Extended abstract for the 20th AIAA Applied Aerodynamics Conference

Parametric Study of a YAV-8B Harrier in Ground Effect Using Time-Dependent Navier-Stokes Computations

Shishir Pandya[¥] **and Neal Chaderjian**[*]
NASA Ames Research Center, Moffett Field, CA 94035

and

Jasim Ahmad[#]
ELORET, Moffett Field, CA 94035

1 Introduction

Flow simulations using the time-dependent Navier-Stokes equations remain a challenge for several reasons. Principal among them are the difficulty to accurately model complex flows, and the time needed to perform the computations. A parametric study of such complex problems is not considered practical due to the large cost associated with computing many time-dependent solutions. The computation time for each solution must be reduced in order to make a parametric study possible. With successful reduction of computation time, the issue of accuracy, and appropriateness of turbulence models will become more tractable.

A parametric study to generate a longitudinal stability and control database on the YAV-8B Harrier aircraft in near-hover conditions has been discussed in a previous paper[1][2]. Two main issues with computing Harrier flow fields are complex geometry and complex flow physics. The Chimera overset grid approach is used to address the geometric complexity. An overset mesh with 67 zones is generated around the YAV-8B to compute the flow field in the vicinity of the aircraft and the ground plane. The flow field with two high-speed rear jets and two front jets impacting the ground from an aircraft fixed at heights between 10 and 30 feet, is complex and time-varying. At each height, computations are carried out at several angles of attack to generate a database of force coefficients. The OVERFLOW code[3] is used to compute the time dependent, viscous flows. Each computation for the generation of the database is costly because it involves long run times to simulate unsteady flow with low dominant frequencies (~1Hz). In a first attempt to generate a database, it took more than one month to generate one solution, without taking grid generation into account. The bottlenecks included the slow decay of numerical transients associated with the solution start-up process, and the lack of scalability to more than 8 processors.

In our previous paper [1], the emphasis is put on reduction of the time to generate a solution. The computations in the earlier paper were done using a first order accurate method in time. Small time steps had to be used in order to maintain algorithm stability to capture the low-speed ambient

---

[¥] Research Scientist, Member AIAA

[*] Research Scientist,Associate Fellow AIAA

[#] Senior Research Scientist

flow. The current paper discusses the use of a dual time-stepping method to further speed-up the process. The method allows the use of larger time steps and is more robust. The robustness allows the solution to be started from free stream (impulsive start) conditions in time-accurate mode. This reduces artificially induced transients allowing for the evaluation of temporal statistics earlier in the computation. The algorithm is now formally second order accurate in time and helps maintain solution accuracy with larger time steps. The dual-time stepping method also eliminates the factorization and linearization errors by iterating between time steps. This dual-time stepping technique and its advantages are briefly described in section 2.2 .

Reduction of computation time can also be achieved by improving the parallel efficiency. The MLP (Multi-level parallelism)[4] version of the OVERFLOW[2] computational fluid dynamics (CFD) code provides a method of grouping mesh zones into groups so that each group requires approximately the same amount of computational work. However, if too few groups are used on a large number of processors, every group is spread over many processors. Since only loop-level parallelism is responsible for parallelizing each group, the goal is to keep the number of processors per group to a minimum in order to get the best fine grain parallel efficiency. Various meshes are split up to ensure both an even distribution of work in domain decomposition and better loop-level parallel efficiency.

The Origin 3000 systems also provide us with a faster platform contributing to the speed-up. This platform has four CPUs per node (four CPUs which share the same main memory chips) instead of the two on the origin 2000 systems. This enables us to use larger groups as any problem's loop-level parallel efficiency will be higher on four processors of an origin 3000 computer than four processors of an origin 2000 machine. Further improvements in the memory access speed, and the cache size and speed also contribute to the faster performance making it possible to scale the problem up to more processors. The present set of cases typically used 112 CPUs per case to generate a static longitudinal stability and control database in one week.

Finally, post-processing many unsteady solutions to determine mean forces and moments for a database can be an arduous task. The large amounts of data generated during a database calculation is automatically stored on a mass storage system. To simplify post-processing, a user interface that allows a user to access and process stored data in an automated manner is improved and utilized. An earlier version of the post-processing tool is discussed in [2]. Addition of a statistical analysis of the lift coefficient history is discussed briefly. Unsteady flow visualization is used to correlate the dominant frequency to the changes in the flow field.

