

Abstract of Paper Proposed for Presentation at the
22nd AIAA Applied Aerodynamics Conference
Providence, RI
16–19 August, 2004
**Automated Parameter Studies Using a Cartesian
Method**

Scott M. Murman*
ELORET
MS T27B
Moffett Field, CA 94035
smurman@nas.nasa.gov

Michael J. Aftosmis†
NASA Ames Research Center
MS T27B
Moffett Field, CA 94035
aftosmis@nas.nasa.gov

Marian Nemec‡
NASA Ames Research Center
MS T27B
Moffett Field, CA 94035
nemec@nas.nasa.gov

1 Introduction

Computational Fluid Dynamics (CFD) is now routinely used to analyze isolated points in a design space by performing steady-state computations at fixed flight conditions (Mach number, angle of attack, sideslip), for a fixed geometric configuration of interest. This “point analysis” provides detailed information about the flowfield, which aides an engineer in understanding, or correcting, a design. A point analysis is typically performed using high fidelity methods at a handful of critical design points, e.g. a cruise or landing configuration, or a sample of points along a flight trajectory.

*Senior Research Scientist, Member AIAA

†Research Scientist, Senior Member AIAA

‡NRC Research Associate, Member AIAA

Current research in CFD is aimed at extending the point analysis to a "set analysis", where CFD simulations are rapidly performed throughout the entire design space - varying both flight conditions and control surface deflections - in order to provide a broader picture of performance[1-3]. Such a set analysis is used both in initial design to quickly estimate performance, and in final design to augment traditional methods such as wind tunnel tests and aerodynamic modeling. The result of a set analysis is both a database containing the aerodynamic performance, and a broad view of critical flow features, such as shock strength and location. A typical CFD aerodynamic database generated from a set analysis currently will contain on the order of $10^4 - 10^6$ datapoints, depending upon the problem requirements. In order to manage such a large number of computations, automated tools are a necessity, not only for generating the data, but also harvesting meaningful results in a post process.

The current work uses a Cartesian, embedded-boundary method[4] to automate the generation of a vehicle aerodynamic parameter study. The Cartesian method provides an efficient and robust mesh generation capability which can handle an arbitrarily-complex geometry description. Recently, a method to generate the watertight surface triangulation required for Cartesian mesh generation directly from a CAD representation of the geometry has been developed[5]. This, combined with the Cartesian embedded-boundary method provides a robust and automatic mesh generation infrastructure which can be utilized through the design process. This meshing scheme is combined with a parallel, multi-level scheme for solving the steady-state Euler equations, either on shared- or distributed-memory architectures[6, 7].

The current abstract briefly describes a modular infrastructure built to automatically perform complex parameter studies about complex configurations which include movable control surfaces. This modular system is built around pre-existing standalone applications (mesh generator, flow solver, force/moment calculator, etc.), using scripts to provide a flexible "glue" between components. A general system of specifying and manipulating control surfaces using the Geometry Manipulation Protocol (GMP)[8] is presented which automatically re-generates appropriate rigid-body configurations from a geometry and control surface description. A sample aerodynamic database including 4700 datapoints for a prototype autonomous biomorphic explorer for Mars[9] is included as a demonstration in the current abstract. This CFD database has been used to develop a neural-network guidance and control system for future flight tests. The proposed paper will also include a discussion of issues related to generating a parameter study using CFD, including steering the process to cluster around critical points (and neglect benign areas), integrating the process with a 3rd-party software database storage (e.g., SQL, ODBC, ...), exploiting symmetry (true and approximate), and building and using multiple levels of fidelity (resolution, geometric, physical model, etc.). The current discussion focuses on determining stability derivatives from static geometry, however future work will also target the incorporation of dynamic derivatives to provide rate of change data[10].

