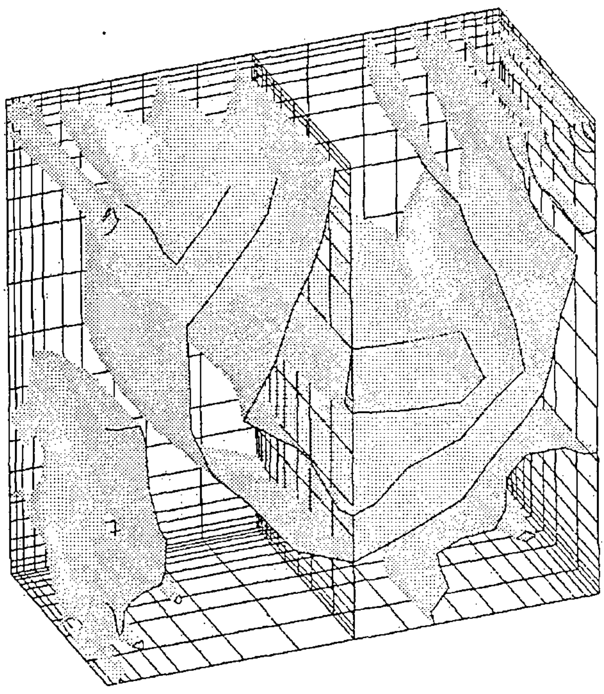


Figure 2. Pressure iso-surfaces for the parallel solution of the lid-driven cavity flow at Re=1000 (t=30) on hexahedral grids using pQ2Q1 elements.

Solutions obtained on two and four subdomain configurations will be presented in the full paper. The full paper will discuss the details of pseudo-second order velocity interpolation and the iterative domain decomposition method used, as well as the effect of the pseudo-biquadratic element on parallel performance. Comparisons of accuracy, computational effort and speed-up for two and four subdomain solutions obtained with psedo-biquadratic element and with regular bilinear hexahedral elements will be included.

REFERENCES

[1] Aslan, A. R., Edis, F.O., Gülçat, Ü., 'Accurate incompressible N-S solution on cluster of work stations', Parallel CFD '98 May 11-14, 1998, Hsinchu, Taiwan

[2] Bercovier, M. and Pironneau, O., 'Error estimates for Finite Element Method solution of the Stokes problem in primitive variables', Numer. Math. Vol. 33 pp 211-224, (1979),

[3] Edis, F. O. and Aslan, A. R. 'Efficient incompressible flow calculations using pQ2Q1 element', Communication in Numerical Methods in Engineering, Vol 14, pp 161-178, (1998)

[4] Edis, F. O. and Aslan, A. R. 'Efficient incompressible flow calculations using pseudo-second-order finite elements', In Numerical Methods in Laminar and Turbulent Flow: Proceedings of the 10th International Conference (Swansea, 1997), Pineridge Press

[5] Edis, F.O. 'Efficient finite element computation of incompressible viscous flows using pseudo-second order velocity interpolation', PhD Thesis, Istanbul Technical University, 1998

[6] Glowinski, R. and Periaux, J., 'Domain decomposition methods for nonlinear problems in fluid dynamics', Research Report 147, INRIA, France 1982

Issues for Large Scale Simulation in the Process Industries

by

*D. R. Emerson and R. J. Blake[1]

Submitted to
Parallel CFD Workshop: *Experiences in Implementation*
June 16-18, 1999, Istanbul, Turkey

Introduction

Fluid flow problems within the Chemical and Process industries are varied and frequently very difficult to solve. At the very least, the flows are turbulent to enhance fluid mixing. Many of the problems involve complex chemistry effects (such as nucleation, surface growth and coagulation) in complex geometries. These problems push the boundaries of current knowledge and some simplification must be made to make the problem tractable. However, many of the everyday problems within these industries are even more complex and typically involve multiphase flows e.g. industrial cyclones involves separation of particles transported in a fluid medium (usually a gas). Additional complications arise when the flows are non-Newtonian. These involve complex mixtures, such as slurries, pastes, gels and blood.
This paper will consider two problems from the Chemical and Process industries where parallel processing has been tried. The first involves modelling of a distillation column. This requires the solution of many Ordinary Differential Equations (ODEs). These systems are large, highly non-linear and frequently very sparse. The second problem involves the production of a white pigment used as an additive in many commercial products ranging from paint through to sweets. The complete process is extremely complex but the fluid modelling concerns the gas-phase reaction that occurs in a circular pipe. The chemistry is very complex but the geometry is very simple and therefore allows the problem to be tractable. The modelling is very compute-intensive and the use of parallel processing has allowed the problem to be tackled in a more routine way.

Modelling the Reactor Process

The equations governing the pigment production are represented by the incompressible Navier-Stokes equations for a turbulent fluid. For the case in question, the k-ε turbulence model was used. The reactor geometry, shown here in 2-D, is illustrated in figure 1.

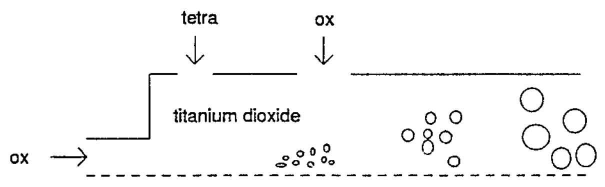


Figure 1: Typical reactor process

* Corresponding author
[1] CLRC Daresbury Laboratory

At the inlet (on the left hand side), high temperature oxygen (1500-2100 K) enters and mixes with titanium tetrachloride, oxygen and trace additives which are critical to the reaction process, which is highly exothermic. The process, as described by Hounslow et al. [1] and Blake and Emerson [2], involves:

- a nucleation process that converts the raw feed-stocks into titanium dioxide nuclei therefore reducing the concentrations of molecular oxygen and titanium tetrachloride and increasing the number of particles in the smallest size class;
- particle growth through a deposition process which is very strongly influenced by trace additives. The growth process obviously depletes the raw feed-stocks and causes particles to jump from one size class to the next as the amount of titanium dioxide is increased;
- particles growing through a coagulation process whose rate is dictated by the frequency of collision and the probability of sticking. This process conserves mass within the size class variables and is important in reducing the standard deviation of the distribution of particles;

The pigment production reaction is strongly exothermic and therefore strongly influences the hydrodynamics through the temperature dependence of the various fluid flow properties such viscosity, density and specific heat. The models can employ anything up to 100 class sizes for high accuracy computations. The calculations of the various source terms describing the nucleation, growth and coagulation are extremely time consuming as the coagulation terms fully couple all the class size variables in each finite volume. It is obvious that three-dimensional computations for systems with this number of class size variables would be completely impractical and uneconomic in the absence of parallel systems.
The talk will discuss the problems of solving large ODEs found in the chemical industry and results from modelling the titanium dioxide process. Future issues and problems will also be presented.

[1] Hounslow, M. J., Ryall, R. L. and Marshall, V. R., (1988), A I Che J, 34, pp. 1821-1832

[2] Blake, R. J. and Emerson, D. R., Proceedings of ECCOMAS '98, Volume 2, pp. 714-719.

Abstract for the Parallel CFD Workshop, Experiences in Implementation
June 16-18, 1999, Istanbul TURKEY

SOME DOMAIN DECOMPOSITION AND PARALLEL ALGORITHM ISSUES FOR THE NUMERICAL SOLUTION OF A CATALYTIC REACTOR

Marc GARBEY, Lyon, FRANCE

Here, we consider the short time catalytic reactor TAP2 and the set of experiments of Mirodatos and Yves Shuurman in the 'Institut de Recherche sur la Catalyse- Lyon'. We will comment on the modelling of this device that is designed to realize the partial oxydation of Methane.

We show that the distributed modeling concept is a way to tackle the complexity of the problem.

We present some domain decomposition solvers for the direct numerical simulation of low Mach number non reactive flow. These results are preliminary results on this pluridisciplinary project.

Parallel Implementation of a Commonly Used Internal Combustion Engine Code on a Cluster of High Performance Workstations

Aytekin Gel[1] and Ismail Celik

Mechanical & Aerospace Engineering Department
West Virginia University, Morgantown, WV

EXTENDED ABSTRACT

Engineering fluid dynamics problems involving the prediction of turbulent flows with direct numerical simulation (DNS) are still beyond the capability of the most state-of-the-art hardware. Large-eddy simulation (LES) is a viable alternative to DNS in which certain portion of the detailed small-scale information resolved in DNS is suppressed. This simplification makes the computational problem more tractable. However, even with LES, today's challenging problems require accurate schemes with higher grid resolution. In addition to these, shorter turn-around time to approximate the instantaneous flow behavior has become more important.

As part of a research effort towards the large-eddy simulation of diesel combustion engines, widely used KIVA-3 code originally developed by Los Alamos National Laboratory [1] was re-structured for parallel execution via message-passing on parallel computers, in particular on distributed-memory platforms. One-dimensional domain decomposition approach initially proposed by Yasar [2] was implemented in KIVA-3 code with substantial improvements (Gel [3]).

KIVA-3 is a general CFD code based on the Arbitrary Lagrangian-Eulerian (ALE) method with enhanced features for internal combustion engine applications. The method is typically implemented in three phases. The first phase is an explicit Lagrangian update of the equations of motion. The second phase is an optional implicit phase that allows sound waves to move many computational cells per time step if the material velocities are smaller than the fluid sound speed. The third and final phase is the remap phase where the solution from the end of phase two is mapped back onto an Eulerian grid if Eulerian approach is selected. Time advancement is similar to many codes that utilize the LES methodology where the convective terms are advanced explicitly and the diffusion terms can be advanced explicitly, implicitly, or semi-implicitly. The degree to which the diffusion terms are implicitly discretized is based on a combination of stability and efficiency considerations. Global timestep is divided into sub-time steps (referred to as sub-cycles) based on Courant condition to advance the convective terms.

The overall parallelization task was subdivided into three phases. In the first phase, all of the essential features of KIVA-3 excluding piston movement, chemical reactions and spray dynamics were parallelized. Results of the first phase are presented in this study with calculated speedup for the benchmark problem executed on an 112 processor SGI Origin 2000 system. The MPI message-passing library was employed due to portability considerations which enabled KIVA-3/MPI to be easily ported on to several other platforms including Cray T3E, IBM SP2 and recently on a Beowulf like cluster of DEC Alpha systems that is being built locally in the department and running under Linux.

Several problems were selected to perform the benchmark runs for speedup and parallel efficiency after the parallel version results were validated with the original KIVA-3 results for the same problems. The particular problem that will be presented in this paper is the turbulent round jet problem where a jet at a velocity of 1500 cm/s and a diameter of 10 cm enters to the three-dimensional rectangular domain through the left wall. This problem was studied by Smith et al. [4] on single processor with a grid of upto 500,000 vertices. The KIVA-3/MPI was used to simulate the same problem up to 32 processors with a maximum grid resolution of 4,370,000 vertices. A Smagorinsky type subgrid-scale

[1]Please send all future correspondance to the first author, e-mail : aike@stokes.mae.wvu.edu

# PEs	Wall Clock (in sec.)	Speedup	Efficiency
1	68,675	1.00	1.00
2	31,466	2.18	1.09
8	8,729	7.87	0.98

Table 1: Speedup & Efficiency for the turbulent jet problem with 160 $\times$ 80 $\times$ 80 cells

Case	Grid Layout	Tot. Vertices	# PEs	Timestep	$\frac{CPU sec}{timestep}$
A	208 x 100 x 100	2,184,840	16	1×10^{-4}	130
B	288 x 122 x 122	4,370,000	32	5×10^{-4}	68

Table 2: Higher grid resolution simulation details

model was facilitated with a substantially improved convection scheme based on central differencing for the convective terms in the momentum equations and Quasi-Second-order Upwind (QSOU) for the density and energy equations. Upto 3 % grid stretching was employed in radial plane (y-z) to capture the core flow and maintain the second order spatial accuracy as close as possible. Table 1 illustrates the speedup and efficiency figures for the grid configuration of 1,024,000 cells.
Two preliminary large scale runs over one million vertices were performed at different grid resolutions. Table 2 shows the grid layout, total number of vertices, total number of processors (PEs) employed and average CPU time required per each timestep of the simulation for each case. Although the simulation of Case A & B are not completed yet, the preliminary results indicate legitimate solutions. In particular, the velocity contour plots from the preliminary results (at t = 0.05 sec) of the developing jet performed on 16 processors for Case A indicate that the symmetry of the jet starts to break-up at about four jet diameters which agrees with the experimental observations (Figures 1 and 2). Original KIVA-3 runs at these grid resolutions that are required for the calculation of the speedup and parallel efficiency for these cases were not performed due to very long turn-around time on single processor. However, based on the speedup figures determined from other benchmark problems, the overall results indicate that for the selected test cases, a speedup close to linear speedup can be achieved even with grids larger than one million vertices. Also, a minimum parallel efficiency of 70-80% is maintained. Although the benchmark runs were performed only on SGI Origin 2000 and not all of the features of KIVA-3 are facilatated, these current speedup results are promising. Furthermore the maximum grid resolution that could be achieved with KIVA-3 has been significantly improved compared to the original version of the code on a single processor.
The implementation of the features excluded in the first phase are under progress and benchmark runs that will incorporate these features will be conducted. Considering the fact that commodity parts based high performance computer clusters are becoming quite popular for the researchers and institutions with limited resources, a detailed discussion will be presented in the final paper to demonstrate the speedup that could be achieved on a 20 node DEC Alpha based Beowulf type cluster and the experiences acquired during the development of the cluster including the difficulties in porting a CFD code onto the cluster.

Acknowledgement: This research has been conducted under the sponsorship of U.S. Department of Defense, Army Research Office through EPSCoR Program (Grant No: DAAH04-96-1-0196).

References

[1] Amsden, A.A., (1993) "KIVA-3: A KIVA Program with Block-Structured Mesh for Complex Geometries",Los Alamos National Laboratory, Technical Report : LA-12503-MS, Los Alamos, NM 87545.

[2] Yaşar, O., (1998), "A Scalable Model for Complex Flows", *Int. J. Computers & Mathematics with Applications*, 35:117–128.

[3] Gel, A., "Improvement of the Efficiency and Accuracy of a Navier-Stokes Solver", Ph.D. Dissertation, Mechanical & Aerospace Engng. Dept., West Virginia University, May 1999.

[4] Smith, J., Celik, I. and Yavuz, I., (1999), "Investigation of the LES Capabilities of An Arbitrary Lagrangian-Eulerian (ALE) Method", AIAA-99-0421 pp. 1–11.