2 Solution Procedure

The solution procedure used to generate a database of time-dependent solutions has been described in references [1][2]. While the Turbulence model and the boundary condition treatment remain unchanged, the grid system, and the numerical algorithm have been changed. These changes are described in this section along with the process used to generate a database of unsteady viscous solutions.

2.1 Grid system

The grid system is modified to improve the load balance when using a large number of CPUs. In the use of MLP, a large number of CPUs for the computation requires each group be assigned more than the ideal number of CPUs for best fine-grain parallel efficiency unless the total mesh size is also very large. There are several ways to increase the loop-level parallel efficiency of each group. However, changing the order of the loops, or changing the stride are not attempted as these changes can be problem dependent. Instead, it is decided to reduce the number of CPUs working on each group by splitting up the large meshes in several pieces(e.g. jet grids). The improvement in parallel efficiency due to grid splitting is discussed in section 3 .

The algorithm in OVERFLOW for determining how much weight each mesh should have is also modified to suit the fact that not all meshes are equal. The viscous meshes generally require more CPU time per grid point than the inviscid meshes. The viscous meshes are also not equal in terms of CPU time required per grid point. This is because the meshes close to the aircraft surface are computed with the thin-layer assumption while other meshes(e.g. jet grids) require full viscous modelling. A weighting system is introduced where by each mesh is given a weight according to the work done by that mesh. If an Euler mesh is the baseline with a weight of 1.0, a thin-layer mesh would have a weight of 1.15. A mesh where all 3 directions are viscous would therefore have a weight of 1.4. These weights combined with the method of computing a load-balance already in OVERFLOW provides a more accurate representation of the work done by the meshes and thus higher parallel performance.

2.2 Dual time-stepping

The dual-time stepping method in OVERFLOW is based on the dual time-stepping methods presented in references [5][6]. The Navier-Stokes equations can be written in conservative form as

$$\frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} + \frac{\partial G}{\partial z} = L(Q)$$

where $Q = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix}$, E, F, and G are the inviscid fluxes and $L(Q)$ represents the viscous terms.

An artificial time term is introduced to the governing equations in order to provide a relaxation (sub-iteration) procedure between physical time steps.

$$\frac{\partial Q}{\partial \tau} + \frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} + \frac{\partial G}{\partial z} = L(Q)$$

In generalized coordinates, this equation is discretized with first order accurate Euler implicit discretization for the artificial time term, second order backwards difference discretization for the physical time terms and central difference discretization for the spatial terms to obtain

$$(I + \Delta\tau\Gamma_e S_p^{-1} A^k \delta_\xi + \Delta\tau\Gamma_e S_p^{-1} B^k \delta_\eta + \Delta\tau\Gamma_e S_p^{-1} C^k \delta_\zeta)\Delta\hat{Q} = \Gamma_e S_p^{-1} R^k$$

where $\hat{Q} = J^{-1}Q$, $S_p = I + \frac{3\Delta\tau}{2\Delta t}\Gamma_e$, $\Gamma_e = \frac{\partial Q}{\partial Q_p}$ (a transformation matrix between conservative and primitive variables), A, B, and C are the flux Jacobians and

$$R^k = -\Delta\tau\left(\frac{3\hat{Q}^k - 4\hat{Q}^n + \hat{Q}^{n-1}}{2\Delta t} + \delta_\xi \hat{E}^k + \delta_\eta \hat{F}^k + \delta_\zeta \hat{G}^k - L(Q)\right) .$$

The variable n is the time step counter while the variable k is the sub-iteration counter. Finally, $\Delta Q = Q^{K+1} - Q^k$. A diagonalized approximate factorization algorithm[7] is used for the solution of this equation. When converged in artificial time, this method is formally second order accurate.

The dual time stepping algorithm has three main advantages. The first is its robustness, which allows the computation to start in a time-accurate manner. It also minimizes the artificially generated transients. The transients with the dual-time method are closer to the physical transients than when the solution is started with a local-time stepping procedure. In the previous paper[1], the solution is briefly started in a steady-state manner using multigrid and local time stepping before transitioning to a time-accurate calculation. Dual-time stepping eliminates this step allowing for meaningful temporal statistics to be obtained more quickly.

