Performance of a Euler Solver Using a Distributed System

Renato S. Silva and Regina C. Almeida * [a]

[a]Laboratório Nacional de Computação Científica
Rua Lauro Müller, 455, 22290-160,
Rio de Janeiro, Brazil

Supercomputers nowadays can be used to solve large-scale problems that come from simulation of industrial or research problems. However, those machines are usually inacessible to most industries and university laboratories around the world. In this work we present an Euler solver to be used in a collection of workstations under PVM that can overcome the use of direct methods on a single processor.

1. Introduction

In the last years, with the advent of successful discretization methods and powerful computer technologies, computer simulation has been seen as a valuable approach to be combined with experimental methods to solve industrial and scientific problems. In this work we are interested in the numerical simulation of compressible Euler flows for which we have to deal with large, sparse and non-symmetric systems. For these types of problems the computational cost is a major concern and so is the accuracy. Obviously, the accuracy is related with the appropriate choice of the numerical method to solve the Euler equations.

Ideally, the numerical method to solve the compressible Euler equations should have good convergence properties and should be stable in the presence of shocks: in those regions of the fluid flow, the variables in the system vary strongly, making the computation very challenging. In that case, it is well known that the approximate solution obtained by using the standard Galerkin finite element method is completely spoiled by spurious oscillations that are spread all over the computational domain. In order to avoid those oscillations, we shall use a very stable Petrov-Galerkin method called CAU Method (Consistent Approximate Upwind Method)[4], that consistently adds to the Galerkin weighting functions two terms. The first one acts over the generalized streamline direction which comes from the SUPG Method (Streamline Petrov-Galerkin Method)[1] and the second one provides the control over the derivatives in the direction of the generalized approximate gradient. This method is also very accurate since the latter perturbation vanishes in the regions where the solution is regular. However, there still be the necessity of using an efficient solver to reduce the computational costs when large problems are concerned.

*Visiting Scholars from TICAM - Texas Institute of Computational and Applied Mathematics, The University of Texas at Austin. During the course of this work the authors were supported by the Brazilian Government fellowship CNPq proc. 201387/93-0 and 200289/93-4, respectively.

It is well known that iterative methods perform better than direct methods for large problems, but to determine the best iterative algorithm to solve non-symmetric systems is still an open issue. However, many conjugate gradient like methods have been widely used with relative success. Included in this class, the GMRES method introduced by Saad and Schultz seems to be a good choice because it seldom breaks down and presents monotonic convergence. But the method possesses a drawback because, in its original version, the orthogonalization process dictates an increasing amount of computational work with increasing iterations. This drawback can be avoided using a restarted GMRES version (GMRES(k)) since it consumes less memory than the original one while keeping its good properties.

A good solver means not only a choice of the best method but a combination between the method and a fast implementation. Recently considerable attention has been focused in heterogeneous distributed systems as a solution for high performance computations. By definition, a heterogeneous distributed system is a collection of different computers loosely connected by a Local Area Network (LAN) or/and a Wide Area Network (WAN). The popularity of these systems can be explained by the increasing performance and lower prices of high speed networks and general-purpose workstations. This means that is much cheaper to form and to maintain these systems than a massively parallel machine (MPM) and, in certain cases, their performances can be similar.

Unlikely MPMs, heterogeneous distributed systems can be formed with different types of processors and with different network types. Besides, they are composed by general-purpose enviroments, which means that it is possible to have different users running their jobs in the machines of the system. As a consequence, the load of each machine and the network traffic can vary at each instant and the choice of the communication pattern has to be done carefully.

Thus, in this work, we present a distributed implementation of the GMRES(k) with right preconditionig, for which the computational cost depends on the dimension of the Krylov Space (k) and on the interface of the domains and we show that this algorithm running on a collection of 16 IBM RISC 6000 workstations can get satisfactory efficiency with a relative small number of degrees of freedom.

