CASE 2: SIMULATION OF A PERIODIC JET IN A CROSS FLOW WITH A RANS SOLVER USING AN UNSTRUCTURED GRID

H. L. Atkins

Computational Modeling & Simulation Branch, NASA Langley Research Center, Hampton, VA 23681-2199

Introduction

A second-order unstructured-grid code, developed and used primarily for steady aerodynamic simulations, is applied to the synthetic jet in a cross flow. The code, FUN3D, is a vertex-centered finite-volume method originally developed by Anderson[1, 2], and is currently supported by members of the Fast Adaptive Aerospace Tools team at NASA Langley. Used primarily for design[3] and analysis[4] of steady aerodynamic configurations, FUN3D incorporates a discrete adjoint capability, and supports parallel computations using MPI.

Solution Methodology

A detailed description of the FUN3D code can be found in the references given above. The code is under continuous development and contains a variety of flux splitting algorithms for the inviscid terms, two methods for computing gradients, several turbulence models, and several solution methodologies; all in varying states of development. Only the most robust and reliable components, based on experiences with steady aerodynamic simulations, were employed in this work.

As applied in this work, FUN3D solves the Reynolds averaged Navier-Stokes equations using the one equation turbulence model of Spalart and Allmaras[5]. The spatial discretization is formed on unstructured meshes using a vertex-centered approach. The inviscid terms are evaluated by a flux-difference splitting formulation using least-squares reconstruction and Roe-type approximate Riemann fluxes. Green-Gauss gradient evaluations are used for viscous and turbulence modeling terms.

The discrete spatial operator is combined with a backward time operator which is then solved iteratively using point or line Gauss-Seidel and local time stepping in a pseudo time. For steady flows, the physical time step is set to infinity and the pseudo time step is ramped up with the iteration count. A second-order backward in time operator is used for time accurate flows with 20 to 50 steps in the pseudo time applied at each physical time step.

For this effort, FUN3D was modified to support spatially varying boundary and initial conditions, and unsteady boundary conditions. Also, a specialized in/out flow boundary condition was implemented to model the action of the diaphragm. This boundary condition is described below in more detail.

The grids were generated using the internally developed codes GridEX[6] for meshing the surfaces and inviscid regions of the domain, and for CAD access; and MesherX[7] for meshing the viscous regions. Grid spacing in on the surfaces and in the inviscid regions are indirectly controlled by specifying sources. The viscous layers are generated using an advancing layer technique. MeshersX allows the user to control the spatial variation of the first step off the surface, growth rates, and the termination criterion by providing small problem dependent subroutines.

Modeling of Diaphragm Boundary

A specialized in/out flow boundary condition is formulated to model the action of the diaphragm at a stationary boundary. This boundary condition is similar to a standard characteristic boundary condition, commonly

applied at far-field boundaries, in that the inviscid boundary flux is evaluated from an intermediate solution state U_i that is computed from the current boundary solution U_b and a prescribed external state U_p. In the weak form implemented in FUN3D, the boundary solution is not replaced by the intermediate solution but is allowed to evolve. The viscous gradients are evaluated using only the boundary and interior solution values, and have no knowledge of the prescribed or intermediate solution states.

At a far-field boundary, the intermediate solution is determined by the characteristic relationships that model the convective and acoustic waves that are assumed to exist between the boundary and the far-field. For example, at a subsonic boundary

$$R^+(U_i) = R^+(U_b), \quad R^-(U_i) = R^-(U_p), \quad and \quad R_c(U_i) = \begin{cases} R_c(U_b) & \text{if } V_n >= 0 \\ R_c(U_p) & \text{otherwise,} \end{cases} \tag{1}$$

where $R^{+(-)}$ denotes the characteristic variable associated with acoustic waves leaving(entering) the domain, R_c denotes characteristic variables associated with convective waves, and V_n is the normal velocity through the boundary, with outflow taken as positive.