The second advantage of the dual-time stepping method is that it allows the use of larger time steps. Stability and accuracy restrictions, due to low-Mach number flow, limit the time step to unacceptably low values. Time steps can be increased by an order of magnitude or more than is possible without the use of dual time stepping. Another issue to be addressed is that the current compressible formulation in OVERFLOW is not suitable for very small Mach numbers. Dual-time stepping can be modified to implement an unsteady preconditioner which addresses the accuracy issues at low Mach numbers. Modified dual-time stepping equation with preconditioning matrix Γ_p can be expressed as

$$\Gamma_p \frac{\partial Q}{\partial \tau} + \frac{\partial Q}{\partial t} + \frac{\partial E}{\partial x} + \frac{\partial F}{\partial y} + \frac{\partial G}{\partial z} = L(Q_p)$$

A preconditioner is not used in the current calculations.

The third and final advantage of the dual-time method is that the sub-iteration procedure eliminates the factorization error in the diagonalized approximate factorization method. It also eliminates the linearization error.

2.3 Parametric Database Generation process

The goal of the database generation process is to achieve as much automation as possible. An object-oriented perl module is used to connect to, store files on, and retrieve files from a mass storage system[2]. This is necessary because the data generated for a database is far greater in size than is available on a local hard disk array.

The module is called by a perl script that runs a single case. The script first sets up database parameters and determines which case it needs to run (Mach number, height, and angle of attack). It then edits a standard OVERFLOW input file to setup the current case conditions. The next step is to copy the appropriate grid and solution (restart) files from the mass storage system to the local

computer. OVERFLOW is than run for up to 8 hours at a time and the newly generated force/ moment history and solution files are stored on the mass storage system.

To run multiple solutions at once and to automate the resubmission process, another perl script is used which finds the set of available nodes from the operating system and sets up which cases will be run on which CPUs. The script also sets up the parameters for each case to be run and subsequently passes these parameters on to the script which runs individual cases. After invoking as many single runs as it can, given the number of CPUs available, the multiple run script proceeds to monitor each job. When a job is completed, it puts an end of completion entry in the standard output. When all jobs are completed, the script determines if the user has specified that it should be resubmitted. If appropriate, the job is resubmitted for further runs else it is terminated.

