44th AIAA Aerospace Sciences Meeting and Exhibit
9 - 12 January 2006, Reno, Nevada

AIAA 2006-929

A Rapid Geometry Engine for Preliminary Aircraft Design

David L. Rodriguez* and Peter Sturdza†
Desktop Aeronautics, Inc., Palo Alto, CA, 94301

A rapid geometry engine (RAGE) has been developed to allow for preliminary design analysis without labor-intensive CAD support. The geometry tool builds complex aircraft configurations using a component-based approach. Basic algorithms for creating the primary components are presented and discussed. Examples of many widely varying geometry models are shown. A select geometry model is analyzed with several aerodynamic analysis methods ranging in fidelity to further demonstrate the versatility of the geometry tool. Example uses of the tool in optimization problems are also presented. Future plans for the geometry engine are also discussed.

I. Background

Preliminary aircraft design methods have advanced tremendously in the past few decades due to rapidly developing computer technology and overall algorithmic improvements. Analysis methods that were once considered only feasible for advanced and detailed design are now available and even practical at the preliminary design stage. Rapid analysis methods also allow for simple and even multidisciplinary optimization methods to be utilized in preliminary design. To fully exploit these advanced analysis and optimization methods, the geometric model of the aircraft must be easily and rapidly generated so as not to inhibit the preliminary design process.

Current geometry generation methods usually involve computer-aided design (CAD) software, although there are some examples of parametric geometry tools such as Boeing's proprietary tool, GGG[1,2] (General Geometry Generator), NASA's RAM[3,4] (Rapid Aircraft Modeler) tool, and Avid's PAGE[5] (Parametric Aircraft Geometry Engine) application. Unfortunately, CAD-based modeling can be detrimental to efficiency as geometry must usually be generated manually and then converted to something usable by an analysis method. Generating the initial geometry model in CAD can be tedious and time-consuming. Converting the CAD model to something useful to a design analysis method can also be laborious and often problematic. The design process efficiency can be hindered significantly if multiple conversions are necessary for different analysis methods. Another shortcoming of CAD-based geometry models is that they are usually built from splined surfaces as opposed to aircraft-centric design parameters. The parameters that define the splined surfaces are rarely directly related to the parameters that actually define the aircraft geometry (such as the wing sweep or thickness). Ideally, an aircraft geometry model would be directly parameterized to allow for quick trade studies and even design optimization. For all these reasons and perhaps more, CAD-based geometry model generation is far from ideal for preliminary design methods.

Another geometry generation technique commonly used in trade studies or optimization is the perturbation method. The method begins with an analysis model created from a CAD design or some other source. Then, to vary the geometry, this initial model is physically perturbed. For example, beginning with a panel code model of a wing, the local camber of an airfoil section can be altered by making the corresponding local perturbations to the original wing surface. Similarly, geometric changes can be made to a Navier-Stokes computational grid by perturbing the surface and local flowfield grid. Often these perturbations are based on smooth shape functions to keep the modified geometry smooth. This method can be very effective and even efficient for small local geometric changes. However, small perturbations are not enough to address gross alterations (such as moving a wing-mounted pylon outboard or changing a wing's vertical location on a fuselage), which are often necessary in conceptual design. In fact, in some cases the surface grid or panel topology may have to be altered making the perturbation technique very difficult to apply and indeed impractical. Not to mention that after the computational grid or mesh is modified, the changes have to then be communicated back into the CAD system, involving more tedium and approximation of the geometry.

Perhaps a better method for creating geometry models is to use the actual parameters that define the aircraft design in the first place. Parametric geometry generation is the process of creating an aircraft model entirely from the set of defining parameters. For example, a simple straight tapered wing geometry can be generated from a very

---

* Engineer/Scientist, Senior AIAA Member.

† Engineer/Scientist, not an AIAA member.

simple set of parameters: the root and tip chords, the wing sweep and span, and an airfoil definition. This parameter set is near the bare minimum required and of course more parameters can always be included; the twist distribution can be specified, for example. Other possible defining parameters include the airfoil thickness, airfoil camber, and wing dihedral. Internal airfoil section definitions could be incorporated adding a whole new set of parameters, including the way the wing geometry is lofted between sections. The key to this geometry generation process is that the geometry model is mathematically built using the design parameters directly as opposed to creating splined surfaces as done with CAD-based software. While CAD models can be as general as necessary for detailed design, preliminary design methods normally do not require fine detail in the geometry definition. On the other hand, by increasing the number of parameters that describe a geometry model, the model can become quite detailed and even general enough for detailed analysis, as will be shown in examples to follow.

There are many applications for a rapid geometry modeler. As already pointed out, preliminary aircraft design is one of the more obvious examples. If a sufficient parameter set is chosen, an entire aircraft configuration can be modeled and analyzed based on the parameter set alone. During the design process, trade studies on specific individual parameters or even functional relations involving a subset of parameters can be performed very easily. Taking this concept a step forward, the parameters can be used as design variables in an optimization framework. Simple aerodynamic shape optimization and even multidisciplinary optimization become practical with a rapid parametric geometry generator. Another less obvious example of application is the quick generation of marketing graphics.

This paper presents a new rapid geometry engine which employs parametric geometry generation. The basic architecture and algorithms of the RAGE (Rapid Aerospace Geometry Engine) tool are first presented. The capabilities of the tool are then demonstrated by presenting examples of various aircraft geometries generated with RAGE. The application of the tool is also demonstrated in a sweep of aerodynamic analyses of varying fidelity on a selected aircraft geometry. Results from previously published optimization problems that utilized RAGE are briefly cited for completeness. Finally, future development plans for the RAGE tool will be presented and discussed.

II. Methodology

The geometry tool is composed in the Java programming language primarily to help ensure platform-independency. The object-oriented capabilities of Java are fully exploited including aggressive use of abstract and interface classes. This allows developers and even users to quickly add new capabilities and extensions to the tool. In effect, users can customize the rapid geometry engine for specific needs. This is possible because of the component-based approach which is presented in the the following section. The basic underlying algorithms of the primary components available in the RAGE tool are first detailed. The following section then discusses the currently available model output formats and some of the analysis tools that can directly use these models.

A. Primary Components

RAGE uses a component-based approach to generate models; a full aircraft configuration is built from a set of simpler components. Currently, the tool can generate basic aircraft components including fuselages, wings, and nacelles. Many other aircraft components can be generated as special versions of these primaries. For example, tails and pylons can be generated as special wing components. External fuel tanks and inlet spikes can be created as special fuselage components. Most geometry components are built from sub-components; for example, wings are lofted from a stack of airfoils. In all cases, the sub-components can also be customized providing even more flexibility. In general, most parts of an aircraft can be built as either an axially splined body or as a lofted stack of airfoils. The following sub-sections describe the underlying methodology for building these primitive aircraft components.

1. Fuselages (Axially-Splined Bodies)

The axially-splined body is used to create components such as fuselages, rocket bodies, external fuel tanks, canopies, and many other types of components. As will be shown in a later section, the concept of the axially splined body can also be used to generate certain nacelle shapes. The body is built mathematically by creating a stack of cross-sections and then lofting in between these sections to create a smooth shape. The cross section is the primary building block of this type of component.

