On the Use of CAD and Cartesian Methods for Aerodynamic Optimization

M. Nemec
NRC Associate, NASA Ames Research Center, MS T27B, Moffett Field, CA 94035, USA
nemec@nas.nasa.gov
M. J. Aftosmis
Research Scientist, NASA Ames Research Center, MS T27B, Moffett Field, CA 94035, USA
T. H. Pulliam
Research Scientist, NASA Ames Research Center, MS T27B, Moffett Field, CA 94035, USA

Extended abstract submitted to ICCFD3, Toronto, July 2004

1 Introduction

Aerodynamic design is inherently a multidisciplinary problem that involves complex surface geometry, competing objectives, multiple operating conditions, and strict design constraints. Consequently, important considerations for an effective optimization framework include: 1) geometry modeling and surface discretization, 2) objective function and constraint evaluation, which includes methods for mesh generation, surface- and volume-mesh perturbation, and flow solution, and 3) the selection of optimization techniques. In modern engineering design environments, the surface geometry is generally represented by a parametric Computer-Aided-Design (CAD) model. Since all downstream analysis and design relies on this representation, the CAD model, accessible in its native environment, should serve as the basis of automated optimization.

Recently, a promising approach has been developed using a standardized Application Programming Interface (API) that allows direct access to the native CAD representation. This approach is *vendor neutral*, i.e. independent of a specific CAD system, and is based on the Computational Analysis and PRogramming Interface (CAPRI) [1, 2, 3, 4, 5]. In addition to providing an effective tool for surface discretization, CAPRI allows the modification of adjustable parameters built into the CAD model. Hence, the design variables and geometric constraints can be intrinsic to the CAD model. Upon a regeneration of the model in response to a parameter change, CAPRI constructs a "water-tight" surface triangulation, which can be automatically refined to obtain a CFD-ready triangulation.

Robust and efficient volume-mesh generation is the next critical part of the optimization framework. Traditional, body-fitted structured and unstructured mesh generation algorithms can be computationally expensive and usually require user supervision. This has motivated the development of mesh-perturbation schemes [6, 7, 8] that are used during the optimization process to modify a given baseline mesh. The location of nodes is tracked as the mesh deforms, which allows the use of fast solution-transfer algorithms and helps maintain a smooth design landscape. Unfortunately, the mesh-perturbation schemes may breakdown and require user intervention for topology and sufficiently large geometry changes.

Cartesian methods offer a promising alternative. The mesh generation is fast, robust, and essentially fully automatic [9, 10]. Due to the decoupling of the surface discretization from the volume mesh, Cartesian mesh generation is virtually insensitive to the complexity of the input geometry. When combined with robust high-fidelity flow solvers, the Cartesian approach provides a unique capability, especially for problems with moving bodies in relative motion [11] and automated optimization [8, 12, 13]. By allowing general topology and radical geometry changes, the optimization algorithm is able to explore new regions of the design landscape that may lead to superior and unconventional designs.

For the problems under consideration here, the most promising optimization algorithms range from autonomous approaches such as evolutionary [14, 15, 16] and finite-difference gradient-based algorithms [17], to methods requiring greater coupling such as the adjoint approach [18, 19, 20] for gradient computations. Furthermore, the use of these techniques in conjunction with pattern-search techniques [21] and approximation methods [22, 23], can help deal with complex design landscapes and reduce the computational cost of the optimization.

The selection of a particular optimizer is problem dependent and involves the classic trade-off between specialization and generality. It is therefore desirable to construct a flexible optimization framework to serve as a test-bed for various strategies and algorithms. Important factors in the integration of an optimization algorithm into such frameworks include: 1) scalability of the optimization technique in a parallel computing environment, 2) degree of coupling among the optimization modules and the high-fidelity solvers within the framework, 3) flexibility in the formulation of objectives and constraints, and 4) effectiveness in multi-modal and noisy design landscapes.

In targeting the design of complex three-dimensional geometry, the presence of noise, or non-smoothness, is unavoidable in the design landscape. The noise stems from three primary sources. First, physical sources such as local flow unsteadiness due to complex geometry may hinder deep convergence of the flow solution. Second, noise due to geometry-representation and discretization, which may be caused by the details of the model construction, the internal characteristics of the CAD system, or the surface tessellation algorithm. Third, noise due to discretization error of the volume mesh. For embedded-boundary Cartesian methods, the intersection of the surface geometry with the volume mesh changes non-smoothly as the geometry evolves during the optimization. Consequently, it is important to evaluate the influence of noise on the optimization algorithm, since the presence of false extrema in the design landscape may slow down and even stall the optimization process.

The objective for this paper is to present the development of an optimization capability for *Cart3D*, a Cartesian inviscid-flow analysis package of Aftosmis *et al.* [10, 24]. We present the construction of a new optimization framework and we focus on the following issues:

- Component-based geometry parameterization approach using parametric-CAD models and CAPRI. A novel geometry server is introduced that addresses the issue of parallel efficiency while only sparingly consuming CAD resources.

- The use of genetic and gradient-based algorithms for three-dimensional aerodynamic design problems. The influence of noise on the optimization methods is studied.

Our goal is to create a responsive and automated framework that efficiently identifies design modifications that result in substantial performance improvements. In addition, we examine the

architectural issues associated with the deployment of a CAD-based design approach in a heterogeneous parallel computing environment that contains both CAD workstations and dedicated compute engines. We demonstrate the effectiveness of the framework for a design problem that features topology changes and complex geometry.

