JOURNAL OF AIRCRAFT
Vol. 36, No. 5, September–October 1999

Multiple-Body Trajectory Calculations Using the Beggar Code

Nathan C. Prewitt*
QuesTech, Inc., Eglin Air Force Base, Florida 32542
Davy M. Belk†
U.S. Air Force Research Laboratory, Eglin Air Force Base, Florida 32542
and
Raymond C. Maple‡
U.S. Strategic Command, Offut Air Force Base, Nebraska 68113

The Beggar code uses blocked and overset grid techniques and combines the grid assembly, flow solution, and six-degree-of-freedom trajectory calculation processes into a single code with reduced user input. Its capabilities to handle moving body problems are demonstrated through a series of store separation trajectory calculations. Single-store separation trajectory calculations at transonic Mach numbers are compared with previously published wind-tunnel data. Supersonic store separation predictions are compared with previously unpublished wind-tunnel data. The generality and ease of use are further illustrated by a three-store ripple release simulation.

Nomenclature

Cp = pressure coefficient
p, q, r = rotation rates
u, v, w = relative velocities
x, y, z = relative c.g. positions

Introduction

THE safe carriage and separation of stores from fighter aircraft is of primary importance to the U.S. Air Force. This process can be modeled using engineering-level methods with experimentally or computationally determined aerodynamic influence functions.[1] These methods often fail to give accurate results when the flow perturbation due to the store is large or when nonlinearities due to shocks, vortical flow, or viscosity dominate. For these cases, wind-tunnel experiments using captive trajectory system (CTS) are often used. This process involves the use of a sting-mounted store placed in the flowfield of the wing and pylon. Forces and moments on the store are measured and the equations of motion are integrated online to move the store. Any ejector or other applied loads are modeled online. Because the trajectory calculation is not done in real time, the flowfield becomes quasisteady at each position of the store. The apparent angle of attack due to store velocity is accounted for by modifying the incidence angle. As shown by Belk et al.,[2] the unsteady aerodynamic forces can be quite different from those obtained by simply modifying the incidence angle when lateral velocities are large. However, CTS-generated trajectories are generally considered to be accurate at typical store ejection velocities.

An alternative to the wind tunnel is the use of computational fluid dynamics (CFD) coupled with a six-degree-of-freedom (6DOF) integration of the rigid-body equations of motion. This approach has several motivations: 1) flow visualization from CFD-based predictions can give insight needed to solve problems, 2) time-accurate aerodynamic loads are calculated, 3) geometry modifications are relatively easy to implement, and 4) long lead times associated with scheduling test facilities and fabricating hardware can be avoided.

Negatives associated with CFD-based methods include 1) difficulty in developing and coupling CFD and 6DOF codes for multibody problems, 2) lack of validated methods, and 3) large computational resources required. Item 1 has been addressed by Maple and Belk[3] and Belk and Maple,[4] where they presented the Beggar code. The Beggar code uses a unique combination of blocked and embedded grids to ease CFD modeling for multiple-body problems, and also combines the 6DOF, moving mesh-handling routines and flow solver into a single code with a single set of user inputs required. The current work addresses item 2 by presenting validation calculations comparing the Beggar code results with wind-tunnel CTS trajectories.

Moving-body trajectory prediction has been performed using unstructured grids[5] and using overset grids.[6–9] The overset methods are more closely related to the current approach, although none use the unique blocked and embedded grid data structure employed by the Beggar code. The overset methods typically use four distinct codes to 1) assemble the overset grid system, 2) calculate the flow solution, 3) integrate the surface pressures to obtain forces and moments, and 4) integrate the equations of motion and move the store. In contrast, the Beggar code integrates all four parts into one code, greatly simplifying usage.

A series of wind-tunnel tests sponsored by Wright Laboratory Armament Directorate were performed at Arnold Engineering Development Center on a generic wing, pylon, and store configuration[10] for the purpose of verifying the accuracy of CFD codes at simulating store carriage and separation. The data set that was produced includes pressure data and trajectory data for a single-finned store at transonic (Mach 0.95) and supersonic (Mach 1.2) conditions. Both cases will be compared with the Beggar code results for validation purposes. The transonic results have been previously published, e.g., Refs. 6 and 7, but the supersonic trajectory results are presented here for the first time.

In addition to the single-store trajectories, a three-store ripple release will be simulated that is similar to that presented by Thoms and Jordan.[7] Unfortunately, experimental data do not exist for this case, but it serves to demonstrate the ease and efficient operation of the Beggar code.

Grid Generation

One of the most time-consuming tasks of any CFD calculation is the grid generation. Block-structured grids require that the solution domain be decomposed into a set of nonoverlapping grids with point continuity between grids. These restrictions make the generation of

Received 2 July 1997; revision received 2 December 1998; accepted for publication 26 January 1999.

*Research Engineer; currently Research Engineer, Advanced Technologies and Studies Support Department, CACI Technology Services. Senior Member AIAA.

†Technical Advisor, Computational Mechanics Branch, Assessments and Demonstrations Division. Associate Fellow AIAA.