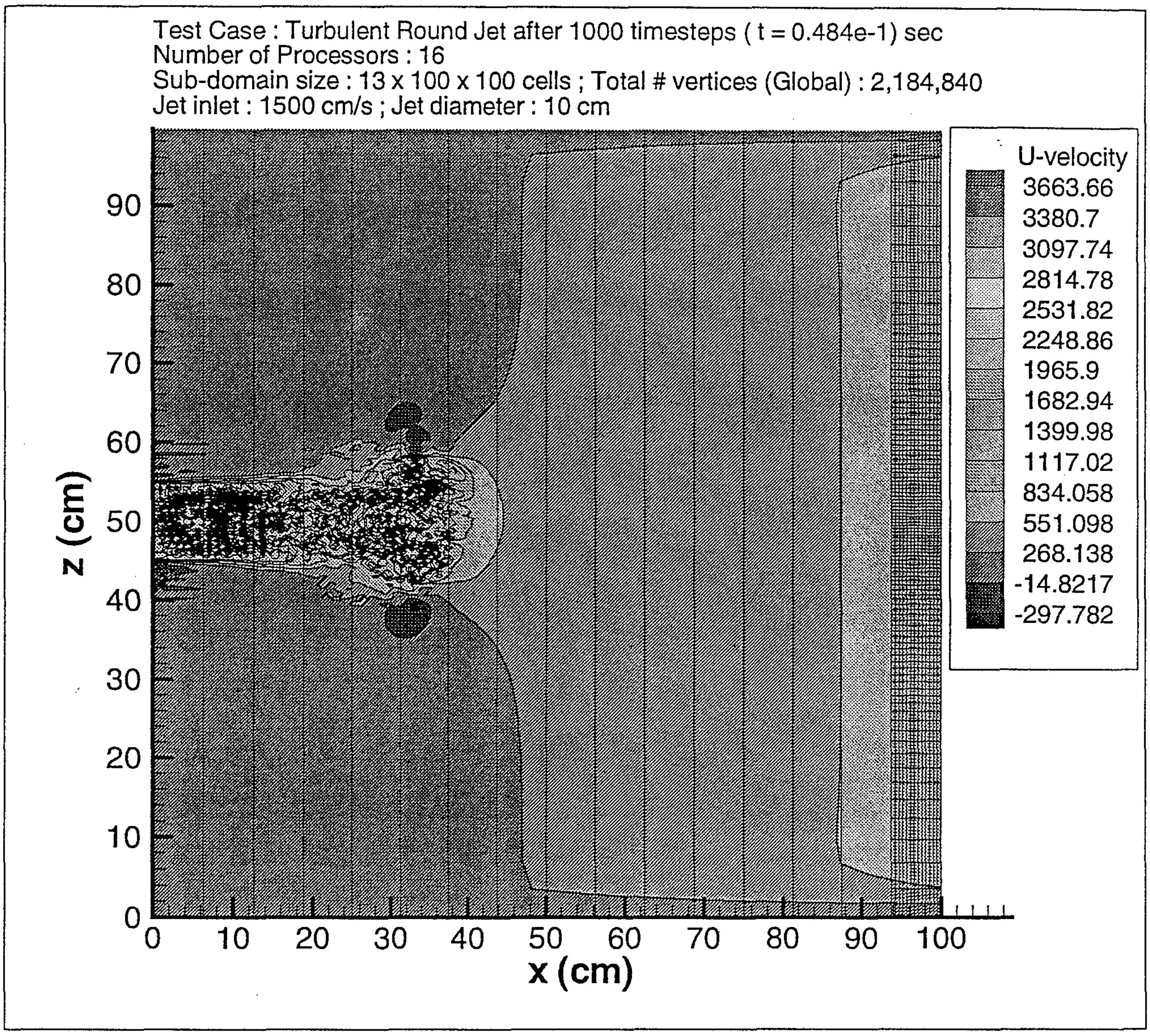


Figure 1: U-velocity contours on y = 0.0 cm plane at t= 0.05 sec (2.1 million vertices)

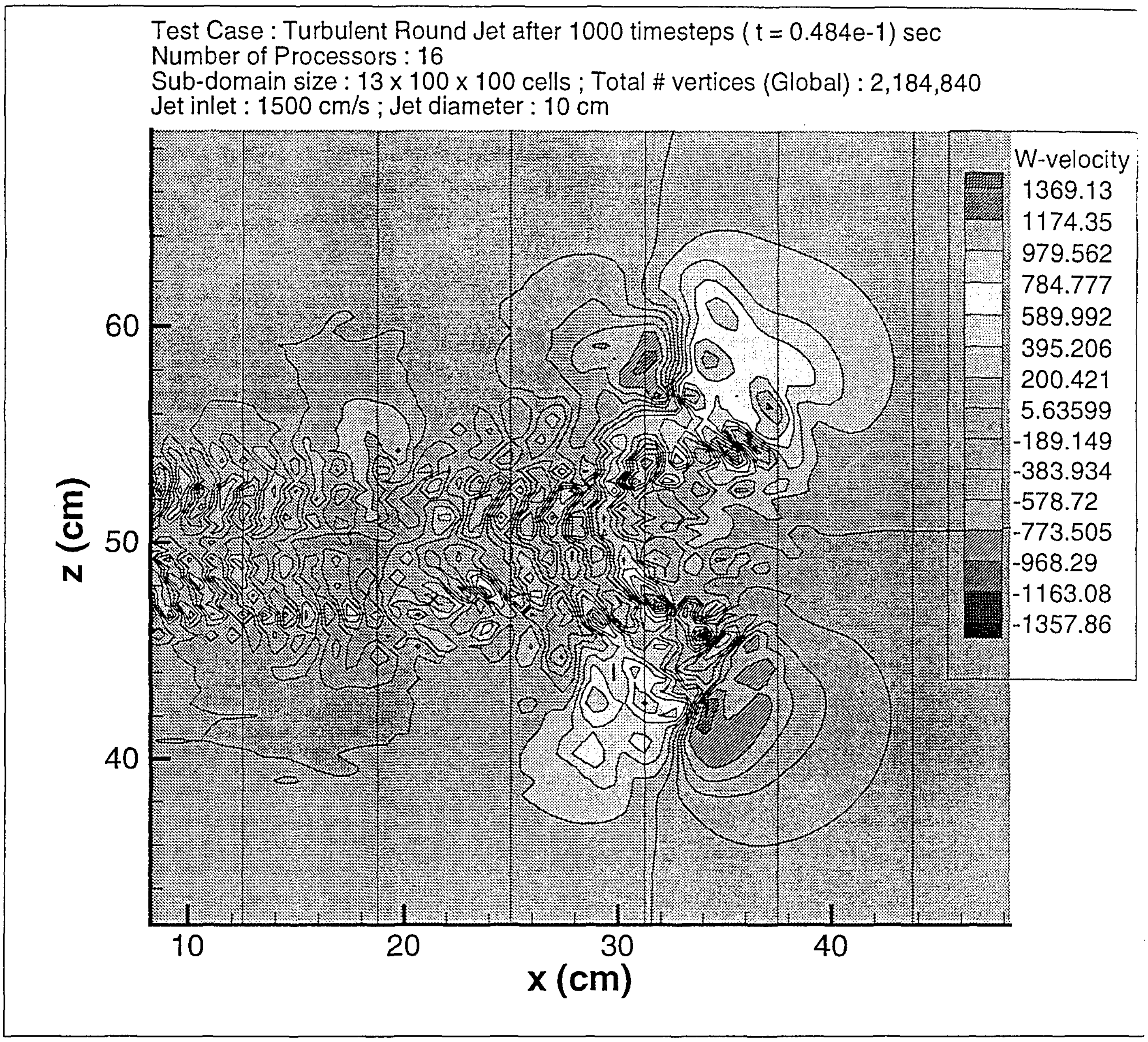


Figure 2: Enlarged view of W-velocity contours on y = 0.0 cm plane at t= 0.05 sec

Parallel Strategies for Agglomeration Multigrid in Finite Element CFD Solvers

by

Y. F. Hu, M. Ashworth and *D. R. Emerson[1]

Submitted to
Parallel CFD Workshop: *Experiences in Implementation*
June 16-18, 1999, Istanbul, Turkey

Introduction

Unstructured CFD solvers, based upon Finite Element Methods, have received much attention over the years. Recently, industrial companies have shown an increasing interest in unstructured grids because they offer the potential of being able to generate complex meshes, such as a complete aircraft, with relative ease. From an industrial viewpoint, this ranks very high in their list of priorities. As computing resources have continued to increase in power, the expectations and demand for quicker solutions have also increased. However, the move towards modelling increasingly complex geometries, such as complete aircraft, requires very substantial computing resources, particularly if the simulation must be carried out within a time suitable for analysis in an industrial context. The use of parallel computers, in this context, can therefore play a crucial role in delivering results of large-scale computations in a reasonable time. This paper will discuss the parallel development of FLITE3D, a three-dimensional unstructured CFD solver used by British Aerospace. A complete suite of routines is available that involves: (i) surface mesh generation; (ii) volume (tetrahedral) mesh generation; (iii) pre-processing (which includes the coarse mesh generation); (iv) CFD flow solver. The flow solver is based upon a Galerkin finite element method. The work to be reported involved the parallelisation of the flow solver and substantial additions to the pre-processor stage.

Governing Equations

The equations governing a compressible inviscid fluid are given by the Euler equations:

$$\frac{\partial U}{\partial t}+\frac{\partial F_i}{\partial x_i}=0 \qquad i=1,2,3$$

where the transpose of U is given by:

$$U=[\rho,\rho u,\rho v,\rho w,\rho\varepsilon]^T$$

and the flux vector F is defined by:

* Corresponding author
[1] CLRC Daresbury Laboratory

$$F = \begin{pmatrix} \rho u & \rho v & \rho w \\ \rho u^2 + p & \rho uv & \rho uw \\ \rho uv & \rho v^2 + p & \rho vw \\ \rho uw & \rho vw & \rho w^2 + p \\ u(\rho\varepsilon + p) & v(\rho\varepsilon + p) & w(\rho\varepsilon + p) \end{pmatrix}$$

In the foregoing notation, u, v and w are the fluid velocity components in the x, y and z directions, and p, ρ and ε are the pressure, density and total specific energy of the fluid, respectively. For a perfect gas, the equation of state is given by:

$$p = (\gamma - 1)\rho\left(\varepsilon - 0.5\sum_{j=1}^{3} u_j u_j\right)$$

The solution is advanced in time by a 5-stage Runga-Kutta explicit time-stepping algorithm. It is well known that explicit schemes have poor convergence properties because they are not very efficient at removing the low frequency errors. However, multigrid schemes can overcome this difficulty by solving the problem on successively coarser meshes where the low frequency errors associated with the current grid level appear as high frequency errors on the coarser mesh. These errors are efficiently removed and the process is continued recursively down to the coarsest mesh level. This approach considerably enhances the convergence properties of the algorithm. For this application, an agglomeration multigrid procedure [1] is used. This method works by fusing, or agglomerating, the control volumes in a heuristic fashion. It has been shown to be very effective for both its convergence and its ability to handle complex meshes [2]. A difficulty associated with multigrid on unstructured meshes is how best to generate the coarse meshes. For structured meshes, this presents minimal difficulty. A number of approaches have been tried for unstructured grids which include: (a) Generating an initial coarse mesh and each subsequent level is refined based upon the original coarse grid. A number of disadvantages that arise from this approach are that the coarsest mesh is pre-determined, the initial grid distribution may have to be quite fine to capture important geometric details or it may not contain sufficient grid points to capture these features. (b) Using a non-nested approach whereby each mesh level is generated independently. The flow data can be transferred between each mesh level by using a piecewise-linear interpolation scheme. The independent generation of grid points does present a problem as there are not, in general, any common points between the successive meshes.
It is important for the coarsest mesh to still capture the geometric details and this becomes particularly difficult in three-dimensions. If critical features are not captured the flow results will clearly be affected. In both the approaches outlined above, the problems associated with the coarsest mesh could lead to such difficulties being encountered. In order to avoid this, a significant burden is placed upon the user. The approach used in FLITE3D, whereby the control volumes are agglomerated together, has been shown to be an effective solution to this problem. An illustration of its convergence properties for a wing-body configuration is shown in figure 1. For this problem, 51737 grid points were used with 302079 tetrahedra. The Mach number was 0.8 and the angle-of attack was set to 2.0 degrees.

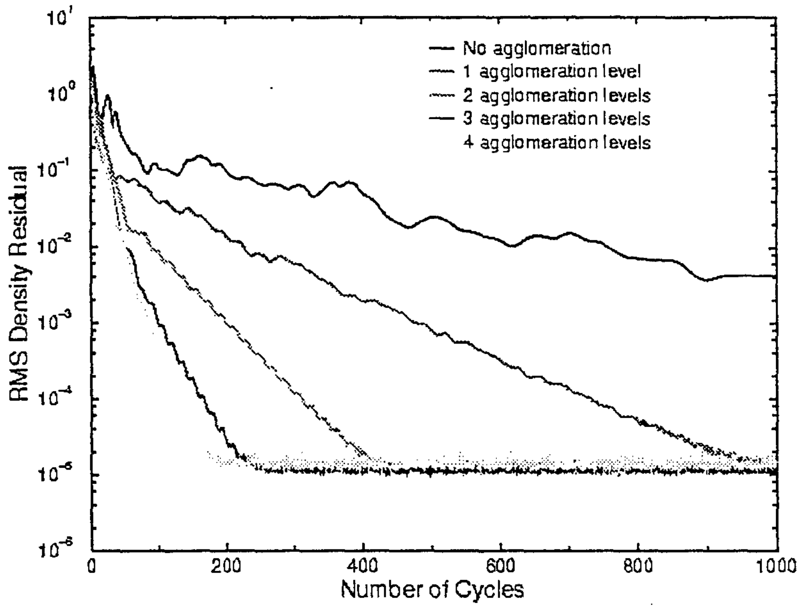


Figure 1: Convergence of the flow solver using agglomeration multigrid

Domain Decomposition

A number of highly efficient packages are now available for partitioning unstructured meshes and Metis [3] has been used for this project. The vast majority of the work has been concerned with the pre-processor stage as a key requirement of the project was minimal intrusion into the flow solver. To allow for portability between various architectures, MPI was used as the message passing standard (although an option to use PVM is also available). The target architecture for British Aerospace was a 128 processor Silicon Graphics Origin but the code has been tested on a range of computer platforms. Figure 2 shows the surface of a Falcon aircraft partitioned for 4 processors:

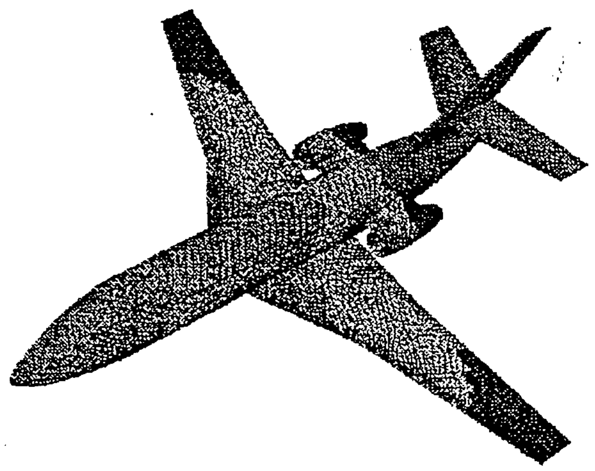


Figure 2: A Falcon aircraft partitioned for 4 processors

The talk will present more in-depth results from a range of distributed memory and shared memory machines and discuss the strategy selected for parallelising the agglomerated multigrid procedure. Relevant industrial issues concerning the parallelisation will also be covered.