An outline of this paper is as follows. In section 2 the CAU method for the compressible Euler equations employing entropy variables is presented. In section 3 the GMRES method and the Additive Scharwrz method used as preconditioner are discussed. Section 4 describes our distributed implementation of methods shown on previous sections. Numerical results are presented in section 5 and the conclusions of this work are drawn in section 6.

2. Simulation of Compressible Inviscid Flows

In this work we are interested in the numerical simulation of the steady state solution of the two-dimensional compressible Euler equations. For such non-linear problems, we get the steady state solution from the limit solution of a time dependent problem. We shall use a stable Petrov-Galerkin formulation, the *CAU method*, in order to prevent instabilities when sharp gradients are presented in the solution [see [4,5] for details].

2.1. Approximate Solution - The CAU Method

We shall formulate the CAU method for the Euler equations by using the time-discontinuous Galerkin method as the basis of our formulation. To this end, consider partitions $0 = t_0 < t_1 ... < t_n < t_{n+1} < ...$ of $\Re^+$ and denote by $I_n = (t_n, t_{n+1})$ the $n^{\underline{th}}$ time interval. The space-time integration domain is the product $\Omega_n = \Omega \times I_n$, $\Omega \subset \Re^d = \Re^2$, with boundary $\overline{\Gamma} = \Gamma \text{ x } I_n$. Denote by Ω_n^e de e^{th} element in Ω_n, $e = 1, ...(N_e)_n$, where $(N_e)_n$ is the total number of elements in Ω_n. For $n = 0, 1, 2...$, let us first introduce the set of kinematically admissible functions

$$S_n^h = \left\{ V^h; V^h \in \left(C^0 \left(\Omega_n \right) \right)^m ; V^h \mid_{\Omega_n^e} \in \left(P^k \left(\Omega_n^e \right) \right)^m ; f^1 \left(V^h \right) \mid_{\overline{\Gamma}_n} = g\left(t \right) \right\}$$

and the space of admissible variations

$$\Im_n^h = \left\{ \widehat{V}^h; \widehat{V}^h \in \left(C^0 \left(\Omega_n \right) \right)^m ; \widehat{V}^h \mid_{\Omega_n^e} \in \left(P^k \left(\Omega_n^e \right) \right)^m ; f^2 \left(\widehat{V}^h \right) \mid_{\overline{\Gamma}_n} = 0 \right\},$$

where f^1 and f^2 are the nonlinear boundary condition transformation, g is a prescribed boundary condition and P^k is the space of polynomials of degree less or equal to k.

With these definitions, the variational formulation using the CAU method consists of: Find $V^h \in S_n^h$ such that for $n = 0, 1, 2...$

$$\begin{aligned}
&\int_{\Omega_n} \widehat{V}^h \cdot \left(A_0 V_{,t} + \widetilde{A} \cdot \nabla V \right) \, d\mathbf{x}dt \quad + \\
&\sum_{e=1}^{(N_e)_n} \int_{\Omega_n^e} (\, A_0 \widehat{V}^h + \widetilde{A} \cdot \nabla \widehat{V}^h \,) \cdot \tau_s^e \, \left(A_0 V_{,t} + \widetilde{A} \cdot \nabla V \right) d\mathbf{x}dt \quad + \\
&\sum_{e=1}^{(N_e)_n} \int_{\Omega_n^e} \left(\left(\widetilde{A} - \widetilde{\theta}^h \right) \cdot \nabla \widehat{V}^h \right) \cdot \tau_c^e \, \left(A_0 V^h + \widetilde{A} \cdot \nabla V^h \right) d\mathbf{x}dt \quad + \\
&\int_{\Omega} \widehat{V}^h(t_n^+) \cdot \, A_0 \, \left(V^h(t_n^+) - V^h(t_n^-) \right) dx = 0 \, , \qquad \forall \, \widehat{V}^h \in \Im_n^h \, .
\end{aligned} \tag{1}$$