‡Research Engineer, USSTRATCOM/J665; currently Research Engineer, Office of the Inspector General, Headquarters, Air Force Materiel Command, Wright–Patterson Air Force Base, OH 45433.

the grid system more difficult for complex geometries; however, once generated, the block-to-block connections can be established easily.

A simplification of blocked grids has been termed patched grids. In a patched-grid system, the domain is decomposed into a set of nonoverlapping grids, but point continuity is no longer required between adjacent grids. This can simplify the grid generation, but communication must now be established through interpolation.

Overset grids allow the solution domain to be decomposed into multiple-structured grids without any requirements of point or slope continuity between grids. Such grid systems are a practical choice for moving body problems because grids are allowed to overlap and proper communication is established by interpolation. The relaxation of the requirements for continuity between grids makes the generation of the complete grid system easier. However, the setup of the proper communication between grids can be difficult.

Because overset grids may overlap regions that are outside the flow domain (such as inside a solid surface), an operation commonly called "hole cutting" must be performed to identify invalid regions and to mark them as holes. These holes create additional boundaries at which the flow solver requires some type of boundary condition (BC). These BCs are supplied through the interpolation of flowfield data. As grids move relative to one another, this process of marking holes and interpolating flowfield data must be repeated.

The Beggar code has the capability of utilizing overset, blocked, and patched grids. The use of blocked grids can reduce the number of points required to decompose a domain and can reduce the amount of work required to set up grid communications.

The specification of block-to-block connections for blocked grids traditionally has taken many user inputs. Likewise, the calculation of interpolation stencils for overset and patched grids, although automated to a large extent, also can take many user inputs to make the problem more manageable and efficient. Beggar automates these grid-assembly functions and thus greatly decreases the amount of user input required.

Figure 1 shows the grid system used in the present study. There are a total of eight grids. The store is made up of a blocked grid system with one grid between each pair of fins. In the Beggar code terminology, this set of blocked grids is called a "superblock." The grids in the superblock communicate with each other via block-to-block connections and with the other grids via interpolation. The pylon and the wing grids represent other superblocks with blocked grid systems. An interface grid is used to enhance communication between the other grids. The total number of points in this grid system is 1.35×10^6.

Problem Specification

The automation of the grid-assembly process greatly reduces the number of user inputs required to specify a given problem. Input is required to read in the grids and to specify the solid surfaces within those grids; however, once this is done, Beggar will automatically identify all block-to-block connections, patched connections, singularity BCs, and freestream BCs. All holes are cut automatically using the solid surfaces and all required interpolation stencils are calculated. The reflection plane BCs are specified through a single user input denoting a point and normal vector. Additional inputs are needed only to specify the flow solution parameters, i.e., Mach, Courant–Friedrichs–Lewy, flux limiter, etc., force and moment calculations, and the 6DOF integration parameters.

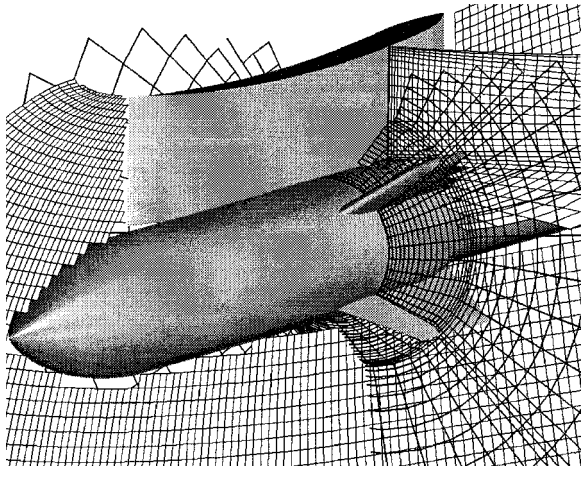

Fig. 1 Blocked store and pylon grids overlapping wing and interface grids.

A useful construct for moving body problems is the dynamic group. Dynamic groups allow one or more superblocks to move as a single entity. The forces and moments needed to drive the motion of the dynamic group are measured by a single "force spec," detailing which surfaces to integrate forces and moments over. Additional inputs needed for the 6DOF calculation are the mass and inertia properties of the dynamic group and a set of dimensional quantities detailing the gravitational force, freestream conditions, and reference lengths.

For the single-store trajectory calculations, the blocked store grid is a single superblock and is associated with its own dynamic group. The solid surface BCs of the store are associated with a force spec that in turn is associated with the store dynamic group. The nondimensional reference length and area are defined in the force spec, whereas the store's mass and inertia properties are defined in the dynamic group. The time of release and ejector forces, duration, and location are also specified with the dynamic group.

The inputs for a multiple-store release are similar to those for a single-store release. The same grids are read multiple times to define multiple superblocks. Each grid is positioned using an initial translation. To cause each store to move independently, each store superblock is placed in its own dynamic group and is driven by its own force spec. A ripple release is created by varying the store release times and the ejector forces.

6DOF Integration

In an effort to reduce the time required to perform the grid assembly, store trajectories have been calculated with the interpolation stencils updated only once for every 20 iterations of the flow solver and the 6DOF.[6,7] This is justified if no grid moves by an appreciable amount during this time, and, therefore, the interpolation stencils are still valid. The combination and efficient implementation of the grid-assembly process with the flow solver, force and moment integration, and 6DOF calculation in Beggar obviates these concerns.