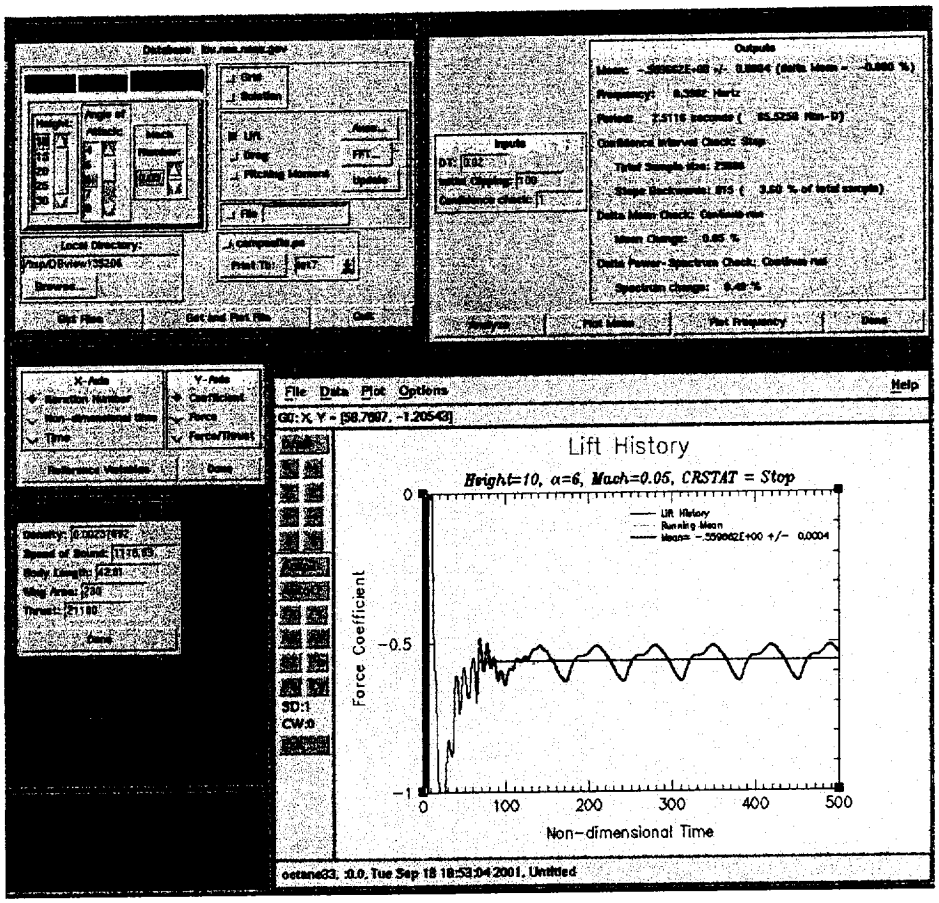

Figure 1: Post-processing interface

The solution process can be further automated by including statistical analysis in the multi-run script which decides when a case is finished. The decision to stop a job is made based on two questions. First, has a mean quantity (e.g. lift) converged to a reasonable value? Second, has this mean value achieved a dominant frequency? Currently, these questions are answered by the user with the help of a post-processing tool described below.

2.4 Post-processing

The first question can be answered with a confidence interval test based on a standard student t-distribution. The second question is much more difficult to answer with certainty as the dominant frequency may not be constant, the solution may prefer to vary between two frequencies, or the flow may be steady. Currently, we examine the results of the fast Fourier analysis through a user interface shown in Figure 1 . The decision to stop a run is made based on how converged the mean is. The frequency content is analyzed, but does not as yet play a large role in stopping the solution. The automation aspect of this method as well as the precise criteria for deciding to stop the run are currently under investigation.

A statistical analysis has been added to the post-processing tool discussed in [2]. Figure 1 shows the user interface to the post processing tool (The FFT analysis window is shown in the upper right corner of the figure). A lift history along with a running mean (in green) and the final mean (in blue) is shown in the lower right corner of the figure. The xy-plot is generated using standard xy-plotting program driven through a pipe from the user interface. This tool gives the user access to the database without having to know anything about how the database is set up on the mass storage machine. The user is asked to choose a database and the parameters to investigate in the main window (top left in the figure). The user can then retrieve grid and solution files and view them interactively. Default views and rakes are setup to make the process simpler for the user. Four default views can also be arranged in a one sheet summary page. An example of this is shown in Figure 2 .

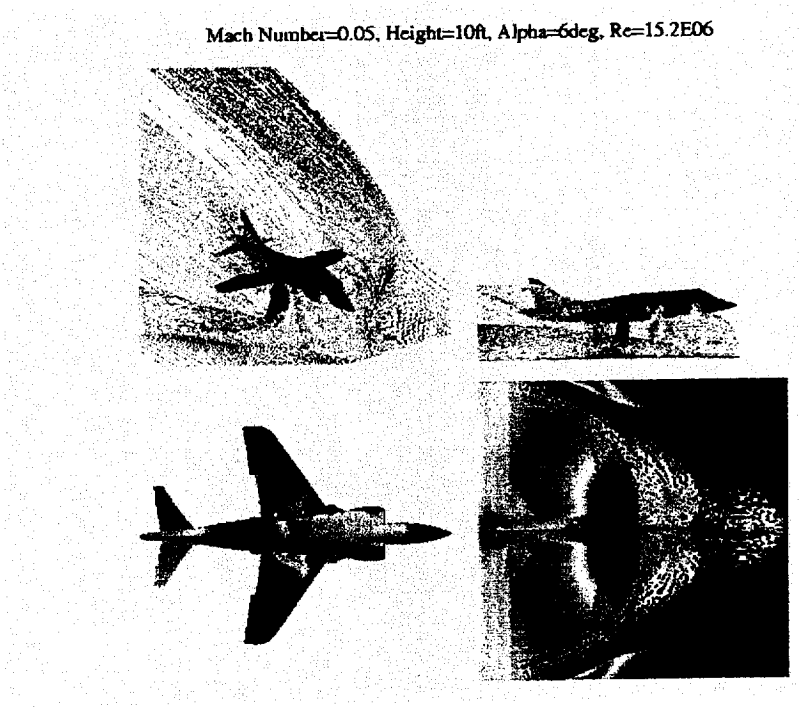

Figure 2: The default views available in the post-processing tool

The post processing tool also lets the user plot lift, drag, and pitching moment histories. The fast Fourier analysis can be performed on a force/moment history to obtain mean and confidence interval information along with a power spectrum to reveal the dominant frequencies. The abscissa can be set to the time step number, non-dimensional time, or physical time. The reference parameters, such as speed of sound, and wing area, used for conversion from coefficient to force can be set in a separate window (shown in lower left). With the use of the reference parameters, the Y-axis on this plot can be assigned to force coefficient, force, or force divided by thrust. The user is also given the option to update the database (i.e. recompute the vehicle forces and moments based on the OVERFLOW output). This ability is especially helpful when the run is in progress.