Cross sections available in the RAGE code vary from simple to complex to general as can be seen in Fig. 1. Perhaps the simplest section is the ellipse where only the width and height become parameters. Another section is the ovate ellipse which introduces a parameter to control “ovateness.” The super-ellipse introduces a parameter (the exponent) that makes the standard ellipse either more square (as shown in Fig. 1) or flatter. The offset super-ellipse includes a parameter that can shift the maximum width of the ellipse vertically. It also allows the upper and lower super-ellipse exponents to be different. Figure 1 shows an example of this type of section which has a flatter ellipse on top, a more square ellipse below, and a downward-shifted maximum width axis. All cross sections can also be split at any location vertically to allow for different widths above and below a wing intersection for example. A gen-

eral cross section shape is also implemented where the user specifies points that are then splined by RAGE (see example in Fig. 1).

The RAGE code creates a fuselage component by taking a stack of cross sections and then axially splining them parametrically. A lofting algorithm is implemented that allows for different kinds of cross sections to be used in the same component. The geometric centers of the cross sections are also splined allowing for fuselage camber in all directions normal to the axis. The algorithm ensures smoothness for all reasonable and even some unreasonable stacks of cross sections. Figure 2 shows the resulting smooth geometry from a significantly varying stack of user-defined cross sections.

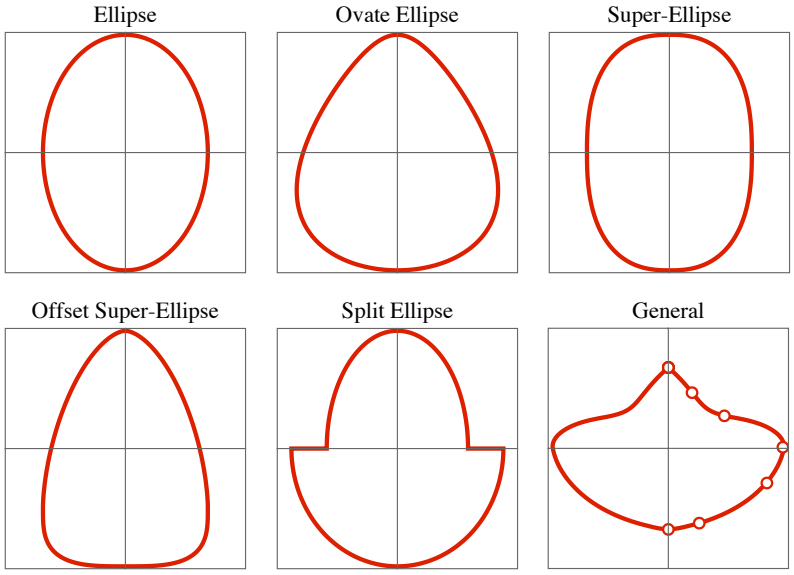


Figure 1. Example cross sections generated by RAGE.

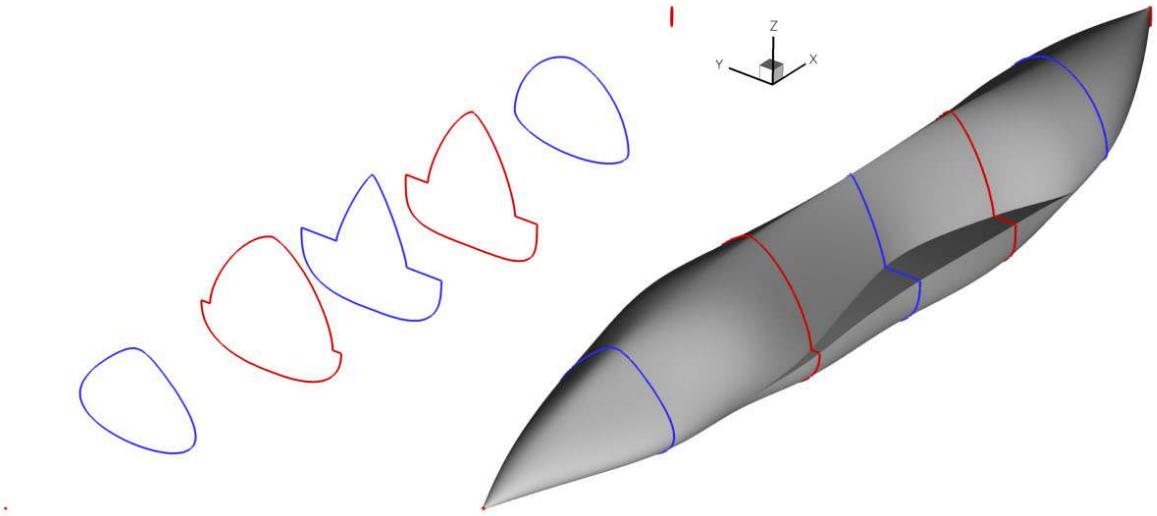


Figure 2. Example smooth fuselage surface generated from a stack of rather different cross section shapes. Note the axially smooth discontinuity boundary created by the split sections where the wing (not shown) intersects.

2. Wings (Lofted Stack of Airfoils)

Wings, pylons, diverters, tails, and other wing-like objects can almost all be generated by this type of component. Wing components are similar to fuselages in how they are built; stacks of airfoil sections are lofted spanwise to create the wing surface. Currently, the lofting algorithm for the wing component is different than that of the fuselage; airfoils are built mathematically and then points on the airfoils are splined spanwise to build the surface. Several lofting methods are available including simple linear lofting and spline lofting for wings with spanwise curvature. Symmetric, asymmetric, and half wings can all be generated. Also, several types of wing cap shapes are available to close off the ends of the wing. Blunt trailing edges are also supported.

Figure 3 shows a wing geometry created using RAGE from a set of initial user-defined airfoils. The inboard section uses a blending of splined and linear spanwise lofting to form a smooth, curved leading edge and a piecewise-linear trailing edge. Other combinations of splined and linear loftings are available. The outboard panel of the wing also has some dihedral which accounts for what appears to be a spanwise discontinuity where the outboard section begins. The inset in Fig. 3 also shows the elliptic wing cap generated by RAGE. The grid is what RAGE