The 6DOF integration of the rigid-body equations of motion is similar to that used in Ref. 9. Quaternions are used to track the state of each dynamic group. After each flow solver step, the forces and moments are integrated. The state of each dynamic group is then updated using a fourth-order Runge–Kutta method. To facilitate the grid-assembly process, transformations between the local coordinate systems of each of the dynamic groups are maintained. These transformations are used to transform all points requiring interpolation stencils to the coordinate system of any overlapping grids. This means that the data structures built for the initial grid assembly can be reused without alteration.

Flow Solver

The basic solution algorithm used in the Beggar code is a Newton–relaxation scheme,[11] used to solve the compressible Euler and Navier–Stokes equations. Equation (1) represents the flow solver main iteration loop:

$$\left[\frac{1}{\Delta t_l} + \left(\frac{\partial R}{\partial \boldsymbol{Q}}\right)^{n+1,m}\right]\Delta \boldsymbol{Q}^{n+1,m+1} = -\left[\frac{\boldsymbol{Q}^{n+1,m} - \boldsymbol{Q}^{n}}{\Delta t_{\min}} + R(\boldsymbol{Q}^{n+1,m})\right] \tag{1}$$

where

$$\boldsymbol{Q}^{n+1,m+1} = \boldsymbol{Q}^{n+1,m} + \Delta \boldsymbol{Q}^{n+1,m+1}, \qquad \boldsymbol{Q}^{n+1,0} = \boldsymbol{Q}^{n}$$

$\boldsymbol{Q}$ is the dependent variable vector, R is the flux imbalance, n denotes the time level, m is the Newton iteration counter, Δt_l is the local time step, and $\Delta t_{\min}$ is the minimum time step. A finite volume discretization is used that is first, second, or third order in space, and first order in time. Flux difference splitting with Roe-averaged variables is used on the right-hand side (RHS). MUSCL extrapolation is used to calculate the flux at cell faces and can be applied on the primitive or the conserved variables. Flux extrapolation is also available for calculating higher-order fluxes at the cell faces. Flux limiters that are available include minmod, van Leer, and van Albada. Steger–Warming jacobians, Roe analytical jacobians, or Roe numerical jacobians can be used on the left-hand side.

Explicit BCs can be used or implicit BCs can be achieved by updating the BCs during the symmetric Gauss–Seidel relaxation solution of Eq. (1).[12] An underrelaxation factor is applied to the implicit BC update to improve stability. The SGS iterations, or inner iterations, are performed on a grid-by-grid basis; whereas the Newton iterations, or dt iterations, are used to achieve time accuracy and are performed on all grids in sequence. This procedure eliminates synchronization errors at blocked and overset boundaries by iteratively bringing all dependent variables up to the t^{n+1} time level.

Steady-state calculations do not use Newton iterations. The first term on the RHS of Eq. (1) becomes zero, and local time stepping is used during the inner iterations. For the time-accurate calculations, a fixed time step $\Delta t_{\min}$ is used to maintain time accuracy, and a local time step Δt_l is used for stability and convergence of the Newton iterations.

The present calculations represent inviscid flow. Unless otherwise noted, the code was run second order using Roe analytical jacobians, primitive variable MUSCL extrapolation, and the van Albada limiter. Implicit BCs are used with six inner iterations and a relaxation factor of 0.6. Two dt iterations are used.

Results

Single Store

Carriage

Figures 2–5 present Cp distributions along the store while in carriage position. Beggar solution data for both the Mach 0.95 and Mach 1.2 cases are compared with wind-tunnel data. These plots show good agreement between the solution generated by Beggar and the wind-tunnel data. The greatest discrepancies are seen in the expansion regions on the upper surface at the back of the store and the lower surface at the ogive–cylinder juncture. Only slight errors in shock location and strength are noticeable. The midbody shocks are consistently displaced aft of the experimental results. These differences can be attributed to viscous effects that are not modeled in the present calculations. Also note that the wind tunnel used a sting, but the current model does not. In general, the carriage solutions compare well with available data and are considered good starting solutions for the trajectory simulations.

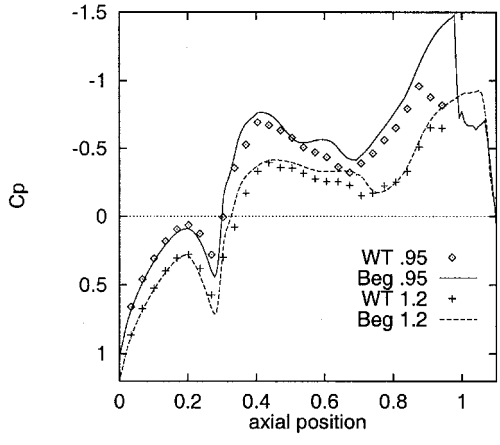

Fig. 2 Carriage *Cp* distributions on store upper surface.