2 Optimization Problem Formulation

The aerodynamic optimization problem consists of determining values of design variables X, such that the objective function $\mathcal{J}$ is minimized

$$\min_X \ \mathcal{J}(X, Q) \tag{1}$$

subject to constraint equations C_j:

$$C_j(X, Q) \leq 0 \qquad j = 1, \ldots, N_c \tag{2}$$

where the vector Q denotes the conservative flowfield variables and N_c denotes the number of constraint equations. The flowfield variables are forced to satisfy the governing flowfield equations, $\mathcal{F}$, within a feasible region of the design space Ω:

$$\mathcal{F}(X, Q) = 0 \qquad \forall\, X \in \Omega \tag{3}$$

which implicitly defines $Q = f(X)$. The governing flow equations are the three-dimensional Euler equations of a perfect gas, where the vector $Q = [\rho, \rho u, \rho v, \rho w, \rho E]^T$.

The objective function defines the goals of the optimization problem, while the constraint equations limit the feasible region of the design space. The constraints may involve performance functionals, such as lift, geometric quantities, such as volumes and thicknesses, and also simple bound constraints for design variables. A modular framework is constructed to solve the optimization problem defined by Eqs. 1–3. An evaluation of the objective function and constraints requires the coupling of several software components that form the analysis module of the framework. These components are outlined in Fig. 1 and are described below. Following the analysis module, we present the optimization algorithms and a detailed description of the optimization framework.

3 CAD-Based Geometry Modeling

In traditional approaches for geometry modeling and regeneration, one begins by either importing a given baseline surface discretization to a geometry parameterization tool, or defining a set of idealized components within a geometry parameterization tool [25, 6, 26, 27]. Most likely, the baseline surface discretization has been generated from an existing CAD geometry. This approach offers fast and accurate regeneration of surfaces and computation of component intersections. Furthermore, the source code is usually available, and hence the computation of design sensitivities (if required) is possible by the use of automatic differentiation or analytically.

However, the geometry parameterization tool is usually tailored to a specific set of allowable topologies, and provides a limited variety of design variables. For example, only wing-body configurations with prescribed design variables for planform and shape changes may be

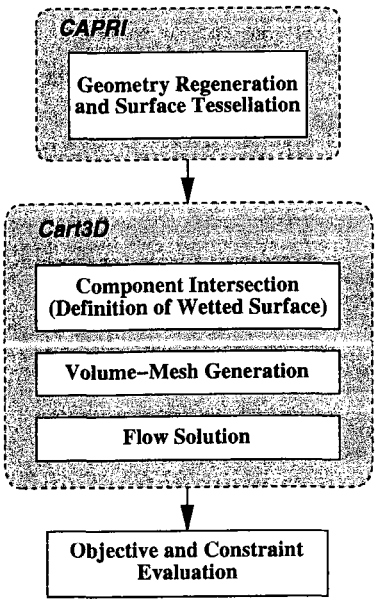


Figure 1: Components of the analysis module

allowed. Such built-in restrictions limit the feasible region of the design space, and consequently, the best design may never be realized. Although it is always possible to improve the parameterization tool with additional code development, this burden becomes prohibitive when faced with complex, integrated configurations and multidisciplinary problems. Ultimately, this effort leads to the development of a specialized in-house tool mimicking aspects of a parametric CAD system.

An alternative approach is to consider the use of a commercial-CAD system [28]. Most present-day CAD software is based on parametric design and feature modeling. A part is constructed by defining features with adjustable parameters. The features define a sequence of operations, for example an extrusion of a sketched cross section, and are organized in the form of a feature tree. This forms the master-model of the part, and different instances of the part can be generated for various parameter values by following the template of the master-model. Individual parts can be grouped in hierarchical dynamic assemblies, which allow relative motion between constituent parts. Furthermore, features not required for the analysis (or design) problem at hand can be suppressed. The potential advantages of this approach include:

- Generality: there are virtually no pre-defined limitations on the complexity of parts and assemblies.
- Consistency: the part is always queried in its native environment, without geometry translation.
- Variable fidelity: through the use of feature suppression, various levels of part abstraction are possible.
- Natural constraints: the feature-based modeling captures the design intent of the part or assembly and therefore can be used to impose natural constraints on the geometry.

Although this approach is conceptually very appealing, the integration of a commercial CAD system into an optimization framework requires careful consideration of the following issues:

1. Parts and assemblies must be created with design modifications in mind. Although this sounds obvious, the selection of parameters for a design study can be a challenging task and the construction of flexible and robust CAD models requires significant CAD-system experience. The geometry parameterization issue is placed well upstream in the design/ analysis process.

2. The use of "legacy" geometry, or geometry with no parametric CAD representation, requires special consideration. Unfortunately, most CFD geometry today belongs to this category.

3. The interface for accessing the parameters of the CAD model depends on the specific CAD system.

4. The efficiency of the geometry updates and the surface discretization depend on the attributes of the proprietary CAD-geometry kernel.

5. Practical issues such as the number of available CAD licenses need to be considered in the design of parallel optimization procedures.

6. The issue of differentiability and the use of mesh-perturbation algorithms for non-smooth changes in the surface discretization.

Items 1 and 2 are organizational issues that are beyond the scope of this work. It is clear, however, that current production and development environments make extensive use of feature-based solid modeling for engineering analysis and design. While the mesh requirements of CFD simulations place unusual demands on the CAD system, it is highly advantageous to leverage its sophisticated modeling capabilities. We use CAPRI [3, 5, 4] to address items 3 and 4, which we discuss in the following section. The architecture of the optimization framework, presented thereafter, mitigates the concerns of item 5. Item 6 remains an open issue. Finite-difference schemes can provide good approximations of sensitivity information, but their dependence on stepsize limits the accuracy, and in some cases the robustness, of this approach [26]. Mesh-perturbation schemes introduce additional difficulties due to the requirement of tracking surface deformations [4].

