98WAC-56

Experience With a Geometry Programming Language for CFD Applications

David C. Fliegel, Thomas P. Dickens, and Andrew P. Winn
The Boeing Company

ABSTRACT

The Boeing Aero Grid and Paneling System (AGPS) is a programming language with built-in geometry features. Accessible through either a graphical user interface (GUI) or through a command line, AGPS can be used by operators with different levels of experience.

Distributed with AGPS are approximately 300,000 lines of macros, or command files, which automate many engineering design and analysis tasks. Most command files were developed to produce inputs to engineering analysis codes such as A502 [1] and TRANAIR [2]. In many cases, command files have been grouped together in AGPS “packages,” which offer users simple menu pick and dialog options to automate entire engineering processes.

INTRODUCTION

In results-oriented environments common at companies such as Boeing, extensive savings and time reduction can be achieved by automating processes, especially those that are cyclic, or iterative. Additional savings can be realized by having technology experts create processes that can be executed by nonexperts.

In the field of CFD (computational fluid dynamics), there is a continuing desire to achieve a user-friendly CFD process. We at Boeing have made significant progress in making our CFD process more user-friendly through repeatable and reusable procedures. In this context, we use “repeatable” to mean rerunning a procedure with the expectation of getting the same results; “reusable” here means the same procedure can be used on a variety of inputs. Our procedures have automated many iterative processes.

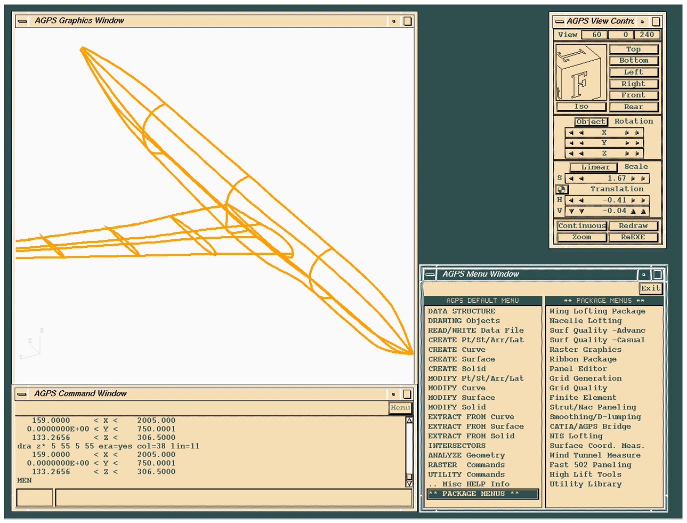
Key to this CFD-automation at Boeing is the use of the Aero Grid and Paneling System (AGPS) geometry language. Over its nearly 20-year evolutionary life, AGPS has proved time and again the strength in providing a general-purpose geometry language to the engineering community. AGPS can be easily used to generate procedures, called command files, that walk users quickly through very complex, geometry-intensive processes. A collection of these command files can be grouped into an AGPS “package,” which automates a CFD process so that it can then be performed by nonexperts. Once a configuration of geometry has been run through a package, modifications to that geometry can be incorporated and very quickly rerun through the process.

In this paper, we provide a brief overview of the AGPS geometry language and our CFD processes, and then detail a number of CFD-related AGPS packages.

SYSTEM OVERVIEW

The Aero Grid and Paneling System (AGPS) is an interactive, three-dimensional, UNIX-based, parametric surface geometry system [3]. It features geometry, grid, and graphics capabilities necessary for effective and rapid CFD code use, including several curve and surface generation options, curve and surface intersectors, raster imaging, structured grid extraction, several supported input formats, and a flexible grid output capability for interfacing with engineering analysis codes [4,5].