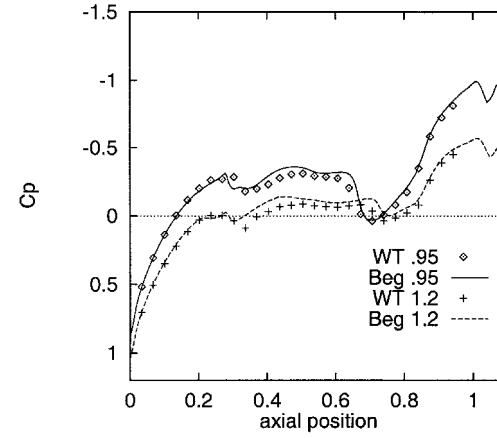

Fig. 3 Carriage *Cp* distributions on store outboard surface.

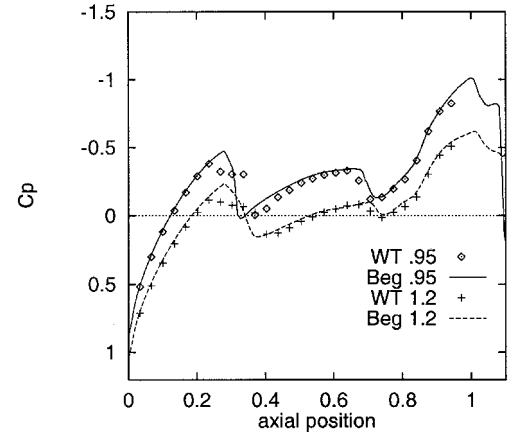

Fig. 4 Carriage *Cp* distributions on store lower surface.

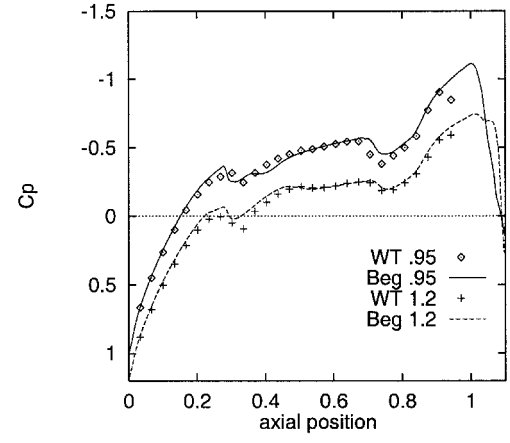

Fig. 5 Carriage *Cp* distributions on store inboard surface.

The Mach 1.2 distributions show a general shift toward higher Cp, and shock locations are shifted farther downstream when compared with the Mach 0.95 case. Because the bow shock created by the relatively blunt leading edge of the wing passes in front of the store, both cases see subsonic flow at the nose of the store. The stronger bow shock for the Mach 1.2 case accounts for the uniform shift in Cp. The shocks seen along the store are due to the acceleration of the flow around the store, pylon, and fin geometry. The forward shock on the upper, inboard, and outboard surfaces of the store is dominated by the position of the pylon leading edge; therefore, its position does not vary significantly with Mach number. However, on the lower surface of the store, the forward shock shows a significant shift in position. The aft shocks are generated by the leading edge

of the fins, and, thus, the shock positions, as seen midway between the fins, show the appropriate shifts with Mach number.

Trajectories

The Mach 0.95 case was run with atmospheric conditions for 26,000 ft altitude, whereas the Mach 1.2 case was run using conditions at 38,000 ft. The store modeled is a 2000-lb bomb with dimensions and mass properties equivalent to those presented in Ref. 7. The same grids were used for both simulations. Both trajectories were computed using a time step corresponding to 0.0005 s. The Mach 0.95 case was run using conservative variable extrapolation, but problems were encountered during the Mach 1.2 solution. Primitive variable extrapolation proved to be more robust and therefore was used for the Mach 1.2 and multiple-body solutions.

Figures 6 and 7 present results from the Mach 0.95 separation trajectory. The coordinate system used is centered at the carriage position of the store c.g. The positive x axis points forward out the store nose, the positive y axis points out the right wing, and the positive z axis points downward. Yaw (PSI) is measured positive nose outboard, pitch (THA) is positive nose up, and roll (PHI) is positive lugs outboard for a store placed on the right wing. The c.g. positions agree well with the experimental data, with the axial position displaying the most discrepancy. This is expected because the viscous drag force is not calculated accurately by an Euler code. The angular positions also show good agreement with wind-tunnel data, although they are more difficult to reproduce accurately due to inaccuracies in the definition of the fin tips and the discretization error in the integration of the forces and moments.

The minor differences in the pitch angle predicted may be due to the lack of a sting in the present model. Initial freestream calculations indicated that the flow was being sufficiently straightened by the fins so that no flow asymmetries would be encountered near the junction of the store and sting. Closer examination of the carriage flow solution revealed significant flow asymmetries in the base flow region due to interference from the wing and pylon. A sting may be needed to accurately reproduce the wind-tunnel results.

Figures 8 and 9 show the position of the c.g. and the angular displacements of the store during the Mach 1.2 trajectory calculation. The position of the c.g. agrees well with the wind-tunnel data with the largest discrepancy again being with the axial position. The angular displacements also agree well with yaw, pitch, and roll all being within 1 deg of the wind-tunnel data after the 0.3-s trajectory.