In the case of a moving diaphragm, the "prescribed" state is that at the diaphragm face, and only the diaphragm velocity is known. If the boundary were moving with the diaphragm, than the intermediate state would be given by

$$V_n = 0, \quad R^+(U_i) = R^+(U_b), \quad and \quad R_c(U_i) = R_c(U_b). \tag{2}$$

Simply applying these conditions at a stationary location, by setting V_n equal to the diaphragm velocity, would result in under specifying the flow during the inflow phase of the simulation. To stabilize the simulation, the latest value of R_c is saved during the outflow phase of the calculation, and reapplied during the inflow phase to give:

$$V_n = \alpha\cos(\omega t), \quad R^+(U_i) = R^+(U_b), \quad and \quad R_c(U_i) = \begin{cases} R_c(U_b) & \text{if } V_n >= 0 \\ \hat{R}_c(U_b) & \text{otherwise,} \end{cases} \tag{3}$$

where $\hat{R}_c$ denotes the value saved from the most recent outflow cycle. In practice, only the entropy is saved and reapplied. The boundary tangential velocity is allowed to evolve without constraint. Also, during the inflow cycle, the entropy is gradually relaxed back toward its initial value.

Implementation and Case Specific Details

The geometry of the cavity is simplified by making the diaphragm flat with the height of the diaphragm chosen so that the "at rest" volume of the cavity is unchanged. To reduce the problem size, the domain is divided at the tunnel centerline and only half of the domain is grided. Although all simulations presented here were performed on this half-domain geometry, it is possible to reflect the grid about the centerline to obtain a symmetric grid for the larger problem. The computational domain extends from 8 jet diameters upstream and to the side of the jet center, and to 16 jet diameters downstream of the jet center.

The fine grid was sized so that the first spacing on the tunnel wall would have a $y^+ \leq 1$. Spacings inside the cavity were based on preliminary simulations with steady blowing applied at the diaphragm face. Griding sources were placed around the lip of the jet and along the jet centerline in an effort to improve the clustering there. Figures 1(a-c) shows three views of the mesh on the symmetry plane that illustrate the mesh distribution near the jet.

A coarse grid was generated simply by doubling all grid sources and modifying the growth rate of the viscous layers. Originally, this grid was intended only to provide rapid turn-around while sorting out boundary and initial conditions. The fine grid has 1457853 tetrahedra and 255426 nodes. The coarse grid has 254046 tetrahedra and 46063 nodes.

Characteristic boundary conditions are applied weakly at the inflow, outflow and top boundaries. Symmetry conditions are imposed at both y=constant boundaries. No-slip conditions are enforced on the tunnel