$V = (V_j)_{j=1}^m$ is the entropy variables vector, m is the number of variables ($m = 4$ for the two-dimensional case); $A_0 = \frac{\partial U}{\partial V}$ is a $(m\text{x}m)$ symmetric and positive-definite matrix; $U^t = [\rho, \rho u_1, \rho u_2, \rho e]$ is the conservation variables vector; ρ is the density; u_i is the velocity in $i^{\underline{th}}$ direction, $i = 1, 2$; e is the total energy density. $\widetilde{A}_i = A_i A_0$ is symmetric, where $A_i = \frac{\partial F_i}{\partial U}$ is the Jacobian matrix, F_i is the Euler flux. Finally, $\nabla^t (\cdot) = \left(I_m \frac{\partial (\cdot)}{\partial x_1}, I_m \frac{\partial (\cdot)}{\partial x_2} \right)$ is the generalized gradient operator where I_m is the $(m\text{x}m)$ identity matrix and $\widetilde{\theta}^h$ is an auxiliary $(md\text{x}m)$ matrix introduced by the CAU Method. The terms in this equation can be identified in order as the standard Galerkin term, the SUPG term, the discontinuity-capturing term provided by the CAU method and the jump term by which the information is propagated from Ω_n to Ω_{n+1}. The SUPG term provides the control over the generalized streamline direction and the $(m\text{x}m)$ matrix of intrinsic time scale τ_s^e is defined as in [8].

Introducing the definition for $\left(\widetilde{A} - \widetilde{\theta}^h \right)$, the CAU term can also be written as

$$\sum_{e=1}^{(N_e)_n} \int_{\Omega_n^e} \max \{ \, 0 \quad , \quad h_c^e \, \frac{\left| \mathcal{L}^h \right|_{A_0^{-1}}}{\left| \nabla V^h \right|_{A_0}} - \frac{\nabla V^{h^t} A \; \tau_s^e \; \mathcal{L}^h}{\left| \nabla V^h \right|_{A_0}^2} \, \} \; \nabla \widehat{V}^{h^t} \, [A_0] \; \nabla V^h \; d\mathbf{x}dt, \tag{2}$$

where h_c^e is the characteristic length in the generalized approximate gradient direction.

3. Preconditioned GMRES(k)

The GMRES method, proposed by Saad and Schultz is a Conjugate Gradient like method for solving general nonsymmetric systems of equations and has gained wide acceptance in solving systems coming from Computational Fluid Dynamics (CFD) problems [8],[9],[14]. Writing the linear system of equations generated by (1) as $Ax = b$, the GMRES derives an approximate solution to x, $x = x_0 + z$, where x_0 is the initial guess and z comes from minimizing the residual $\|b - A(xo + z)\|$ over the Krylov subspace $span\left[r_0, Ar_0, A^2r_0, ..., A^kr_0\right]$, where k is the dimension of the space and $r_0 = b - Ax_0$. The major drawback of GMRES is that increasing k the amount of memory required per iteration increases linearly and the computational work increases quadratically . In order to control the amount of work and the memory requirements a restarted version (GMRES(k)) is used. This means that after k orthogonalization steps all variables will be cleaned and the obtained solution will be taken as the initial solution for the next k steps.

The crucial element for a successful use of this version is based on the choice of the parameter k. If it is too small, the convergence may slow down or the method can not converge, but with large value the work and the memory requirements would turn this method useless.

Another important point is the choice of the preconditioner. We choose the Addive Scharwz Method (ASM) [11] which not only can increase the convergence rate but has a high degree of parallelism. The main idea of the ASM is that we can solve the preconditiong step of the GMRES algorithm [13] $z_j = M^{-1}v_j$, by partioning the domain Ω in p overlaping subdomains Ω_i and then approximate A^{-1} by $M^{-1} = \sum_{i=1}^{P} R_i^t\ A_i^{-1}\ R_i$ where A_i is the local submatrice formed by $A_i = R_i\ A\ R_i^t$ with R_i and R_i^t the restriction and extension matrices. It can be noticed that the preconditioning step can be solved in parallel with each processor solving a smallest system of equations. There are two forms to solve the local problems: a direct solver or a iterative solver, like multigrid. At this time we decided to use a direct solve, the LU decomposition method from the Lapack package[16].