AGPS provides a geometry programming language featuring more than 170 commands. The program offers two interactive modes of interface: textual and graphical (Figure 1).



```
   159.0000      < X <    2005.000
  0.0000000E+00 < Y <    750.0001
   133.2656      < Z <    306.5000
dra z* 5 55 5 55 era=yes col=38 lin=11
   159.0000      < X <    2005.000
  0.0000000E+00 < Y <    750.0001
   133.2656      < Z <    306.5000
MEN
```

Figure 1. AGPS Graphical User Interface

In the graphical interface, Users may enter commands interactively through the menu (the figure's lower right window) or through the command input line in the text window (lower left). As AGPS executes a command, it echoes the command to the text window. Executed commands may be recalled easily to the command line, where they may be edited if desired and executed again. Users may render selected geometry for viewing in the Draw window (upper left), and choose an orientation for viewing by manipulating the Viewcube (upper right). By clicking with the mouse on the various Viewcube buttons, users choose scale, viewing angle, and translation vertically and horizontally with respect to the viewing perspective.

If the graphical interface is not desired or is unavailable, AGPS offers a text mode command line, which also features a simple text mode menu. More detailed information about the AGPS user interfaces may be found in the references [3,4,5].

PROGRAMMING IN AGPS

The AGPS language capability allows sequences of steps to be recorded in a command log file and replayed. Users may also capture interactive graphical AGPS sessions through a journaling capability, in which screen picks as well as commands are recorded. The parametric nature of the internal data structure allows users to swap new geometry or design parameters into the recorded log or journal file in place of the original geometry or design parameters and rerun the entire command sequence.

The AGPS programming language uses many features common to other computing languages, such as DO loop statements, IF-THEN-ELSE and WHILE constructs, CASE statements, numeric and character symbols, trigonometric functions, and subroutines. These features can be combined with AGPS commands to yield concise, efficient, automated macro instruction sets we call "command files," as in Figure 2. Users can customize the menus in their command files to create sophisticated, standalone, interactive command procedures we know as "packages." Command files and packages automate common design processes, thereby allowing repetitive tasks that were once tedious and time consuming to be accomplished quickly and easily.

```
!  CURVE.COM
!  USAGE:
!@CURVE(ZDATA=DATAIN,ZCURVE=CUROUT)
!  PURPOSE:     Fits curves to strings
!  INPUT:       ZDATA = List of strings
!  OUTPUT:      ZCURVE = List of
!  curves fit through strings
$CALL GET_TYPE(ZDATA,NT)
$IF NT < 0 THEN
   $WRITE ' ZDATA DOES NOT EXIST'
$ELSE
   $CALL GET_LENGTH(ZDATA,NS)
   $IF NS < 1 THEN
      $WRITE 'LIST ZDATA IS EMPTY'
   $ELSE
      CLS ZCURVE F=
      $FOR I=1 TO NS DO
         $CALL GET_TYPE(ZDATA.<I>,NT1)
         $IF NT1 <> 9 THEN
            $WRITE I, 'ANCESTOR OF \
            ZDATA IS NOT A STRING'
         $ELSE
            FCV QCURVE ZDATA.<I> CUBIC
            ATL ZCURVE QCURVE
            REN QCURVE
         $ENDIF
      $ENDDO
   $ENDIF
$ENDIF
```

Figure 2. Elements of a Typical Command File

AGPS instructions are categorized by type: system directive, comment, command file, or command. Generally, commands create or manipulate geometry objects. System directives create or manipulate symbols, which store numeric or text data. Language constructs such as FOR-DO-ENDDO and IF-ELSE-ENDIF are system directives.

The initial character of each line indicates which type of instruction follows: a "$" for system directives, "!" for comments (comments are ignored by AGPS), or "@" for executable command files or log files. All other lines are assumed to be commands and thus need no distinguishing initial character. The first three characters of command lines are abbreviations of the full command names; e.g., CLS stands for the Create-List command. Character strings following the command abbreviations are parsed into the individual command keywords. The language allows for flexible keyword input order.

Note in the command files example the incorporation of the parametric data structure through the "." notation. This convention allows objects to be specified not by their names but by their logical relationships. We call these "ancestor" relationships; thus "ZDATA.<I>" refers to the ith ancestor of the object "ZDATA." At run time the user can swap any object in place of ZDATA by using a simple string substitution. All other objects are then specified by their ancestor relationships to ZDATA or its replacement.

Example string substitutions for ZDATA and ZCURVE are demonstrated in line three of the command file in Figure 2.