2 Component Infrastructure

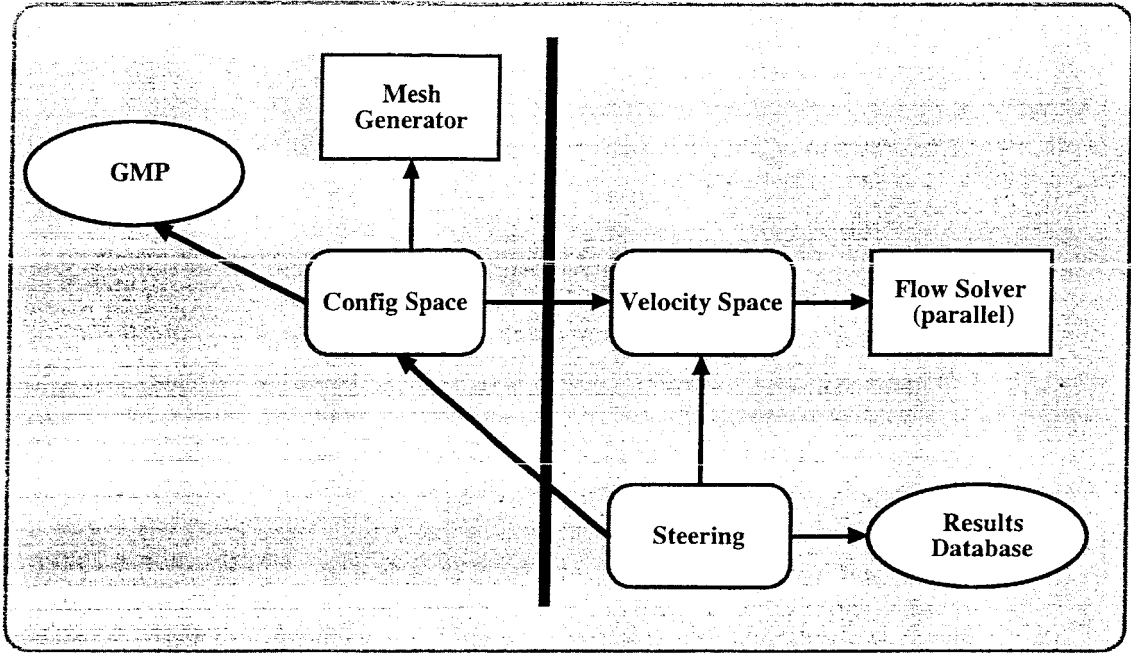


Figure 1: Schematic of the modular set of software components for performing an automated CFD parameter sweep. The stand-alone applications are shown in red, and the control scripts which glue the applications together in green.

Figure 1 shows a schematic of the modular set of software components used for building a CFD aerodynamic database. The stand-alone applications are shown in red, and the control scripts which glue the applications together in green. More details on the specifics of the infrastructure will be provided after a brief overview. Note that several low-level support scripts which are common to all of the tools are not described. The entire parameter space is decomposed into a two-level hierarchy. At the lowest level is a *velocity space* which contains the wind vector for each data point. This is characterized by the Mach number, angle of attack, and sideslip angle (M_∞, α, β). The next higher level contains a *configuration space* which represents all of the possible geometric configurations (control surface deflections, δ_c) being tested. A single element of the configuration space contains all elements of the velocity space. The rationale for this decomposition is the modular re-use of components. Each element of the velocity space uses a fixed computational mesh, and is thus independent of the configuration space, which requires a modified geometry and mesh for each element. In other words, a parameter study can be computed using the velocity space in isolation, or as a lower-level to a broader configuration space, without requiring modification of the velocity space software components or architecture. The *steering software* links the configuration space, velocity space, and the results database into

a coherent unit. The steering software monitors the results and determines which element of the parameter space to compute next, and this command is sent to the configuration space control script (or directly to the velocity space control script if only a single configuration is being examined). The configuration control then passes this to the velocity space. This hierarchy of command control continues through the process, with each control script being responsible for handling input and output from their isolated section of the process. This is similar to an object-oriented framework, though here a rigid object-oriented interface is not maintained.

Since the elements of the process infrastructure are independent, each can evolve in isolation. For example, the steering software can be as simplistic as iterating through a uniform parameter space following the matrix element ordering, or as complex as leveraging 3rd-party neural-network toolkits. This flexibility is key to providing a system which can grow as more experience is gained with building aerodynamic databases using CFD. Currently the steering software is intentionally straightforward. In the final paper a prototype of more intelligent decision-making tools will be provided.

The emphasis of the current work is on an automated process, hence the user interfaces (UIs) are at the highest level of the component hierarchy. The analyst is charged with providing inputs to the system, and harvesting results, however the control of the process is the purview of the script system. This is necessary, as managing or steering tens of thousands of computations by hand is not practical or desirable. The understanding is that if more details are required about a certain isolated point, or set of points, the analyst will perform a point analysis at those critical points.

One implicit assumption with the implementation of the infrastructure outlined in Fig. 1 is that there are no conflicts for CPU resources between the isolated parts of the process. In other words, it's assumed that the lightweight processes of generating a mesh, or adding an entry into the results database, will not interfere with the more compute-intensive flow solver processes. In practice this is not a restrictive assumption. Most compute resources either provide a front-end machine, or isolate a CPU or set of CPUs, which are responsible for job scheduling, interactive tty sessions, etc. Since the cost of mesh generation, etc. is amortized over thousands of flow solver runs, and these tasks run concurrent with the flow solver, these lightweight processes do not adversely impact the overall parallel efficiency.