[1] M. H. Lallemand, H. Steve, and A. Dervieux, 'Agglomeration for the Three-Dimensional Euler Equations', A.I.A.A.J., 33(4), 633-640, 1995.

[2] V. Venkatakrishnan and D. J. Mavriplis, 'Unstructured Multigridding by Volume: Current Status', Computers Fluids, 21(3), 397-433, 1992.

[3] G. Karypis and V. Kumar, 'A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs', Tech. Rep. 95-035, Dept. Computer Science, University of Minnesota, Minneapolis, MN 55455, 1995.

Efficient Parallelization of an Unstructured Grid Solver: A Memory-Centric Approach

D. K. Kaushik* D. E. Keyes†

1 Introduction and Motivation

With teraflops-scale computational modeling expected to be routine by 2003-04, under the terms of the Accelerated Strategic Computing Initiative (ASCI), there is an increasing need for highly scalable solvers. Since the gap between cpu speed and memory response time is widening further, we need to adopt a memory-centric view of the computation. Until automated tools like parallel compilers or source-to-source translators can discover enough of the locality (to minimize the vertical memory hierarchy traffic) and concurrency (with as little horizontal memory traffic as possible) latent in most scientific computations, their manual expression is the only alternative for achieving high performance. We present parallelization and performance tuning experiences with a three-dimensional unstructured grid Euler code (compressible and incompressible) from NASA, which we have parallelized in the PETSc [3] framework.

2 Memory-Centric Approach

We view a PDE computation as predominantly a mix of loads and stores with embedded floating point operations (FLOPs). Since FLOPs are cheap, we concentrate on minimizing the memory references and emphasize strong sequential performance as one of the factors needed for aggregate performance worthy of the theoretical peak of a parallel machine. We use *interlacing* (creating spatial locality for the data items needed close in time), *structural blocking* for a multicomponent system of PDEs (cutting the number of integer loads significantly,

*Computer Science Department, Old Dominion University, Norfolk, VA 23529-0162 and Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439-4844, kaushik@cs.odu.edu. Supported in part by Wallace Givens Fellowship from Argonne National Laboratory and by a GAANN fellowship from the U. S. Department of Education.

†Computer Science Department, Old Dominion University, Norfolk, VA 23529-0162 and ICASE, NASA Langley Res. Ctr., Hampton, VA 23681-2199, keyes@icase.edu. Supported in part by the National Science Foundation under grant ECS-9527169, by NASA under contracts NAS1-19480 and NAS1-97046, and by the U.S. Department of Energy under ANL subcontract 982232402 and LLNL subcontract B347882

and enhancing reuse of data items in registers), and *vertex and edge reorderings* (increasing the level of temporal locality).

The basic philosophy of any efficient parallel computation is "owner computes," with message merging and overlapping communication with computation where possible via split transactions. Each processor "ghosts" its stencil dependences on its neighbors' data. Grid functions are mapped from a global (user) ordering into contiguous local orderings (which, in unstructured cases are designed to maximize spatial locality for cache line reuse.) Scatter/gather operations created between local sequential vectors and global distributed vectors, based on runtime-deduced connectivity patterns.

3 Parallel Newton-Krylov-Schwarz Solvers and Software

Our framework for an implicit PDE solution algorithm, with pseudo-timestepping to advance towards an assumed steady state, has the form: $(\frac{1}{\Delta t^{\ell}})\mathbf{u}^{\ell} + \mathbf{f}(\mathbf{u}^{\ell}) = (\frac{1}{\Delta t^{\ell}})\mathbf{u}^{\ell-1}$, where $\Delta t^{\ell} \to \infty$ as $\ell \to \infty$, $\mathbf{u}$ represents the fully coupled vector of unknowns, and the steady-state solution satisfies $\mathbf{f}(\mathbf{u}) = 0$.

Each member of the sequence of nonlinear problems, $\ell = 1, 2, \ldots$, is solved with an inexact Newton method. The resulting Jacobian systems for the Newton corrections are solved with a Krylov method, relying directly only on matrix-free operations. The Krylov method needs to be preconditioned for acceptable inner iteration convergence rates, and the preconditioning can be the "make-or-break" feature of an implicit code. A good preconditioner saves time and space by permitting fewer iterations in the Krylov loop and smaller storage for the Krylov subspace. An additive Schwarz preconditioner [4] accomplishes this in a concurrent, localized manner, with an approximate solve in each subdomain of a partitioning of the global PDE domain. Applying any preconditioner in an additive Schwarz manner tends to increase flop rates over the same preconditioner applied globally, since the smaller subdomain blocks maintain better cache residency, even apart from concurrency considerations. Combining a Schwarz preconditioner with a Krylov iteration method inside an inexact Newton method leads to a synergistic parallelizable nonlinear boundary value problem solver with a classical name: Newton-Krylov-Schwarz (NKS) [6]. Combined with pseudo-timestepping, we write ΨNKS.

We employ the PETSc package [3], which features distributed data structures — index sets, vectors, and matrices — as fundamental objects. Iterative linear and nonlinear solvers, implemented in as data structure-neutral a manner as possible, are combinable modularly, recursively, and extensibly through a uniform application programmer interface. Portability is achieved through MPI, but message-passing detail is not required in user code. We use MeTiS [7] to partition the unstructured grid.

4 Parallel Port of FUN3D

The demonstration code, FUN3D, is a tetrahedral vertex-centered unstructured grid code developed by W. K. Anderson of the NASA Langley Research Center for compressible and incompressible Euler and Navier-Stokes equations [2, 1]. FUN3D uses a control volume discretization with variable-order Roe schemes for approximating the convective fluxes and and a Galerkin discretization for the viscous terms. Our parallel experience with FUN3D is with the incompressible/compressible Euler subset thus far, but nothing in the solution algorithms or software changes with additional physical phenomenology. Of course, convergence rate will vary with conditioning, as determined by Mach and Reynolds numbers and the correspondingly induced grid adaptivity. Furthermore, robustness becomes more of an issue in problems admitting shocks or making use of turbulence models. The lack of nonlinear robustness is a fact of life that is largely outside of the domain of parallel scalability. In fact, when nonlinear robustness is restored in the usual manner, through pseudo-transient continuation, the conditioning of the linear inner iterations is enhanced, and parallel scalability may be improved. In some sense, the Euler code, with its smaller number of flops per point per iteration and its aggressive pseudotransient buildup towards the steady state limit may be a *more*, not less, severe test of scalability.

5 Results and Discussions

5.1 Sample Sequential Performance

As observed in [8] for the same unstructured flow code, data structure storage patterns for primary and auxiliary fields should adapt to hierarchical memory through: (1) interlacing, (2) blocking of degrees of freedom (DOFs) that are defined at the same point in point-block operations, and (3) reordering of edges for reuse of vertex data. For vertices we have used Reverse Cuthill McKee (RCM) ordering [5], which is known for reducing cache misses.

Table 1 shows the effect of these techniques on one processor of SGI's Origin 2000. The combination of the three effects can enhance overall execution time by a factor of 5.7 (a table comparing several architectures is available in [9]). To further understand these results, we carried out hardware counter profiling on R10000 processor. Figure 1 shows that edge reordering reduces the TLB misses by two orders of magnitude while secondary cache misses (which are very expensive) are reduced by a factor of 3.5.

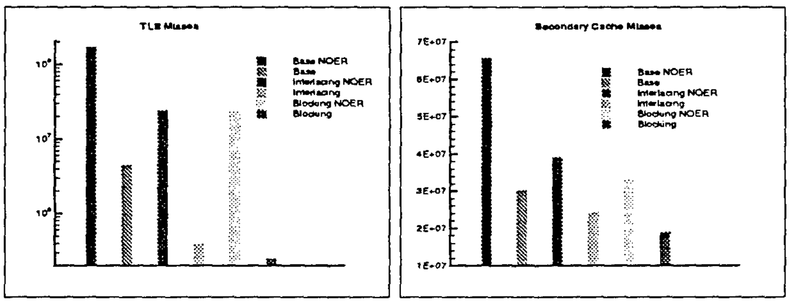
5.2 Parallel Scalability Studies

Parallel scalability of PETSc-FUN3D is shown in Figure 2 for a tetrahedral grid with 2.8 million vertices running on up to 1024 Cray T3E processors. We see that the implementation efficiency of parallelization is 82% in going from 128 to 1024 processors. Also the Mflop/s per processor are close to flat over this range. This further emphasizes the fact that good serial performance is

Table 1: Flow over M6 wing; fixed-size grid of 22,677 vertices (90,708 DOFs incompressible; 113,385 DOFs compressible); MIPS R10000, 250MHz, cache: 32KB data / 32KB instr / 4MB L2. Activation of a layout enhancement is indicated by a "×" in the corresponding column. Improvement ratios are averages over the entire code; different subroutines benefit to different degrees.

Enhancements			Results			
Field Interlacing	Structural Blocking	Edge Reordering	Incompressible Time/Step	Incompressible Ratio	Compressible Time/Step	Compressible Ratio
			83.6s	—	140.0s	—
×			36.1s	2.31	57.5s	2.44
×	×		29.0s	2.88	43.1s	3.25
		×	29.2s	2.86	59.1s	2.37
×		×	23.4s	3.57	35.7s	3.92
×	×	×	16.9s	4.96	24.5s	5.71

Figure 1: TLB and secondary cache misses corresponding to the data in Table 1. "NOER" denotes no edge ordering, otherwise edges are reordered by default.

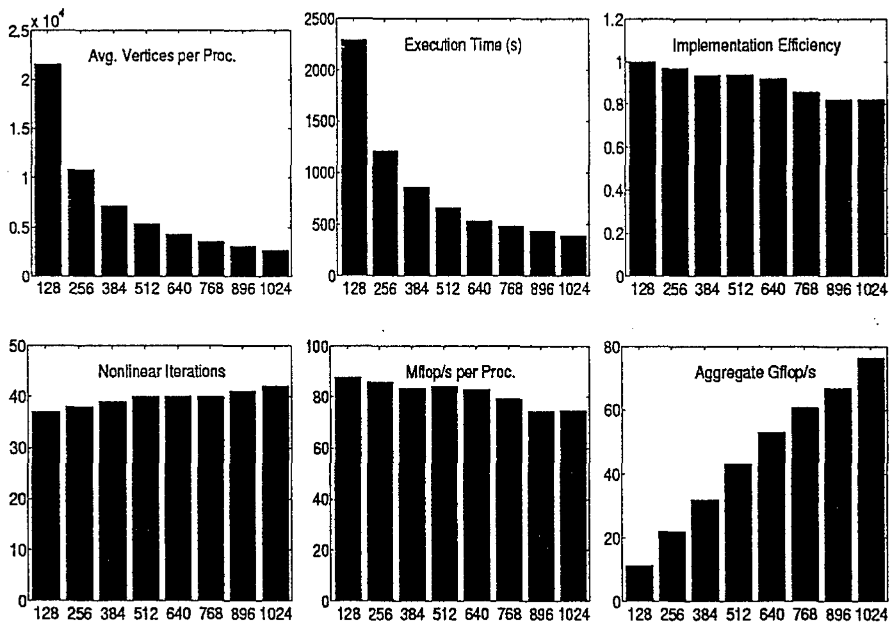


necessary for good parallel performance. Observe, as well, that the number of nonlinear iterations varies only from 37 to 42 over an 8-fold increase in the number of processors. This is much less serious degradation than predicted by the linear elliptic theory, see [10]).

6 Conclusions and Future Work

Unstructured implicit CFD solvers are amenable to scalable implementation, but careful tuning is needed to obtain the best product of per-processor efficiency and parallel efficiency. We [9] and others have solved problems of millions of vertices on hundreds of processors at rates approaching hundreds of gigaflop/s, and we believe such performance is extensible, with further effort, to the teraflop/s regime. In the future, we hope to enhance per-processor per-

Figure 2: Parallel performance for a fixed size grid of 2.8 million vertices run on upto 1024 Cray T3E 600 MHz processors



formance through improved spatial and temporal locality and the exploitation of "processors in memory" (PIM). We also hope to enhance parallel efficiency through algorithms that synchronize less frequently, and through multiobjective partitioning, which equidistributes communication (surface) work as well as computational (volume) work.

Acknowledgements

The authors thank W. Kyle Anderson of the NASA Langley Research Center for providing FUN3D. Satish Balay, Bill Gropp, Lois McInnes, and Barry Smith of Argonne National Laboratory co-developed the PETSc software employed in this paper. Computer time was supplied by DOE (through Argonne and NERSC) and Silicon Graphics.

References

[1] W. K. Anderson and D. L. Bonhaus. An implicit upwind algorithm for computing turbulent flows on unstructured grids. *Computers and Fluids*, 23(1):1–21, 1994.

[2] W. K. Anderson, R. D. Rausch, and D. L. Bonhaus. Implicit/multigrid algorithms for incompressible turbulent flows on unstructured grids. *Journal of Computational Physics*, 128:391–408, 1996.

[3] Satish Balay, William Gropp, Lois Curfman McInnes, and Barry Smith. The Portable, Extensible,Toolkit for Scientific Computing (PETSc) ver. 22. http://www.mcs.anl.gov/petsc/petsc.html, 1998.

[4] X. C. Cai. Some domain decomposition algorithms for nonselfadjoint elliptic and parabolic partial differential equations. Technical Report 461, Courant Institute, New Haven, Connecticut, 1989.

[5] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *ACM Proceedings of the 24th National Conference*, 1969.

[6] W. D. Gropp, L. C. McInnes, M. D. Tidriri, and D. E. Keyes. Parallel implicit pde computations. In A. Ecer, D. Emerson, J. Periaux, and N. Satofuka, editors, *Proceedings of Parallel CFD'97*, pages 333–344. Elsevier, 1997.

[7] G. Karypis and V. Kumar. A fast and high quality schema for partitioning irregular graphs. *SIAM Journal of Scientific Computing*, 20(1):359–392, 1999.

[8] D. K. Kaushik, D. E. Keyes, and B. F. Smith. On the interaction of architecture and algorithm in the domain-based parallelization of an unstructured grid incompressible flow code. In C. Farhat, et al., editor, *Proceedings of Proceedings of the 10th International Conference on Domain Decomposition Methods*, pages 311–319. Wiley, 1997.