4 Role of CAPRI

4.1 CAD-Model Regeneration

CAPRI exposes the master-model feature tree of the CAD model and allows direct modification of parameters (non-driven values) within that tree. A detailed overview of CAPRI's extensive capabilities is given in Ref. [5]. An alternative to CAPRI is the direct use of "Developer Toolkits" that are available for most CAD systems, but this approach sacrifices CAD-vendor neutrality.

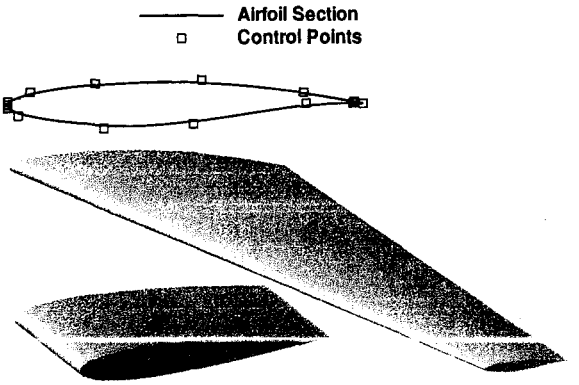


Figure 2: Example of two instances of a generic-wing CAD model. A B-spline airfoil parameterization is shown at the top of the figure.

Most design variables are associated directly with values exposed in the feature tree. An exception is surface shape modification, which requires the access to feature information at a high level of detail. For example, the control-point locations for individual curves are required. CAPRI is able to expose non-dimensional curve data points of sketched features, which can be modified to generate new surfaces. For example, these may be the data points of airfoil sections that are lofted to define a wing, or fuselage cross-sections. Note that the use of each data point as a design variable would lead to very large optimization problems and potentially non-smooth curves. To circumvent this difficulty, we use B-spline curves to define each cross-section. The shape design variables are associated with the B-spline control points and are external to the CAD system. This additional level of indirection can easily accommodate other approaches, such as the Hicks-Henne shape functions [4].

Fig. 2 shows an example of two very different instances of the same parametric-CAD model for a generic wing part. The generic model consists of the typical planform parameters that include surface area, aspect ratio, taper ratio, sweep, and root-section and tip-section twist. A shape parameterization example is shown at the top of Fig. 2, where a cubic B-spline with 15 control points is used to closely approximate the RAE-2822 airfoil. The root and tip airfoil sections are linearly lofted to generate the wings shown.

4.2 Automatic Surface Tessellation

After modifying and regenerating the CAD-model, CAPRI provides a surface triangulation for each component. The triangulation is refined based on three measures of quality[3]: 1) triangle edge length, 2) the deviation of an edge from the underlying CAD model, and 3) a dihedral angle bound between adjacent triangles. The triangulation algorithm is highly robust, but in certain instances the resulting triangulations for a component subject to small shape perturbations may be significantly different. This may introduce noise into the optimization problem, which we discuss further in the Results section.

Recently, a new triangulation algorithm has been added to CAPRI that provides a more uniform, right-triangle based tessellation. An example of coarse triangulations is shown in Fig. 3 for a dramatic change in surface shape. The algorithm identifies component faces that

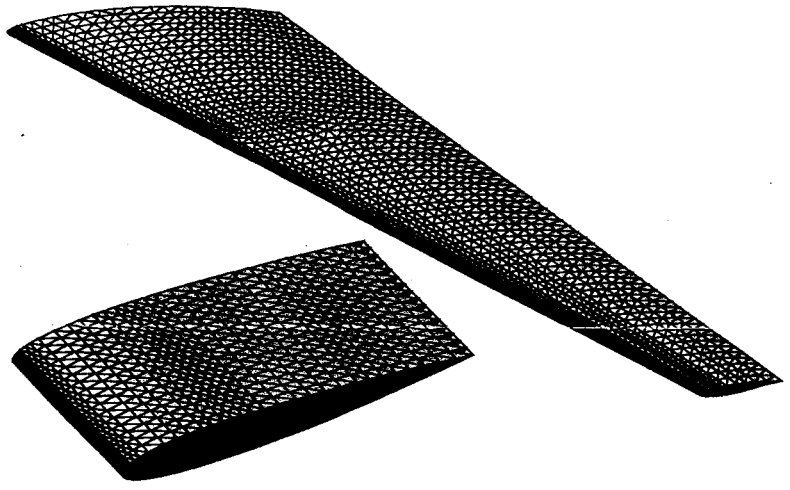


Figure 3: Examples of right-triangle based tessellations for large shape deformations

qualify for such triangulations, and otherwise reverts back to the quality triangulation. The new triangulation algorithm is less sensitive to small geometry perturbations.

5 Mesh Generation and Flow Solution

The extraction of a wetted surface from a set of intersecting components is the next task of the analysis module (see Fig. 1). Since CAD-solid representations typically rely on the use of parametric B-splines (or NURBS), the computation of component intersections can be costly within the CAD system. In the present approach, the components are intersected after the surface discretization. This operation is performed efficiently as a part of the component-based approach of *Cart3D* [10]. It should be noted that CAPRI caches an associated triangulation with each component. This caching avoids unnecessary re-triangulations for components that are not modified or experience only rigid body motion during the design process.