2.1 Velocity Space

The implementation of the velocity space control script is straightforward. The control script sets the velocity space parameters which are inputs for the flow solver, creates any required directory and file structures, and then executes the solver. After the solver finishes, the aerodynamic coefficients are returned as results of the control script to the process which executed the velocity space script. These coefficients are selected by the user, and can contain items such as hinge moments, bending moments, etc., in addition to the 6 aerodynamic coefficients for the entire configuration. The

flow solver can execute either on a single processor or in parallel. Filling a parameter space involves computing many essentially identical problems, hence the parallelism is usually exploited on this coarser-grained level, not at the finer level of the flow solver, however there is nothing which prohibits combining both fine- and coarse-grained parallel strategies. The Cartesian solver does not require case-specific inputs for each set of flow conditions, which greatly simplifies the velocity space sub-system. The solver contains a hierarchy of robustness levels, with each increasing level requiring more computing resources. The run strategy simply starts with the least robust scheme, and proceeds up the hierarchy if a computation is numerically unstable. At the highest level of robustness it is always possible to maintain numerical stability, though it is still possible to generate spurious results, for example when computing an inherently unsteady problem with a steady-state method. In practice these pathological cases are relatively easy to filter, for example by monitoring convergence in residuals and computed aerodynamic loads. The velocity space control script is responsible for creating the flow solver runtime environment. Several interfaces are provided, including a simple interactive runtime process, job management directly through the portable batch system (PBS)[11], and an interface to the web-based AeroDB job control software[3]. Again, since the velocity space control scripts are self-contained they can evolve along with new runtime environments without effecting the remainder of the process.

2.2 Configuration Space

One powerful feature of the Cartesian method is the ability to automatically create a computational mesh about varying geometric configurations, as has been demonstrated for computing bodies in relative motion[12, 13] and aerodynamic shape optimization[14]. Moving control surfaces within a static configuration space also requires the ability to “re-mesh” changing geometric configurations. The complete set of control surfaces and their possible deflections, or orientations, makes up the configuration space. This configuration space is described using the ConfigSpace datatype from the GMP protocol. GMP is a set of rules and datatypes for manipulating geometry for CFD applications. Only a brief overview is provided here, and more information can be found in [8?]. Each ConfigSpace is defined by a set of Parameters (control surfaces), which are usually a single water-tight component (or groups of components) in the surface triangulation of the geometry. These Parameters take a finite number of discrete States, which are specified as rigid-body motions relative to the complete Configuration. The allowable rigid-body motions are extremely general and flexible. The GMP specification is stored in an XML file, which can be parsed by the application control script to determine the number of Parameters, how many States are specified for each Parameter, etc. The application control script for the configuration space thus looks very similar to the velocity space script, except that the matrix of parameters are control surfaces and deflections, rather than changes to the velocity vector. The ConfigSpace datatype also allows the specification of groups

of Parameters and States, for instance a coarse-, medium-, and fine-grained set of control surface deflections. A middleware layer between the GMP specification and the Cartesian applications actually does the work of translating the GMP Parameters and States into a unique geometric configuration.

2.3 Post-processing

Post-processing a large aerodynamic database usually employs two strategies: creating line or carpet plots of aerodynamic coefficients against the independent variables ($M_\infty, \alpha, \beta, \delta_c$), or viewing and analyzing pre-defined flow visualization results, such as surface contours or cutting planes, using a matrix visualization strategy. Either of these methods may be used in an interactive or batch-processing mode depending upon the needs. The current process has been used successfully with both approaches, and sample results will be presented in the next section.

One of the strengths of a modular process design is the ability to use 3rd-party tools when practical. These tools are used in many parts of the process in Fig. 1, e.g. PBS, GMP, etc., including limited use of database software packages based on SQL or the ODBC API. The storage and search requirements for a CFD aerodynamic database are relatively modest, so a tree-based database storage solution is not a necessity, however the use of a standardized database does have benefits. Since database APIs are static and supported by a large user community, a CFD database can be portable across many application domains rather than focused on a niche. For example, a database with an SQL or ODBC interface can easily be ported to web services. Similarly, a supported API allows different domains, such as G&C, structures, and aerodynamics to speak a common language, and build their individual tools on a known stable system. Further, the use of a database software package allows the results of a CFD aerodynamic database to evolve to include items such as flowfield images, documentation, or cutting planes, which would be inconvenient to implement into a simpler format.