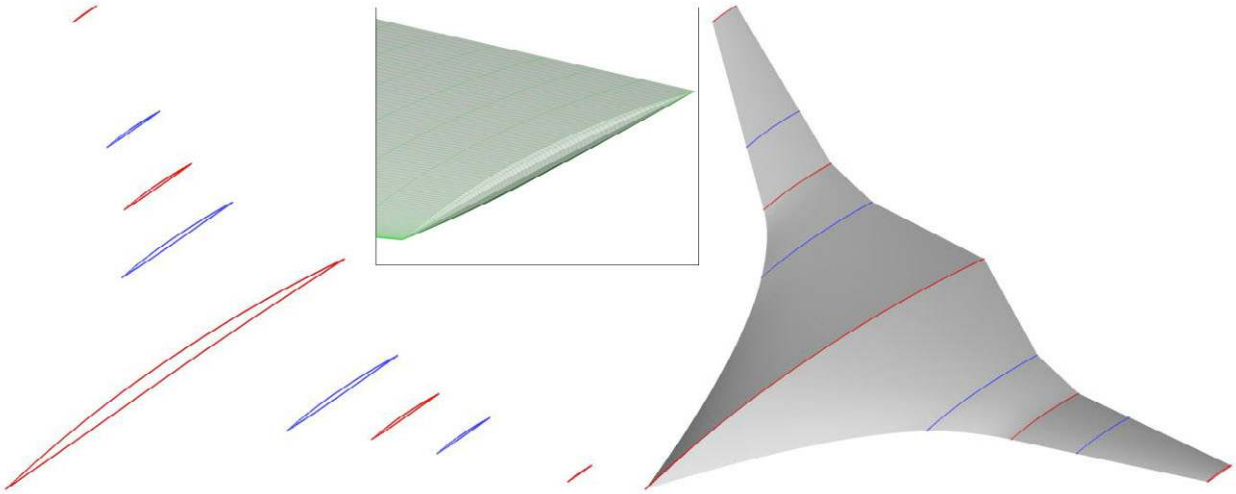


Figure 3. Example of wing lofted from initial user-defined airfoil sections. Note the leading edge is a smooth curve while the trailing edge is a series of line segments. Also note the inset showing the elliptic wing cap.

actually generates in this case, so the cap is actually faceted; more points could be added to the mathematically smooth cap if necessary.

3. Nacelles

In the current version of RAGE, nacelles are built as two separate axially-splined bodies and then attached at the nacelle leading edge. Figure 5 depicts this method with an example supersonic nacelle. The blue cross sections along with the black leading and trailing edges are used to define the interior duct of the nacelle. Likewise, the red cross sections and the black edges define the outer geometry. This method for creating nacelles is ideal for supersonic nacelles since it preserves the sharp leading edge that is normally necessary. Any type of cross section which is available for axially-spline bodies is also available for nacelles. The defining sections can also be yawed or canted about their centers allowing for beveled leading and trailing edges. This capability also allows for two-dimensional inlet design.

Another method for creating nacelles which is being added to the RAGE code treats the nacelle as a circular wing. Airfoil sections are defined at several angles around the axis and then lofted circumferentially to create the full nacelle. This method will be more appropriate for subsonic nacelles because it easily generates blunt leading edges.

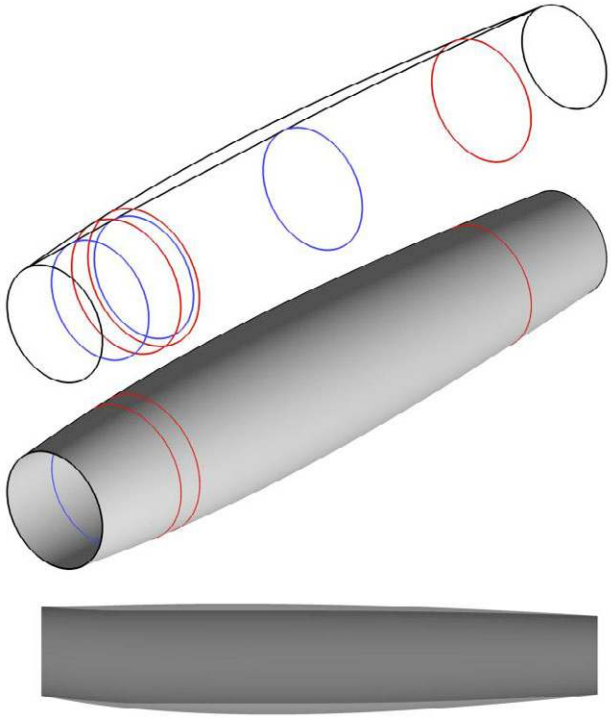


Figure 5. Example nacelle generated from a stack of internal and external sections.

B. Output Analysis Models

In its current infant state, RAGE can output two geometry model formats for aerodynamic analysis. These output formats are geared toward advanced CFD methods. While the RAGE internal geometry model definition is truly a mathematical surface definition, this representation is not terribly useful in discrete CFD analyses. Therefore, the geometry engine is capable of outputting a geometry as an array of points that can be directly input to a CFD analysis method.

One output format is the almost universally used PLOT3D[6] format based on the NASA visualization tool created in the infancy of CFD. The RAGE tool creates a structured grid on each geometry component and then writes

the geometry model as a multi-block surface grid for use in CFD analysis. The spacing of this structured mesh is user-controlled; points can be clustered at the leading edge of a wing for example. Spanwise clustering is also possible. Figure 6 shows a typical PLOT3D output geometry. Note the grid on this model was intentionally kept very sparse for clarity. However, the fineness and even local distribution of the grid points can be easily controlled with RAGE for accurate CFD analysis. Also note the clustering of points at the wing leading edge demonstrating the capability to control grid spacing.

Referring back to Fig. 6, note that the component grids are independent of one another; the fuselage and wing have not even been intersected. Different CFD methods have different requirements for the surface grids. The CART3D[7] Euler analysis tool could use this geometry model directly as the triangulation and intersection of components are performed internally in the code. A structured overset grid could also be created directly from this output model with the exception of collar grids which are usually necessary where components intersect. However, some overset grid generation methods (such as the Chimera Grid Tool[8,9] set developed at NASA Ames) can generate the collar grids semi-automatically from an original non-intersected model such as that in Fig. 6. For this reason, the PLOT3D output model is very useful for running overset Euler and Navier-Stokes methods such as the OVERFLOW[10] code developed primarily at NASA.

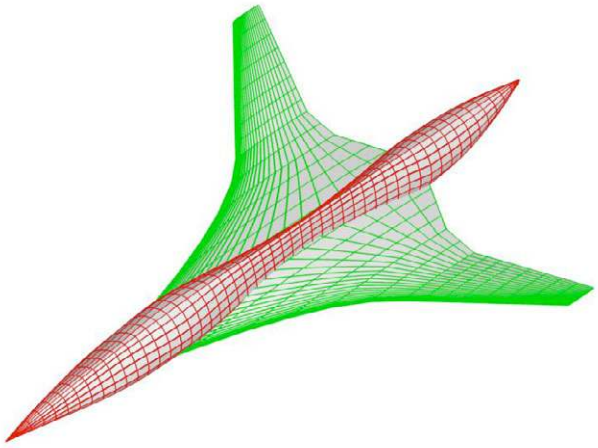


Figure 6. Example PLOT3D output model from RAGE.