4. Distributed Implementation

Recently there has been much interest in using workstations to form distributed systems. With the quickly evolution of desktop workstations, they can off-load jobs from saturated vector computers, often providing comparable term around time at a fraction of the cost. With high-speed networks the workstations can also serve as an inexpensive parallel computer. Another factor in favor of distributed compunting is the availability of many lightly loaded workstations. These otherwise wasted idle cycles can be used by a distributed computation to provide speedups and/or to solve large problems that otherwise could not be tackled.

Similar to other Conjugate Gradient like iterative methods, the most expensive part of the GMRES algorithm is the preconditioner. Using the ASM the preconditioner step is reduced to a series of local sequential problems. But now the question is: *how to* *partition the data among the processors?* The significant fact for this type of system is that the common network used, Ethernet and FDDI, allows all processors to logically communicate directly with any other processor but physically supports only one '*send*' at a time. So we chose one-dimensional partition of the domain and we associate each one with a different processor, reducing the overhead on '*startups*' in the network and the number of neighbors. This type of partition will lead the way that the matrix-vector products will be done.

With this distribution each processor will have a part of the initial residual vector r_0 and part of the matrix-vector result, the v_i vectors. Thus, the orthogonalization will be performed in parallel, where the partial results ($\|r_0\|$,$\|v_i\|$, $\beta_{i+1,j}$) will be exchange to complete the process.

Obviously this procedure requires a greater number of communications. That is increased by the fact that an Element-by-Element structure is being used to storage the matrix which implies that to evaluate this operation we need to do some communications between nearest-neighbors to obtain the contribution of the interface degrees of freedom. However this choice will be better to solve large problems with adaptive procedures because only fixed number of scalar variables, equal to the Krylov dimension, will be exchanged.

Two types of communication are used on this implementation: the nearest-neighbor and one global communication. For the first one we use simple '*sends*' and '*receives*' between neighbors. In the global communications we will use the Bucket Collect algorithms [15].

Recently several tools have been developed to support distributed computing like PVM, MPI and others [12]. The PVM, *Parallel Virtual Machine* is chosen because it can connect a heterogeneous group of workstations without difficulty.

5. Numerical Results

In this section we convey some numerical results obtained by applying the proposed methodology on the solution of a variety of compressible Euler flow problems. All computations were done using piecewise linear elements in space and constant in time. This leads to a time-marching scheme which has low order of accuracy in time but has good stabilitiy properties, being appropriate for solving steady problems.

5.1. Oblique Shock Problem

This problem deals with an inviscid supersonic flow (Mach number M=2) over a wedge at an angle with respect to the mesh. It is illustrated in Figure 1 where the dashed line shows the position of the shock determined analytically.

The Figure 2 which contains the elapsed time of the solver for differente problem sizes is obtained using a collection of sixteen IBM RISC 6000 workstations model 3BT connected by FDDI network. Obviously for small degrees of freedom (dof) the performance is not satisfactory for small numbers of processors. But when the dof increases the performance increases too.

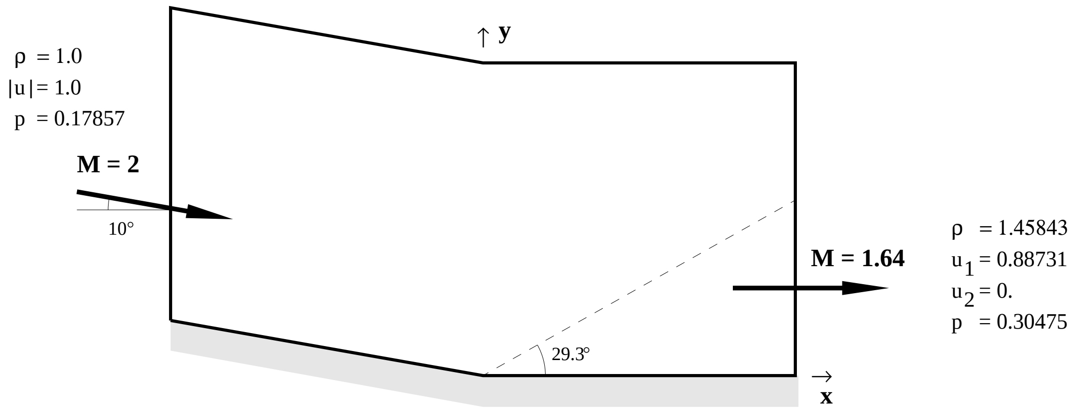

Figure 1. Oblique Shock Problem: problem statement.