Figures 10 and 11 show the Mach 1.2 solution results of store c.g. velocity (u, v, and w) and angular rates (p, q, and r). The large downward velocity and nose-up pitch rate are due to the ejectors.

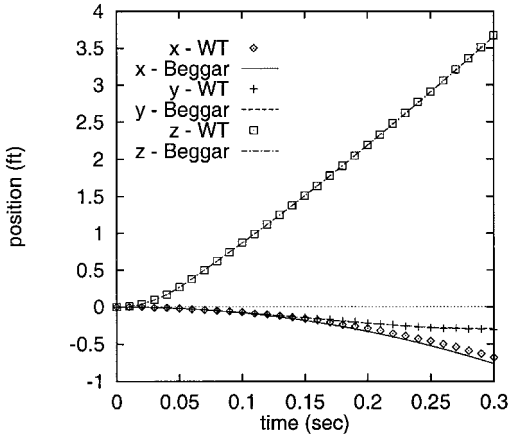

Fig. 6 Mach 0.95 trajectory c.g. positions.

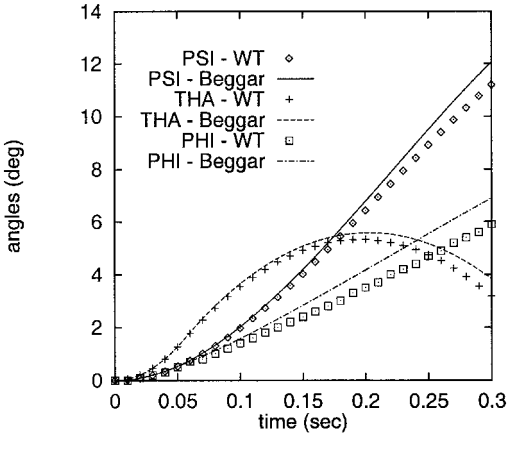

Fig. 7 Mach 0.95 trajectory angular positions.

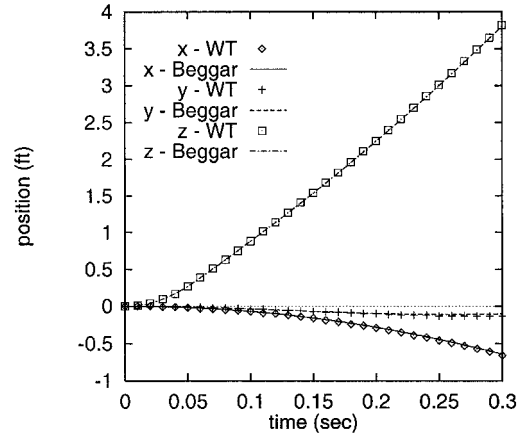

Fig. 8 Mach 1.2 trajectory c.g. positions.

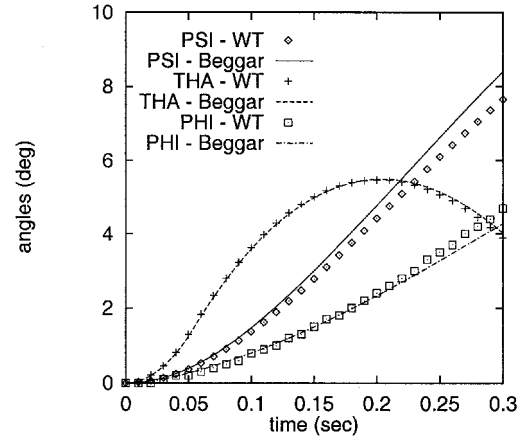

Fig. 9 Mach 1.2 trajectory angular positions.

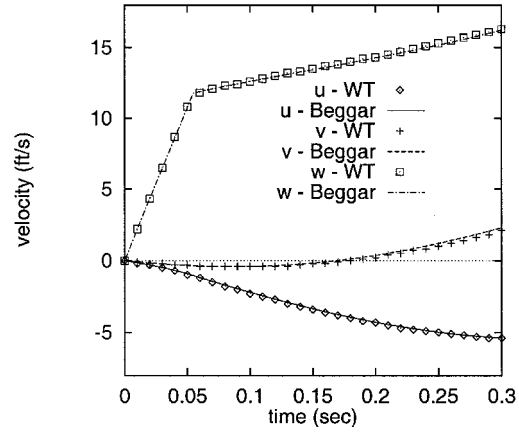

Fig. 10 Mach 1.2 trajectory velocities.

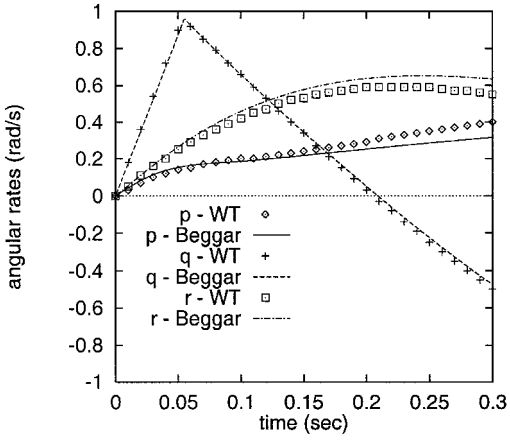

Fig. 11 Mach 1.2 trajectory angular rates.