Two methods of performing interactive flow visualization with a CFD database have been explored. The first uses the ParaVista software package[15] to view a matrix of parametric cases (c.f. Fig. 2). Here, rather than looking at a single solution, a range of solutions can be viewed and interrogated. ParaVista allows the database to be stored locally or accessed remotely through a web server portal. A similar idea is the NASA hyperwall of computer displays and graphics cards to interactively display a set of solutions[16]. The user interacts with software which controls the entire hyperwall of displays rather than multiple windows within a single display.

Another important aspect of utilizing a CFD aerodynamic database is performing a flight simulation by coupling a guidance system and 6-DOF package with the aerodynamic performance data. An example is presented in Fig. 3, where an aerodynamic model built from a CFD database and theoretical considerations is compared to a live-fire missile test. The trajectory predicted by the database is in excellent agreement with the flight-test data until the missile performs a high-g maneuver near time =

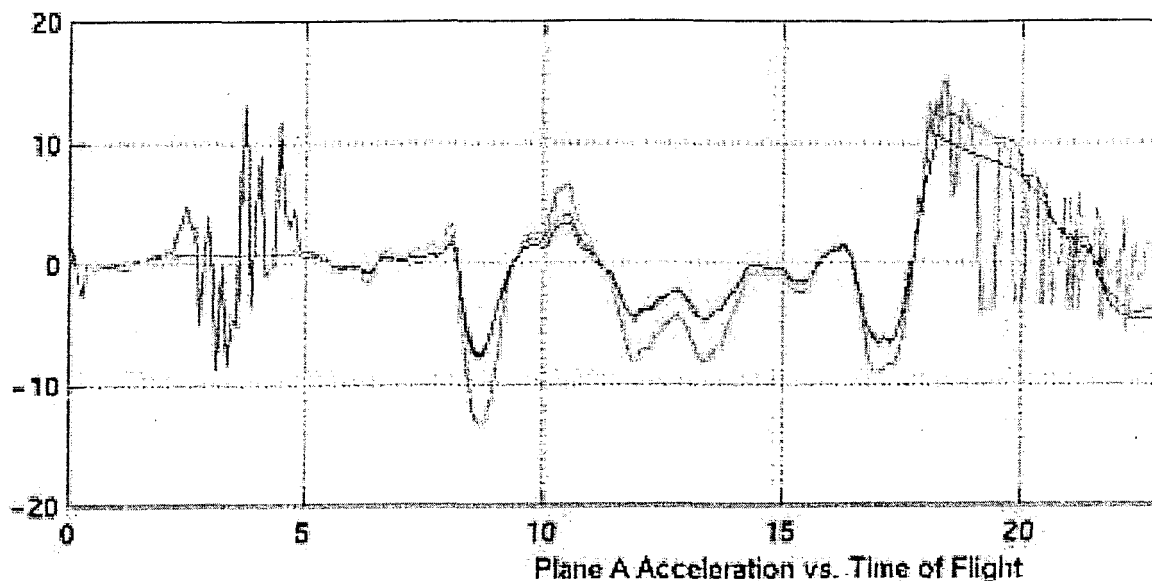


Figure 3: 6-DOF trajectory simulation using a CFD aerodynamic database compared to a live-fire missile test.

3 BEES Flyer Example

The bio-inspired engineering of explorations systems (BEES) flyer is envisioned as a small, autonomous platform with sensing and control systems mimicking those of biological systems, for scientific exploration on the surface of Mars[9]. The current BEES flyer is a delta-wing with twin vertical tails and two elevons which provide pitch and roll control of the aircraft (c.f. Fig. 4). Since the flying wing design is unstable in pitch, the control system must constantly provide stability by adjusting the elevons. A neural-network-based adaptive flight control system is being developed to provide this, however accurate stability and control (S&C) information is required in order to develop such a system.

The Cartesian method described in Sec. 2 was used to compute a parametric set of solutions over the expected flight domain. S&C derivatives were extracted from the database and integrated with the neural flight control system. Each elevon was positioned independently, and both were allowed to range over $[-10, 20]$ deg. of pitch with 7 discrete settings. Eight Mach numbers, six angles of attack, and two sideslip angles were computed for a total of 4704 datapoints and 5 independent parameters. This database is a prototype for use in initial development of the neural network system, and to provide an overview of the flow features. As the BEES flyer design is refined, so can the database be augmented to provide a more refined view of the aerodynamics near the expected design operating conditions.