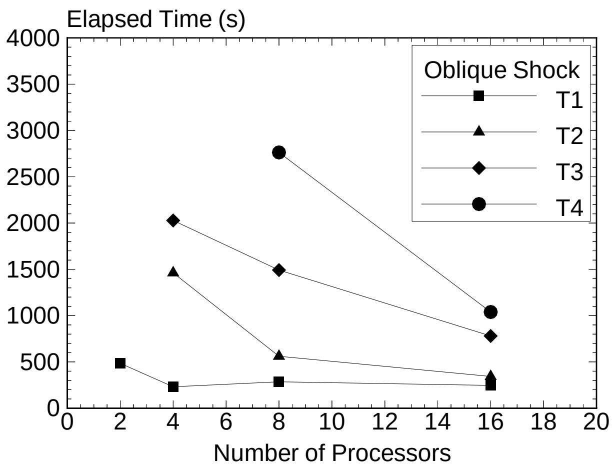

Figure 2. Oblique Shock Problem: elapsed time (s), T1 = 2,244 dof, T2 = 4,420 dof,T3 = 6,596 dof, T4 = 8,580 dof.

In Table 1 we compare the new solver with a widely used direct solver , the frontal solver, runing on a IBM RISC model 590. It can be seen that the solver introduced here in this paper is 1.8 times faster. It is important to remember that 8,580 dofs is still considered a small problem for iterative methods.

8,580 dof	
Serial (590)	Distributed (p = 16)
Frontal	ASM - GMRES(k)
31.3 min	17.3 min

Table 1. Comparison of Direct and Distributed Solver

5.2. Shock-Reflection Problem

This two dimensional steady problem consists of three flow regions separated by an oblique shock and its reflection from a wall.

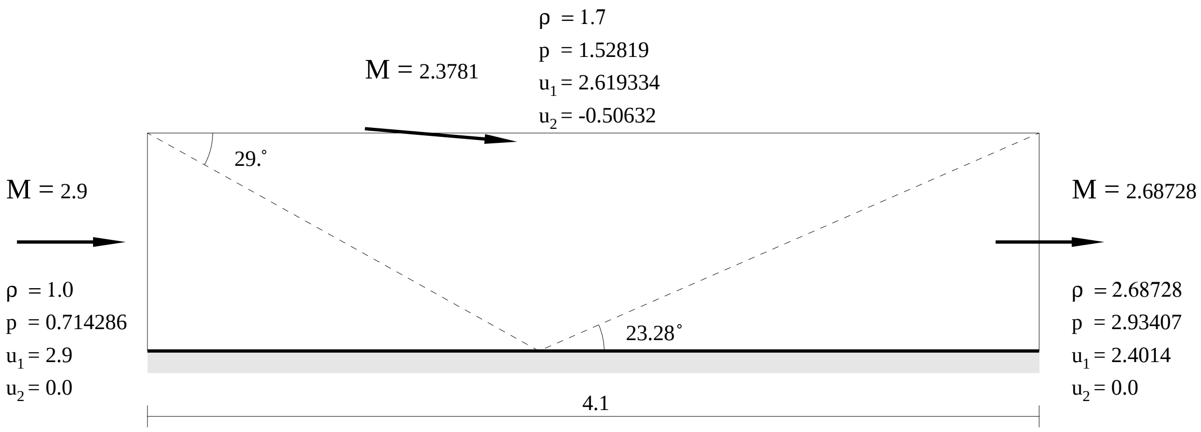

Figure 3. Shock-Reflextion: problem statement.