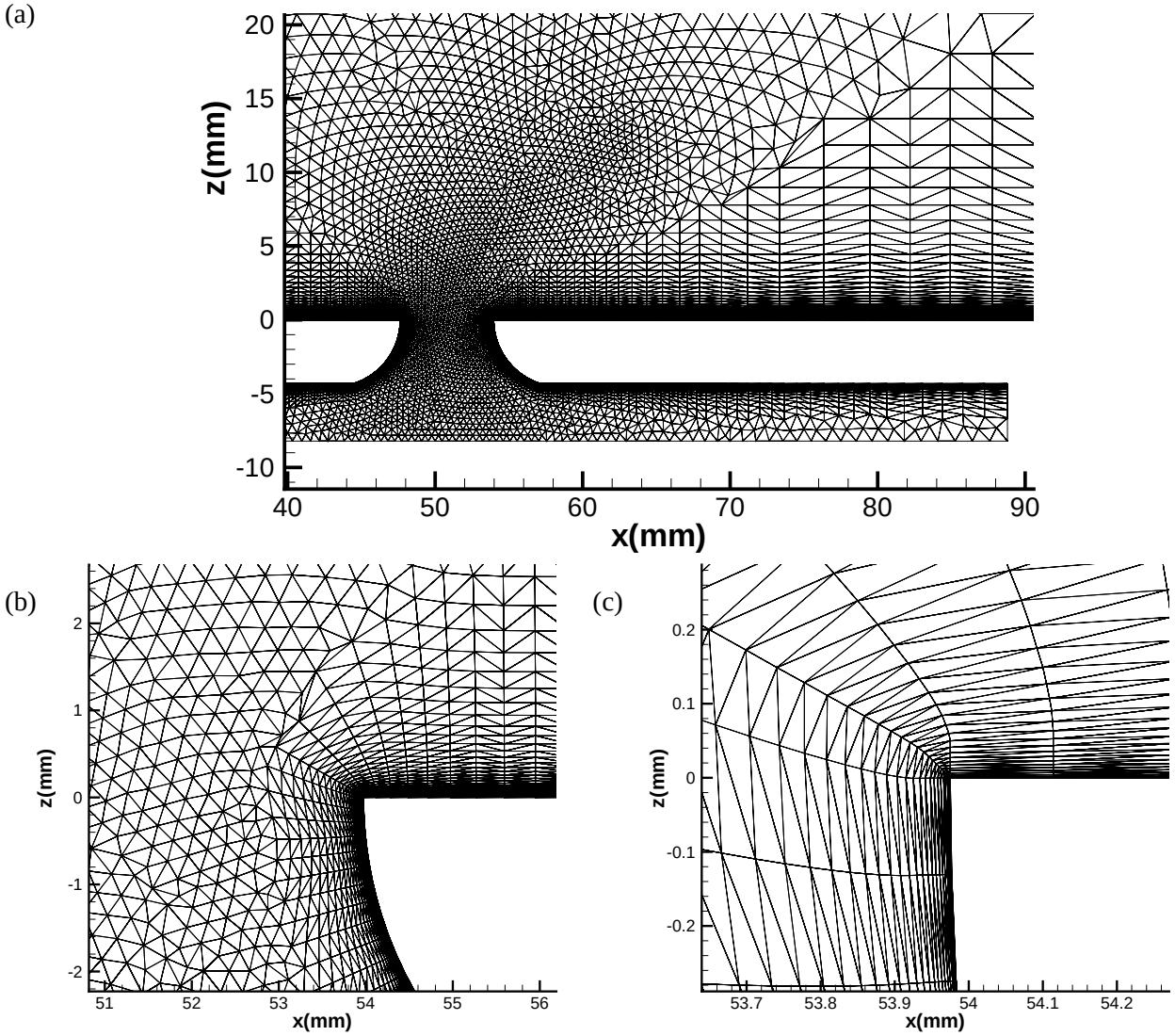

Figure 1: Grid on the symmetry plane: a) overall view, b) near jet exit, c) near corner of jet exit.

floor, the top wall of the cavity, and in the jet contraction. At these no-slip boundaries, the temperature is set to the adiabatic wall temperature, and the density is determined from the continuity equation. The side walls of the cavity are treated as inviscid walls, which FUN3D enforces weakly. The in/out flow boundary condition described in the previous section is applied on the lower wall of the cavity.

The inflow and initial conditions in the tunnel region were generated by performing a separate boundary layer simulation with FUN3D and extracting the solution at the appropriate Reynolds number. Initially, there is no flow in the cavity and the pressure is set to the freestream value. The simulation was started impulsively with the diaphragm velocity (normalized by u_∞) specified as $-0.007\cos(\omega t)$. The simulations used 720 time steps per period, with results saved every 5 degrees of phase. The long time averages requested by the workshop were computed by averaging these 5-degree samples. The fine grid simulations required 35 hours per period when using 16 intel processors; the coarse grid simulations required less than 11 hours per period when using 8 processors.

The fine and coarse grid simulations produced noticeably different solutions in several respects. Figures 2 (a) and (b) show the fine grid solution history at a location 0.17mm upstream of the jet center. Although the vertical velocity component W settles quickly to a periodic solution, the streamwise velocity