[9] D. K. Kaushik, D. E. Keyes, and B. F. Smith. Newton-Krylov-Schwarz methods for aerodynamic problems: Compressible and incompressible flows on unstructured grids. In C. -H. Lai, et al., editor, *Proceedings of the 11th International Conference on Domain Decomposition Methods.* Wiley, 1998.

[10] B. F. Smith, P. Bjorstad, and W. Gropp. *Domain Decomposition.* Cambridge University Press, 1996.

Submitted to Parallel CFD Workshop, June 16-18, 1999
Istanbul Technical University, Istanbul, Turkey

EFFICIENCY STUDIES OF A PARALLEL SUBSTRUCTURING ALGORITHM ON DIFFERENT PLATFORMS

S. Kocak*, H.U. Akay, and A. Ecer

CFD Laboratory
Department of Mechanical Engineering
Indiana University-Purdue University Indianapolis
Indianapolis, Indiana USA

ABSTRACT

Introduction

Effectiveness of a parallel algorithm is usually assessed via studying speedup, efficiency, and communication versus computation times. The most commonly used assessment criterion among these is the speedup curve, which represents the elapsed time spent for execution of a program using various number of processors that are normalized with respect to the time elapsed to run with one processor. Here, the elapsed time is the total time spent for execution of a program, which is summation of computation and communication times. Computation time is related to CPU speed of a processor and communication time is related to message passing speed between processors. Therefore, the speedup computations of parallel algorithms are dependent on not only to algorithms but also the platforms which they are tested on. To make a realistic assessment of a parallel algorithm or to compare different algorithms, the properties of the platforms that are used must also be taken into consideration. Here, it should be noted that a parallel algorithm's computation versus communication load is the key, which makes either the CPU or the message passing speed of a platform dominant for elapsed time. For example, to compare speedups of two parallel algorithms with considerable communication load, where one algorithm is tested on a platform with 100Mb ethernet and another platform with 10Mb ethernet, is not realistic. Since the message passing rates are very different, the effect of communication speed might be considerable in communication bound algorithms. Currently, there are various workstation architectures with different operating systems (e.g., UNIX, LINUX, WINDOWS/NT), message passing software (e.g., PVM, MPI), and message passing interfaces (e.g., ethernet, fast ethernet). Due to the variety of

*Permanent Address: Department of Civil Engineering, Cukurova University, Adana, Turkey architectures available, a parallel algorithm's efficiency may strongly depend upon the properties of the workstation it is tested on.

Present Study

In order to investigate the effect of platforms on efficiency of parallel algorithms, we tested a substructuring (Schur complement) based finite element parallel algorithm (e.g., Farhat and Wilson [1]) on different platforms. We used our previously developed finite element computer program for solving Poisson equations as test cases [2]. A Gausssian elimination based direct equation solver is used. A given computational domain is partitioned into subdomains using a greedy-based grid-partitioning algorithm. Since in this algorithm the interface equations are inherently implicit, it is more computationally dominant than the explicit domain-decomposition algorithms. Message passing becomes dominant only after large number of processors are used.

Results to be Presented

The communication and computation timing results obtained on different platforms will be presented at the time of the *PCFD Workshop.* Platforms to be tested will include workstations and PCs with Unix, Linux and Windows/NT operating systems and different network characteristics.

References

1. C. Farhat and E. Wilson, "A New Finite Element Concurrent Computer Program Architecture," *International Journal for Numerical Methods in Engineering,* 24, (1987), 1771-1792.

2. S. Kocak, H.U. Akay, and A. Ecer, "Parallel Implicit Treatment of Interface Conditions in Domain Decomposition Algorithms," *Proceedings of Parallel CFD '98*, Edited by A.C. Lin, et al., Elsevier Science, Amsterdam, 1999 (in print).

Abstract provided for the Parallel CFD Workshop in Istanbul, 1999

APPLICATION OF EVOLUTIONARY ALGORITHMS TO AIRFOIL DESIGN

M. Meinke, A. Meijering, S. Mählmann

Aerodynamisches Institut, Wüllnerstraße zw. 5 und 7, 52062 Aachen, Germany
Tel.: +49-241-804821, E-mail: meinke@aia.rwth-aachen.de

One of the classical goals in airplane development is the minimization of the aerodynamic drag. One possible way to decrease the drag is to delay the transition to the turbulent boundary layer, i.e. to move the transition point as far downstream as possible. This can be achieved with an appropriate profile shape. The basic numerical methods to determine such profile shapes are presented and their application on parallel computers is demonstrated in this paper. The optimization algorithm applied is an evolutionary strategy [1], which can be implemented easily with a high parallel efficiency on parallel computers. This method is robust and well suited also for the detection of global minimums, but requires a large number of evaluations of the quality function. The determination of the transition point and profile drag is therefore carried out here with a computationally cheap Euler-boundary layer solution (MSES, [2]). The characteristics of the resulting optimized shapes are checked with the help of a solution of the Navier-Stokes equations and a transition prediction based on a e^n-method. An example for an airfoil optimization is shown in Fig. 1 for a NLF-0416 profile, which shows the pressure distribution and the predicted transition point for the initial and optimized airfoil shape. The parameter for which the optimization was carried out is a combination of the transition point location and the length of the laminar separation bubble. For a Mach number of $Ma_\infty = 0.4$ and a Reynolds number of $Re_\infty = 1.66 \cdot 10^6$ the transition point could be shifted from 42% cord length to 62%. The drag coefficient was reduced by about 14% compared to that of the original profile. Such an optimization typically requires 60-100 generations with a population size of 20 individuals. This optimization was run on a Hitachi SR2201 with 20 processors within less than 2-3 hours computing time.

Currently, these investigations are extended to transonic and off-design flow conditions. In addition, experiments are carried out with a wind tunnel model with a flexible upper surface to confirm the results obtained with the numerical methods.

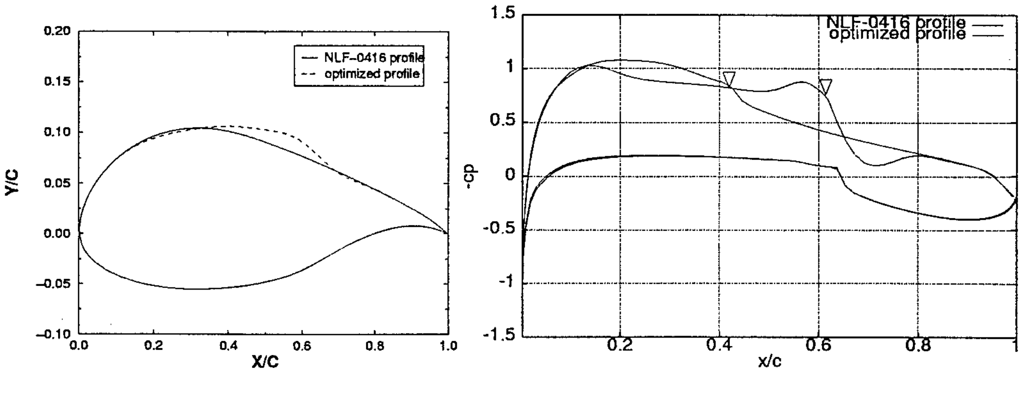


Figure 1: Shape and the corresponding pressure distribution for the NLF-0416 profile and the optimized profile shape. Triangles indicate the transition point location.

References

[1] T. Bäck. *Evolutionary Algorithms in Theory and Practice.* Oxfort University Press, New York, 1996.

[2] M. Drela. A user's guide to MSES 2.95. *MIT Computeral Aerospaces Science Laboratory,* 1996.

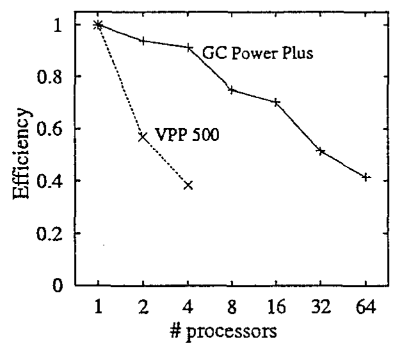
PARALLELIZED MULTIGRID ACCELERATED SOLUTION SCHEMES FOR COMPRESSIBLE AND INCOMPRESSIBLE FLOWS

M. Meinke

Aerodynamisches Institut, Wüllnerstraße zw. 5 und 7, 52062 Aachen, Germany
Tel.: +49-241-804821, E-mail: meinke@aia.rwth-aachen.de

Parallelized solution methods for the Navier-Stokes equations for compressible and incompressible flows are discussed. The main focus is on explicit multigrid accelerated schemes. The implementation and performance of such algorithms are discussed and compared to those of implicit methods. Results of several recent applications are presented, which were obtained on high-performance vector-parallel architectures e.g. Nec-SX4-32 or SNI/Fujitsu VPP300-8 and also on workstation and PC-clusters running Linux.

Different parallelized algorithms for the simulation of incompressible flow are compared: an explicit and implicit scheme both based on pressure correction methods and an implicit scheme based on the concept of artificial compressibility. Within the implicit schemes a dual-time stepping technique is used, so that the time accuracy can simply be chosen by a backward difference approximation. The solution of the linear systems of equations, is carried out with a locally preconditioned (ILU) Bi-CGStab method and, alternatively, with a block Gauß-Seidel line-relaxation scheme. All methods are formulated for node-centered non-staggered grids. The momentum interpolation of Rhie and Chow is used to avoid an odd-even decoupling of the pressure field in case of the pressure corrections methods. Parallelization is achieved with a message passing library (PVM, MPI). The parallel efficiency of the solver for the linear system of equations is shown in Fig. 1 for constant local and constant global problem size and one iteration step. The efficiency remains almost constant if more than eight processors are used, which indicates that the algorithm is applicable also on massively parallel systems. The efficiency for constant global problem size decreases especially fast on vector parallel machines, since the arithmetic speed is reduced considerably with decreasing vector length. In Fig. 2 the convergence history for the solution of a linear system of equation, which is very asymmetric and not well conditioned, is shown for an increasing number of processors. The influence of the local preconditioning can be recognized clearly. For 64 processors about two times the iterations are required to reach the same residual drop compared to the single processor calculation. Since the linear equation considered here is one of the worst cases for the solver, the overall solution scheme still has an acceptable efficiency.



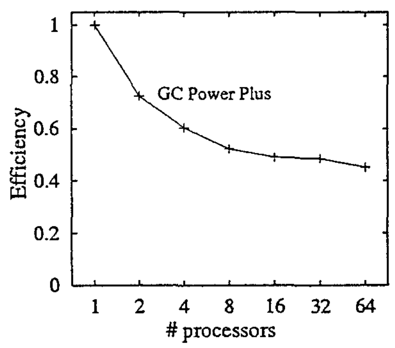
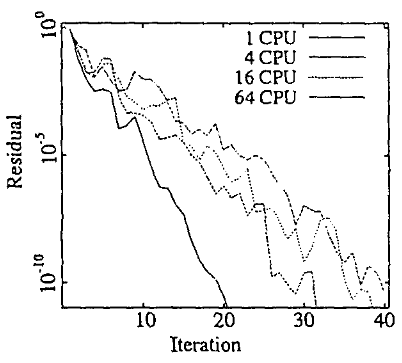


Figure 1: Efficiency of the parallelized, preconditioned BI-CGStab solver for a constant global problem size of 33×33×17 grid points (left) and a constant local problem size of 33×33×(5 × # processors) grid points.

In Fig. 2 a comparison of the performance of the three different schemes is shown. It indicates that in this case the implicit schemes are much more efficient than the explicit scheme, since the



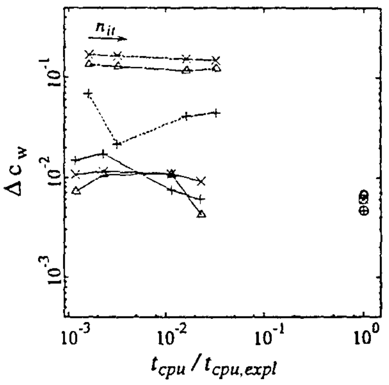


Figure 2: Residual of the iterative solution of one equation system with the Bi-CGStab method for different numbers of processors on the GC Power Plus (right). Relative error in the drag coefficient for a impulsively started flow around a circular cylinder at a Reynolds number of 40 as a function of the computing time (based on the time used by the explicit scheme) for three different physical time levels and increasing number of iterations n_{it} in the implicit scheme. o: explicit pressure correction scheme, —: implicit pressure correction scheme, - - -: implicit artificial compressibility scheme.

time step can be chosen much larger than in the explicit scheme ($\Delta t_{implicit}/\Delta t_{explicit}$=$10^4$). In addition the implicit pressure correction scheme seems to provide a better accuracy at the same computational cost than the implicit scheme based on the artificial compressibility concept.

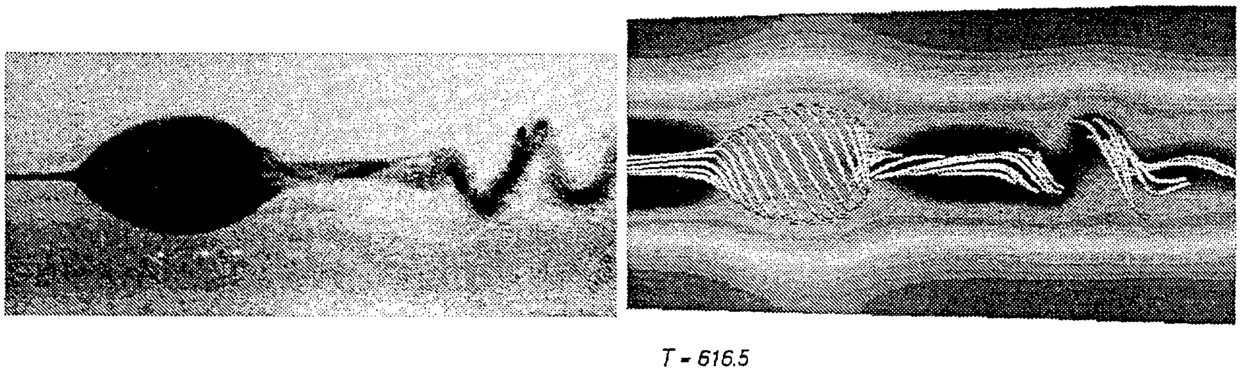


Figure 3: Visualization of vortex breakdown in water flow for Re = 4200 (left), streaklines taken from [1]. Visualization of the numerical solution of vortex breakdown for Re=3220 (right): vortex lines and pressure distribution in grey scales.

The solution schemes for incompressible flow were applied to the simulation of vortex dominated flows. Examples presented include the interaction of vortex rings and the simulation of vortex breakdown in a slightly diverging pipe as shown in comparison to experimental flow visualization of Faler [1], in Fig. 3.