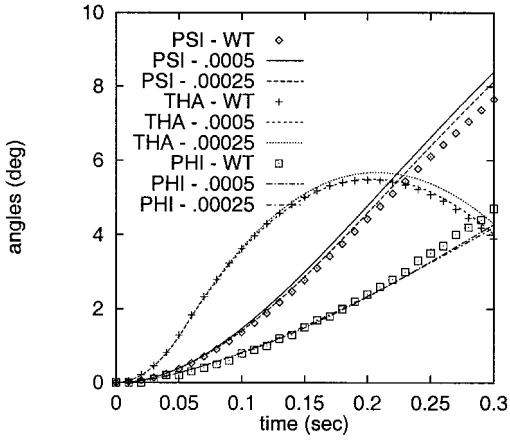

Fig. 12 Mach 1.2 time-refinement study.

The change in slope of these curves indicates the point where the ejectors were turned off. These plots also show that the store initially drifts inboard before moving outboard and moves downstream while rolling outboard and yawing outboard.

Approximately two-thirds of the way through the trajectory, the solver failed due to a large recirculating region created by flow along the span of the wing trying to turn the sharp corner at the bottom of the pylon. Because the store was sufficiently separated from the pylon, the solution was restarted with the pylon solver options set to first order. This added enough dissipation to allow the solution to continue and produced no visible effects on the trajectory of the store.

To help verify the results for the Mach 1.2 case, the trajectory was also run with 3 dt iterations and a 0.00025-s time step. This represents a refinement of the time integration. Figure 12 shows the resulting angular displacements of the store plotted with the wind-tunnel data and the original solution results. The yaw angle moved closer to the wind-tunnel data while the pitch angle moved farther away from the wind-tunnel data and the roll angle remained virtually unchanged. Because the changes were less than 0.5 deg after the 0.3-s trajectory, the 0.0005-s time step with 2 dt iterations seems adequate for producing acceptable results.

Multiple Stores

For this case, the configuration presented by Thoms and Jordan[7] is reproduced. The stores are positioned as if loaded on a triple-ejector rack and are released in a bottom–outboard–inboard sequence with a 0.04-s delay between each release. The release conditions are for Mach 0.95 at 26,000 ft altitude. All stores have equal mass and inertia properties. Ejector forces are equivalent to those used in the single-store calculations, with the exception that the shoulder stores' ejectors are placed at 45-deg angles with respect to vertical. Because the single-store trajectories were dominated by gravitational and ejector forces, the store mass and moments of inertia were each decreased by a factor of 2 in an effort to increase the effects of the aerodynamic forces.

To ensure grid assembly once the shoulder stores started moving, fin-tip interface grids were added for the two fins closest to the pylon. Two interface grids were also added to extend the outer boundaries of the shoulder stores. These grids were needed because the wing/pylon interface grid, which was originally generated for the single-store trajectories, was not large enough to cover holes cut in the wing grid as the shoulder stores move downward.

The trajectory was run using the same 0.0005 time step that was used in the single-store trajectories. Figure 13 shows a time history trace of 0.32 s of the three-store ripple release.

Figures 14–19 present the c.g. location and angular position of the three stores during the ripple release trajectory calculation. The origin of the coordinate system is at the initial position of the c.g.

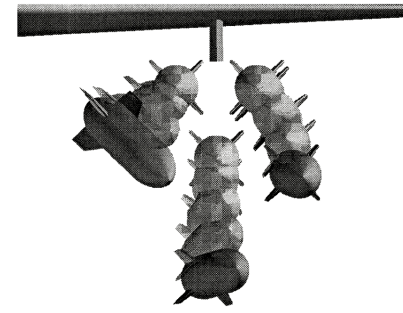

Fig. 13 History of three-store separation.

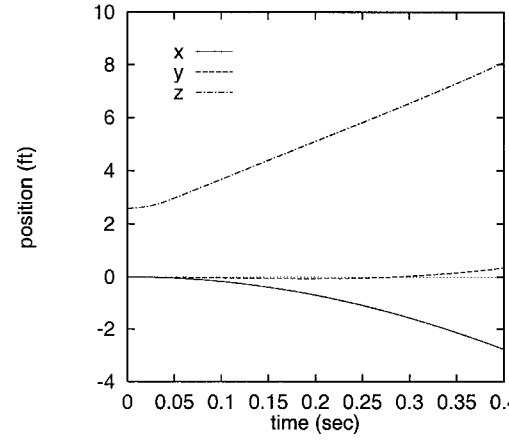

Fig. 14 Bottom store c.g. positions.

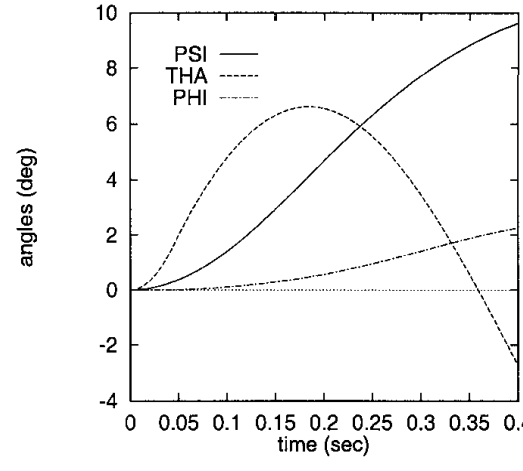

Fig. 15 Bottom store angular positions.