THE DESIGN CYCLE AND AGPS

The design engineer is often faced with an iterative process of configuration optimization. This optimization process cycle (Figure 3) involves four basic steps [4,5]. The process begins with surface geometry definition, the output of which is geometry representing the initial surface configuration. AGPS has sufficient geometry creation capability to serve as the principal geometry tool in our CFD process; many other geometry packages could serve in this capacity.

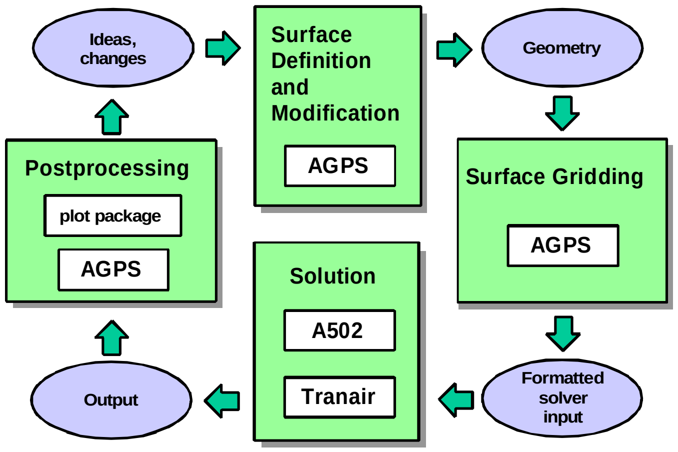


Figure 3. CFD Design Process

The second step is surface gridding, or extracting from the surfaces a set of points that represent those surfaces. Where field grids are desired the gridding process is more complicated, but AGPS is used in field grid preparation. We discuss this below, and more may be found in the references [6,7].

The gridding output is a text file, which is formatted to match the engineering analysis code input specifications. Here AGPS offers two advantages over many other packages. First, the points that AGPS extracts lie identically on the surfaces they represent; no approximation is involved, enabling accurate representation of the real geometry. Second, AGPS can be programmed easily to write the points to almost any output format. This flexible output file formatting capability allows AGPS to communicate geometry to virtually any analysis code.

The third step, analysis, can be performed anywhere, on such diverse platforms as a remote supercomputer or a local workstation. Its product is a formatted output file, which is then fed to one or more postprocessing packages. AGPS can be useful in postprocessing through its raster graphics capability and solution interpolation commands.

AGPS reads several standard neutral output file formats, making it compatible with many CFD codes, among them A502 (the Boeing version of the PANAIR Technology Program, or PANAIR) and TRANAIR. AGPS can also read free-field format. This capability, when combined with other AGPS commands to interrogate and reformat free-field input, enables AGPS to interpret unfamiliar formats without requiring extensive source code modification.

The result of postprocessing is either the engineer's satisfaction with the current geometry design, or more often, ideas for an improved design. If improvements are desired the design cycle begins again with modifications to the original geometry.

In the Boeing AGPS development and support group, we like to say that the first configuration is designed "by hand," meaning that every step requires user input and direction. Sometimes only one such design cycle is needed to converge on an acceptable configuration. In such a case, process automation yields minimal savings. When multiple iterations of the design cycle are necessary, though, automation is crucial to productivity.

AGPS users have been writing command files and packages to automate iterative design processes like the one shown in Figure 3 for more than 15 years. Through its links to the workstation operating system, AGPS can even be programmed to drive the entire design cycle without user intervention, as we will show. In the following material we detail several AGPS command files and packages that illustrate the program's usefulness in reducing design cost and cycle time.

AGPS PACKAGES

Distributed with AGPS are approximately 300,000 lines of command files (macros) that are used to rapidly perform tasks automatically. The majority of these command files are "packages" (groups of command files) developed to produce inputs to CFD flow codes (e.g., A502 [1] and TRANAIR [2]). These packages accept surface lofts and point distribution information, grid the surfaces, and produce formatted flow solver input files. Total turnaround time in most cases is less than an hour. This section will look at some of these packages in use today and others currently under development.

AUTOMATED PANELING PACKAGES.

AGPS includes a number of procedures organized as sets of custom menus that automate the paneling process for CFD.