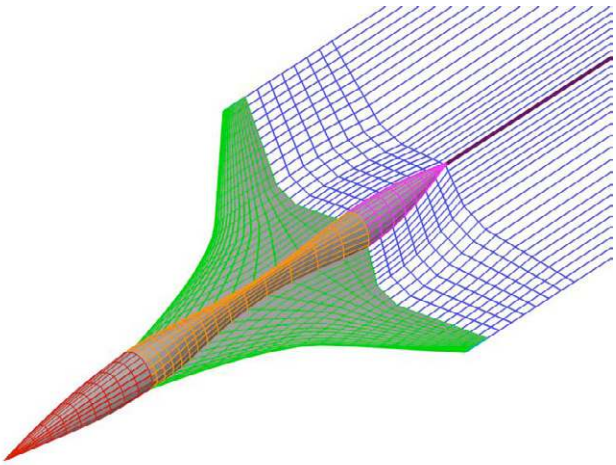


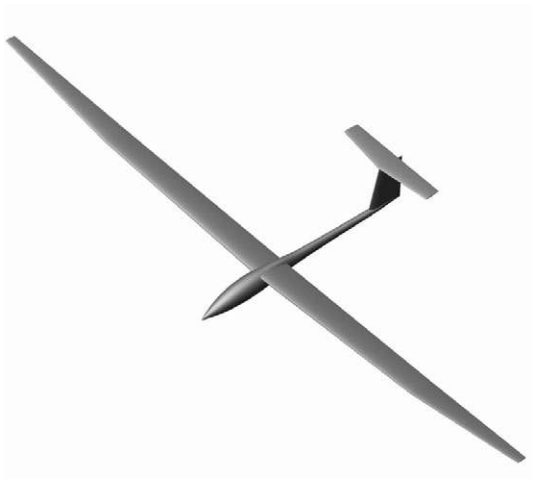
Figure 7. Example A502 output model from RAGE.

Since some aerodynamic analysis codes require intersected geometries, RAGE has the capability to intersect components as well. Panel methods are an example of analyses that require intersected, point-match surface grids. A502[11], also known as PanAir, is one of the few commercially available panel methods that can solve both subsonic and supersonic flows. RAGE can produce intersected geometries for bodies with multiple wings and tails. The various surfaces can be split into multiple pieces (or networks) as required for the panel code. For instance, the wing is typically split into an upper and lower surface, and a separate wing tip cap. Wakes can also be automatically generated behind wing surfaces and stitched along the fuselage. Blunt-body base networks and associated wakes are also automatically added to the geometry. Control of wake placement is minimal at this time; however that is one of the black arts of panel codes, so it is a planned future enhancement to the code.

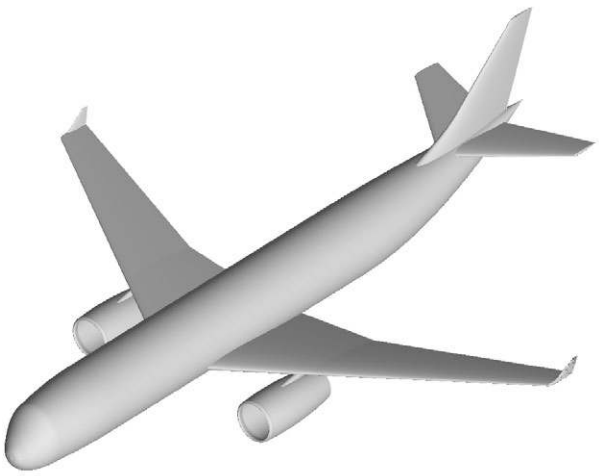
An example of a wing-body A502 geometry is shown in Figure 7, with each network depicted in a different color. RAGE allows significant control of panel spacings in both directions. For instance, on the wing, the usual clustering of panels near the leading and trailing edges is easy to manipulate. The spanwise spacing of panels is also user-adjustable, to allow a tighter spacing near the wing tip or near a break in the wing planform, for instance. If there are breaks in the planform, the wing can also be split into multiple networks in the spanwise direction, if necessary.

III. Applications

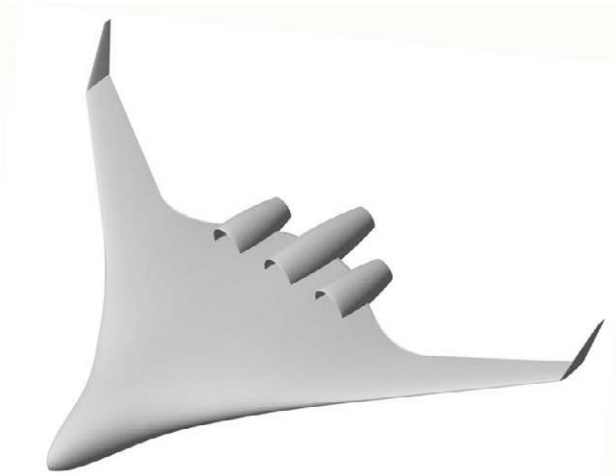
Even though RAGE is in its infancy, the geometry engine is already capable of generating a wide variety of aircraft geometries from the very simple to the very complex. Figures 8a-f give six example geometry models that were all generated with the RAGE tool. Note the fine detail in some of the geometry models including winglets, pylons, and even rocket nozzles. The ability to model fine details allows an engineer (or perhaps optimizer) to perform detailed design of aircraft components. The designer can even analyze the geometry with several methods of varying fidelity. The following sections demonstrate this feature.



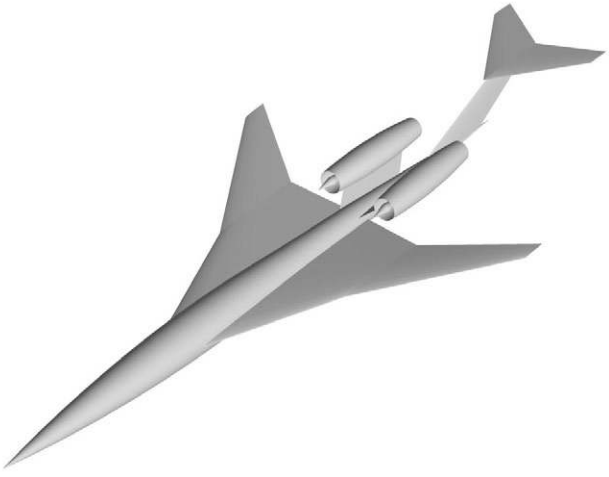
(a) sailplane



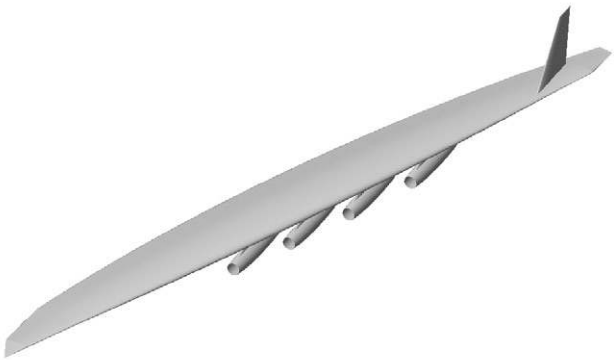
(b) subsonic transport



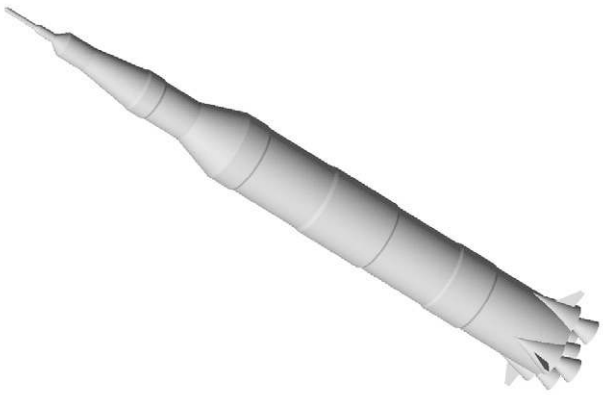
(c) blended-wing-body



(d) supersonic business jet



(e) oblique flying wing



(f) rocket

Figure 8. Example geometry models created by the RAGE tool.