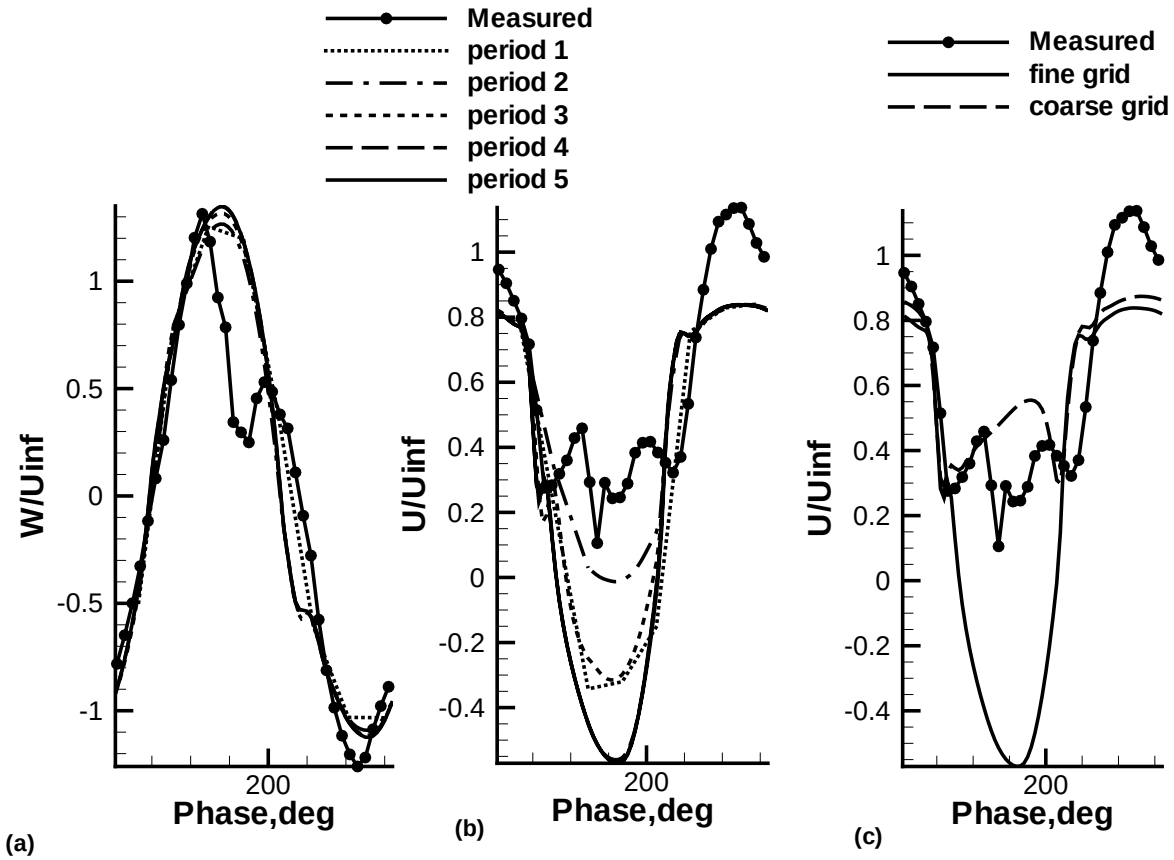

Figure 2: Solution at 0.17mm upstream of jet center: a) and b) fine grid startup history of W and U, respectively; c) comparison of U.

component U requires 4 periods. In the coarse grid simulation, the solution requires even longer, 6 periods, before the U velocity component becames periodic. The fine and coarse grid simulations give similar results for W, but the U velocity components, compared in fig. 2(c), have completely different character during the blowing phase. The coarse grid produces results similar to the experiment, while the fine grid predicts a large negative streamwise velocity component during the blowing cycle. The contour plots of U, shown in fig. 3, indicate that this negative streamwise velocity is due to a vortical behavior that develops in the jet exit flow.

Acknowledgments

The author would like to acknowledge Bill Jones and Mike Park for their assistance and suggestions with grid generation, and Bob Biedron, Beth Lee-Rausch and Eric Nielson for their assistance and advice in modifying and running FUN3D.

References

[1] Anderson, W. Kyle and Bonhaus, Daryl L. “An Implicit Upwind Algorithm for Computing Turbulent Flows on Unstructured Grids,” *J. Computers and Fluids*, Vol. 23, No. 1, pp. 1–21, 1994.