Wing/Body/Tail Paneling

In the early phases of airplane design, there is considerable need for rapid paneling of cruise wing/body/tail configurations. Without an automated procedure, the aerodynamicist would need to generate paneling by hand for each small modification to the design. This process would not allow the designer to look at many configurations in the short time period allotted. The Wing/Body/Tail package provides very rapid paneling for

A502 or TRANAIR for subsonic or supersonic aircraft to address this problem.

This AGPS package allows users to generate paneling for A502 or TRANAIR input in as little as 5 minutes, allowing for many iterations on the design (Figure 4). It can be used with wing/body, wing/body/horizontal-tail or wing/body/horizontal- and vertical-tail surfaces. The user needs only to provide the input surfaces and an optional spacing. If the user has no spacing file, he or she will be interactively prompted to provide spacing inputs that will be saved into a spacing file that can be easily edited and rerun if necessary. With the spacing file concept, the user has the ability to tailor the paneling and to modify it rapidly. All point arrays are guaranteed to have outward-pointing normals and exact abutments (required for input into the solvers). Trailing wakes off of the wing and body are generated automatically, and the user has the option of creating boundary layer wakes if needed. The user can also select whether to prepare input decks automatically.

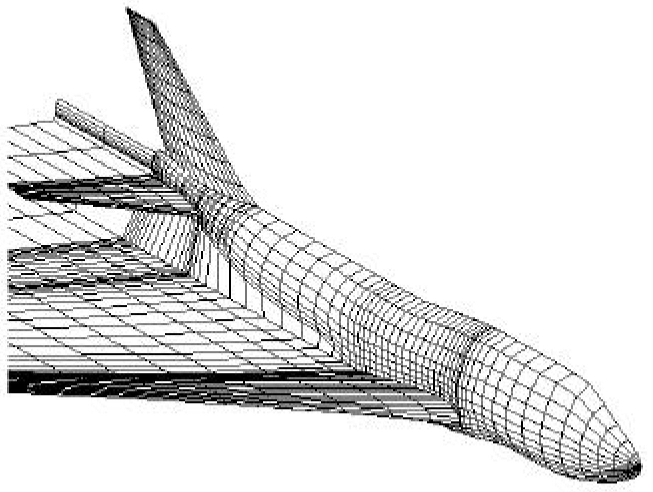


Figure 4. Paneling From Wing/Body/Tail Package

Variations of this package handle other configurations. For example, one version produces paneling that contains wing leading edge and trailing edge flaps, horizontal elevators and/or rudder cutouts. Also available is the ability to handle a wing in which the lower surface passes beneath the body surface. Currently in work is the capability to handle a high-wing design as well as a T-tail.

Strut/Nacelle Paneling

Once the wing/body/tail paneling is complete, the next phase of design is to incorporate struts and nacelles. Without an automated procedure, this process is extremely difficult and time consuming. Before the Strut/Nacelle Paneling package was available, each change to the strut or nacelle definition required weeks of work to hand-generate new paneling. The Strut/Nacelle Paneling package automates this procedure so that the aerodynamicist can rapidly analyze many different designs. The package is designed to take wing/body paneling from the Wing/Body/Tail package and add one or two engines and struts to the generated half model paneling (Figure 5).

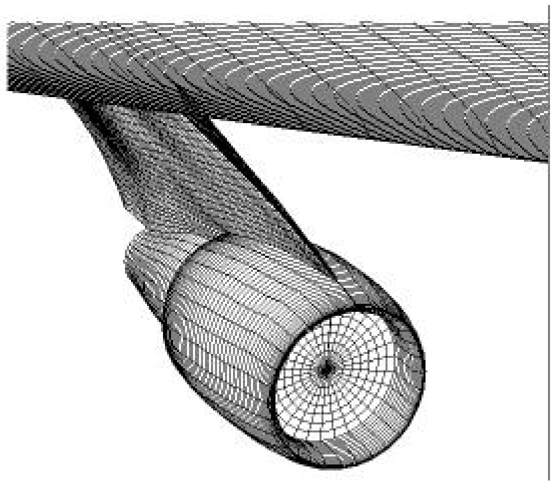


Figure 5. Paneling From Strut/Nacelle Package

The user can run either in an interactive environment, which allows him or her to step through each section of the paneling, or in a batch mode, by providing inputs in a file and completing the paneling automatically. Example input files are available for the new user as a starting point and are easily modified to use different options. Many options are available for differing configurations. For example, the user could have an inboard strut that intersects the wing surface completely on the lower wing surface while the outboard strut wraps onto the upper wing surface.