The BEES flyer geometry was obtained from a CAD solid model, and is shown with a cutting plane through the computational mesh in Fig. 5. Each mesh contains approximately 1.5M cells, with a finest refinement of 1.5 mm. As the elevons change

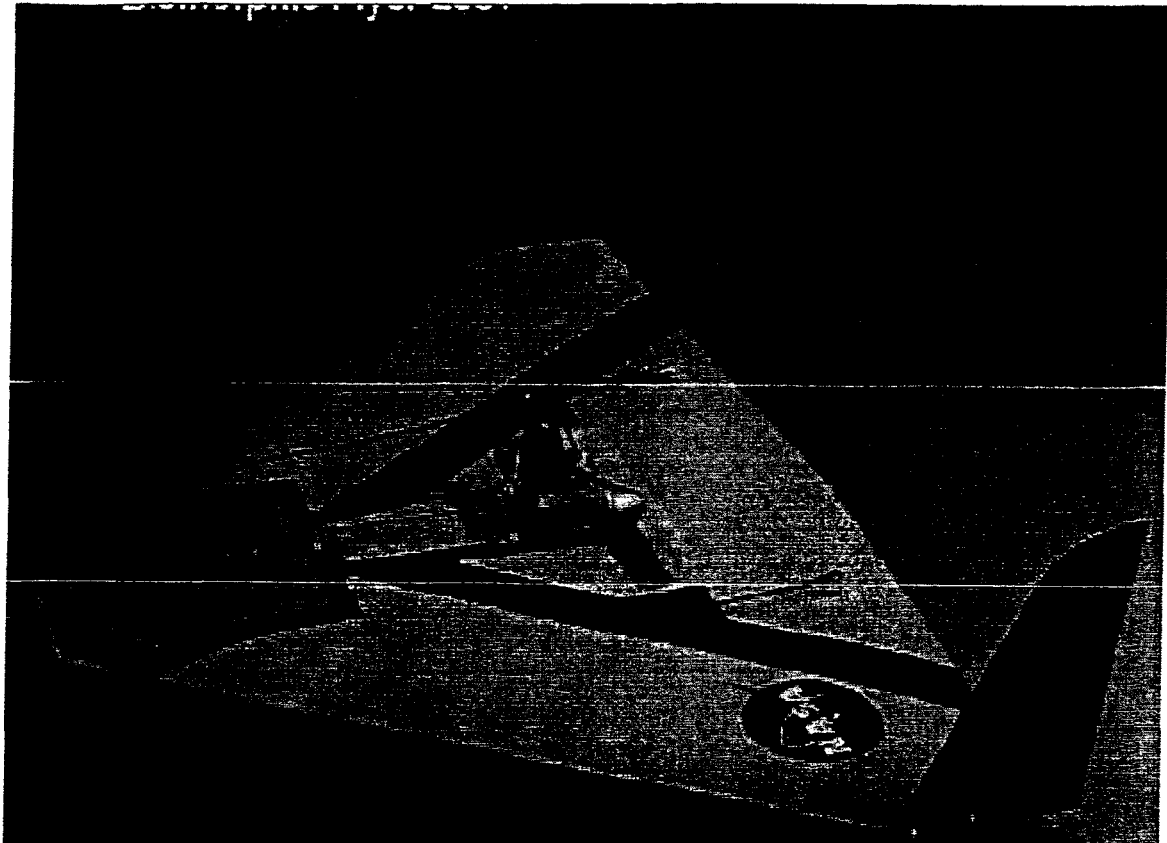


Figure 4: Prototype of the BEES flyer geometry (c.f. Ref. [9] for more details).

position, the Cartesian meshing scheme responds to the change in surface definition and refines and coarsens the mesh appropriately. This causes the total number of cells in the mesh to vary around 1.5 million depending upon the elevon settings. All computations were run on a 256-CPU SGI Origin 2000 machine with 400 MHz MIPS4 processors. Each individual datapoint required 9.3 hours of CPU time, and the entire database was filled in 9 days of wallclock time. During the database build the machine was shared with other users in a typical production environment. In fact, the machine was taken down for hardware maintenance twice during the 9 days without impacting the automated process.

Examples of the solutions are shown in Fig. 6 for two transonic Mach numbers. The elevons are deflected asymmetrically. As the freestream Mach number increases the shock on the wing upper surface and engine canopy moves aft from the leading edge and gets stronger. The asymmetry in the aft loading due to the elevon deflections is also evident. The large drag due to the strong shock adversely impacts the range of the vehicle, and hence the preliminary design cycle successfully identified a problem and solution early in the process.