A. Analysis of an Example Geometry

To demonstrate the current output capability of the RAGE tool, one sample geometry model was selected for analysis with several methods of widely varying fidelity. This sample geometry, a swept wing supersonic business jet design, is shown in Fig. 8d and was provided by the work completed in Ref. 12. The full geometry consists of a fuselage, wing, T-tail empennage, nacelles with center-spike inlets, and pylons. Four analysis methods were selected to analyze the geometry model generated by RAGE. As will be discussed in the section on future work, a good deal more analysis and other output models are planned for RAGE.

1. Linear Aerodynamic Analysis

The first analysis of this geometry was performed with LinAir[13], a linear vortex-lattice method. Note that RAGE did not output this model directly, but enhancements are currently being made to allow it to do so. However, to demonstrate the planned wide range of output models, the LinAir model is included here for completeness.

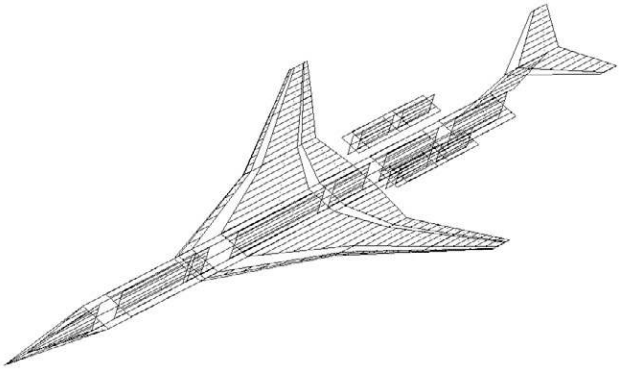


Figure 9. LinAir model of example geometry.

Since LinAir, like all vortex-lattice methods, only models lifting surfaces, it does not directly model wing or fuselage thickness. As shown in Fig. 9, wings and tails are flattened and fuselages and nacelles are modeled with independent vertical and horizontal elements. Also, wing camber, twist and control surface deflections are input as boundary conditions rather than explicitly appearing in the geometry. Most vortex-lattice models require similar simplifications of the geometry. Additionally, there are restrictions on the allowable paneling of the surfaces such as the spanwise spacing of panels on a canard and wing to avoid undesirable interactions between the canard's trailing wake and the control points on the wing.

Due to these large differences in geometry definitions between vortex-lattice codes and higher-fidelity CFD analyses, it is a laborious process to generate LinAir model from a complete 3D geometry, or vice-versa. But with RAGE's parametric description of the geometry the process could be greatly simplified.

2. Nonlinear Panel Method

A panel method (A502) was used for the second analysis method of this demonstration. Unfortunately, the panel method is not robust with nacelles in supersonic flow, so they are not included in this analysis. Wakes were generated automatically by RAGE for the wing and horizontal tail. Fuselage and vertical tail wakes can also be generated if necessary. Wing-fuselage intersections were also automatically performed by RAGE, forcing the wing and fuselage networks to match point for point, even in the wing wake. Additionally, freestream flow properties and reference values can be input into RAGE, permitting a complete A502 deck to be generated. The path from a RAGE-defined geometry to an A502 solution is practically automatic.

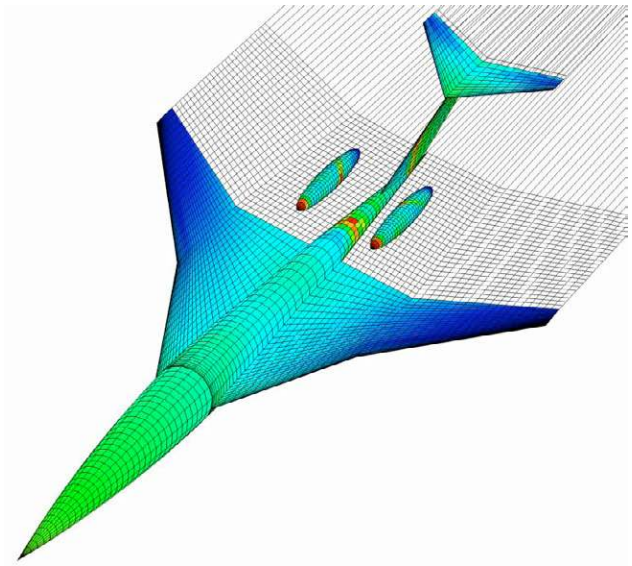


Figure 10. RAGE A502 model and solution on example geometry. Contours are local pressures.

The nacelles for this analysis are built as axisymmetric bodies to at least model their effect on the rest of the configuration. Fig. 10 shows an A502 solution on the RAGE generated model. Note that the wakes actually terminate very far downstream but are not shown entirely for clarity. The contours shown are of surface pressure. As the A502 code improves, nacelles and pylons can be modeled with RAGE as well.

3. Cartesian Euler Method

Increasing in fidelity, the CART3D Euler package was used to analyze the full geometry. Because the CART3D package provides almost automatic grid generation of very complex geometry, the RAGE code is ideal for this type

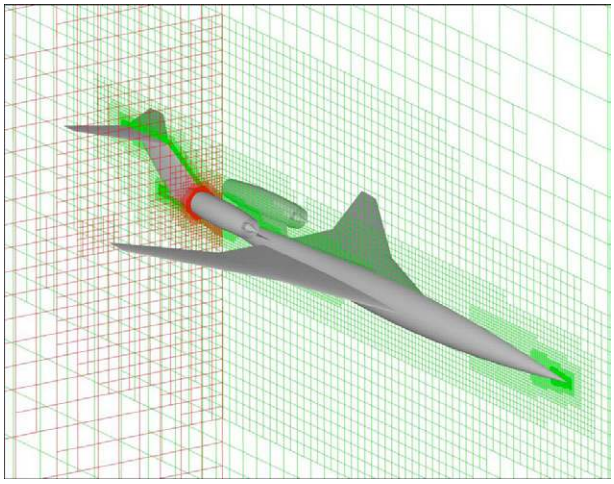


Figure 11. Cartesian mesh generated on RAGE geometry model for CART3D.

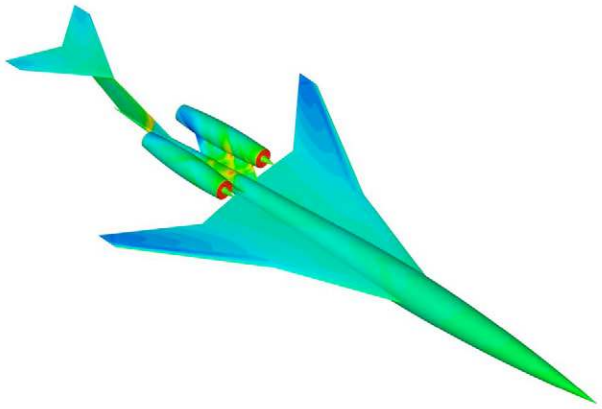


Figure 12. CART3D Euler solution on example geometry. Contours are of surface pressure.

of analysis. The computational grid used for this analysis is shown in Fig. 11. The pressure contours on the aircraft from the CART3D analysis are depicted in Fig. 12.