Cartesian volume meshes are generated by repeated cell division of an initial coarse mesh[10]. A parallel multilevel method is used to solve the steady-state Euler equations. The spatial discretization is second order accurate using van Leer's flux vector splitting in conjunction with either Minmod or Venkatakrishnan's flux limiters, see Aftosmis *et al.* [24] for details.

6 Optimization Algorithms

We cast the optimization problem as an unconstrained problem by lifting the side constraints, Eq. 2, into the objective function using a penalty method. The constraint imposed by the flow-field equations, Eq. 3, is satisfied at every point within the feasible design space, and consequently these equations do not explicitly appear in the formulation of the optimization problem. We investigate the genetic algorithm of Holst and Pulliam [16], and an unconstrained BFGS quasi-Newton algorithm coupled with a backtracking line search [17, 29, 20]. The objective function gradient is evaluated using central-differences. We "warm-start" the finite-

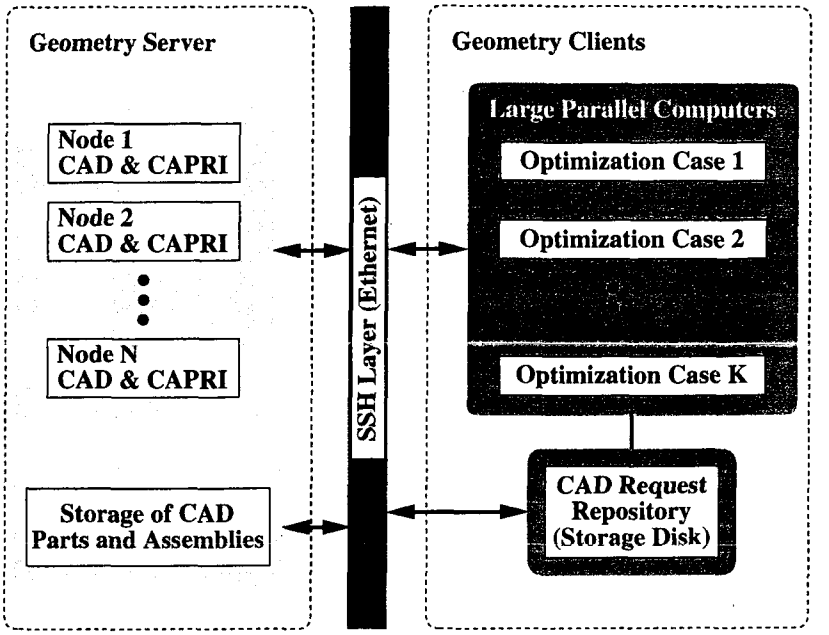


Figure 4: Layout of the interface between optimization processes, or geometry clients, on the right side and the distributed geometry server on the left side.

difference gradient computations from the base-state solution, saving roughly 25 to 50% when compared with the standard full-multigrid startup. The solution-transfer algorithm is described by Aftosmis *et al.* [30].

7 Optimization Framework

The synthesis of individual modules into an automated and efficient optimization framework is a challenging software design problem. A number of sophisticated frameworks exist, such as the DAKOTA toolkit [31], which provide a flexible and general approach for linking analysis tools with optimization techniques in large parallel computing environments. In order to have a direct control over the layout of the framework, and therefore quickly evaluate different parallel architectures, we pursue the development of a custom framework.

The first part of the framework addresses the coupling of the CAD/CAPRI module with the optimization process. Figure 4 shows the layout of this distributed client-server interface. The optimization along with the analysis module are executed in a queue system of large compute engines. At each iteration of the optimization process, CAD geometry requests are generated for different parameter values and these are placed in a central repository (right side of Fig. 4). Independent of the optimization runs, a geometry server is initiated that consists of multiple CAD nodes (left side of Fig. 4). The nodes process the geometry requests by retrieving the required parts or assemblies from a specified storage location, regenerating the CAD models, and providing surface triangulations for the optimization processes. Since the geometry requests are independent, we expect the geometry server to achieve nearly linear scalability.

Initially, it may appear that in order to obtain an efficient geometry server, the number of

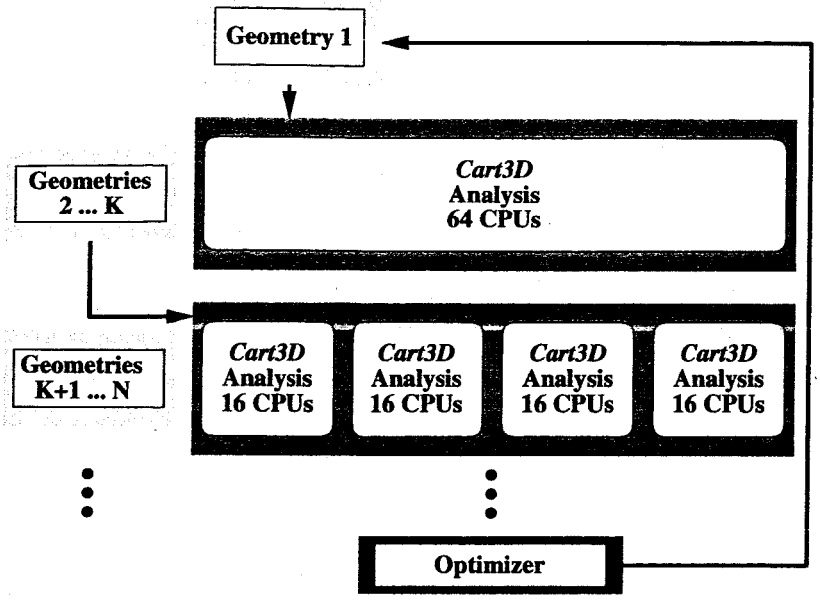


Figure 5: Dynamic allocation of processors to mask the latency of CAD geometry processing (based on 64 CPUs as an example)

available CAD nodes should match the number of geometry requests from all optimization processes. In practice, the number of CAD nodes is limited by the number of available CAD licenses, as each node consumes one license. An immediate concern is that the CAD nodes become the bottleneck of the optimization process, idling the processors of the compute engines. *One of the driving requirements in the design of the present geometry interface is to maintain the efficiency of the optimization process when only a handful of licenses are available, yet remain scalable should the number of licenses increase.*