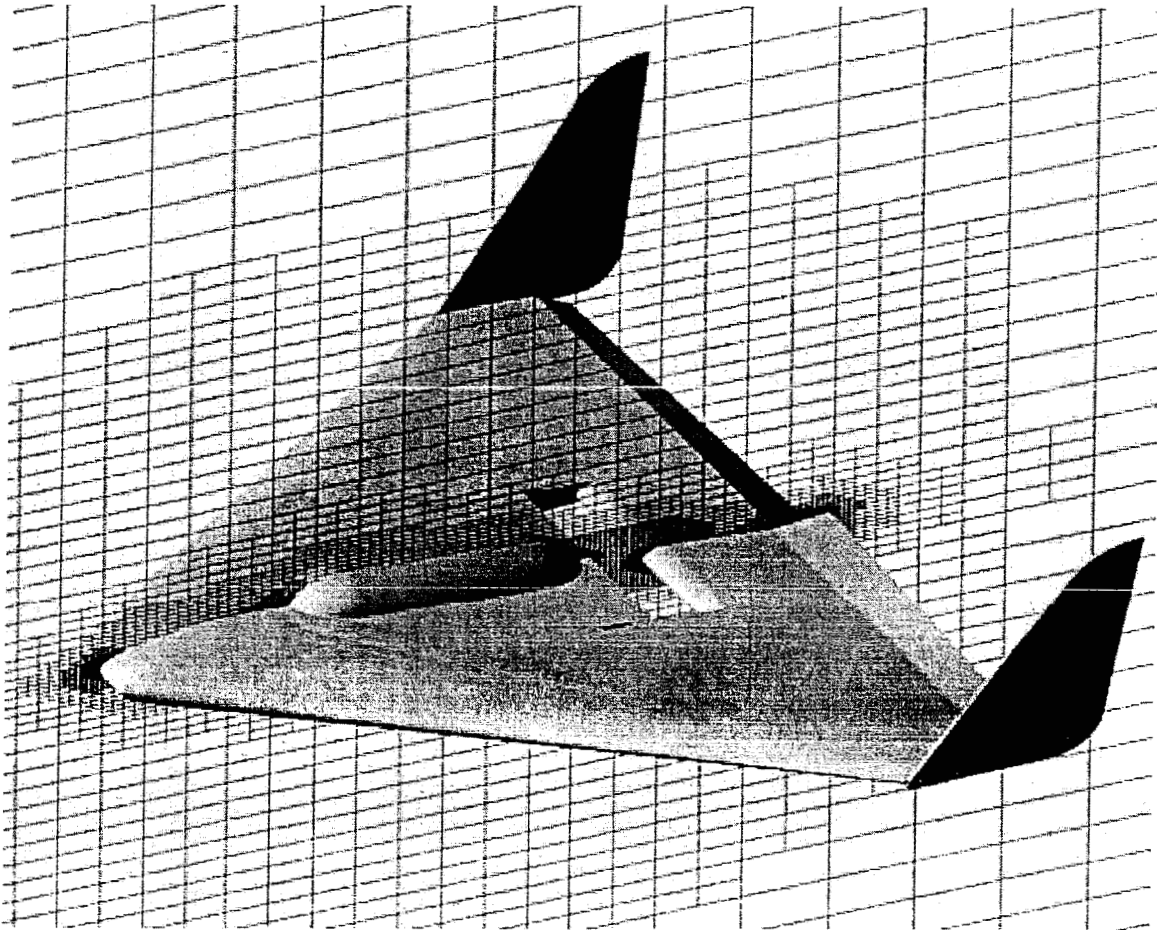


Figure 5: BEES flyer geometry with a cutting plane through the Cartesian mesh. The triangulated surface geometry was generated directly from a CAD solid model. The Cartesian mesh generation automatically refines near large geometric curvature.

4 Future Work

A modular process for performing parametric studies about a configuration using a Cartesian method has been described. This process leverages existing stand-alone applications for performing isolated steady-state simulations, and glues them together with control scripts to provide the functionality for performing a set analysis. A novel part of this process is the automatic handling of general control surfaces deflections based upon simple, user-specified inputs. With the basic infrastructure in place and tested, improvements can be initiated. The proposed paper will include details of different steering strategies. Steering is a critical component of developing an aerodynamic database as it directly impacts both efficiency, by optimizing the benefit for a fixed amount of work, and the accuracy by placing datapoints where there are abrupt changes in the S&C derivatives. An example from Ref. [3] of lift vs. Mach number and angle of attack for a liquid glide-back booster is shown in Fig. 7. Except