The flexibility and robustness of CART3D has steered the development of RAGE to be able to model very fine details of an aircraft. This geometry is actually quite simple for CART3D to analyze and was completed in less than an hour from start to finish. The robustness of the combination of RAGE and CART3D makes it a very attractive and powerful preliminary design tool, especially when optimization is involved as will be demonstrated in a later section.

4. Overset Navier-Stokes Analysis

The final demonstration was initiated but not completed to conserve time and computer power. The RAGE code was used to output surface grids for use in developing an overset Navier-Stokes grid to run in OVERFLOW. The resulting grid set is shown in Fig. 13. Note that the Chimera Grid Tool package was used to generate the volume grids and all the collar grids at component intersections. However, the initial surface grids were output directly from RAGE, speeding up the process tremendously. While the grid has not been debugged or even tested, it was completely developed in less than a day using the combination of RAGE and the Chimera tool set. An attractive feature of this method is that surface grids can be quickly altered and regenerated with RAGE. With new surface grids, volume grids can be regenerated in very little time using input files and scripts for the Chimera tool set. This speeds the grid linking process up tremendously, especially since surface spacing can be controlled directly with RAGE.

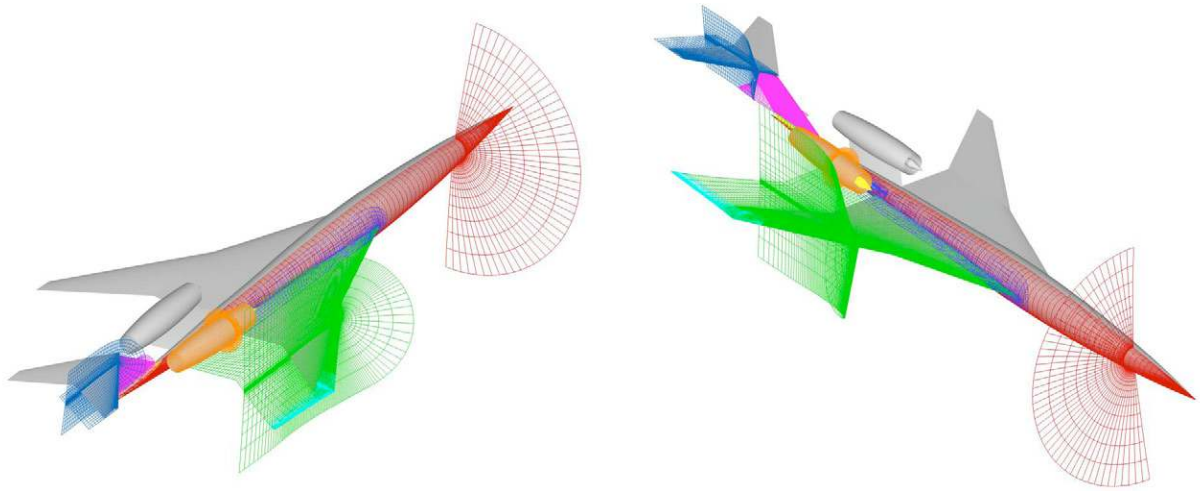


Figure 13. Overset grid system generated on example geometry. Major surface grids were directly output from RAGE while collar grids were generated from these surfaces with the NASA Chimera Grid Tools.

Using RAGE to regenerate surface grids also reduces errors that are introduced when surface grids are re-splined by grid generation tools; the surface grid always remains true to the mathematically defined geometry.

As previously stated, the solution for this Navier-Stokes grid was not completed. OVERFLOW solutions can often be time-consuming and labor-intensive to ensure a quality solution. Several other proprietary geometries have been analyzed successfully with RAGE and OVERFLOW. Work continues on RAGE to improve the quality of these output surface grids. Future versions may even include the ability to generate collar grids speeding up the process even more.

B. Optimization

Perhaps the most advantageous feature of the RAGE tool is its applicability to optimization. Because geometry is parameterized using a fundamental approach, the tool can easily be linked with an analysis method and optimizer of the designer's choice to create very powerful preliminary design tool. Practical design variables are readily accessible streamlining the process; even unique design variables can be selected with little effort. This applicability to optimization is demonstrated in previously published work. Excerpts from these publications are given here for completeness.

1. Optimization of a Supersonic Business Jet

Reference 14 details the optimization of a supersonic business jet using the CART3D Euler analysis method and a response-surface-based optimization method. The baseline design, shown as a RAGE model in Fig. 14, was first generated using the PASS[15] preliminary design tool. A series of optimizations was applied to the geometry, including modifications of the fuselage area distribution, the wing twist and camber distribution, the nacelle and pylon orientation, and the pylon camber. The geometry was optimized for minimum drag at fixed cruise lift. As described in Ref. 13, the response surface method worked well for a limited number of design variables and did lower the drag of the aircraft significantly. Some final results from the optimization work are shown in Fig. 15 as drag polars of the baseline and optimized designs. Note the marked improvement in inviscid cruise drag after the three optimizations were completed. More details of the optimization process and results are given in Ref. 14.

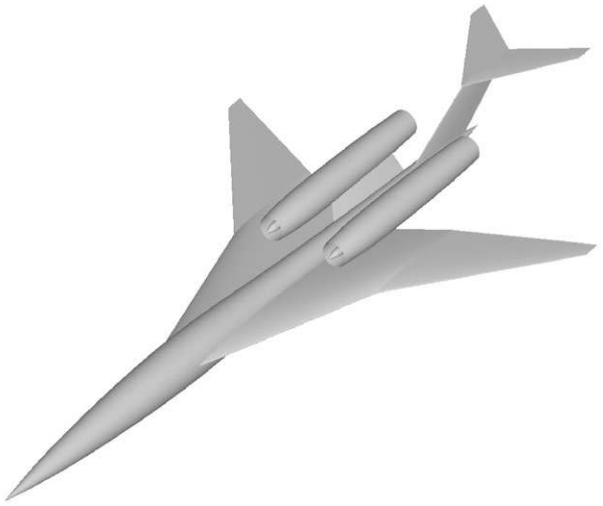


Figure 14. RAGE model of supersonic business jet design used in example optimization.

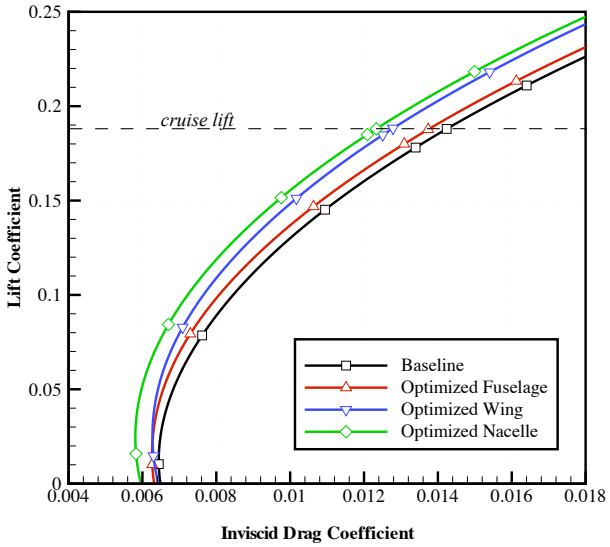


Figure 15. Optimization results on supersonic business jet.