The inputs to the Strut/Nacelle Paneling package are the wing/body/tail paneling, along with the wing/body surfaces, and, for each engine, a strut surface, interior and exterior nacelle surfaces, interior and exterior primary (core) surfaces, and an optional plug surface. Fan and core plumes are generated automatically to maintain constant area cross sections. An optional text input file can be used to automate interactive portions of the procedure.

The procedure checks all surface orientations and corrects them if necessary. The wing paneling is modified to incorporate the intersection with the strut surface. The strut paneling is defined by the existing points on the wing and the points are carried over to the nacelle, allowing for exact abutments. Fan faces and exits are generated automatically, as are strut and plume wakes. The wing wakes are modified to handle the addition of the struts and nacelles. Input decks can be generated automatically. All these features allow the user to generate complete input decks in as little as 30 minutes.

Using the Strut/Nacelle package in combination with the Wing/Body/Tail package allows the nonexpert AGPS user to generate complete CFD input decks with wing/body/tail, struts, and nacelles in less than an hour. Again, without these automated procedures, analysis time would be, and has been, on the order of weeks.

Fast502 Paneling

The packages described above are useful for the aerodynamicists working in high-speed applications, but engineers working the low-speed or high-lift configurations need additional procedures. Recently a new package was developed to address these requirements: Fast502 (A502 is the Boeing version of PANAIR, hence the acro-nym Fast502). The Fast502 High Lift Paneler (shown in Figure 6) is an interactive, 3D, commercial-transport-oriented, high-lift-configuration paneling package. It can currently handle wing/body/ strut/nacelle configurations with leading edge devices, single- or double-slotted flaps, sealed or gapped leading edges, gapped trailing edges, and two or four engines. The package was originally developed to support preliminary design trade studies and rough performance estimates, but it is also useful for preparing models for many other tasks that require 3D flow field information.

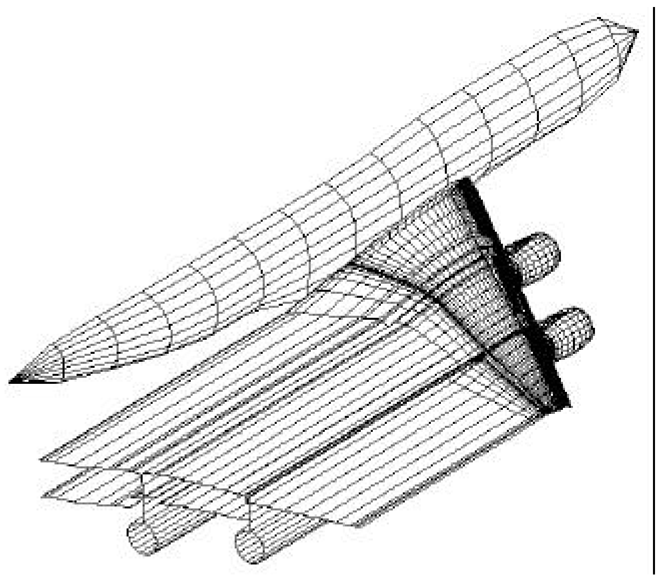


Figure 6. Paneling From Fast502 Package

The Fast502 Paneling package can produce a simple 3D model with only days of initial setup time for typical configurations. Incremental changes can often be made in minutes. Without these procedures, it used to take weeks to months to panel each configuration.

Autopaneling Package

One issue that the packages described above do not address is the problem of having a large number of surfaces define the configuration. Those packages assume that the user has surfaces in a specific form, e.g., a single wing surface and a single body surface. Often the engineer is provided instead with dozens or hundreds of surfaces. This typically happens when surfaces are imported from a different CAD system. We have seen approximately 90 surfaces defining a fighter configuration and more than 700 surfaces defining a helicopter!