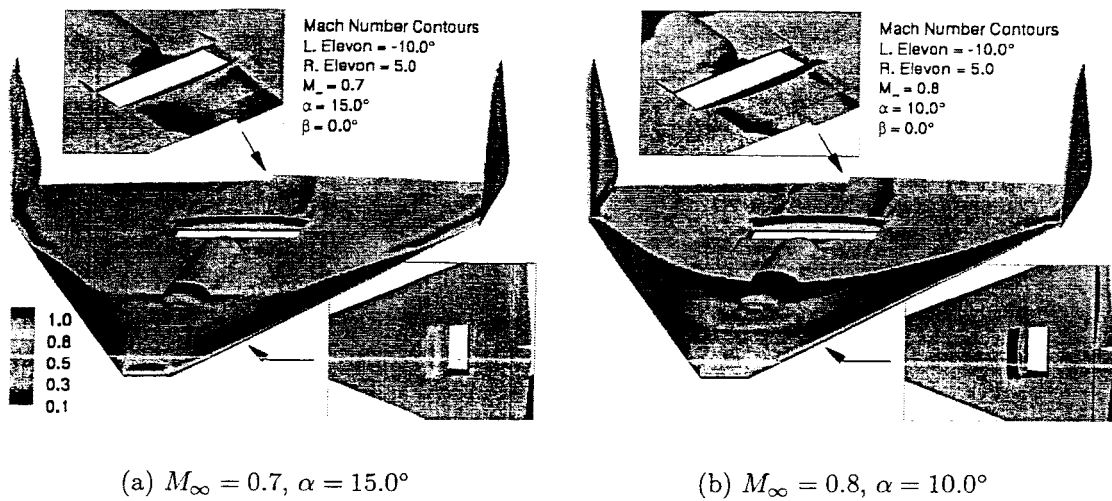


Figure 6: Surface Mach number for the BEES flyer. Red is high, blue is low.

near Mach 1.0, the variation of the derivatives is relatively benign. Refining near the sonic line throughout the angle of attack range, and coarsening elsewhere would be more optimal. This example is straightforward, however the design space with control surface deflections becomes more complicated, and methods that can automatically adjust are desired.

Another important topic is the merging or augmenting of databases using isolated refined datapoints. Two examples will illustrate the problem. The first is performing a parametric study early in the design cycle, and still leveraging that information later in the cycle after the design has changed. A second example is the use of multiple levels of fidelity, with the lower-fidelity tools being chosen for the majority of the design space, and higher-fidelity datapoints added at selected or critical areas. In both of these examples the idea of incrementally modifying the design response surface with the higher-fidelity data is appealing. This approach can provide higher-fidelity performance at the cost of a low-fidelity analysis. The proposed paper will also include a discussion of this topic.

References

- [1] Yarrow, M., McCann, K.M., DeVivo, A. and Tejnil, E., "Production-Level Distributed Parametric Study Capabilities for the Grid," in *Proceedings of Grid 2001 2nd International Conference on Grid Computing*, 2001.
- [2] Murman, S.M., Chaderjian, N.M., and Pandya, S. A., "Automation of a Navier-Stokes S&C Database Generation for the Harrier in Ground Effect," AIAA Paper 2002-0259, Jan. 2002.