The simulation of compressible flows is carried out with an implicit dual-time stepping method and an explicit Runge-Kutta scheme. For the solution of the linear system of equations in the implicit scheme an explicit multigrid accelerated scheme was used. In addition a scheme similar to the previously described methods for incompressible flows was applied. The performance of the multigrid accelerated scheme is shown in Fig. 4. The high performance on vector machines was achieved by formulating long vector loops which operate on all grid points within one grid

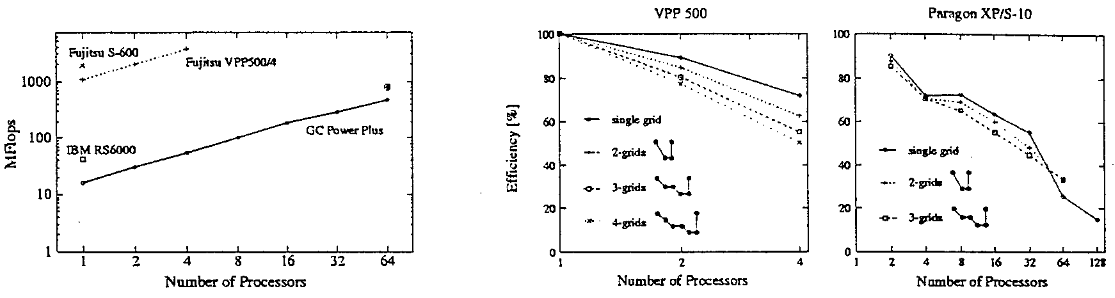


Figure 4: Performance of the explicit scheme for different hardware platforms. Parallel efficiency of the multigrid accelerated scheme for two different parallel computers.

block and grid level.

In addition a hybrid scheme will be presented, which is advantageous for unsteady flows. The explicit scheme is used in the major part of the domain of integration, except where the stability restriction requires a time step which is smaller than the desired resolution. In these regions, usually near rigid walls, an implicit solution used. Tests have shown that this scheme can be more than two times faster than the pure implicit scheme.

The solution schemes for compressible flows were applied to various flow problems like the simulation of the external flow around the model of a space transportation system with external combustion. In this case the flow is assumed to be in chemical equilibrium, so that a single mixture fraction variable is sufficient for the description of the chemical reactions.

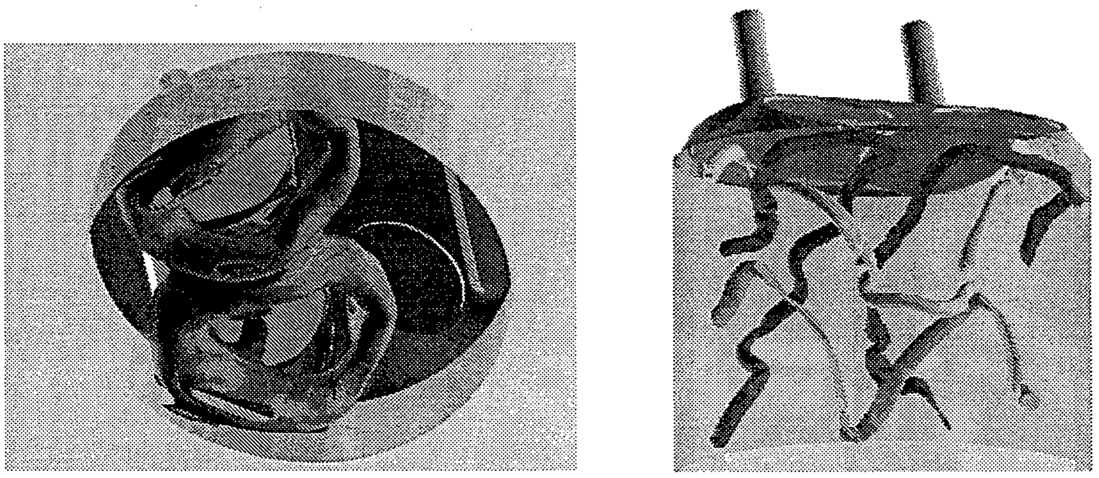


Figure 5: Vortex structures during the intake and compression stroke in a 4-valve IC engine. Surface of constant pressure at a crank angle of 60^o (left), selected vortex lines at a crank angle of 180^o (right).

An application on moving grids is shown in Fig. 5. The flow in a realistic piston engine is simulated on a grid, which is refined and coarsened during the intake and compression stroke. This simulation enabled the determination of the main vortex structures, which develop in the unsteady flow field.

The last applications presented are the simulation of turbulent flows with the help of LES. For this purpose a purely explicit scheme is used with a modified AUSM method which exhibits a low level of numerical dissipation. In Fig. 6 a round jet issuing in a still fluid is presented. In this case a turbulent pipe flow is simulated in parallel to the jet flow computation in order to provide turbulent inflow conditions at each time level. More details will be presented on the Workshop.

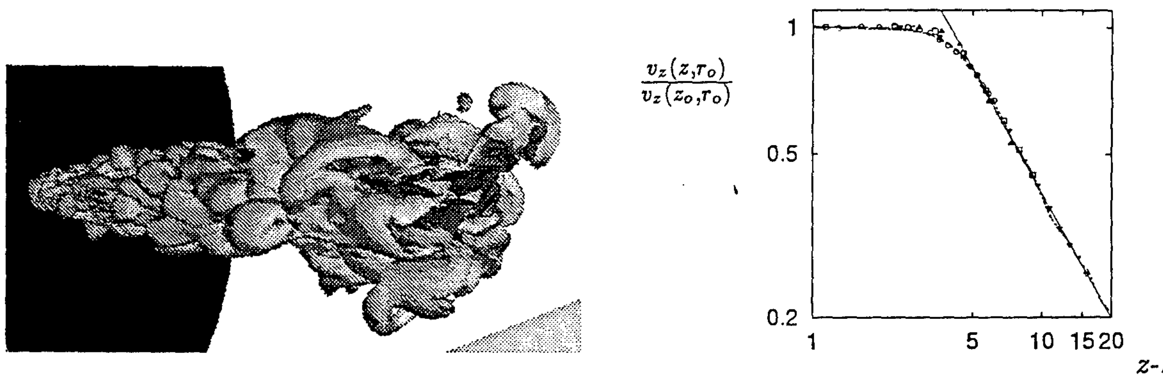


Figure 6: LES of a turbulent round jet at a Reynolds number of 5000. Surface of constant vorticity of the instantaneous flow field. Grey scales denote streamwise velocity. Decay of the centerline velocity for Re=5000 and Re=27000 in comparison to experimental data from [8, 10] and others.

References

[1] J. H. Faler and S. Leibovich. Disrupted states of vortex flow and vortex breakdown. *Phys. Fluids*, 20:1385–1400, 1977.

[2] J. Hofhaus. *Numerische Simulation reibungsbehafteter, instationärer, inkompressibler Strömungen- Vergleich zweier Lösungsansätze.* Diss., Aerodyn. Inst. RWTH Aachen, 1997.

[3] M. Kropp. *Reagierende Über- und Hyperschallumströmung eines Raumtransportsystems mit Außenverbrennung.* Diss., Aerodyn. Inst. RWTH Aachen, 1998.

[4] M. Meinke, A. Abdelfattah, and E. Krause. Simulation of piston engine flows in realistic geometries. In J. J. C. P. Kutler, J. Flores, editor, *15th International Conference on Numerical Methods in Fluid Dynamics*, pages 195–200. Springer Verlag, June 1996.

[5] M. Meinke and E. Krause. Simulation of incompressible and compressible flow on vector-parallel computers. In A. Ecer, D. Emerson, J. Periaux, and N. Satufoka, editors, *10th International Conference on Parallel CFD Developments and Applications of Technology, Hsinchu, Taiwan.* Elsevier, May 1998. To be published.

[6] M. Meinke, T. Rister, F. Rütten, and A. Schvorak. LES of free and wall bounded turbulent flows. In K. W. Morton, editor, *Lecture Notes in Physics, 16th International Conference on Numerical Methods in Fluid Dynamics, Arcachon, France*, volume 515, pages 201–206. Springer Verlag, July 1998.

[7] M. Meinke, C. Schulz, and T. Rister. LES of Spatially Developing Jets. In R. Friedrich and P. Bontoux, editors, *Computation and Visualization of Three-Dimensional Vortical and Turbulent Flows. Proceedings of the Fifth CNRS/DFG Workshop on Numerical Flow Simulation*, volume NNFM 64, pages 116–131. Vieweg Verlag, 1998.

[8] P. A. Monkewitz, D. W. Bechert, B. Barsikow, and B. Lehmann. Self-excited oscillations and mixing in a heated round jet. *J. Fluid Mech.*, 213:611–639, 1990.

[9] C. M. Rhie and W. L. Chow. Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA J.*, 21(11):1525–1532, 1983.

[10] S. Russ and P. J. Stykowski. Turbulent structure and entrainment in heated jets: The effect of initial conditions. *Phys. Fluids*, A 5(12):3216–3225, Dec. 1993.

ABSTRACT

PARALLEL COMPUTATION OF THERMOCAPILLARY BUBBLE AND DROP MIGRATION IN ZERO GRAVITY

Assistant Prof. Dr. Selman Nas
Istanbul Technical University,
Faculty of Aeronautics and Astronautics,
Department of Space Engineering,
80626 Maslak,Istanbul TURKEY
E-mail: nas@itu.edu.tr

The thermocapillary migration of many bubbles and drops in zero gravity is studied in two and three dimensions. The full Navier-Stokes equations and the energy equation for the temperature, are solved for the fluid inside and outside of the bubbles and drops by a front tracking/finite difference method. It is found that two bubbles, initially oriented arbitrarily with respect to the temperature gradient, tend to line up, side by side, perpendicular to the temperature gradient in both two and three dimensions.In the low Re and Ma number region drops behave like the bubbles but when both the Re and Ma are high, they tend to line up in tandem. Numerical simulation of a large mono-dispersed bubble system show that the bubbles form horizontal layers. Three-dimensional simulations confirm this tendency to form layers while the simulations of bubbles in polydispersed systems show the same behavior. Unlike two dimensional simulations, three dimensional simulation of a poly dispersed system shows that different sizes of bubbles form different layers. In three dimensional computations, an IBM SP2 parallel computer having 32 nodes is used to achieve the solution by using domain decomposition method.

A HIGH-ORDER ACCURATE, THREE-DIMENSIONAL EULER SOLVER ON DISTRIBUTED COMPUTERS

Yusuf Özyörük*
Department of Aeronautical Engineering
Middle East Technical University, 06531 Ankara, Turkey

Summary

In this paper, the porting of an originally data parallel, high-order accurate, three-dimensional (3D) Euler solver onto distributed memory computing platforms is described. This code was originally developed for predicting ducted fan noise on massively parallel computers (e.g. Connection Machine CM-5) using the single-instruction, multiple-data paradigm (SIMD). The ported code is intended to handle 3D wing geometries with appropriate modifications. In the SIMD approach there is a single domain mesh and an instruction is carried out for all data segments (i.e. on all processors), whereas in the present approach the domain is decomposed into smaller parts and the computational work is distributed over more than one processor to be performed simultaneously. The essential element of parallel processing is data communication among the processors and this is realized through the Message Passing Interface (MPI) standard in the present work. The distributed-memory parallel code has been tested on various platforms including a cluster of Pentium processors that are on a local area ethernet network.

1 Introduction

With the emergence of high-speed massively parallel computers and high-speed networking [1], great advances have been achieved in computational sciences. Among these are computational fluid dynamics (CFD) and computational aeroacoustics (CAA) that have found extensive application in aerospace engineering. However, owning and housing such high-technology tools is very costly and therefore, among the scientists there has been a growing trend to advocate more cost-effective parallel computing [2]. One way of accomplishing this is to utilize off-the-shelf computers, such as Pentium processors [3] or workstations that are readily available in a university environment or a company.

One application of parallel processing in CFD is the computation of highly vortical tip flowfields of helicopter blades [4]. Accurate simulations of such flowfields are important for accurate predictions of blade-vortex interaction phenomena, and consequently the unsteady blade loadings [5] and their resultant noise [6].

The present work attempts to carry out numerical simulations of 3D, fixed-wing flowfields using a high-order CFD algorithm with parallel processing. High-order algorithms are less dissipative and therefore more appropriate for calculating tip-vortex evolutions both in time and space. It is useful to assess the capabilities of a numerical algorithm against predicting fixed-wing tip-vortex flowfields (e.g. Ref. [7]) before it could be extended to more complicated cases, such as of a rotating blade [8].

For this purpose, a spatially and temporally fourth-order accurate algorithm that had been originally written for predicting turbofan inlet noise was modified to handle 3D wing geometries. Capabilities for parallel processing in distributed computing environments were also included in this code. This algorithm and the parallel processing approach are briefly described below. Then some preliminary results are presented, which is followed by some conclusions.

*Associate Professor, Phone:++90-312-210 4275, Fax:++90-312-210 1272, E-mail: yusuf@ae.metu.edu.tr

2 Numerical Algorithm

In this section, the features of the algorithm that pertain to 3D wing aerodynamics problems are described briefly. The ducted fan noise code [9, 10] was first modified to handle 3D wing geometries using the structured H-H mesh topology with appropriate boundary conditions implementations. The time-dependent Euler equations are solved in a body conforming curvilinear coordinate system. A fourth-order accurate central difference scheme is employed for spatial discretization and a four-stage Runge-Kutta time discretization is used to integrate the semi-discrete fluid equations in time. The scheme is augmented with artificial dissipation to suppress high-frequency spurious oscillations. Jameson type adaptive dissipation [11] is used for this purpose. At the artificial boundaries of the domain (i.e. far-field) characteristic based boundary conditions are used. For rapid steady-state calculations a multigrid convergence acceleration technique is employed with a 3-stage sawtooth cycle [12].

3 Domain Decomposition Strategy

A domain decomposition algorithm automatically decomposes the lower and upper blocks of the H-H mesh into as many subdomains as desired. This decomposition is performed in an (i, j, k)-ordered fashion with NT_i subdomains in the i-direction, NT_j subdomains in the j-direction, and NT_k subdomains in the k-direction, respectively, totalling to a number of $NT = NT_i * NT_j * NT_k$ subdomains. This 3D decomposition approach is preferred because it minimizes the ratio of the number of surface cells to the number of volume cells, reducing the communication overhead for the subdomain boundary data compared to the volume work (floating point operations). The computational grid and work for each subdomain is then assigned to its respective processor according to the following scheme:

```
do $taskid = 0, NT - 1$
    $task_k = taskid/(NT_i * NT_j) + 1$
    $task_j = (taskid - (task_k - 1) * NT_i * NT_j)/NT_i + 1$
    $task_i = taskid - (task_j - 1) * NT_i - (task_k - 1) * NT_i * NT_j + 1$
    $taskid_{ijk}(task_i, task_j, task_k) = taskid$
enddo
```

where *taskid* indicates the processor identification number for the processor which is responsible for the $(task_i, task_j, task_k)$ subdomain work.