This is a classic problem of latency. To avoid the geometry processing bottleneck, we mask the latency of the CAD nodes by dynamically allocating the available processors of the optimization process to the number of completed surface triangulations. Figure 5 illustrates this on an example with 64 processors. At the start of each design iteration, all processors are dedicated to the solution of the first returned surface triangulation from the CAD nodes. This is the base state of the gradient method and the first chromosome of the genetic algorithm, denoted as "Geometry 1" in Fig. 5. Note that there is a brief idling of all processors, which could be avoided by implementing an asynchronous optimization approach. Upon completion of the first geometry analysis, we check the number of completed surface triangulations. These are processed by the CAD nodes while the analysis of the first geometry is performed on the compute engine, denoted as "Geometries $2 \ldots K$" in Fig. 5. The number of processors is distributed among the completed surface triangulations and multiple analysis modules are executed on subsets of the available processors. This cycle repeats until all geometry requests are analyzed. For example, the optimization process may have 64 processors available and if 4 surface triangulations are completed by the CAD nodes, we can execute 4 analysis modules in parallel with 16 processors per module, see Fig. 5.

It is important to note that the geometry intersection and volume mesh generation algorithms of the analysis module (see Fig. 1) are serial algorithms, while the flow solver is an efficient parallel solver [24]. The dynamic, coarse-grained parallelism used during each design iteration provides not only concurrent execution of serial tasks, but also ensures high parallel efficiency of the flow solver by limiting the number of processors available to each analysis module. Studies by Eldred *et al.* [32] demonstrate that such multilevel parallelism significantly improves the scalability of optimization frameworks.

The worst case scenario occurs when the wall-clock time required for the processing of a geometry request exceeds the time for completion of the flow solution when all processors are used. If only one CAD node is available, then this CAD node would not be able to feed the compute engine with geometries without processor idle time. This situation is unlikely, since CAD model regeneration and tessellation tasks have computational complexity of $\mathcal{O}(N^2)$, while volume mesh generation and flow solution tasks are $\mathcal{O}(N^3)$.

The CAD nodes are typically distributed among available engineering workstations. They could also be executed on a single parallel machine or the compute engine itself. The individual nodes are fully independent. Hence, the system is tolerant of node crashes and it is easy to add or delete nodes. The nodes are "greedy", that is, they compete for geometry requests by checking the CAD repository. In order to avoid race conditions between nodes for the same geometry request, a node must first acquire a lock on the CAD repository. Once a lock is obtained, the node searches for the oldest geometry request and releases the lock. This process is further complicated by the fact that all communications between the node, the CAD-request repository, the part storage location, and the compute engines are performed using secure-shell commands (see Fig. 4). Once a geometry request is processed, the node notifies the optimization process that the surface triangulation is ready. The surface triangulation is pulled from the node by the optimization process when the analysis of that particular configuration is required. This facilitates the downloading of the surface triangulations in parallel.

8 Results and Discussion

For this abstract, we present only a simple three-dimensional configuration to demonstrate the effectiveness of the new optimization framework. All geometry models are constructed using the Pro/ENGINEER CAD system.

8.1 3-D Design Example

The second design example is based on the configuration shown in Fig. 6. This generic model is a CAD assembly of five parts consisting of a fuselage with a bluff base, a wing, a canard, a canted tail, and an engine cluster. The wing and canard are constructed from the same CAD model, which was also used in the first design example and is shown in Fig. 2. At the assembly level, the wing and canard parts are "attached" to the fuselage via two parameters, their horizontal and vertical locations, respectively. These parameters are constrained to intersect the projection of the fuselage on the symmetry plane within the CAD system. This simple construct avoids non-physical configurations, for example wings that detach from the fuselage during the optimization, even if the fuselage shape and dimensions change.

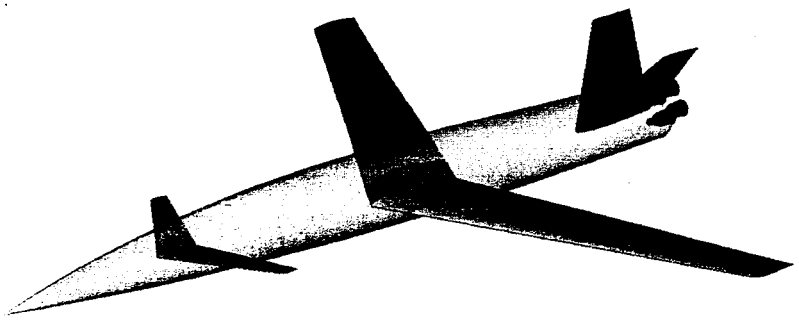


Figure 6: Model configuration for the second design example (before component intersection)

Table 1: Average CPU time for CAD-model regeneration and tessellation (600 MHz R14000 SGI Octane Workstation, Pro/ENGINEER kernel)

Part	Tessellation Algorithm	CAD-Model Regeneration (s)	Tessellation (s)	Number of Triangles
Fuselage	Quality-based	2.0[a]	93.3	$\approx 41,000$
Wing	Right-triangle	3.0[b]	16.5	$\approx 50,000$

[a] No shape-section change, only global parameter modifications
[b] Shape-section change and planform parameter modifications

Before presenting optimization results, we characterize the performance of the optimization framework. We focus on the analysis module, see Fig 1, as this is the most expensive part of the framework. Table 1 presents average CPU timing results for the CAD model regeneration and surface triangulation using CAPRI. The timings for the fuselage and wing parts are representative of any other component in the assembly. The CAD-model regeneration times are slightly faster for changes that do not require shape modifications, i.e. no profile section changes. It is clear from Table 1 that CAD-model regeneration times are not a significant expense even for problems with many design variables.