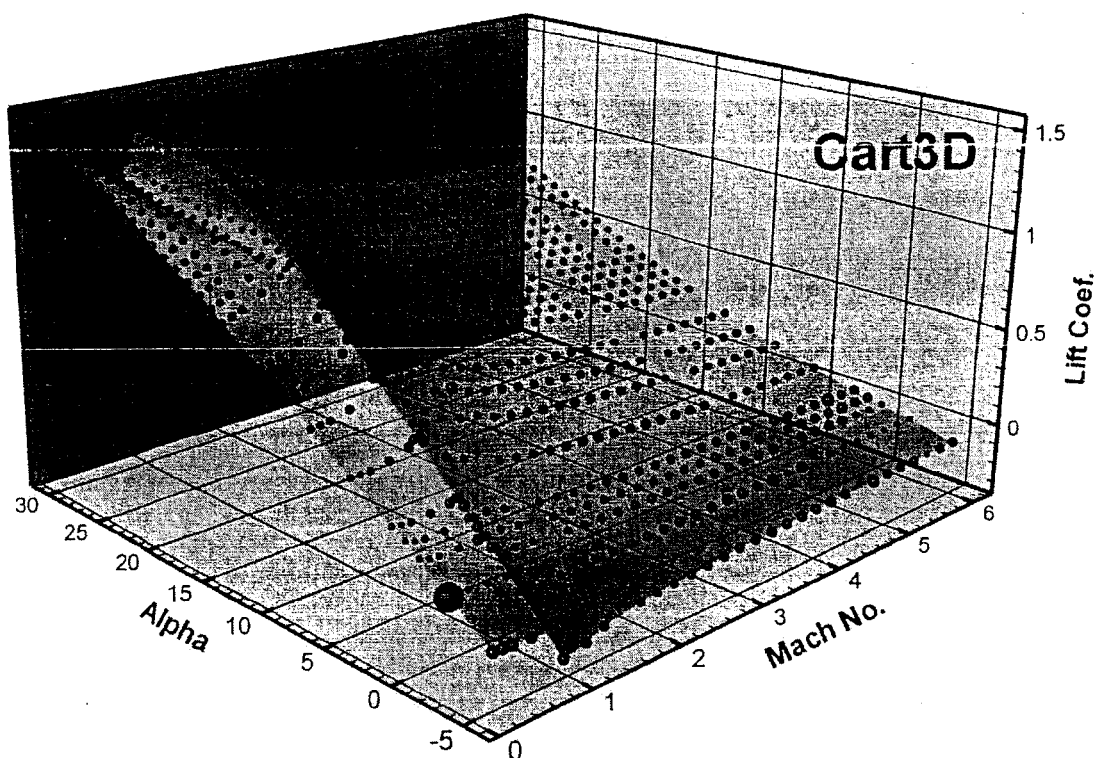


Figure 7: Variation of lift for a liquid glide-back booster configuration (from Ref. [3]).

- [3] Rogers, S. E., Aftosmis, M. J., Pandya, S. A., Chaderjian, N. M., Tejnil, E. T., and Ahmad, J. U., "Automated CFD Parameter Studies on Distributed Parallel Computers," AIAA Paper 2003-4229, June 2003.
- [4] Aftosmis, M.J., Berger, M.J., and Melton, J.E., "Robust and Efficient Cartesian Mesh Generation for Component-Based Geometry," *AIAA Journal*, 36(6):952-960, June 1998.
- [5] Haimes, R. and Aftosmis, M.J., "On Generating High Quality "Water-tight" Triangulations Directly from CAD," in *Meeting of the International Society for Grid Generation (ISGG) 2002*, June 2002.

- [6] Aftosmis, M.J., Berger, M.J., and Adomavicius, G., "A Parallel Multilevel Method for Adaptively Refined Cartesian Grids with Embedded Boundaries," AIAA Paper 2000-0808, Jan. 2000.
- [7] Berger, M.J., Aftosmis, M.J., Marshall, D.D., and Murman, S.M., "Performance of a New CFD Flow Solver using a Hybrid Programming Paradigm," *submitted to the Journal of Parallel and Distributed Computing*, 2003.
- [8] Murman, S.M., Chan, W.M., Aftosmis, M.J., and Meakin, R.L., "An Interface for Specifying Rigid-Body Motion for CFD Applications," AIAA Paper 2003-1237, Jan. 2003.
- [9] Soccol, D.D., Chahl, J.S., Le Bouffant, J., Garratt, M.A., Mizutami, A., Thurrowgood, S.A., Ewyk, G., and Thakoor, S., "A Utilitarian UAV Design for NASA Bioinspired Flight Control Research," AIAA Paper 2003-461, Jan. 2003.
- [10] Murman, S.M., Aftosmis, M.J., and Berger, M.J., "Numerical Simulation of Rolling-Airframes Using a Multi-Level Cartesian Method," AIAA Paper 2002-2798, June 2002.
- [11] "Portable Batch System." <http://pbs.mrj.com>.
- [12] Murman, S.M., Aftosmis, M.J., and Berger, M.J., "Implicit Approaches for Moving Boundaries in a 3-D Cartesian Method," AIAA Paper 2003-1119, Jan. 2003.
- [13] Murman, S.M., Aftosmis, M.J., and Berger, M.J., "Simulations of 6-DOF Motion with a Cartesian Method," AIAA Paper 2003-1246, Jan. 2003.
- [14] Nemec, M., Aftosmis, M.J., and Pulliam, T.H., "CAD-Based Aerodynamic Design of Complex Configurations Using Cartesian Methods," AIAA Paper 2004-0113, Jan. 2004.
- [15] "ParaVista." <http://www.viz-solutions.com/ParaVista.htm>.
- [16] Sandstrom, T., Henze, C., and Levit, C., "The hyperwall," in *Proc. of IEEE International Conference on Coordinated & Multiple Views in Exploratory Visualization*, pp. 124-133, 2003.