Sometimes it is impossible or impractical to put these configurations into a form our current automated procedures can handle. This leads the engineer to the unenviable task of trying to generate paneling by hand. This process can take months, obviously not allowing for minor changes to configurations to be analyzed in any kind of timely manner. The process of cutting surfaces and relofting to provide the needed inputs can lead to approximation errors.

The AGPS team has developed a new package to address this problem. The only assumption in this Autopaneling package is that the surfaces are trimmed so that there are no overlaps. The inputs are a list of surfaces and two tolerances: a surface gap tolerance and a grid spacing. The advantage of this package is that it distributes grid points exactly on the original lofted surfaces with no approximation.

The first part of the Autopaneling package defines a matrix of topology information, describing which surfaces are attached to which other surfaces and which edges of the surfaces need to match the others. From the matrix of topology information and the point distribution information, a linear programming code returns point distributions for each edge. The point distributions are then passed back to the command file to generate surface grids for each surface. Adjoining edges are automatically point-matched and ready to be output in whatever flow code format is needed.

This new Autopaneling package will provide AGPS users with the capability to rapidly panel any configuration imaginable without needing to combine surfaces to fit existing automated procedures. As an example of the time savings this procedure provides, a fighter configuration consisting of 90 surfaces was paneled (Figure 7) in less than an hour using this procedure; the task would normally have taken months by hand.

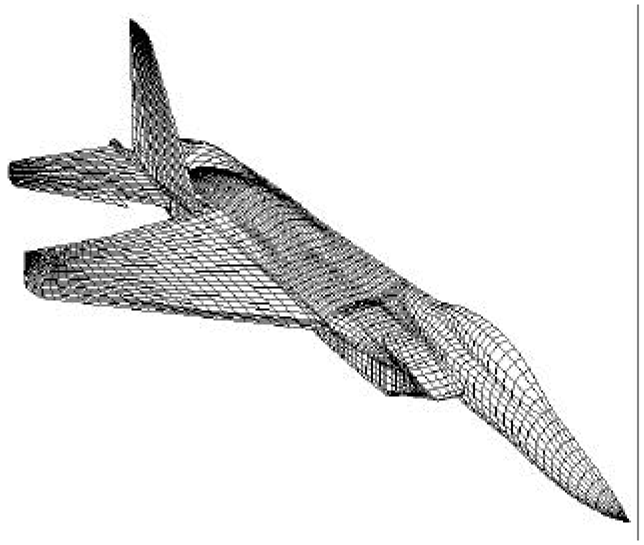


Figure 7. Paneling From Autopaneling Package

Navier-Stokes Package

The above packages are generally used to define surface grids or paneling for panel codes. A new package is being developed to address the need for volume grid generation for Navier-Stokes flow codes. The package will rely on surface grids generated from the above procedures (Wing/Body/Tail and Strut/Nacelle packages). From this definition, the Navier-Stokes package will automatically generate block faces and topology mapping information to be output to a volume grid generation code. Expected savings from using this package is a reduction in the time needed to generate volume grids from months to a couple of hours.

GENERAL PANELING TOOLS

Sometimes it is necessary to create or modify existing arbitrary paneling or grids by hand. There are two packages in AGPS that address this issue. The Panel Editor package allows simple panel modification or array abutments to be done in an easy-to-use manner, and the Grid Generation package provides more than 200 procedures to do anything from surface and curve definition to surface grid or volume grid generation.

Panel Editor Package

Occasionally, the AGPS user needs to hand-modify an existing paneling to fix abutments or to add or delete rows or columns in existing arrays. The Panel Editor package provides a very simple process for performing these tasks through a menu system. This enables the user to perform common tasks rapidly without having to understand the command language of AGPS. The package consists of various tools that make it possible to read and write data files and modify existing arrays. In the example in Figure 8, the two arrays on the left ("Before") have been modified ("After") so that they match along the abutment.