The computational mesh is divided equally so that a load balance is achieved to prevent some of the processors from idling while the others are still working to finish their part. If the computational mesh is not equally divisible, then the best effort for this is made and the remaining grid points are included in the subdomains in which the least boundary work exists. This way an indirect load balancing is tried to be achieved.

4 Data Communication

Since the fourth-order spatial discretization scheme requires a five-point stencil at each grid point to evaluate a spatial derivative, information is needed from two adjacent cells to a subdomain interface. This is realized by use of two ghost cells in each direction at the boundaries of a subdomain. Data are then communicated between two neighboring subdomains launching library calls from the MPI standard [13]. Specifically blocking send and receive calls are used to exchange data between two processors.

The code has been written in Fortran 90 which makes it convenient to send and receive multi-dimensional array segments in this process and therefore, there is no need to pack data into 1D arrays unlike some of the other communication libraries (e.g. Parallel Virtual Machine, PVM) require.

A master processor is chosen initially and all input/output (IO) is done over this processor. The flow and case control parameters are read in by this processor and this information is broadcast to all processors.

The same processor reads the grid data (and the solution from a previous run if restart) and broadcasts each processor's share according to the decomposition information. Similarly information on the residual, its maximum value and location is sent from each processor to the master and the master compares these and calculates the global L_2 norm of the residual and prints all this information as the solution progresses.

5 Results

This code was run on two different parallel computing platforms first, namely the SGI Power Challenge and IMB SP2, at the Pennsylvania State University. Later with some further modifications it was run successfully on a cluster of Pentium II processors (350 MHz, 128 MB of Ram on each) housed in the Department of Aeronautical Engineering at the Middle East Technical University. These Pentiums run a Linux operating system and are connected with a 10 MBit/s network (ethernet). We employ the LAM (Local Area Multicomputer [14]) implementation of the MPI communication standard for parallel processing on this system. The LAM provides an MPI programming environment for heterogenous computers on a network in general.

Figure 1 illustrates the lower block of a 3D H-H mesh around a rectangular wing made up of NACA 0012 sections. This mesh was decomposed into a total of 8 subdomains, 2 in each direction. Care is taken in general in the decomposition of the computational mesh for load balancing. Figure 2 shows the resultant pressure contours for a flow speed of $M_\infty = 0.117$ and angle of attack of $\alpha = 5^o$. It is clear that the pressure contours are very smooth in the domain indicating that the data was communicated correctly among the processors, which is extremely crucial for accurate simulations. These results were obtained running the code on an 8-processor SGI Power Challenge. The identical results were obtained on IBM SP2.

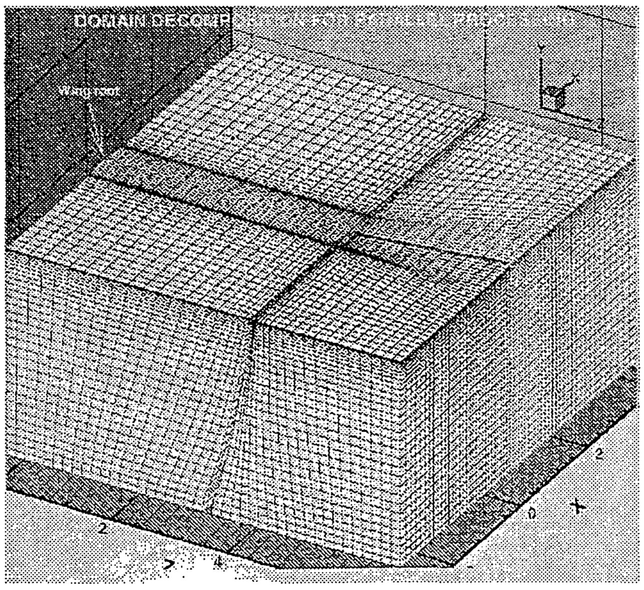


Figure 1: Lower block of the H-H mesh decomposed into 4 subdomains.

As mentioned above this code was also run on a cluster of Pentium II's. Figure 3 indicates the convergence speed-up via parallel processing on these Pentiums for attaining a 2D flowfield around the NACA 0012 airfoil at $M_\infty = 0.5$ and $\alpha = 0^o$ conditions. It is evident from the figure that doubling the number of processors has resulted in a reduction the CPU time by a factor of almost 2, indicating a linear speed-up.

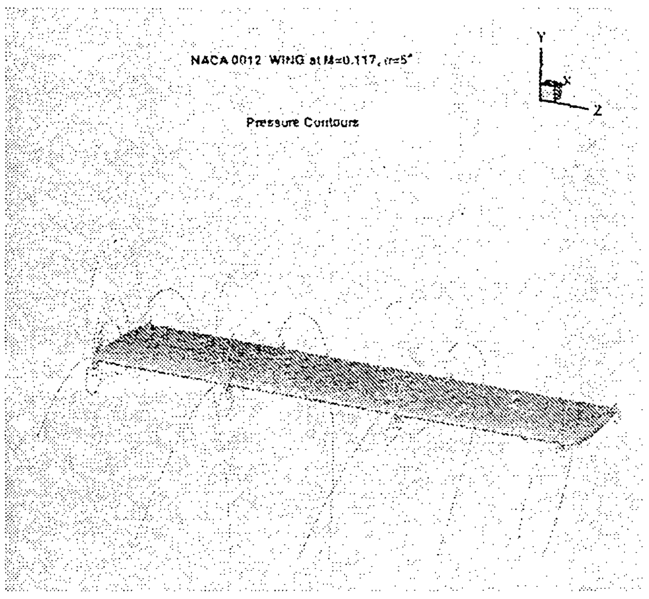


Figure 2: Pressure contours around the NACA 0012 wing at $M_\infty = 0.117$ and $\alpha = 5^o$.

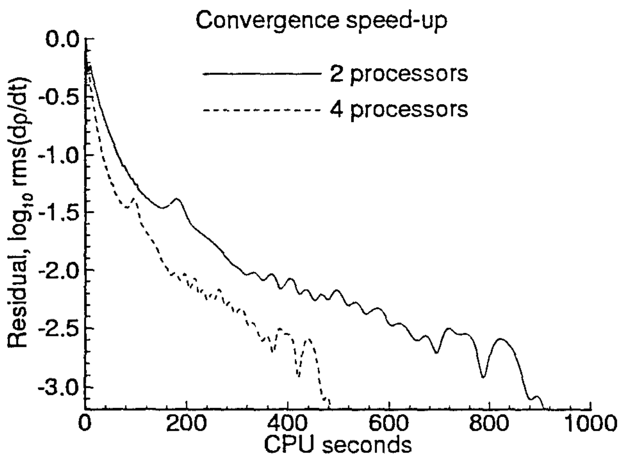


Figure 3: Convergence speed-up on Pentium II processors.

6 Concluding Remarks

A spatially and temporally fourth order CAA code has been modified for parallel computations of 3D flowfields of fixed wings. A domain decomposition approach is employed for distributed computing using the MPI standard. This code has been tested on various platforms, including a cluster of Pentium II processors. The preliminary results indicate that the parallel code is able to use Pentiums effectively.

References

[1] Hwang, K. *Advanced Computer Architecture: Parallelism, Scalability, Programmability.* McGraw-Hill, Inc., 1993.

[2] *Proceedings of Supercomputing'97: High Performance Networking and Computing*, San Jose, CA, USA, November 15-21, 1997.

[3] Cost-effective computing array home page. http://www.cocoa.psu.edu.

[4] Thompson, T. L., Komerath, N. M., and Gray, R. B. Visualization and measurement of the tip vortex core of a rotor blade in hover. *Journal of Aircraft*, **25(12)**, pp. 1113–1121, December 1988.

[5] Sheffer, S. G., Alonso, J. J., Martinelli, L., and Jameson, A. Time-accurate simulation of helicopter rotor flows including aeroelastic effects. AIAA Paper 97-0399, AIAA 35th Aerospace Sciences Meeting and Exhibit, Reno, NV, 6–10 January 1997.

[6] Baeder, J. D., Gallman, J. M., and Yu, Y. H. A computational study of the aeroacoustics of rotors in hover. *Journal of the Americal Helicopter Society*, **42(1)**, pp. 39–53, January 1997.

[7] Wake, B., and Choi, D. Investigation of high-order upwinded differencing for vortex convection. AIAA Paper 95-1719, Proceedings of the AIAA 12th Computational Fluid Dynamics Conference, San Diego, CA, June 1996.

[8] Srinivasan, G. R., Baeder, J. D., Obayashi, S., and McCroskey, W. J. Flowfield of a lifting rotor in hover: A navier-stokes simulation. *AIAA Journal*, **30(10)**, pp. 2371–2378, October 1992.

[9] Özyörük, Y., and Long, L. N. A new efficient algorithm for computational aeroacoustics on parallel processors. *Journal of Computational Physics*, **125(1)**, pp. 135–149, April 1996.

[10] Özyörük, Y., and Long, L. N. Computation of sound radiating from engine inlets. *AIAA Journal*, **34(5)**, pp. 894–901, May 1996.

[11] Swanson, R. C., and Turkel, E. Artificial dissipation and central difference schemes for the Euler and Navier-Stokes equations. AIAA Paper 87-1107, 1987.

[12] Özyörük, Y., and Long, L. N. Multigrid acceleration of a high-resolution computational aeroacoustics scheme. *AIAA Journal*, **35(3)**, pp. 428–433, March 1997.

[13] Snir, M., Otto, S. W., Huss-Lederman, S., Walker, D. W., and Dongarra, J. *MPI, The Complete Reference*. MIT Press, 1996.

[14] Lam home page, http://www.lsc.nd.edu/lam/.

PARALLEL COMPUTATION OF MULTI-PASSAGE CASCADE FLOWS WITH OVERSET GRIDS

Ismail H.Tuncer*
Middle East Technical University
06531 Ankara, TURKEY

INTRODUCTION

In Computational Fluid Dynamics applications, multiple workstations on a local network may be combined into a parallel computer, and this virtual parallel computer may be used to solve a single flow problem. Such a parallel processing environment is an attractive alternative to a supercomputer or to a dedicated computer with multi-processors. It also improves the productivity of otherwise idle workstations. Furthermore, multiple-grid systems used in CFD provide an easily exploited parallel solution algorithm by solving the equations on each grid independently on a separate workstation, with boundary information exchanged on inter-grid interface surfaces.

The current trends in gas turbine design towards higher aerodynamic blade loading and slender blades demand an accurate and detailed study of aeroelastic behavior of compressor and turbine blades. Considerable experimental and computational efforts are currently being made to predict unsteady cascade flows and blade flutter in turbomachinery. It is, therefore, of great interest to develop solution methods to compute dynamic response of blades with an acceptable level of accuracy and speed.

Unsteady cascade flows as the blades undergo in and out-of-phase vibrations are currently being computed by solving the Euler and Navier-Stokes equations. In the computation of out-of-phase cascade flows, the periodic boundary conditions are implemented either by discretizing the multi-passage domain based on the Inter-Blade Phase Angle (IBPA) or by imposing temporal periodicity in a single passage domain. The latter method is known as *direct store*[1]. Although the *direct store* method reduces the computational domain to a single blade passage, it in fact linearizes the periodic boundary condition in time. In addition, a periodic convergence may take a large number of periods to be computed on a single passage domain, and a large storage may be required to store the periodic boundary information in time[2].

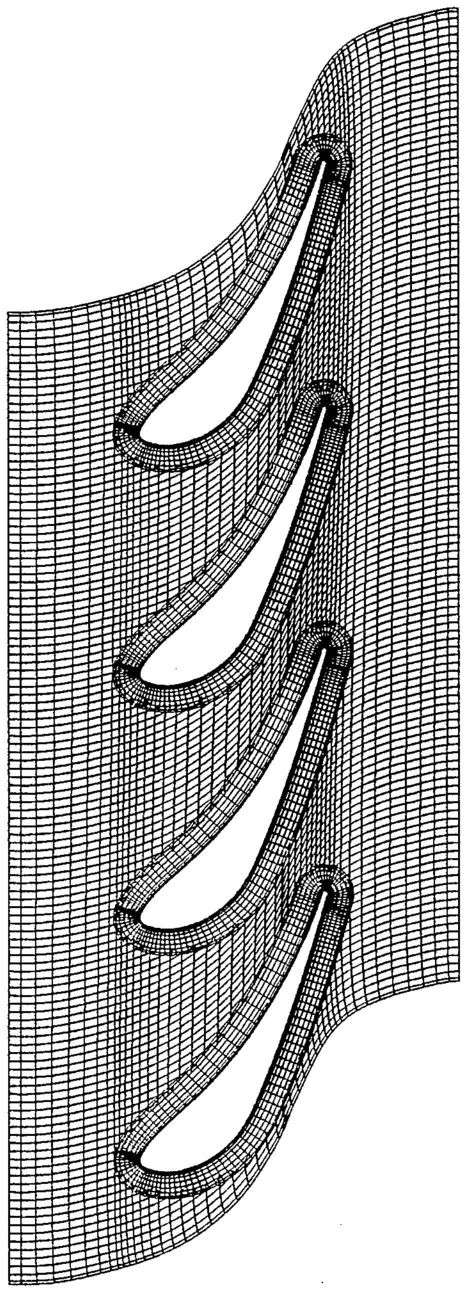


Fig. 1 Overset grid system for a four-passage compressor cascade. (STC No. 4 configuration)

*Assoc. Prof., Dept. of Aeronautical Engineering, Tel: (90-312) 210-4289, Fax: (90-312) 210-1272, E-mail: tuncer@ae.metu.edu.tr

SUMMARY AND RESULTS

In this study, a Navier-Stokes solver is modified to compute multi-passage cascade flows on overset grids in a parallel processing environment. The multi-passage flow domain is discretized with a rectangular background grid and viscous grids around each blade which are overset onto the background grid. The background grid covers the whole multi-passage domain. Figure 1 shows a discretized four-passage flow domain over the turbine cascade, STC No. 4, which has been investigated experimentally and extensive data are available[3]. The blade grids are free to move with respect to the background grid, and a periodic plunging motion of blades with a phase shift is imposed independently. The main advantage of this approach lies in its versatility to resolve the multi-passage computational domain with high quality, mostly orthogonal subgrids, and in the application of simple periodic boundary conditions. Furthermore, it imposes almost no restriction on the plunging motion of the blades. The implicit solution on the background grid also improves the time accuracy of the solution.

We have already computed unsteady flows over the multi-passage compressor cascade, *Fourth Standard Configuration, STC No. 4* [7], as the blades undergo periodic plunge oscillations. The plunge oscillations are defined as

$$h_n(t) = A \sin[2\pi f t + (n-1)\,\phi]$$

where A is the amplitude of the plunging motion, and ϕ is the phase shift between blades. For a four-passage cascade flow, $\phi = \pm 90$. n is the blade index, where $n = 1$ refers to the bottom blade. The blade grids are of 266×25 size, and the background grid is of 91×226 size.