The CPU time for surface triangulation is greatly influenced by the choice of the triangulation algorithm. For the fuselage, CAPRI uses the quality-based triangulation algorithm. This is in contrast to the wing surfaces, where the right-triangle tessellation algorithm is used. To further elucidate the performance reported in Table 1, the quality-based triangulation algorithm generates roughly 500 triangles per CPU sec., while the right-triangle tessellation algorithm generates roughly $3,300$ triangles per CPU sec. While the time required for surface triangulation is not prohibitive, it is important to avoid all unnecessary re-triangulations during the optimization. This is accomplished by caching an associated baseline triangulation for each part prior to the optimization and tracking parameter changes. For example, we tag design variables that control relative motion between components, since a change in these parameters does not require surface re-triangulation.

Table 2 presents average timing results for individual components within the *Cart3D* anal-

Table 2: Wallclock times for individual components of the *Cart3D* module (600 MHz R14000 SGI Origin 3000)

Component	Time (s)	Algorithm
Mesh Generation[a]	132.0	Serial
Flow Solution[b]	455.0	Parallel
Mesh Solution Transfer	26.0	Serial

[a] Includes component intersection (definition of wetted surface), mesh generation, flow-solver domain decomposition, and multigrid coarse-mesh generation

[b] Using 64 processors

ysis module. The volume mesh contains roughly 1.5 million cells for a half-span model of the configuration and 64 processors are used to obtain the flow solution. The time for the mesh-solution transfer algorithm used to "warm-start" finite-difference gradient computations is also shown.

Valuable information regarding the CPU efficiency during a design iteration is obtained by comparing Tables 1 and 2. For example, suppose that we have only one CAD license available and that the design problem of interest involves design variables associated with both the fuselage and wing. Then, the timings in Tables 1 and 2 indicate that the time required to complete a CAD-model regeneration and surface triangulation is a factor of six smaller than the time required for a flow solution. This means that by the time the analysis module completes the flow solution of the first chromosome of the GA or the base-state of the gradient method, six new surface triangulations are ready for analysis. By subdividing the available CPUs of the optimization process, we execute multiple analysis modules in parallel to enhance the parallel efficiency of the optimization framework.

We consider the optimization problem of attaining a nearly zero pitching moment coefficient for the configuration shown in Fig. 6 by optimizing the canard control surface. The lift coefficient is constrained by the initial lift of the configuration. The design variables are the control surface aspect ratio, twist, and position along the center line of the fuselage. The problem has two local optima, the tail or canard configuration, with the canard configuration as the global optimum due to an aft location of the center of gravity. For optimization using the genetic algorithm, the canard area is also a design variable. This introduces the possibility of a topology change in the design space, since the resulting wetted surface may not include a control surface, as shown in Fig. 7. We use 16 chromosomes for each generation of the genetic algorithm. For the gradient-based quasi-Newton algorithm, the control surface area is kept constant and we start from a canard configuration, i.e. the control surface is positioned in front of the center of gravity. The freestream Mach number is 0.85 and the angle of attack is 1.0 deg.

The objective function uses a quadratic-penalty formulation, with a target lift coefficient of 0.222 and a target pitching moment coefficient of 0.001. The initial pitching moment is

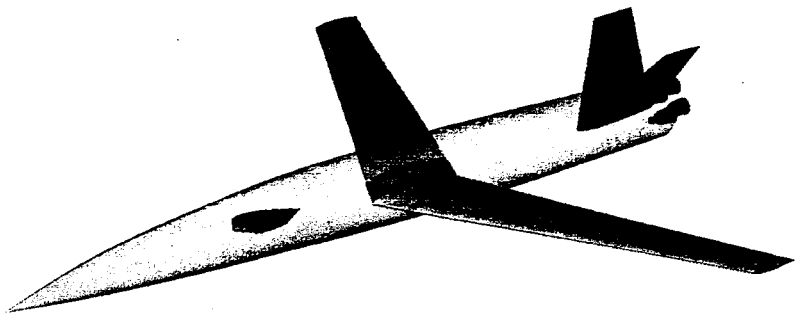


Figure 7: Example configuration where the control surface is not part of the wetted surface

-0.0714. Figure 8 shows the convergence history of the objective function. Note that the label "Design Iteration" refers to the number of generations evaluated by the genetic algorithm, and the number of objective function and gradient evaluations by the quasi-Newton algorithm. Both optimization methods trim the configuration at the given flight conditions. The gradient has been reduced by almost three orders of magnitude. The genetic algorithm converges within six design iterations, requiring only 96 function evaluations. The quasi-Newton algorithm requires 56 function evaluations.