3 Results

To assess the goals of speed and accuracy, we present two studies. First to show the time accuracy, we show that the time dependent solution is changing very little with time step refinement. Second, a scalability study is presented to show that the current method is faster on a given number of CPUs. The total speed up achieved is demonstrated by comparing the solution time to previous solutions of the same problem. Finally, the computed lift coefficient for the 35 cases in the database is presented along with a method for extending the database using monotonic cubic splines. All 35 solutions were generated on the SGI Origin 2000/3000 machines using the MLP version of OVERFLOW in one week.

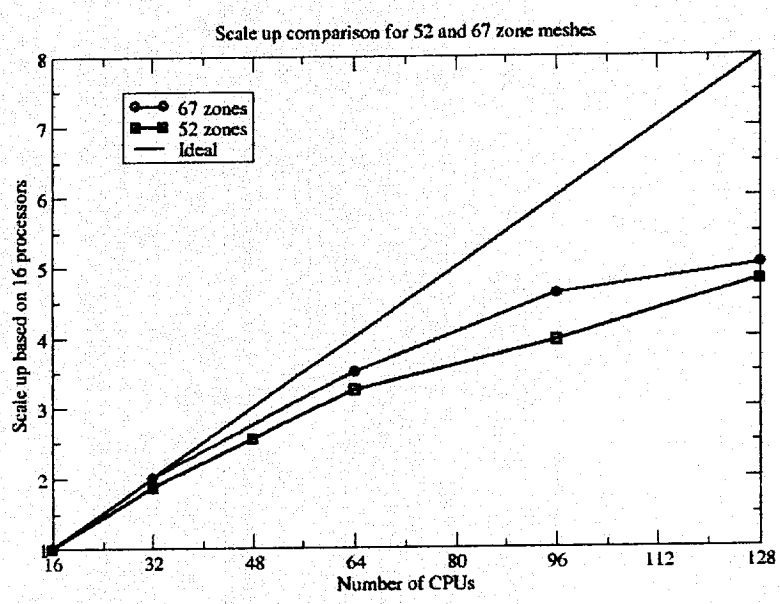

Figure 3: Scalabilty comparison of 52 and 67 zone solutions

3.1 Time accuracy

A time accuracy study will be performed and presented in the final paper. The study will involve restarting an existing solution with half, quarter and eighth the time steps to show time step independence.

3.2 Solution of 35 cases in one week on 952 processors

A database of solutions was generated on 840 processors of two SGI Origin 3000 machines and 112 CPUs of an origin 2000. Speed up issues are addressed by examining the speed caused by change in the mesh, the algorithm, and the computer architecture. To asses the speed up associated with the new grid system, the same problem was run on the same machine with the old mesh (52 zones) and the new mesh (67 zones). Figure 3 shows the speed up of both the 52 zone mesh and the 67 zone mesh. This figure shows there was some improvement using more zones. The time for 16 CPUs is used as the reference value. The 67 zone solution scales linearly up to 32 processors. The problem does not scale linearly beyond that because of low work load for each processor due to the small problem size of 3.2million grid points. However, the scale up is excellent up to 64 processors where the solution takes only 15% more time compared to 16 CPU equivalent time. At 96 CPUs, the solution takes 30% more time and is still better than many implicit parallel CFD codes reported in the literature. The parallel scale up improves even further on the origin 3000 systems due to the higher number of CPUs per node, the faster access to memory and the larger cache size. This will be shown in detail in the final paper.