Figure 2 shows the steady state Mach number contours computed at $\alpha_{inlet} = 50.0°$, $P_0 = 219600\ Pa$, $T_0 = 329.68° K$, and $P_{exit} = 91400\ Pa$. As shown, the overset grid solution is continuous across the subgrids. The shock is also resolved across the inter-grid boundaries. The computed steady pressure distribution on blade 1 agrees well with the experimental data and SAFE1[8] solution. SAFE1 Navier-Stokes solver employs a single passage domain on a single O-type grid with the *direct store* interblade boundary conditions. In the coarse grid solution, the size of the blade grids is reduced to 134×25.

Next, unsteady flow at $\phi = -90°$, $f = 150\text{Hz}$, $A = 0.0030c$ is computed for more than three periods of the periodic plunging motion. Figure 4 shows the first harmonic, and the phase shift of the unsteady surface pressure distribution on blade 1. The present solution and the SAFE1 predictions are again in good agreement.

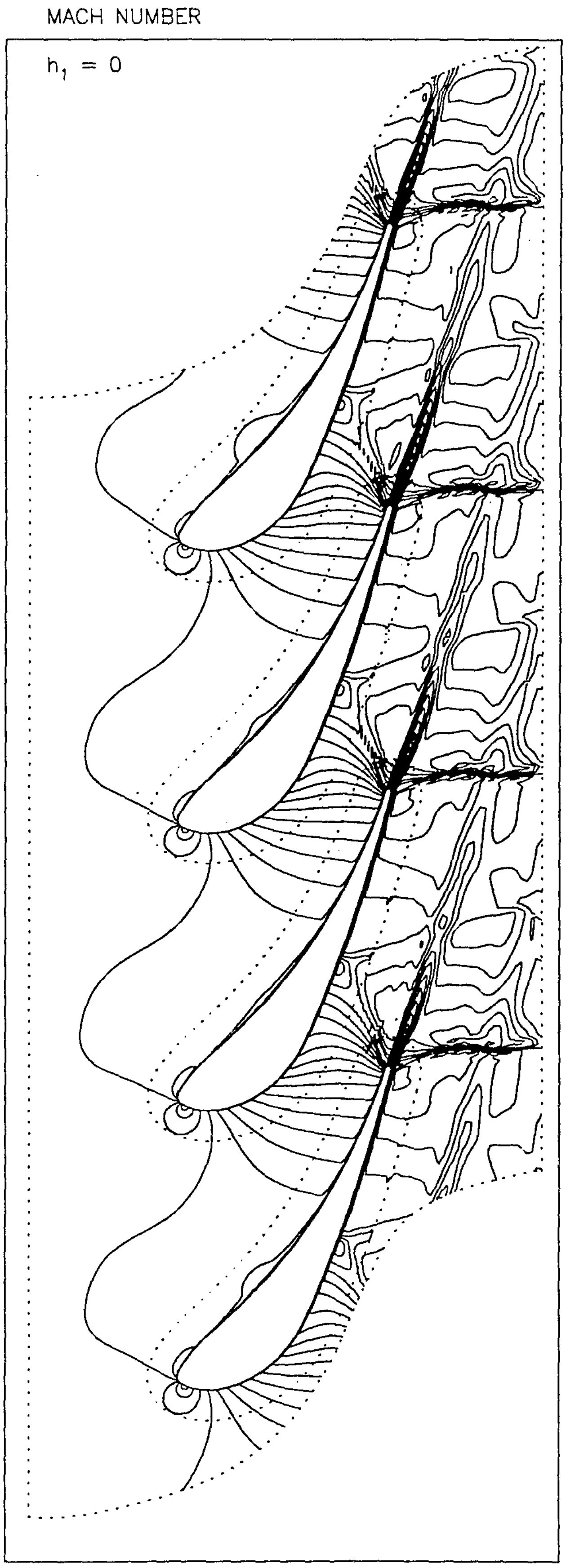


Fig. 2 Steady state solution

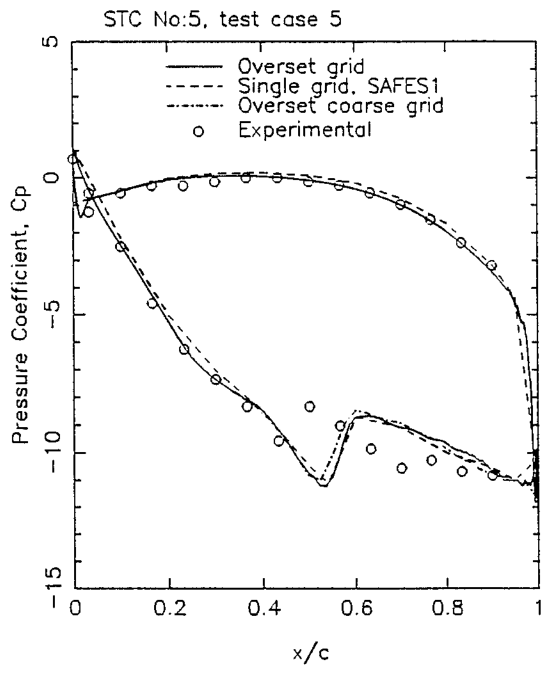


Fig. 3 Steady surface pressure distribution on a blade

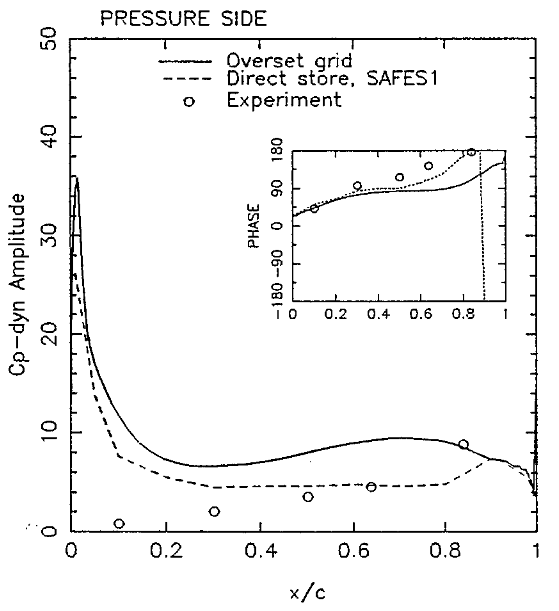


Fig. 4 First harmonic and phase shift of the unsteady pressure distribution on blade 1

As a continuation of this study, the present solver is parallelized using the PVM library routines in a master-worker paradigm. In addition to distributing the solution on each blade grid to a worker processor, the master processor also applies the inter-grid boundary conditions. The parallel processing environment consists of Pentium-II workstations, which are connected over a local Ethernet network using TCP/IP protocol, and run Linux operating system.

In the final paper, multi-passage cascade flows will be computed in parallel and the computed solutions will be compared with the serial solutions already obtained on a single processor. The efficiency and scalability of the parallel computations, load balancing between processors, latency in message and data passing will be investigated in detail. Retarded application of the inter-grid boundary conditions, and its sensitivity to the retardation steps will also be studied.

REFERENCES

1. Erdos, J.I, Alzner, E., and McNally, W., 1981, *Numerical Solution of Periodic Transonic Flow Through a Fan Stage*, AIAA-Journal, Vol. 15.
2. Abhari, R.S., Giles, M., 1995, *A Navier-Stokes Analysis of Airfoils in Oscillating Transonic Cascades for the Prediction of Aerodynamic Damping*, ASME-Paper 95-GT-182.
3. Bölcs, A., and Fransson, T.H. (eds.), 1986, *Aeroelasticity in Turbomachines - Comparison of Theoretical and Experimental Cascade Results*, Communication de Laboratoire du Thermique Appliquée et de Turbomachines, No. 13, EPFL Lausanne, Switzerland.
4. Tuncer, I.H, June 1996, *A 2-D Navier-Stokes Solution Method with Overset Grids,* ASME paper 96-GT-400.
5. Tuncer, I.H and Sanz, W., 1997, *Computation of Multi-Passage Cascade Flows with Overset and Deforming Grids*, ASME Paper 97-GT-021.
6. Tuncer, I.H, 1997, *A 2-D Unsteady Navier-Stokes Solution Method with Overset Moving Grids,* AIAA Journal, Vol. 35, No. 3, March 1997, pp. 471-476.
7. Tuncer, I.H., Weber, S. and Sanz, W., June 1998 , *Investigation of Periodic Boundary Conditions in Multi-Passage Cascade Flows Using Overset Grids,* ASME Paper 98-GT-11.
8. Weber, S., Benetschik, H., Peitsch, D. and Gallus, H.E., 1997 *A Numerical Approach to Unstalled and Stalled Flutter Phenomena in Turbomachinery Cascades*, ASME Paper 97-GT-102.

What about load imbalance?

R.L. Verweij [1] , A. Twerda, T.W.J. Peeters
Department of Applied Physics, Delft University of Technology, Lorentzweg 1, 2628 CJ Delft, The Netherlands.

1 Introduction

In parallelisation of CFD codes one usually assumes that if the number of operations per PE is the same for all PE's then the computing time per PE is also similar. However, the paper reports on our observation that load imbalance is only partly caused by the amount of work per PE. We show that other effects contribute to the load imbalance and should be taken into account.

The case under consideration is the numerical simulation of turbulent reacting flows in industrial glass-melting furnaces. These simulations require advanced models of turbulence, combustion and radiation in conjunction with sufficiently fine numerical grids to resolve important small scale interactions. Currently, many of the numerical simulation have to be performed with relatively simple models due to the fact that they are very CPU- and memory-demanding, hampering an accurate prediction. Parallel processing is regarded nowadays as the promising route by which to achieve desired accuracy with acceptable turn-around time.

2 Description of the physics and numerics

The simulation of turbulent combustion involves the solving of the 3D incompressible (variable density) stationary conservation laws for mass, momentum, energy and species, together with the hydrodynamic and caloric equations of state. Models for turbulence, combustion and radiation are applied. A detailed description of turbulent combustion modelling for furnaces can be found in [1, 4].

The modelled equations are discretised using the Finite Volume Method on a colocated, Cartesian grid [3]. The linearised systems are solved using the SIP-algorithm. Full multi-grid [3] is used to improve the convergence behaviour of the multi-block code.

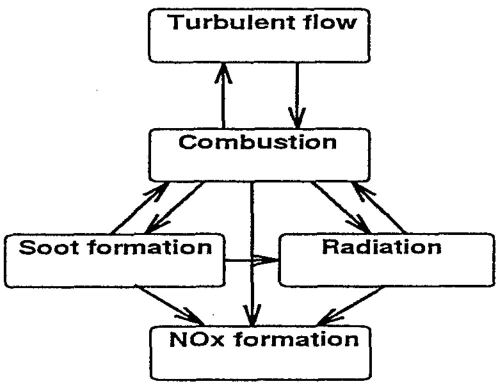


Figure 1: Complexity of the simulations

For the domain decomposition the grid-embedding technique [2] is used. This means that one global (coarse) grid is defined, and the domains are defined as (refined) subdomains of this coarse grid.

3 Description of parallel strategy and brief description of the architecture, parallel tools and environments used.

Static load balancing is performed; every domain contains (approximately) the same amount of grid-points and the amount of work per grid point was constant for all points. Exactly one domain is computed per processor, so that the sequential single-block program could easily be used for multi-block computations. The SPMD (Single Program, Multiple data) programming model was used.

To maintain a portable code a flexible change from one message passing library to another was considered compulsory. This was achieved by using generic subroutines for all communication and parallel statements. Five subroutines were written which contain all communication-library-specific statements: *parstart* and *parstop*, to start and stop a parallel program respectively, and check whether machine settings are consistent ; *parsend* and *parrecv* for point-to-point communication, and *parglobal* for all global communication (global sums, global maxima, barriers, broadcasts and gathers are used currently).

[1] Corresponding author. `R.Verweij@tn.tudelft.nl`

MPI was used as a communication protocol. However, during the development, PVM and MPI were adopted on clustered workstations and the IBM SP2 and the (machine specific) message passing tool SHMEM on the CRAY T3E. All computations performed in this paper were performed on a CRAY T3E AC80-128, using the 600 MFL DecAlpha, available at the Delft University.

4 Results

Before discussing the load imbalance in these domain-decomposition based codes figure 2 shows the speedup of 2D and a 3D fixed-size problem on several gridsizes. The code scales better for finer grids, as

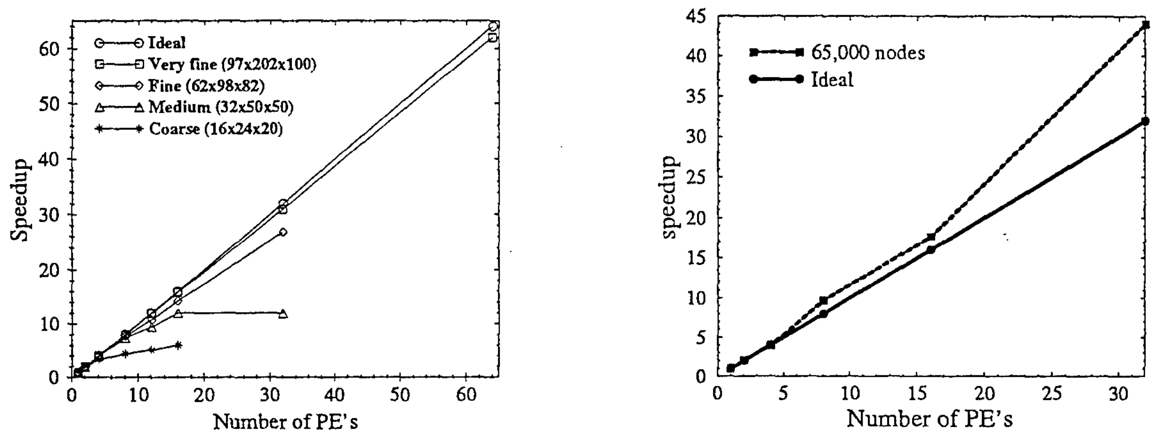


Figure 2: Measured speedup for several gridsizes. **left**: 3D **right**: 2D

is expected. The super-linear speedup of the 2D version can be explained by assuming better cache-use, because of the two dimensionality of the arrays.