Figures 9(a) and 9(b) show the initial and final designs for the quasi-Newton algorithm. The control surface converged to the minimum allowable forward location on the fuselage (8% of fuselage length), a twist angle of 2.98 deg., and an aspect ratio of 6.03. Note that the control surface area is fixed at 60.0 during the optimization. Figure 9(c) shows the final design using the genetic algorithm. For this case, the optimization converged to the upper bound of the control surface area, which is 60.0, a forward location of 8.2% of fuselage length, a twist angle of 3.41 deg., and an aspect ratio of 4.36. The difference in the two designs indicates that the optimization problem does not have a unique solution. There may be many control surfaces that trim this configuration and further constraints are required to define a unique problem.

9 Acknowledgments

The authors gratefully acknowledge Robert Haimes (MIT) for his assistance with CAPRI, in particular the implementation of the new tessellation algorithm and master-model improvements. The authors would also like to thank Peter Gage (NASA Ames), Alexander Te (NASA Ames), and Curran Crawford (MIT) for their help with parametric geometry models. This work was performed while the first author held a National Research Council Research Associateship Award at the NASA Ames Research Center.

References

[1] R. Haimes and G. Follen. Computational analysis programming interface. In Cross, Eiseman, Hauser, Soni, and Thompson, editors, *Proceedings of the 6th International Conference on Numerical Grid Generation in Computational Field Simulations*, University of Greenwich, 1998.

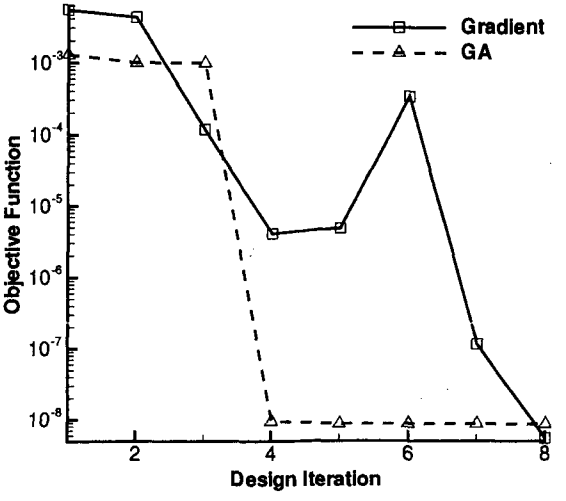


Figure 8: Objective function convergence

[2] M. J. Aftosmis, M. Delanaye, and R. Haimes. Automatic generation of cfd-ready surface triangulations from cad geometry. AIAA Paper 99–0776, Reno, NV, January 1999.

[3] R. Haimes and M. J. Aftosmis. On generating high quality "water-tight" triangulations directly from cad. Technical report, Meeting of the International Society for Grid Generation, (ISGG), Honolulu, HI, June 2002.

[4] J. J. Alonso, J. R. R. A. Martins, J. J. Reuther, R. Haimes, and C. Crawford. High-fidelity aero-structural design using a parametric cad-based model. AIAA Paper 2003–3429, Orlando, FL, June 2003.

[5] R. Haimes and C. Crawford. Unified geometry access for analysis and design. Technical report, 12th International Meshing Roundtable, Santa Fe, NM, September 2003.

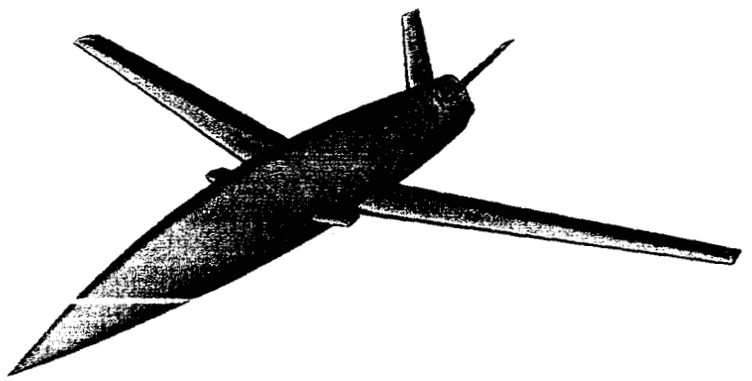
[6] J. J. Reuther, A. Jameson, J. J. Alonso, M. J. Rimlinger, and D. Saunders. Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, part 1. *Journal of Aircraft*, 36(1):51–60, 1999.

[7] E. J. Nielsen and W. K. Anderson. Recent improvements in aerodynamic design optimization on unstructured meshes. *AIAA Journal*, 40(6):1155–1163, 2002.

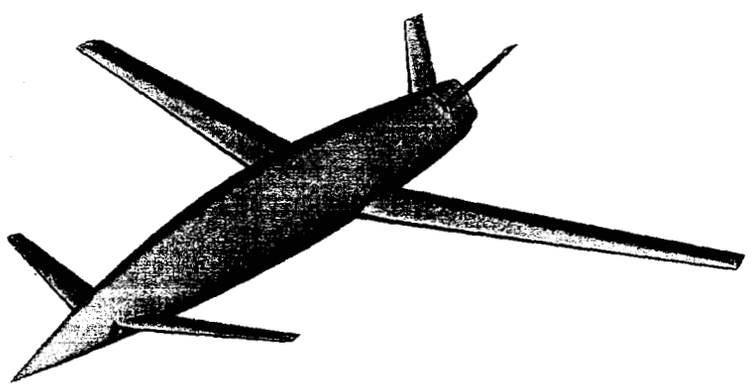
[8] S. E. Cliff, S. D. Thomas, T. J. Baker, and A. Jameson. Aerodynamic shape optimization using unstructured grid methods. AIAA Paper 2002–5550, Atlanta, GA, September 2002.