2. Multidisciplinary Optimization of a Supersonic Inlet

The RAGE tool was coupled with CART3D, NEPP[16] (a propulsion simulator), and a nonlinear simplex optimization method to optimize the performance of an isolated axisymmetric inlet. The inlet, which consists of a double-cone spike inside a sharp axisymmetric nacelle, was optimized for maximum performance. Several definitions of this performance were studied as described in Ref. 17, along with all other details of the optimization method and results. The results given here are simply excerpts from this publication. Figure 16 shows Mach contours of the inlet flowfield of two different optimized geometries. The results provided some interesting guidelines for designing

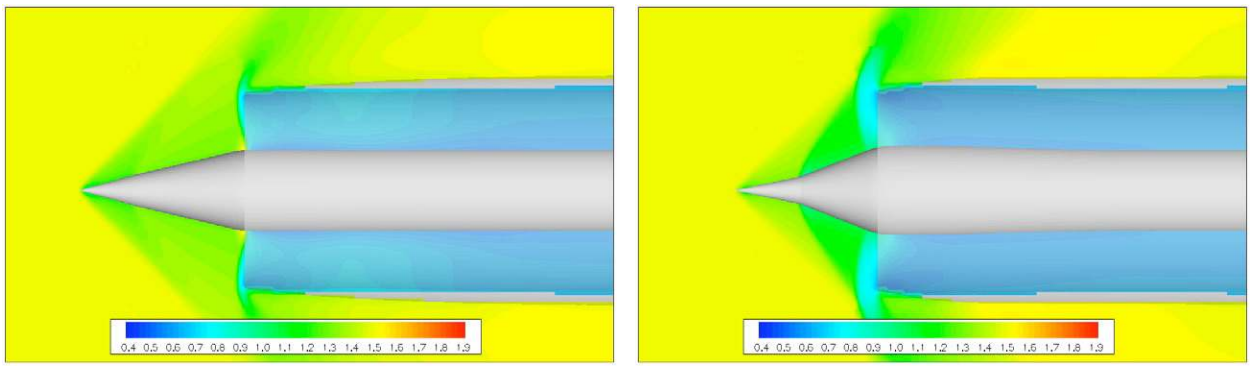


Figure 16. Mach contours on two inlets optimized using the RAGE tool. The left inlet was optimized for minimum fuel burn while the right was optimized for maximum pressure recovery.

spike inlets that operate at low supersonic Mach numbers. This example further demonstrates the versatility of a rapid geometry engine.

3. Multidisciplinary Optimization of an Oblique Wing Business Jet

Reference 12 discusses the application of the RAGE tool in the design of an oblique wing supersonic business jet. PASS and a CART3D-based design process were used to optimize the oblique wing design along with a more conventional design for comparison. During the process, a large number of gross variations on the geometry were necessary necessitating the use of RAGE. An optimized result from Ref. 12 is shown in Fig. 17.

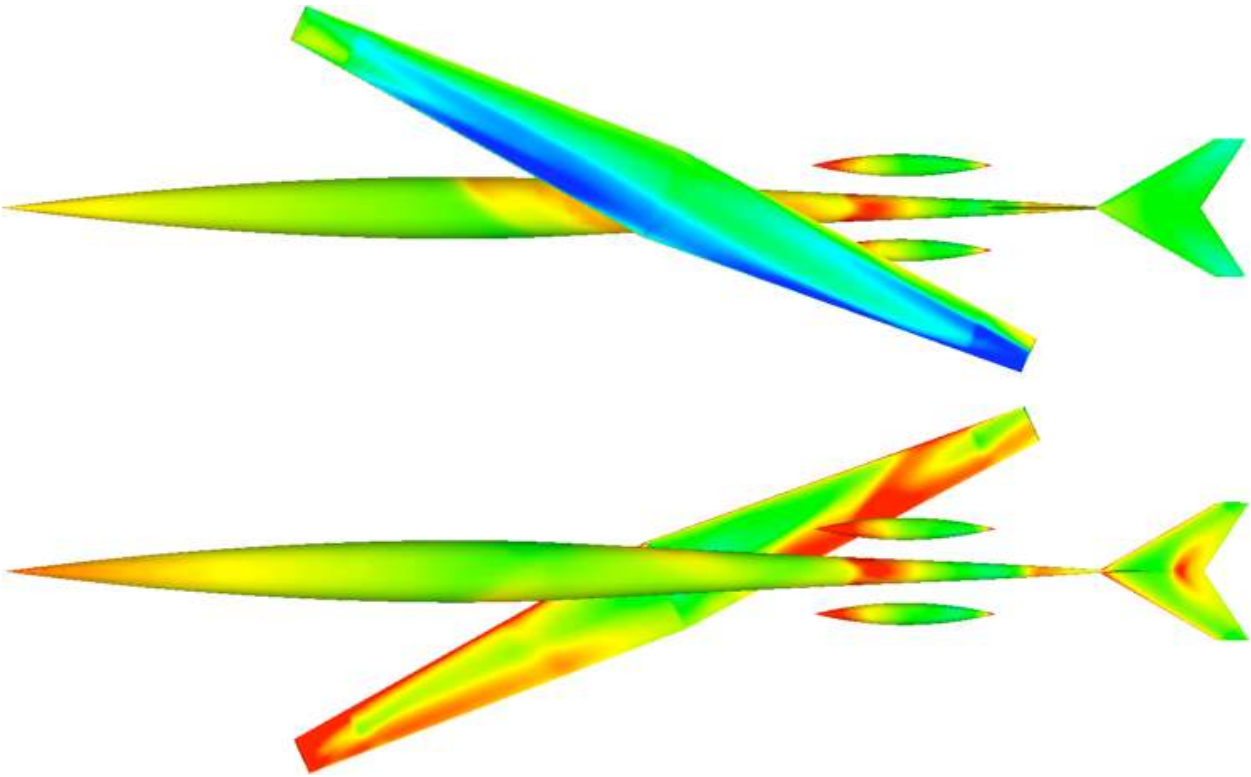


Figure 17. Optimized oblique wing business jet analyzed with RAGE and CART3D. Contours are of surface pressure.

IV. Future Work

RAGE is still a very preliminary code. While the tool is easy to use by its developers, several improvements must be made to make the tool truly versatile and efficient for the average user. This section describes some of the improvements and enhancements that are planned for this tool.

The geometry modeling algorithms will continue to be improved. More options for building geometry components will be added to better model complex shapes. These improvements will be mandatory as new problems and geometries will continue to challenge the limits of the tool. For example, the ability to build fuselages with multiple loftings must be added as manufacturing requirements often affect the actual shape. Occasionally one section of a fuselage must be a straight cylinder while the rest is a smooth, curved surface. This is just one example of an algorithmic improvement that will be added to the tool. The ability to model deflected flaps would be another powerful feature. As users continue to find cases that exceed the limits of RAGE, further enhancements will be incorporated.