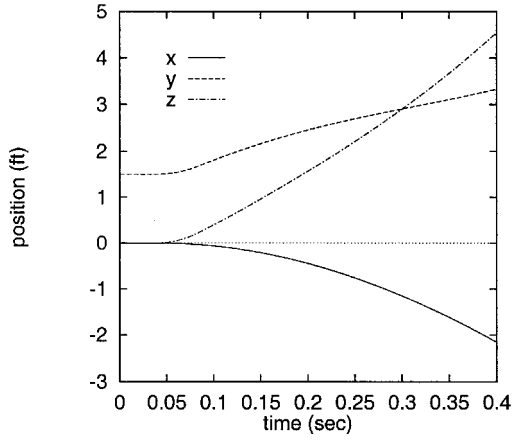

Fig. 16 Outboard store c.g. positions.

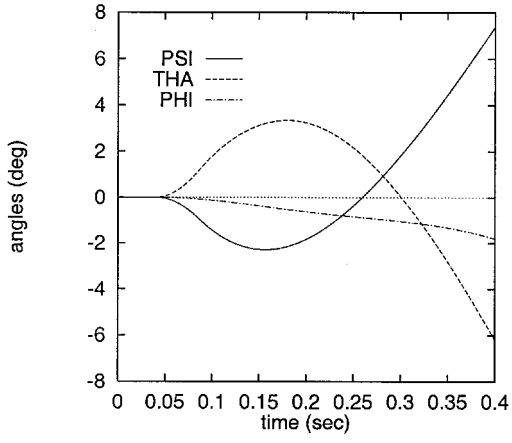

Fig. 17 Outboard store angular positions.

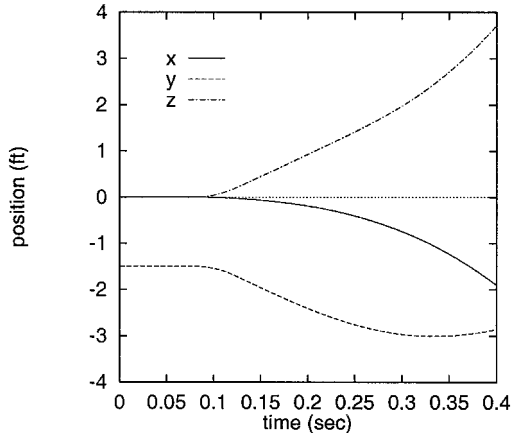

Fig. 18 Inboard store c.g. positions.

for the single-store trajectories; thus, the bottom store is initially displaced in the positive z direction and the shoulder stores are displaced in the y direction.

The results are qualitatively similar to those presented by Thoms and Jordan.[7] A comparison of the carriage force and moment coefficients to those published by Thoms and Jordan shows good agreement with the largest discrepancies being a sign change on the normal force coefficient of the outboard store and a 30% smaller pitching moment for the outboard store. The inboard store also had a 10% smaller pitching moment and a 10% larger yawing moment.

The difference in the small normal force coefficient has relatively no effect on the position of the outboard store; however, the smaller nose-down pitching moment translates into a slightly larger maximum pitch angle and slightly longer time before the pitch angle goes

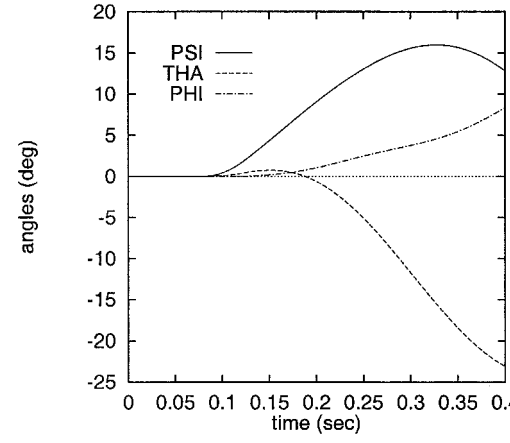

Fig. 19 Inboard store angular positions.

negative. The smaller nose-down pitching moment for the inboard store means that this store rotates slightly nose up before turning nose down; whereas Thoms and Jordan[7] showed the inboard store rotating nose down from the beginning of the release. The yaw angles of the inboard stores agree well despite the difference in the carriage yawing moments; however, the roll angles show significant differences despite good agreement between the carriage rolling moments.

Some of these differences may be attributed to the larger 0.0005-s time step used in the present study as compared with the 0.0002 s time step used by Thoms and Jordan. The difference in updating the interpolation stencils may also have some effect. Different grids can produce significantly different results; in particular, the fin tip regions appear to be important for this configuration. However, experimental data are not available for accurate comparison.

Timing Information

The results presented were run on several different machines including a Cray C90, a Cray Y-MP, and an SGI Onyx R8000. The single-store trajectories were initially assembled on the Cray Y-MP, and the unsteady flow calculations were completed on the Cray C90. This initial assembly took ~241.5 s of CPU time. After each flow solution step the grids were reassembled, taking an average of 52.2 s. The flow solution took ~273.5 s per iteration when using Roe analytical jacobians, 2 dt iterations, and implicit BCs with six inner iterations.