The timming results for this problem are very similar to the previous example. In Table 2 the comparation with the frontal solver is shown, and again our algorithm is faster by a factor 1.7.

12,740 dof	
Serial (590)	Distributed (p = 16)
Frontal	ASM - GMRES(k)
68.3 min	38.4 min

Table 3. Comparison of Direct and Distributed Solver

6. Conclusions

In this paper we have shown a distributed solver for solving CFD problems, in particular the compressible Euler equations using a stable Petrov Galerkin method written in entropy variables, designed using a space-time finite element formulation. The numerical results showed that this solver is appropriate for distributed systems like a collection of workstations connected by a token-ring or bus based network. We have shown that this type of aproach can be efficient and it can be superior or faster than the usual solvers even for the small size problems solved in this paper.

REFERENCES

1. A. N. Brooks and T. J. R. Hughes, Streamline Upwind Petrov-Galerkin Formulations for Convection-Dominated Flows with Particular Emphasis on the Incompressible Navier-Stokes Equations, *Comput. Methods Appl. Mech. Engrg.* 32 (1982) 199–259.
2. T. J. R. Hughes and M. Mallet, A New Finite Element Formulation for Computational Fluid Dynamics: III. The Generalized Streamline Operator for Multidimensional Advective-Diffusive Systems, *Comput. Methods Appl. Mech. Engrg.* 58 (1986) 305–328.

3. C. Johnson, U. Nävert and J. Pitkaranta, Finite Element Methods for Linear Hyperbolic Problems, *Comput. Methods Appl. Mech. Engrg.* 45 (1984) 285–312.
4. R. C. Almeida and A. C. Galeão, The Generalized CAU Operator for the Compressible Euler and Navier-Stokes Equations, *8th International Conference on Numerical Numerical Methods in Laminar and Turbulent Flows* (1993).
5. R. C. Almeida and A. C. Galeão, An Adaptive Petrov-Galerkin Formulation for the Compressible Euler and Navier-Stokes Equations, accepted for publication in *Comput. Methods Appl. Mech. Engrg.*
6. A. Harten, On the Symmetric Form of Systems of Conservation Laws with Entropy, *J. Comp. Physics* 49 (1983) 151–164.
7. T. J. R. Hughes, L. P. Franca and M. Mallet, A New Finite Element Formulation for Computational Fluid Dynamics: I. Symmetric Forms of the Compressible Euler and Navier-Stokes Equations and the Second Law of Thermodynamics, *Comput. Methods Appl. Mech. Engrg.* 54 (1986) 223–234.
8. F. Shakib, Finite Element Analysis of the Compressible Euler and Navier-Stokes Equations, *Ph. D. Thesis, Stanford University* (1988).
9. M. Mallet, A Finite Element Method for Computational Fluid Dynamics, *Ph. D. Thesis, Stanford University* (1985).
10. R. C. Almeida and R. Silva, A Stable Petrov-Galerkin Method for Convection-Dominated Problems, (submitted to *Comput. Methods Appl. Mech. Engrg.*)
11. Patrick Le Tallec, Domain Decomposition Methods in Computational Mechanics, *Computational Mechanics Advances* 1 (1994).
12. Geist A., Beguelin A., Dongarra J., Jiang W., Manchek R. and Sunderam V., PVM: Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing, *The MIT Press*
13. Youcef Saad, A Flexible Inner-Outer Preconditioner GMRES Algorithm, *iUniversity of Minnesota Supercomputer Institute Research Report* 91/279 (1991)
14. William D. Gropp and David E. Keyes, Domain Decomposition Methods in Computational Fluid Dynamics, *Inter. Journal for Numerical Methods in Fluids* 14 (1992) 147–165
15. M. Barnett, R. Littlefield, D. G. Payne and Robert A. van de Geijn, On the Efficiency of Global Combime Algorithms for 2-D Meshes With Wormhole Routing, *TR-93-05, Department of Computer Sciences, University of Texas* 1993
16. E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammariling, A. McKenney, S. Ostrouchov and D. Sorensen, *LAPACK Users' Guide, Second Edition*, SIAN, 1995