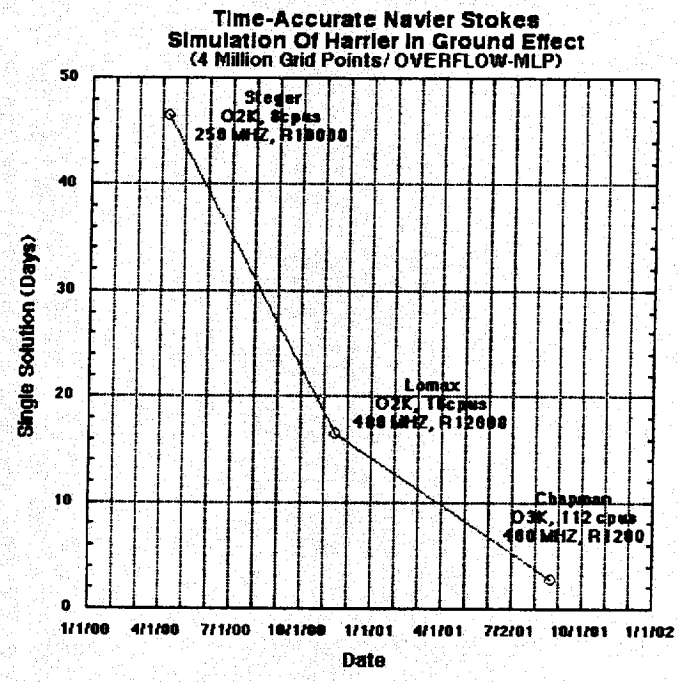

Figure 4: Process speed up as a function of time

The computation time is also reduced by the use of the dual-time stepping algorithm and by use of the faster communications architecture of the SGI Origin 3000 machines. The improvement of solution time for 4 million grid points over the past two years is shown in Figure 4 . The first data point corresponds to the time-dependent solution of a simplified Harrier geometry. Only the fuselage, wing, inlet and jet exits were modelled along with the ground plane. The half body meshes are made of an average of 2.5 million grid points in 37 zones. The solution was performed on the SGI Origin 2000(Steger) at NASA Ames research center which has 128 MIPS R10000 CPUs rated at 250MHz. This was the production machine in March/April 2000 when this baseline database was generated.

The second data point in the figure corresponds to the full geometry computation. The full geometry includes the empennage and a deployed wing flap. The mesh consists of 3.1million grid points at the lowest vehicle height of 10 feet and 3.6million grid points at the highest vehicle height of 30 feet. These grid points are distributed in 52 overset zones. It was performed on the Origin2000(Lomax) at NASA Ames research center which has 512 MIPS R12000 CPUs which run at 400MHz. Larger grid systems are required at higher heights to capture the larger space between the aircraft and the ground plane. All grids are automatically generated from a base grid system.

The last data point in Figure 4 corresponds to the same geometry as the one run on Lomax, but the mesh was changed to achieve better scalability on up to 128 processors as noted in an earlier section. This computation was carried out on an Origin 3000(Chapman) on 112 CPUs per case. Each CPU is a 400MHz MIPS R12000 as was the case for Lomax. The machine based speed up comes from the faster communication, more/faster cache, and a new node board arrangement where each node holds 4 CPUs that share one memory bank instead of the 2 CPUs on Lomax.

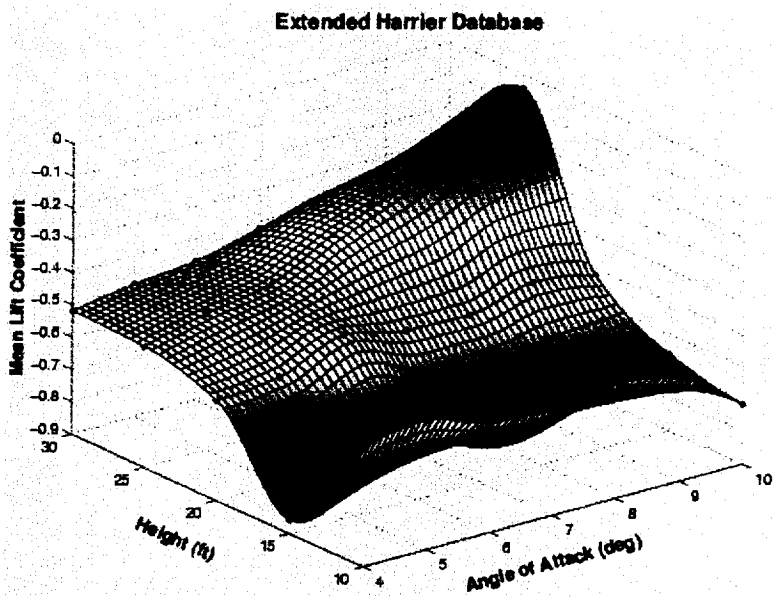

Figure 5: Lift as a function of vehicle height and angle of attack

The initial 35 database solutions are represented in Figure 5 by the surface plot of the Mean coefficient of lift plotted against vehicle height and angle of attack. The data is extended to 2500 points by a monotonic cubic spline procedure. The cubic spline and the trends seen will be discussed in detail in the paper.