The three-store ripple-release case was run on the SGI Onyx R8000. This case took ~254 s for the initial assembly and an average of 119 s for each reassemble. Using Roe analytical jacobians, the flow solution took ~3640 s per iteration or 1.8e-3 s per cell per iteration. Changing to Steger–Warming jacobians reduced this to 1333 s per iteration or 6.8e-4 s per cell per iteration. This represents a 63% improvement in execution speed without any adverse effects on the trajectory.

The Beggar code has not been significantly optimized for vector machines and these execution times represent large computer resources when simulating large moving-body problems. However, note that the grid assembly times were relatively consistent across the different platforms and their cost is significantly less than the flow solution.

Conclusions

Single-store trajectory calculations were presented. The Mach 0.95 case was compared with previously published wind-tunnel data. The Mach 1.2 case was compared with previously unpublished wind-tunnel data. Both cases compared favorably with the available data.

A Mach 0.95 three-store ripple-release trajectory calculation was presented and was qualitatively similar to the results of Thoms and Jordan. The extension of the input for a single-store case to a multiple-store case was outlined.

Guidelines for the efficient operation of the flow solver were established. Two dt iterations and an unsteady time step of 0.0005 s were used successfully for these trajectory calculations. Implicit BCs with 4–6 inner iterations and a relaxation factor of 0.6 were also used. Primitive variable extrapolation was more robust than conservative variable extrapolation, and the reduction to first order in space was used cautiously to circumvent problems with inviscid flow around sharp corners. Significant execution time can be saved by using Steger–Warming jacobians during the unsteady flow calculations.

The Beggar code presents a usable and efficient environment for simulating moving body problems. The integration of all necessary functions simplifies the simulation process, whereas the automation and efficient implementation of the grid assembly process significantly reduces the amount of user input required and allows the interpolation stencils to be updated each time a grid is moved.

Acknowledgments

This work was performed under the funding and oversight of Wright Laboratory Armament Directorate and the U.S. Air Force Seek Eagle Office. Support was provided in part by a grant of computer time from the Department of Defense High Performance Computing Center, U.S. Army Corps of Engineers Waterways Experiment Station Cray C90. The authors thank L. E. Lijewski, R. D. Thoms, and J. K. Jordan for their cooperation; and Magdi Rizk for his work in enhancing the Beggar flow solver.

References

[1]Jordan, J. K., "Computational Investigation of Predicted Store Loads in Mutual Interference Flow Fields," AIAA Paper 92-4570, Aug. 1992.

[2]Belk, D. M., Janus, J. M., and Whitfield, D. L., "Three-Dimensional Unsteady Euler Equations Solution on Dynamic Grids," *AIAA Journal*, Vol. 25, No. 9, 1987, pp. 1160, 1161.

[3]Maple, R. C., and Belk, D. M., "Automated Set Up of Blocked, Patched, and Embedded Grids in the Beggar Flow Solver," *Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, edited by N. P. Weatherill, P. R. Eiseman, J. Häuser, and J. F. Thompson, Pineridge, Swansea, Wales, UK, 1994, pp. 305–314.

[4]Belk, D. M., and Maple, R. C., "Automated Assembly of Structured Grids for Moving Body Problems," AIAA Paper 95-1680, June 1995.

[5]Lohner, R., and Baum, J. D., "Three-Dimensional Store Separation Using a Finite Element Solver and Adaptive Remeshing," AIAA Paper 91-0602, Jan. 1991.

[6]Lijewski, L. E., and Suhs, N., "Time-Accurate Computational Fluid Dynamics Approach to Transonic Store Separation Trajectory Prediction," *Journal of Aircraft*, Vol. 31, No. 4, 1994, pp. 886–891.

[7]Thoms, R. D., and Jordan, J. K., "Investigation of Multiple Body Trajectory Prediction Using Time Accurate Computational Fluid Dynamics," AIAA Paper 95-1870, June 1995.

[8]Yen, G.-W., and Baysal, O., "Dynamic-Overlapped-Grid Simulation of Aerodynamically Determined Relative Motion," AIAA Paper 93-3018, July 1993.

[9]Meakin, R. L., "Computations of the Unsteady Flow About a Generic Wing/Pylon/Finned-Store Configuration," AIAA Paper 92-4568, Aug. 1992.

[10]Lijewski, L. E., "Transonic Euler Solutions of a Wing-Pylon-Finned Body Configuration Using Blocked and Overlapped Grid Schemes," AIAA Paper 91-2854, Aug. 1991.

[11]Whitfield, D. L., "Newton-Relaxation Schemes for Nonlinear Hyperbolic Systems," Engineering and Industrial Research Station, Rept. MSSU-EIRS-ASE-90-3, Mississippi State Univ., MS, Oct. 1990.

[12]Rizk, M. H., "The Use of Finite-Differenced Jacobians for Solving the Euler Equations and for Evaluating Sensitivity Derivatives," AIAA Paper 94-2213, June 1994.