[9] M. J. Aftosmis. Solution adaptive cartesian grid methods for aerodynamic flows with complex geometries. Lecture notes, von Karman Institute for Fluid Dynamics, Series: 1997-02, Brussels, Belgium, March 1997.

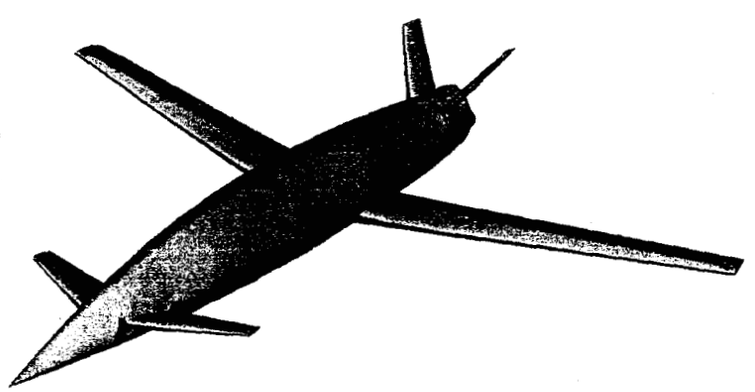
[10] M. J. Aftosmis, M. J. Berger, and J. E. Melton. Robust and efficient cartesian mesh generation for componenet-based geometry. *AIAA Journal*, 36(6):952–960, 1998.



(a) Initial configuration for gradient-based optimization



(b) Final configuration for gradient-based optimization



(c) Final configuration for genetic algorithm

Figure 9: Surface Mach number ($M_\infty = 0.85$, $\alpha = 1°$). Mach numbers above 1.3 are red and Mach numbers below 0.5 are blue.

[11] S. M. Murman, M. J. Aftosmis, and M. J. Berger. Implicit approaches for moving boundaries in a 3-d cartesian method. AIAA Paper 2003–1119, Reno, NV, January 2003.

[12] D. L. Rodriguez. Response surface based optimization with a cartesian cfd method. AIAA Paper 2003–0465, January 2003.

[13] A. Dadone and B. Grossman. Efficient fluid dynamic design optimization using cartesian grids. AIAA Paper 2003–3959, Orlando, FL, June 2003.

[14] S. Obayashi. Aerodynamic optimization with evolutionary algorithms. In R. A. Van den Braembussche and M. Manna, editors, *Inverse Design and Optimization Methods, Lecture Series 1997-05*, Brussels, Belgium, 1997. von Karman Institute for Fluid Dynamics.

[15] N. Marco, J.-A. Désidéri, and S. Lanteri. Multi-objective optimization in CFD by genetic algorithms. Technical Report 3686, Institut National De Recherche En Informatique Et En Automatique (INRIA), France, April 1999. Also see www.lania.mx/~ccoello/EMOO/EMOObib.html.

[16] T. L. Holst and T. H. Pulliam. Aerodynamic shape optimization using a real-number-encoded genetic algorithm. AIAA Paper 2001–2473, Anaheim, CA, June 2001.

[17] J. E. Dennis Jr. and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, N.J., 1983.

[18] A. Jameson. Aerodynamic shape optimization using the adjoint method. Lecture notes, von Karman Institute for Fluid Dynamics, Brussels, Belgium, February 2003.

[19] J. Elliott and J. Peraire. Constrained, multipoint shape optimisation for complex 3D configurations. *Aeronautical Journal*, 102(1017):365–376, 1998.

[20] M. Nemec and D. W. Zingg. Newton–Krylov algorithm for aerodynamic design using the Navier–Stokes equations. *AIAA Journal*, 40(6):1146–1154, 2002.

[21] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1):1–25, 1997.

[22] N. M. Alexandrov, E. J. Nielsen, R. M. Lewis, and W. K. Anderson. First-order model management with variable-fidelity physics applied to multi-element airfoil optimization. AIAA Paper 2000–4886, September 2000.

[23] Y. S. Ong, P. B. Nair, and A. J. Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA Journal*, 41(4):687–696, 2003.

[24] M. J. Aftosmis, M. J. Berger, and G. Adomavicius. A parallel multilevel method for adaptively refined cartesian grids with embedded boundaries. AIAA Paper 2000–0808, Reno, NV, January 2000.

[25] E. F. Charlton. *An Octree Solution to Conservation-laws over Arbitrary Regions (OSCAR) with Applications to Aircraft Aerodynamics*. PhD thesis, University of Michigan, 1997.

[26] J. A. Samareh. Survey of shape parametrization techniques for high-fidelity multidisciplinary shape optimization. *AIAA Journal*, 39(5):877–883, 2001.

[27] J. A. Samareh. Novel multidisciplinary shape parametrization approach. *Journal of Aircraft*, 38(6):1015–1023, 2001.

[28] J. C. Townsend, J. A. Samareh, R. P. Weston, and W. E. Zorumski. Integration of a cad system into an mdo framework. NASA TM 1998–207672, May 1998.

[29] J. J. Moré and D. J. Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Transactions on Mathematical Software*, 20(3):286–307, 1994.

[30] M. J. Aftosmis, M. J. Berger, and S. M. Murman. Applications of space-filling-curves to cartesian methods for cfd. AIAA Paper 2004–1232, Reno, NV, January 2004.

[31] M. S. Eldred, A. A. Giunta, and B. G. van Bloemen Waanders. *DAKOTA, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 3.1 Users Manual.* Sandia National Laboratories, 2003. http://endo.sandia.gov/DAKOTA/software.html.

[32] M. S. Eldred and W. E. Hart. Design and implementation of multilevel parallel optimization on the intel teraflops. AIAA Paper 1998–4707, 1998.