The set of available output model types also must be expanded. Ideally, the RAGE tool would be able to output geometry models for most analysis methods available to the preliminary designer. More CFD output models will be added to the code including improved overset surface grids and triangulated surfaces for unstructured CFD methods. An active LinAir model output capability would also prove useful in many instances. Naturally a complex geometry would have to be simplified intelligently to be able to create the LinAir model. Another more ambitious enhancement would be to allow RAGE to output a "CAD-friendly" representation of the geometry. Once a designer feels his aircraft geometry is ready for more detailed design, a geometry file could be quickly created and handed to the CAD expert for further design work.

Of course, RAGE in itself does not have to be limited to aerodynamic analyses. Expanding the method to address other disciplines in aircraft design would prove to be extremely useful. Outputting structural analysis models from a RAGE-defined geometry would be a powerful feature. This capability would allow the designer to perform aeroelastic design work and even optimization with a truly common geometry. Ultimately, RAGE could be expanded to produce models for heat-transfer and perhaps even electromagnetic analysis. The ability to perform different kinds of analysis on one geometry definition is a very intriguing feature.

In terms of user-friendliness, the most useful enhancement would be the addition of a graphic-user-interface (GUI). Since RAGE is written in the Java language, platform-independent toolboxes already exist for creating GUI elements. This feature should allow for a relatively rapid integration of a GUI with RAGE.

To fully exploit the RAGE tool, the aircraft designer would need to apply optimization methods. The addition of an integrated optimization tool set that would work seamlessly with RAGE would indeed prove to be a powerful tool. Ideally, the designer would have a plethora of optimization schemes at his disposal including simplex methods, gradient-based schemes, and genetic algorithms. The user would be able to quickly set up an optimization problem by selecting design variables, objective functions, and constraints with minimal effort. Trade studies and sensitivity analyses would also be possible with this type of approach.

To simplify integrating RAGE with optimizers or other user-supplied codes, an extensible data input interface is planned. Currently RAGE reads its inputs from a human-readable text input file. Some prefer more computer-readable formats such as XML or a variety of databases such as the CAFFE[18] framework for multidisciplinary design. Multiple data input/output interfaces including user-customizable formats will be straightforward to add to RAGE due to Java's object-oriented programming capabilities. Each routine within RAGE that expects input could query a database or possibly trigger a file read and be oblivious to the details of the input format. For each file format or database, customizable reader objects can be instantiated at run-time. This approach would permit RAGE to act as a drop-in replacement for a legacy geometry-generation code in an existing optimization or CFD package It also allows for the same RAGE executable to be integrated with high-fidelity analyses and optimization frameworks in order to guarantee consistent geometry definitions.

V. Conclusions

A powerful geometry engine has been developed primarily for aerodynamic analysis in preliminary aircraft design. The RAGE tool allows the designer to bypass any labor-intensive CAD work and directly analyze the geometry. Currently only a few analysis models are available, but future plans will expand the output capabilities of the tool. Though improvements and enhancements continue to be implemented to the internal geometry algorithms, RAGE can already generate complex geometries of gliders, conventional airplanes, revolutionary aircraft designs, and even rockets. Results from a sweep of analyses of varying fidelity on one geometry model were presented demonstrating its current versatility in terms of analysis. RAGE has also been used successfully in several optimization problems. The rapid geometry engine has proven to be a vital tool for preliminary and even detailed design work by allowing the designer to manipulate and analyze the geometry directly and efficiently. Enhancing the tool to expand capability and improve user-friendliness along with the addition of an optimization tool set will truly make RAGE a powerful tool for preliminary aircraft design and analysis.

Acknowledgments

The authors would like to acknowledge several people who were crucial in the development of the RAGE tool. Professor Ilan Kroo and Desktop Aeronautics, Inc. provided guidance, impetus, and most importantly, funding for this project. Mathias Wintzer of Desktop Aeronautics also helped debug by providing many test cases for RAGE. He is also responsible for the birth of an online documentation detailing the ever-expanding features of RAGE. The Aerion Corporation in Reno, Nevada also continually challenged the limits of the RAGE tool compelling the authors to continue to make significant improvements to the tool.

References

1. Cramer, E. J. and Gablonsky, J. M., “Effective Parallel Optimization of Complex Computer Simulations,” AIAA 2004-4461, August 2004.
2. Bowcutt, K., “ A Perspective on the Future of Aerospace Vehicle Design,” AIAA 2003-6957, December 2003.
3. McCormick, D. J., “An Analysis of Using CFD in Conceptual Aircraft Design,” M.S. Thesis, Dept. of Mechanical Engineering, Virginia Polytechnic Institute, Blacksburg, VA, 2002.
4. Gloudemans, J. R., Davis, Paul C., and Gelhausen, Paul A., AIAA-1996-0052, January, 1996.
5. “AVID LLC - AVID PAGE,” URL: http://www.avidllc.biz/design_tools/AVID_PAGE [cited 5 January 2006].
6. Wakata, P. P., Buning, P. G., Pierce, L., and Elson, P. A., “PLOT3D User’s Manual,” NASA TM 101067, 1990.
7. Nemec, M., Aftosmis, M.J., and Pulliam, T.H., 42nd AIAA Aerospace Sciences Meeting and Exhibit, Jan. 2004.
8. Chan, W. M., “The OVERGRID Interface for Computational Simulations on Overset Grids,” AIAA-2002-3188, June 2002.
9. Chan, W. M., Rogers, S. E., Nash, S. M., Buning, P. G., and Meakin, R. L., “User’s Manual for Chimera Grid Tools, Version 1.6,” NASA Ames Research Center, September, 2001.
10. Buning, P. G., Jespersen, D. C., Pulliam, T. H., Klopfer, G. H., Chan, W. M., Slotnick, J. P., Krist, S. E., and Renze, K. J., “OVERFLOW User’s Manual, Version 1.8s,” NASA Langley Research Center, November, 2000.
11. Carmichael, R. L. & Erickson, L. I., “PANAIR –A Higher Order Panel Method for Predicting Subsonic or Supersonic Linear Potential Flows about Arbitrary Configurations”, AIAA-81-1255, June 1981.
12. Wintzer, M., Sturdza, P., and Kroo, I. M., “Conceptual Design of Conventional and Oblique Wing Configurations for Small Supersonic Aircraft,” AIAA-2006-0930, Jan. 2006.
13. “LinAir 4 User’s Manual,” URL:http://desktopaero.com/manuals/LinAir_4_Manual.pdf [cited 5 January 2006].
14. Rodriguez, D. L., "Response Surface Based Optimization with a Cartesian CFD Method," AIAA-2003-0465, January 2003.
15. Kroo, I. M., “An Interactive System for Aircraft Design and Optimization,” AIAA-92-1190, Feb. 1992.
16. Klann, J. and Snyder, C., “NEPP Programmer’s Manual,” NASA TM-106575, 1994.
17. Rodriguez, D. L., "Multidisciplinary Optimization of a Supersonic Inlet Using a Cartesian CFD Method," AIAA-2004-4492, August 2004.
18. Antoine, N., Kroo, I., Willcox, K., and Barter, G., “A Framework for Aircraft conceptual Design and Environmental Performance Studies,” AIAA-2004-4314, August 2004.