3.3 Unsteady Flow Visualization

Unsteady flow visualization is performed on several cases in the database in an attempt to explain the variations in flow properties in terms of the changes in the flow conditions. A software tool, Graphics Encapsulation library (GEL), was developed at NASA Ames research center and relies on "out-of-core" visualization technology. This method is an excellent tool to visualize the large amount of time dependent data from one simulation on a work station. With limited memory, this is usually a difficult task. However, "out-of-core" algorithms can predict which part of the data will be needed. The unused data can reside on disk instead of loading the entire data file in core [8][9].

The dominant frequency in the flow can be correlated to a flow feature with the help of unsteady flow visualization. For example, at a height of 10ft and angle of attack of 6° (lift history shown in Figure 1), the jets are steady but the dominant frequency in the lift history correlates very well with the movement of the fountain vortex. For identification, Figure 6 shows a typical harrier flow field. Unsteady flow visualization will be used to identify dominant flow structures and will

Figure 6: Depiction of typical frame of unsteady flow visual-

be reported in the final paper.

4 Summary and Conclusions

A database of 35 time dependent viscous solutions is generated in one week on modern computational platforms with improved algorithms. This database was extended to 2500 solutions by a monotonic cubic spline procedure. A speed up of over 20x is achieved in a span of about one and half years. A new tool (GEL) is used for unsteady flow visualization to create movies of the time-dependent flows so the dominant frequencies can be correlated to the dominant features of the flow. The time consuming post-processing step is further improved with a better user interface and the addition of statistical analysis to compute mean forces and moments and to examine the dominant frequencies in the flow.

However, the use of statistical analysis to determine when a run is complete remains a topic of further research. The current project has made the computation of high fidelity time dependent viscous solutions more affordable and practical by using improved computing technologies and state-of-the-art computing platforms. The harder questions of accuracy and turbulence modelling on problems with geometry and flow complexities such as the harrier aircraft can now be more readily addressed.

5 References

[1] Chaderjian, N. M., Pandya, S., Ahmad, J., and Murman, S. M., "Parametric Time-Dependent Navier-Stokes Computations for a YAV-8B Harrier in Ground Effect," AIAA Paper No. 2002-0950, To be presented at the AIAA 40th Aerospace Sciences Meeting & Exhibit, Reno, NV, January 14-17, 2002.

[2] Murman, S. M., Chaderjian, N. M., and Pandya, S, "Automation of a Navier-Stokes S&C Database Generation for the Harrier in Ground Effect," AIAA Paper No. 2002-0259, To be presented at the AIAA 40th Aerospace Sciences Meeting & Exhibit, Reno, NV, January 14-17, 2002.

[3] Buning, P. G., Jespersen, D. C., Pulliam, T. H., Chan, W. M., Slotnick, J. P., Krist, S. E., and Renze, K. J. "OVERFLOW User's manual," NASA.

[4] Taft, J. R., "Multi-Level Parallelism, a simple highly scalable approach to parallelism for CFD," HPCCP/CAS workshop Proceedings, (C. Schulbach, editor), 1998

[5] Pulliam, Thomas H., "Time accuracy and the use of implicit methods," AIAA Paper No. 93-3360

[6] S. Venkateswaran and Merkle, C. L., "Dual time stepping and preconditioning for unsteady computations," AIAA Paper No. 95-0078

[7] Pulliam, T. H., and Chausee, D. S., "A diagonal form of an implicit approximate-factorization algorithm," Journal of Computational Physics, Vol. 39, No. 2, 1981, pp. 347-363

[8] Sandstrom, T. A., Chaderjian, N. M. "The Power of Unsteady Flow Visualization," Gridpoints, Vol. 2, No. 2, Summer, 2001, pp. 16-17,20.

[9] Cox, M. B., Ellsworth, D., "Application-controlled demand paging for out-of-core visualization," IEEE Visualization '97 (Roni Yagel and Hans Hagen editors) pp.235-244.