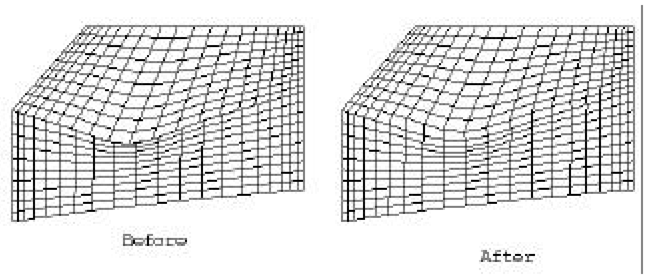


Figure 8. Correction of Array Abutment Using Panel Editor Package

Grid Generation Package

The AGPS Grid Generation package makes use of the grid extraction and modification features of AGPS and uses interactive graphics for selection of objects. The primary functions of the Grid Generation package include:

1. Defining block edges with a wide variety of distribution functions.
2. Generating surface grids and volume grids (Figure 9).
3. Generating block face field grids.

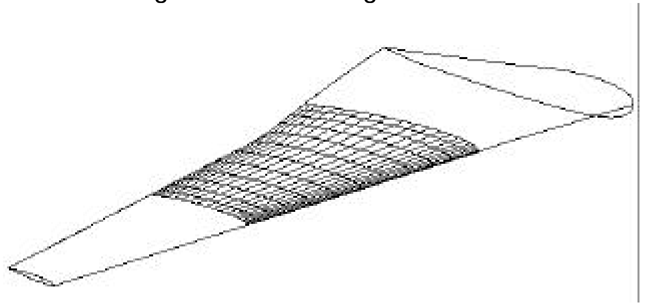


Figure 9. Surface Paneling From Grid Generation Package

This package can be used by a novice for general paneling or grid-generation tasks that are not currently handled by automated procedures. Inputs to the package can be from AGPS save files, IGES files, or existing grid files. The user can choose from more than 100 menu selections to perform various tasks, from typical CAD-type surface definitions to surface grid or volume grid generation. There are also several customized solver interfaces, along with options to save the current session so that the user can return to the session later on and pick up where he or she left off.

POSTPROCESSING

Many grid generation programs do not provide the tools needed for postprocessing or visualizing of CFD results. AGPS supports all aspects of the CFD process (except the actual CFD solution), from geometry definition to surface and volume grid generation to postprocessing. Once the flow solution is produced, the user can read the results back into AGPS and generate color fringe plots displaying any of the properties contained in the solution file (Figure 10). For example, the user can read in pressure values and display them as a color property overlaying the paneling. Other capabilities include contour plots, field grid properties, isosurfaces, vector fields, and stream lines.

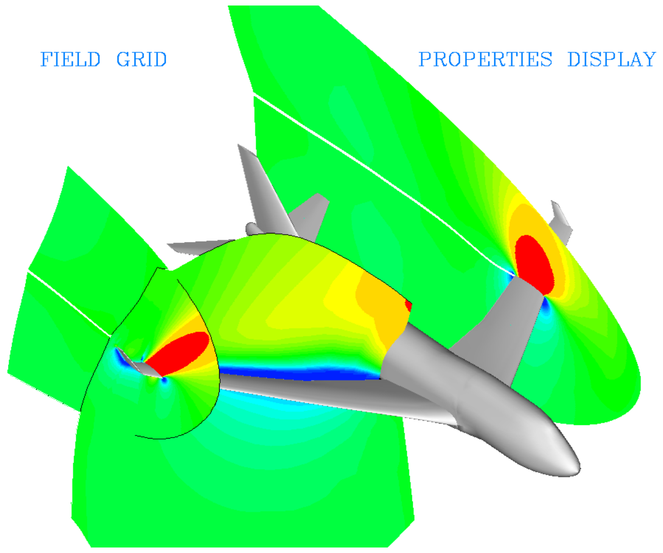


Figure 10. Field Grid Properties

FURTHER AUTOMATION OF THE CFD PROCESS

HIGH LIFT TOOLS PACKAGE

Further cycle time reductions in the CFD design process can be achieved by having AGPS drive the flow solver; one place we do this is in the High Lift Tools package. This package addresses the needs of the aerodynamicist to run other 2D and 3D CFD analysis tools. It provides tools for the high lift designer to manipulate geometry, run a flow solver, and graphically evaluate the solutions without needing to know the specific details of the input and output files for each code.