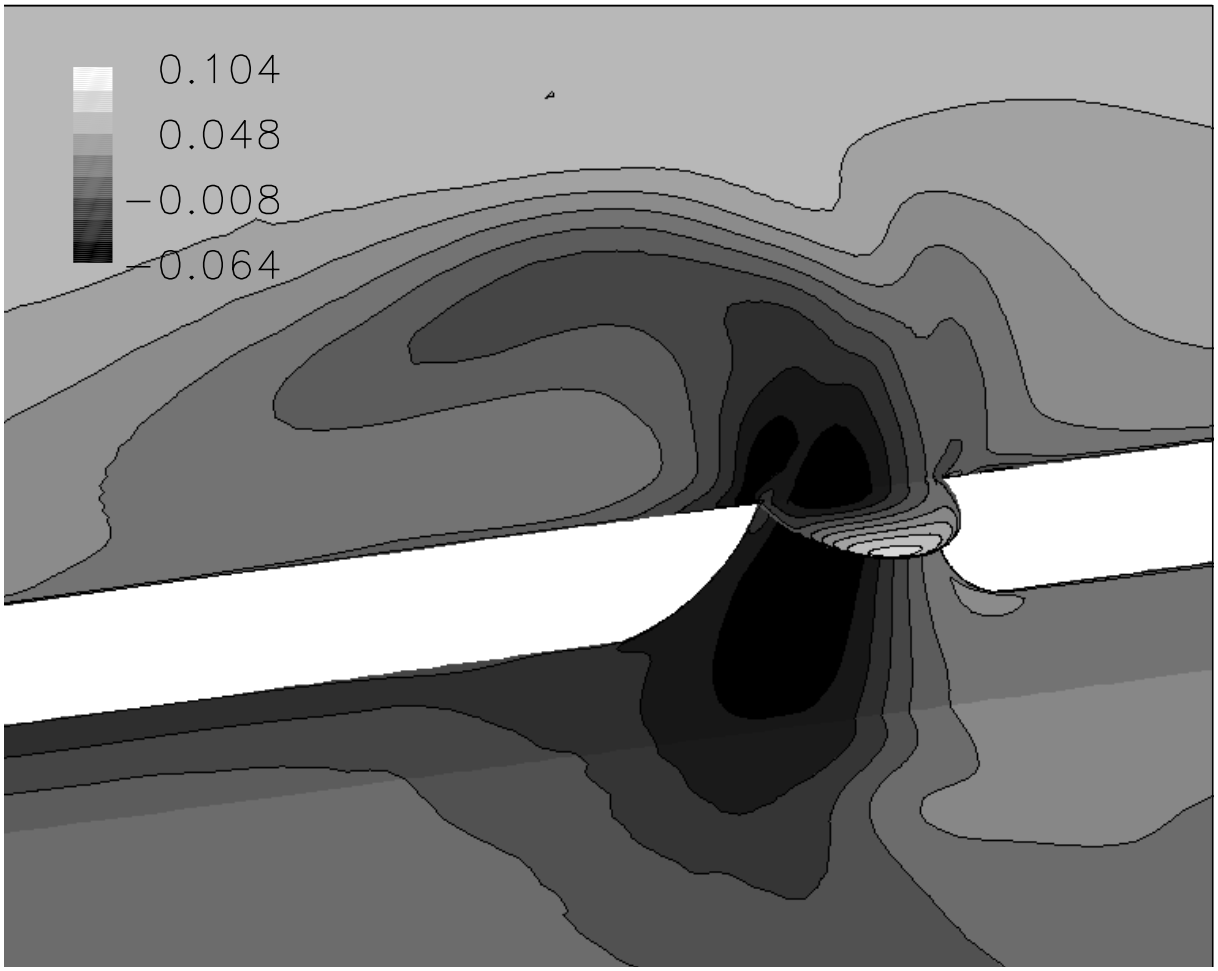

Figure 3: Streamwise velocity contours from fine grid simulation during peek blowing: phase = 160.

[2] Anderson, W. K. and R. D. Rausch and Bonhaus, D. L. ”Implicit/Mulitgrid Algorithms for Incompressible Turbulent Flows on Unstructured Grids,” *J. Computational Physics*, Vol. 128, pp. 391-408, 1996.

[3] Neilson, Eric, J. and Lu, James and Park, Mike A. and Darmofal, David L “An Exact Dual Adjoint Solution Method for Turbulent Flows on Unstructured Grids,” AIAA paper 2003-0272, 2003, to appear in *J. Computers and Fluids*.

[4] Lee-Rausch, E. M. and Mavriplis, D. J. and Rausch, R. D. “Transonic Drag Prediction on a DLR-F6 Transport Configuration Using Unstructured Grid Solvers,” AIAA Paper 2004-0554, 2004.

[5] Spalart, P. R. and Allmaras, S. R. “A One-Equation Turbulence Model for Aerodynamic Flows,” AIAA paper 92-0429, 1992.

[6] Jones, William T. “GridEx - An Integrated Grid Generation Package for CFD,” Proceedings of the 16th AIAA Computational Fluid Dynamics Conference, AIAA Paper 2003-4129, 2003.

[7] Park, Micheal A. “Three-Dimensional Turbulent RANS Adjoint-Based Error Correction,” AIAA paper 2003-3849, 2003.