At first glance the speedup seems completely predictable. The single PE performance for the code was approximately 100 MFLOPS, although the cache-use was quite modest (less than 1 operation per load). A detailed analysis on subroutine level has been given in table 1 for the very fine grid simulation on 64 blocks (decomposed in $4 \times 4 \times 4$ domains). Two things are noteworhty. First the amount of time

Table 1: Percentage of total CPU time spent in different code parts for a particular block in the very fine grid turbulent combustion simulation

Function	Tot. time	% tot	# calls
MATRIX COEFFICIENTS	692.907	23.2	6000
MATRIX SOLVER	607.905	19.9	6750
BARRIER WAITING TIME	587.952	19.3	64218
SOURCE TERMS	561.929	18.4	6000
RADIATION MODEL	147.714	4.8	155
BND. COND. EXTERN	60.837	2.0	752
BND. COND. INTERN	34.465	1.1	9754
PT2PT COMMUNICATIONS	18.740	0.6	117024
GLOBAL COMMUNICATIONS	3.671	0.1	2853

spent in communication is completely negligible. In total less than 1% of the total time is spent in actual communication. This involves all hardware issues and message passing library depending stuff. It can hence be concluded that on a CRAY T3E with very fast network and large bandwidth it is completely superfluous to search for the fastest protocol available. This was indeed confirmed by using the SHMEM message passing subroutines, which did not yield any improvement over the MPI calls. But second, the total barrier waiting time (the percentage of idle time for this block) is almost 20%.

Analysis showed that this large percentage stems from the fact that in our study the complex geometry is modelled with the porosity method, which marks the cells that lie outside the fluid domain. In figure

3 we plotted the relative idle time of all PE's n the 64-block run together with the number of cells in that block that lie outside the furnace ('closed' cells). The blocks which contain many closed cell are obviously quicker than other blocks, and hence this causes significant load imbalance. The best thing to do is rearrange the blocks such that every block has the same amount of 'not-closed' cells, but this would involve quite complicated block decomposition and has therefor not yet been tried.

Apart from this there is still an average load imbalance of approximately 7%. This has to be contributed to the fact that even if blocks have the same amount of points, still the total amount of work differs.

A first reason for this is that some blocks have more neighbours than others, which implies that some blocks will updating more boundaries than others. To test this assumption 200 iterations were done on a coarse (16 × 24 × 20) grid. This grid was split into four blocks in two different configurations, 2 × 2 × 1 and 4 × 1 × 1. In the first decomposition all blocks have exactly two neighbours (the rectangular decomposition), in the second one the middle two blocks have two neighbours (the sliced decomposition), and the outer blocks one. Every block contains the same amount of points. Both decompositions lead to exactly the same converged solution, but the timings are quite different, as shown in table 2. The rectangular decomposition yields much better timing results than the sliced decomposition. This confirms the assumption that in the sliced decomposition, the two middle blocks have much more communications to do, creating an huge load-imbalance in that part of the code. This effect becomes smaller if the number of points per blocks becomes bigger. Note that the total communication time is approximately equal in both cases, and is small compared to the entire program time. The smaller parallel working time in the rectangular decomposition is because of better cache use.

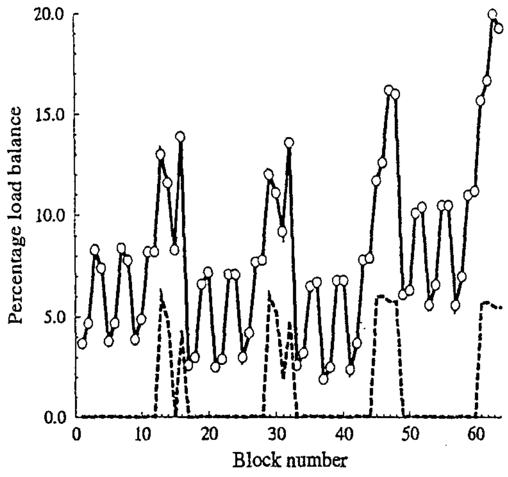


Figure 3: Idle time for every processor in the very fine grid simulation (solid line). Number of closed cells (dotted line).

Table 2: Load imbalance for complete problem

Time needed (s)	sliced decomp.		rectangular decomp.	
Total program time	2150		1700	
Parallel working time	1140	(53%)	1241	(73%)
Total barrier waiting time	606	(28%)	106	(6%)
Total communication time	85	(4%)	77	(5%)

Still the 6% load imbalance is found, even for the 'perfect' decomposition. This remaining 6% should be contributed to other effects, like different cahce-behaviour on different PE's. Of course these effects are small, but if thousands of iterations are performed, which is not uncommon in turbulent combustion simulations, this effect might be severe.

5 Conclusions indicating results to be presented in the full paper.

Parallelisation by domain decomposition is a straightforward and efficient way to parallelise combustion codes. However, load imbalance is a underestimated problem in most CFD applications as it arises in several ways. The speedup itself is not a good indicator for load imbalance, as load imbalance depends on the fastest PE, whereas speedup is determined by the slowest PE's. We hope to be able to analyse the remaining reasons for load imbalance in June.

References

[1] G.P. Boerstoel. *Modelling of gas-fired furnaces.* PhD thesis, TU Delft, may 1997.

[2] P. Coelho, J.C.F. Pereira, and M.G. Carvalho. Calculation of laminar recirculating flows using a local non-staggered grid refinement system. *Int. J. Num. Meth. Fl.*, 12:535–557, 1991.

[3] J.H. Ferziger and M Perić. *Computational Methods for Fluid Dynamics.* Springer-Verlag, Berlin, 1996.

[4] T.W.J. Peeters. *Numerical modeling of turbulent natural-gas diffusion flames.* PhD thesis, TU Delft, September 1995.

[5] M. Perić, M. Schäfer, and E. Schreck. Computation of fluid flow with a parallel multigrid solver. In *Parallel Computational Fluid Dynamics 1991*, pages 297–312. Elsevier Science Publishers. B.V., 1992.

ABSTRACT

Parallel Adaptive Flow Solution on Unstructured Mesh

Erdal YILMAZ, M.Şerif Kavsaoğlu
Middle East Technical University
Aeronautical Engineering Department
Ankara-TURKIYE

Introduction

Computing power of present days are challenging with not only massively parallel machine architecture but also clusters of heterogeneous computer networks. In addition to machine power, computing algorithms bring efficient use of these resources.

In this study, implementation of an Euler flow solver has been done on a parallel machine, IBM-SP2. Validation of the flow solver has been presented in previous works [1,2]. It has been shown that very complex flow fields can be covered. This paper gives an idea about how solution adaptive re-meshing can be achieved on parallel environment. The flow field is a high pressure jet flow coupled with high speed external freestream. In such field there are strong gradient regions where adaptation efficiency can easily be monitored. Solution is started with a mesh having smooth distribution of spacing. Then, three adaptation sequences are applied. After final adaptation, solver runs for fully converged solution. Computation efficiency is compared with that on a single machine.

Flow Solution

Governing flow equations are compressible Euler equations. Finite volume technique is used to discretize flow equations written for a control volume. Unstructured triangular grids are used to define flow domain in space. Edge based connectivity is used to store grids. Artificial dissipation terms are used to damp oscillations due to numerical solution technique and to capture shock waves. Residual averaging technique is employed to accelerate numerical solution. Direct multi-step time stepping is used to advance solution in time domain. It is similar to Runge-Kutta time stepping algorithm. Euler equations in conservative form are given as

$$\frac{\partial}{\partial t}\int_{\Omega} U d\Omega + \oint_{S}(F dy - G dx) + \alpha \int_{\Omega} H d\Omega = 0$$

where U is unknown vector for density, velocity and energy, F and G are convective fluxes, H is source term for axisymmetric flow and α is a flag to turn on source term.

Adaptation

The grid points are usually desired to be dense in regions where there will be large gradients, such as near shock waves and leading edge of an airfoil, and relatively sparse in regions where the solution is expected to vary slowly. This leads to the adaptive grid generation procedure in which the grid generator and flow solver interacts. By using flow solution sensor values are obtained for all grid point through which new spacing of the previous grids are found. In this paper, gradients of mach number is used to find sensor values After that using new grid spacing values, grid generator is activated to find new grid. Three adaptation sequences are used with just two hundreds time stepping in flow solver. Then using final adapted mesh almost three thousands of time stepping are achieved.

Parallel Solution

In unstructured mesh, there are several partitioning techniques which give minimal communication overhead. In this work efficient partitioning is not point of interest since parallel adaptation is aimed. Therefore a very simple method of partitioning is used. At interface of each mesh block overlapping of boundary is used. By this way no averaging is performed, instead the values of unknown vector is imported from its neighbors. Test case is a high pressure jet with supersonic external flow. The detail of this case study can be found in ref. [2,3]. Mesh of the flow field is given in figure-1. Initial mesh with four partition is presented in figure-2. It has 3217 number of points with almost smooth distribution from wall and symmetry axis to external boundary. Figure-3 and 4 give adapted mesh and mach contour in final adaptation.

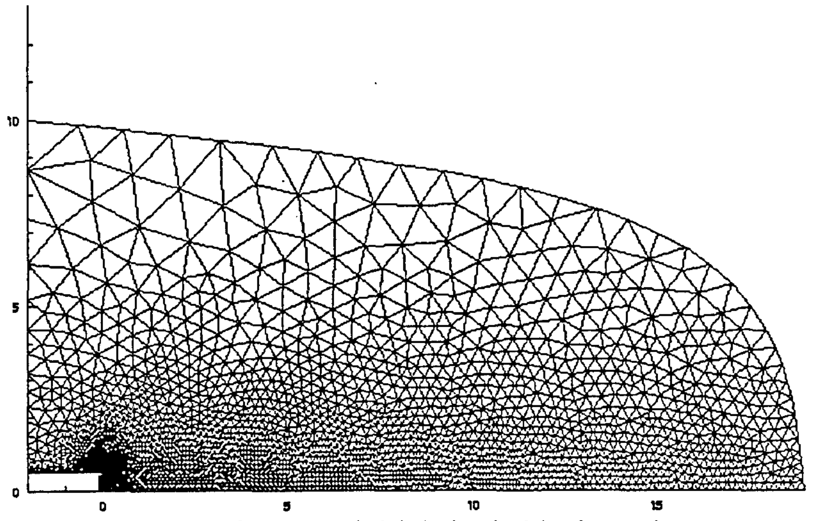


Figure-1 : initial single block mesh

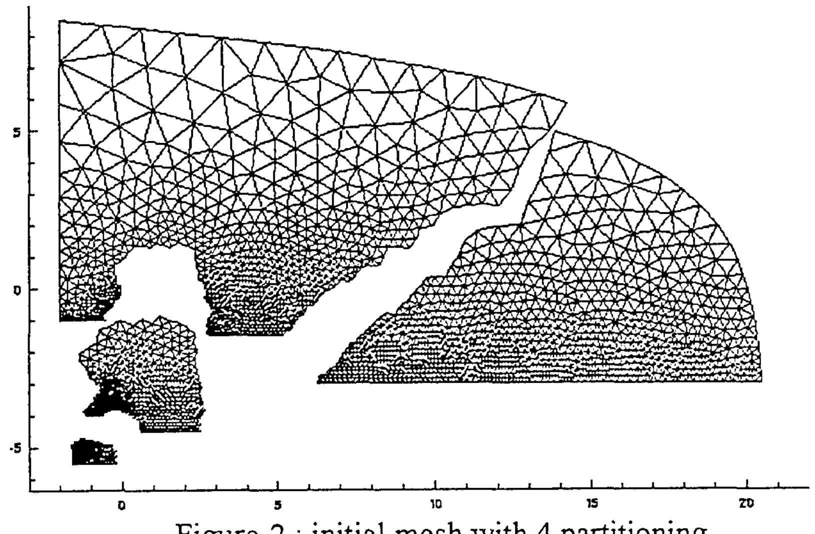


Figure-2 : initial mesh with 4 partitioning

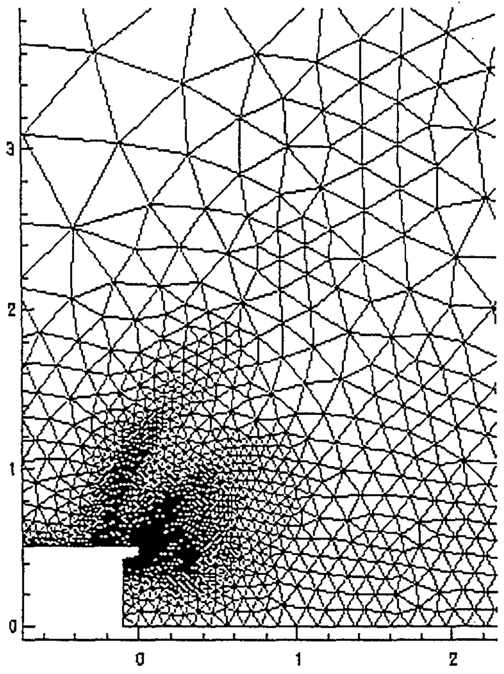


Figure 3 : adapted mesh around jet exit

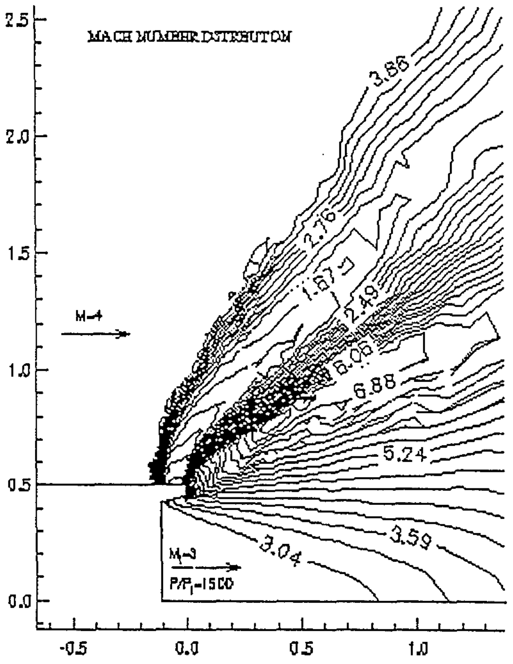


Figure 4 : solution around exit after adaptation

References

[1] YILMAZ, E., " Two Dimensional and Axisymmetric Euler Solution on Unstructured Grids", M.S. Thesis, Middle East Technical University, Ankara-Türkiye, 1994

[2] YILMAZ, E. and KAVSAOĞLU, M.Ş., "Numerical Solution of Axisymmetric Jet Plume Coupled with External Freestream",ICAS-96-7.10.4, Sept.8-13,1996, Sorrento-Napoli-ITALY.

[3] YILMAZ, E. and KAVSAOĞLU, M.Ş., "Euler Solution of Axisymmetric Jets in Supersonic Flow", Journal of Spacecraft and Rocket, Vol. 35, No:1, 1998.

Abstract for the Parallel CFD Workshop, 16-18 June 1999, Istanbul-Turkey

USAGE OF THE NEW OPENMP STANDARD IN PARALLELIZATION

Igor Zacharov
Senior HPC Technical Specialist
Silicon Graphics European Headquarters
Chemin des Avouillons 30, 1196 Gland, SWITZERLAND

OpenMP is the industry standard for parallel processing on Shared Memory Machines. This report contrasts OpenMP to the previous attempts to agree on a common parallelization paradigms, including message passing tools. A practical usage of the OpenMP is discussed for recent parallelization projects.