1990 N91-18982

NASA/ASEE SUMMER FACULTY FELLOWSHIP PROGRAM

MARSHALL SPACE FLIGHT CENTER
THE UNIVERSITY OF ALABAMA

MULTI-BLOCKING STRATEGIES FOR THE INS3D INCOMPRESSIBLE NAVIER-STOKES CODE

Prepared By:	Boyd Gatlin
Academic Rank:	Assistant Professor
University and Department	Mississippi State University Aerospace Engineering
NASA/MSFC:	
Laboratory:	Structures and Dynamics
Division:	Aerophysics
Branch:	Computational Fluid Dynamics
MSFC Colleague:	Dr. Paul K. McConnaughey

Contract No:	NGT-01-002-099 The University of Alabama

MULTI-BLOCKING STRATEGIES FOR THE INS3D

INCOMPRESSIBLE NAVIER-STOKES CODE

Boyd Gatlin
Mississippi State University

INTRODUCTION

With the continuing development of bigger and faster supercomputers, computational fluid dynamics (CFD) has become a useful tool for real-world engineering design and analysis. This is particularly true in cases where experimental data is too expensive or impossible to obtain. However, the number of grid points necessary to resolve realistic flow fields numerically can easily exceed the memory capacity of available computers. In addition, geometric shapes of flow fields, such as those in the Space Shuttle Main Engine (SSME) power head, may be impossible to fill with continuous grids upon which to obtain numerical solutions to the equations of fluid motion.

The solution to this dilemma is simply to decompose the computational domain into sub-blocks of manageable size. Computer codes that are single-block by construction can be modified to handle multiple blocks, but *ad hoc* changes in the FORTRAN have to be made for each geometry treated. For engineering design and analysis, what is needed is generalization so that the blocking arrangement can be specified by the user. This enables the engineer or analyst to concentrate on the flow problem and its solution rather than on programming.

INS3D is a computer program for the solution of steady, incompressible flow problems. Developed at NASA Ames Research Center, it is used frequently to solve engineering problems in the CFD Branch at Marshall Space Flight Center. INS3D uses an implicit solution algorithm and the concept of artificial compressibility to provide the necessary coupling between the pressure field and the velocity field. The thrust of the work reported here is the development of generalized multi-block capability in INS3D.

BLOCKING STRATEGIES

Central to the concept of multi-block computations is the requirement that only one block occupy core memory at any time. After each time step or iteration, the current block is written to an out-of-core device such as magnetic disk, and the next block is read in. This process is carried out over all blocks and then is repeated until the solution converges to some tolerance. Clearly, the issue of communication among the blocks is immediately raised, since the solution must have the same order of accuracy at block interfaces as it does over the rest of the field. In addition, the solution must be continuous and smooth.

An obvious approach is to construct grids such that one or more surrounding layers of points of one block overlaps into the adjoining blocks. For a central-difference algorithm with second-order spatial accuracy, such as INS3D, only one such overlapping layer is needed. Points along the block interface can then be computed as field points in the usual way. The overlapping points which are interior field points in the adjoining block serve as boundary values in the current block, and vice versa. This amounts to an implicit treatment of block interfaces at the same order of accuracy as that over the rest of the field.

However, this mutual overlapping of all adjoining blocks results in a non-unique solution along block interfaces, since these points are computed in both blocks. In principle, the values would be identical at complete convergence, but as a practical matter this is not true. One treatment used in this work is to simply average the two values in order to obtain a unique solution on the interfaces.

A more sophisticated approach is to have only one of two adjoining blocks overlap into its neighbor. Under this approach, the block interface is computed in only the current block, and the interface then serves as a true boundary for the neighboring block. This requires considerably more programming logic to keep track of the relationship among several interfacing blocks. In fact, more work remains to be done on this issue for cases where several blocks share a common corner.

Since INS3D is a central-difference scheme, artificial dissipation terms are needed to damp out oscillations resulting from the odd-even decoupling inherent in the differencing molecule. At block interfaces, these can be evaluated as one-sided differences, as is done at natural boundaries. However, it was found that central differencing of these terms leads to more computational stability at block boundaries.

The effect of using overlapping solution values is, of course, to increase the size of the block to be solved by at most two in each coordinate direction. An important drawback to out-of-core storage of overlap values is that each time a block is to be computed, it must first be loaded into core, and, in the general 3D case, its boundary values must be updated by reading each of the six or more surrounding blocks and extracting the overlapping layers, one block a at a time. The resulting I/O operations would quickly become prohibitive.

A more efficient method is to maintain in-core arrays of block interface and overlap layers. This does not result in any significantly greater core storage than would be necessary if blocks were swapped in and out of core in order to extract boundary values, since that approach would require that two blocks occupy internal storage simultaneously. However, it is much faster, since the I/O operations are drastically reduced.

USER INPUT

The relationships among the blocks in a domain are specified by user input through the NAMELIST utility, consistent with that in the original, one-block version of INS3D. In principle, arbitrary blocking configurations and boundary conditions can be treated without program modifications. In practice, however, there will inevitably be cases which will have to be treated individually through modifications to subroutines which set initial conditions and boundary conditions.

In setting up the relationship between any two blocks, one is take to be BLKA, the current block, and the other BLKB, the neighboring block. The common interface is then specified by STARTA and ENDA in BLKA and by STARTB and ENDB in BLKB, where START and END are indices of the starting and ending points of the block boundary. Each BLK is assumed to have six faces, identified in BLKA, for example as SURFA=___, where ___ indicates the face. The numbering scheme for faces is such that SRFA=1 indicates the face corresponding to the minimum value of J, the number one index in INS3D. SRFA=6, then, would refer to the face in BLKA where the number three index, L, is maximum.

Boundary conditions for all blocks are established in the same way. Block interfaces are simply treated as another type of boundary condition. For each block face, its boundary conditions are established with a simple user flag. For example, BCTYPE=1 imposes no-slip on a block boundary, while BCTYPE=4 causes the boundary to be computed using mutually overlapping layers with BLKB. Several BCTYPE options will be eliminated when the code is released to MSFC, since they invoke conditions used only experimentally during the development.

RESULTS AND SUMMARY

Only preliminary results were obtained during the Summer Fellowship period. Several two-dimensional cases were run successfully, but the persistent presence of pressure disturbances in the neighborhood of block interfaces indicates that there are subtle problems yet to be addressed. Indications are that a way must be found to effectively smooth the pressure field at block interfaces. A good possibility would be to install adjustable damping terms in the pressure equation in the neighborhood of block interfaces.

A few three-dimensional cases were computed, including a straight circular duct and a circular duct with a 90-degree bend. Both cases diverged because of contamination of the solution at block interfaces. In order to try to isolate the problem, the straight circular duct was computed with inviscid conditions. With no errors in the code, this case should have converged immediately, since the initial conditions specified were, in fact, the correct solution. Instead, the solution diverged very slowly. Analysis of the results indicated a false pressure gradient arising at the block interface. The source of this problem was not discovered by the end of the fellowship period.

However, the basic algorithm and underlying blocking strategy are now fully developed and working in INS3D. With additional collaborative efforts between the author and engineers in the MSFC CFD Branch, the remaining difficulties can be resolved.