A Newton-GMRES Method for the Parallel Navier-Stokes Equations

J. Häuser[a], R.D. Williams[b], H.-G. Paap[c], M. Spel[d], J. Muylaert[d] and R. Winkelmann[a]

[a]Center of Logistics and Expert Systems, Salzgitter, Germany
[b]California Institute of Technology, Pasadena, California
[c]Genias GmbH., Regensburg, Germany
[d]ESTEC, Noordwijk, The Netherlands

1. INTRODUCTION

CFD is becoming increasingly sophisticated: grids define highly complex geometries, and flows are solved involving very different length and time scales. The solution of the Navier-Stokes equations must be performed on parallel systems, both for reasons of overall computing power and cost effectiveness.

Complex geometries can either be gridded by completely unstructured grids or by structured multiblock grids. In the past, unstructured grid methods almost exclusively used tetrahedral elements. As has been pointed out in [1] and recently in [2] this approach has severe disadvantages with regard to program complexity, computing time, and solution accuracy as compared to hexahedral finite volume grids. Multiblock grids that are unstructured on the block level but structured within a block provide the geometrical flexibility and retain the computational efficiency of finite difference methods.

In order to have the flow solution independent of the block topology, grids are slope continuous, and in the case of the N-S solutions an overlap of two points in each coordinate direction is provided. This causes a memory overhead: if N is the number of internal points in each direction for a given block, this overhead is the factor $(N+4)^3/N^3$. The overhead is caused by geometrical complexity, i.e., to generate a block topology that aligns the flow with the grid as much as possible [3].

Since grid topology is determined by both the geometry and the flow physics, blocks are disparate in size, and hence, load balancing is achieved by mapping a group of blocks to a single processor. The message-passing algorithm must be able to handle efficiently the communication between blocks that reside on the same processor, meaning that there can only be one copy operation involved. Message passing (PVM or MPI) is restricted to a handful of functions that are encapsulated, and thus full portability is achieved. In addition, the code is written in ANSI-C to guarantee portability and provide significant advantages over Fortran (see for example [4]). A serial machine is treated as a one-processor parallel machine without message passing. Therefore the code, *Parnss* [5], will run on any kind of architecture. Grids generated by *GridPro* [7] or *Grid**[1] (Plot3D format) can be directly used.

Available parallelism (the maximum number of processors that can be used for a given

---

Proceedings of the Parallel CFD '95 conference, Pasadena, California, July 1995

problem) is determined by the number of points in the grid: a tool is available to split large blocks, if necessary.

Regarding the solution algorithm for the Navier-Stokes equations, an explicit algorithm is easiest, but is not as efficient as relaxation schemes in calculating the steady state. Often relaxation schemes are used, but it should be remarked that even these methods may not converge for highly stretched grids with large aspect ratios (10^6), as is needed in most viscous flows.

Thus, we shall use implicit methods for these grids, with a linear solver chosen from the Krylov family. To make good use of these techniques, we need an effective and efficient preconditioner; this paper is about the convergence properties of the Navier-Stokes code for different preconditioners used on parallel architectures.

2. Solving the N-S Equations

An implicit step of the discretized N-S equations can be cast in the form of a set of nonlinear equations for the flow variables, whose numerical solution is by a Newton scheme. The computational kernel is a linear solve with the matrix being the Jacobian of the equations. There are numerous schemes for such solves, such as Jacobi relaxation or lower-upper triangular split (Gauss-Seidel). As mentioned above, these schemes are slow to converge for a stiff system, caused by widely varying temporal and spatial scales.

The numerical solution proceeds in five major stages: in this work, stage (4) will be described in some detail.

1. *Topology:* Perform domain decomposition of the solution domain.
2. *Grid generation:* Create a high-quality grid within each domain. Spatial discretization reduces the N-S equations to a set of ODE's.
3. *Explicit Solution:* Advance explicitly in time by using a two step Runge-Kutta scheme.
4. *Implicit Solution:* Advance the solution implicitly in time with the backward Euler scheme, thus requiring solution of nonlinear equations, which can be solved by a Newton or quasi-Newton algorithm, which in turn requires solving sets of linear equations: we use preconditioned GMRES for these.
5. *Root Polishing:* For steady state solution, use a Newton iteration to derive the steady state, which is equivalent to an implicit step with infinite time step.

Investigations are underway to determine if it is possible to relax the accuracy with which the nonlinear equations in (4) are solved, yet still obtain robust and accurate convergence through stage (5).

2.1. Krylov Subspace Methods

In what follows, we describe the CG method, because it is the basis for the *Generalized Minimal Residual (GMRES)* technique, as used in this paper