The package is divided into 2D Analysis Tools, 3D Analysis Tools, 3D Lofting Tools, and 3D Positioning/ Cutting Tools. The 2D Analysis Tools menu provides the capability to manipulate 2D geometry, run 2D-airfoil CFD codes, and graphically display the results. The 3D Analysis Tools menu provides similar capabilities to work with 3D geometry and run 3D codes; presently a vortex-lattice method is supported. The 3D Lofting Tools menu provides a set of tools for quickly creating a set of high lift surfaces given a wing surface and the required planform corner point information. The 3D Positioning/ Cutting Tools menu provides many different tools for positioning and cutting high lift surfaces.

TOTAL PROCESS AUTOMATION

In some cases, postprocessing and geometry updating can also be automated, resulting in even greater cycle time reduction. Some work has already been done in this area. Thomas Dickens and Arvel Gentry devised a system to use AGPS as a real time geometry monitor, manipulator, and interrogator for other codes [8]. They coupled AGPS directly with the flow solver TRANAIR, which has an inverse design capability.

In a design phase, TRANAIR makes small deformations to a seed geometry grid to produce a new geometry that drives the flow solution toward predefined constraints, in this case, surface pressure distributions. To achieve large geometry deformations, the solver's output grids are fed back into AGPS after each design iteration, and new surfaces are fit to them. The new surfaces are gridded and sent back to the flow solver. The cycle repeats until the initial input flow constraints are met.

Before the coupling of AGPS and TRANAIR, users regenerated surface grids themselves following each design phase, then reran TRANAIR to update the solution. The new procedure allows users to set the constraints once and let AGPS and TRANAIR do the rest of the work automatically, reducing the total cycle time from days to hours.

CONCLUSION

We have outlined the AGPS geometry programming language and have provided examples of using the programming capabilities of AGPS to create CFD-supporting procedures. These AGPS "packages" are used to automate the CFD process, and have contributed to greatly reducing design cycle time.

Packages that address specific areas of the CFD process have been presented, including wing/body/tail paneling, strut and nacelle paneling, and A502 and Tranair input file generation. The autopaneling capability allows many-surface definitions to be used in the CFD process. Volume grids are generated using the Navier-Stokes package. More general capabilities are offered in the Panel Editor and the Grid Generation packages. CFD postprocessing is also available through AGPS packages.

We have experienced favorable results in taking this approach in process automation for CFD tasks. Further work continues to create AGPS packages to address other specific areas in the field of CFD, as well as work in other geometry-intensive applications.

ACKNOWLEDGMENT

The authors thank Charlene Scammon for her graphics support, helpful suggestions, and thorough editing of this paper.

REFERENCES

1. Saaris, Gary R., “A502I User’s Manual—PAN AIR Technology Program for Solving Problems of Potential Flow about Arbitrary Configurations,” Boeing Document No. D6-54703, 1992.
2. AIAA 87-0034, S. S. Samant, et al., “TRANAIR: A Computer Code for Transonic Analyses of Arbitrary Configurations,” AIAA 25th Aerospace Sciences Meeting, Reno, Nevada, January 12–15, 1987.
3. AIAA-87-2902, David K. Snepp and Roger C. Pomeroy, “A Geometry System for Aerodynamic Design.”
4. AIAA-91-0800, W. K. Capron and K. L. Smit, “Advanced Aerodynamic Applications of an Interactive Geometry and Visualization System.”
5. Arvel E. Gentry, “Requirements for a Geometry Programming Language for CFD Applications,” from Proceedings, Software Systems for Surface Modeling and Grid Generation Workshop, NASA Langley Research Center, April 28–30, 1992.
6. T. Y. Su, T. J. Kao, and R. Appleby, “An Interactive Multi-Block Grid Generation System,” published in the Proceedings of the NASA Workshop on Surface Modeling and Grid Generation Software Systems, held at NASA Langley, April 28–30, 1992.
7. AIAA 96-1995, T. Y. Su, N. J. Yu, and W. M. Wilkinson, “Multiblock Grid Generation Process for Complex Configuration Analysis using Navier-Stokes Solvers.”
8. T. P. Dickens, “Cooperative Solutions Coupling a Geometry Engine and Adaptive Solver Codes,” Surface Modeling, Grid Generation, and Related Issues in Computational Fluid Dynamics (CFD) Solutions, NASA Conference Publication 3291, May 9–11, 1995.