



An Idealized, Single Radial Swirler, Lean-Direct- Injection (LDI) Concept Meshing Script

*Anthony C. Iannetti
Glenn Research Center, Cleveland, Ohio*

*Daniel Thompson
Ohio Wesleyan University, Delaware, Ohio*

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI Program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- **TECHNICAL PUBLICATION.** Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.
- **TECHNICAL MEMORANDUM.** Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.
- **CONTRACTOR REPORT.** Scientific and technical findings by NASA-sponsored contractors and grantees.
- **CONFERENCE PUBLICATION.** Collected

papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- **SPECIAL PUBLICATION.** Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.
- **TECHNICAL TRANSLATION.** English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include creating custom thesauri, building customized databases, organizing and publishing research results.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at <http://www.sti.nasa.gov>
- E-mail your question via the Internet to help@sti.nasa.gov
- Fax your question to the NASA STI Help Desk at 301-621-0134
- Telephone the NASA STI Help Desk at 301-621-0390
- Write to:
NASA Center for AeroSpace Information (CASI)
7115 Standard Drive
Hanover, MD 21076-1320

NASA/TM—2008-215048



An Idealized, Single Radial Swirler, Lean-Direct- Injection (LDI) Concept Meshing Script

Anthony C. Iannetti
Glenn Research Center, Cleveland, Ohio

Daniel Thompson
Ohio Wesleyan University, Delaware, Ohio

National Aeronautics and
Space Administration

Glenn Research Center
Cleveland, Ohio 44135

July 2008

Acknowledgments

This work was funded under the Aeronautics Mission Directorate by the Fundamental Aeronautics Subsonic Fixed Wing and Supersonic Fixed Wing programs.

This report is a formal draft or working paper, intended to solicit comments and ideas from a technical peer group.

Trade names and trademarks are used in this report for identification only. Their usage does not constitute an official endorsement, either expressed or implied, by the National Aeronautics and Space Administration.

This work was sponsored by the Fundamental Aeronautics Program at the NASA Glenn Research Center.

Level of Review: This material has been technically reviewed by technical management.

Available from

NASA Center for Aerospace Information
7115 Standard Drive
Hanover, MD 21076-1320

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161

Available electronically at <http://gltrs.grc.nasa.gov>

An Idealized, Single Radial Swirler, Lean-Direct-Injection (LDI) Concept Meshing Script

Anthony C. Iannetti
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

Daniel Thompson
Ohio Wesleyan University
Delaware, Ohio 43015

Summary

To easily study combustor design parameters using computational fluid dynamics codes (CFD), a Gridgen Glyph-based macro (based on the Tcl scripting language) dubbed BladeMaker has been developed for the meshing of an idealized, single radial swirler, lean-direct-injection (LDI) combustor. BladeMaker is capable of taking in a number of parameters, such as blade width, blade tilt with respect to the perpendicular, swirler cup radius, and grid densities, and producing a three-dimensional meshed radial swirler with a can-annular (canned) combustor. This complex script produces a data format suitable for but not specific to the National Combustion Code (NCC), a state-of-the-art CFD code developed for reacting flow processes.

Introduction

Lean direct injection (LDI) is a NASA-developed, low-emissions gas turbine combustion concept. Although this concept produces reduced emissions in flame tube tests, the mixing and combustion processes must be understood to optimize the concept (refs. 1 to 3). Computational fluid dynamics (CFD) programs allow LDI design studies on computers. However, in past studies, the geometries were fixed because of the difficulties in creating a computational mesh (refs. 4 and 5). To better optimize the LDI concept using current CFD codes, parametric geometry manipulation is a necessary requirement.

Although parametric geometries may be easily created by using computer-aided design (CAD) tools like Pro Engineer (Parametric Technology Corp.), creating a suitable efficient mesh for CFD design studies is difficult. Coupling parametric solid models to a mesh generator and creating a good quality mesh are still a research problem. To solve this problem, a Gridgen (Pointwise, Inc., ref. 6) Glyph macro dubbed BladeMaker has been developed for the meshing of a simplified, idealized, LDI radial swirler. BladeMaker is capable of taking in a number of parameters, such as blade width, blade tilt with respect to the perpendicular, swirler cup radius, and grid densities, and producing a meshed radial swirler. This LDI mesh can then be used as input to a CFD code, preferably the National Combustion Code (NCC).

Briefly, The NCC is a state-of-the art CFD program specifically designed for combustion processes. A short summary of the features of NCC follow: the use of unstructured grids (ref. 7); massively parallel computing—with almost perfectly linear scalability (refs. 8 and 9); a dynamic wall function with the effect of adverse pressure gradient (ref. 10); low-Reynolds-number wall treatment (ref. 11); a cubic, nonlinear k-epsilon turbulence model (refs. 12 and 13); and stiff laminar chemistry integration. Recently, viscous low-speed preconditioning (refs. 14 and 15) has been added to improve the low-speed convergence of the NCC in viscous regions. The combination of these features is usually not available in other CFD codes and gives the NCC an advantage when one is computing turbulent, reacting flows.

NOTES ON FORMATTING AND NAMING: This document uses different fonts to indicate specific code items. Sample code will always be shown in 12-point Courier New font face and will be boxed. Commands to be submitted to a shell and file names will be in unboxed 12-point Courier New. Inline Glyph code or commands will also be in this font, but not boxed. Glyph variables and procedure names will be shown in 10-point Arial when not in sample code.

Using BladeMaker

Creating a Project Directory

It is suggested that you create a directory for your case inside the BladeMaker directory. This suggestion is made because modification of the script and input files will be required. If you begin to have problems with a file, you will always have the original to go back to. Once you have created a project directory, you will want to copy the files `param.in` and all the files with the extension `sh` into it. These files can be found in the BladeMaker root directory.

Parameter File Construction

All specifications for your geometry and mesh are made in the parameters file, which is called `param.in` by default. The parameter file that comes with BladeMaker is shown in figure 1 for quick reference.

```
ASW_TYPE
NCC
ASW_DIM
3
swirler_rad
2.0
bladevol_rad
6.0
manifold_rad
9.0
num_blades
8
blade_ang
15.0
blade_rad
0.1
height_s
1.0
basic_con_dim
5
blade_con_dim
10
wall_ds
0.001
height_cc
15.0
fname
/your/working/directory/patran.out
*EOF*
```

Figure 1.—Default parameter file, `param.in`.

TABLE 1.—DESCRIPTIONS AND LIMITS IN PARAMETER FILE

Variable name	Description	Limit		Data type
		Lower	Upper	
ASW_TYPE	Analysis software package	Any valid Gridgen analysis software type. See Gridgen documentation for more information.		
ASW_DIM	Number of dimensions in geometry	3	3	Positive integer
swirler_rad	Radius of swirler cup	1.0	Less than bladevol_rad	Positive real
bladevol_rad	Distance from center of swirler cup to edge of blade volume	Greater than swirler_rad	Less than manifold_rad	Positive real
manifold_rad	Radius of entire radial swirler	Greater than bladevol_rad	Infinite	Positive real
num_blades	Number of blades in swirler	These two variables are inversely proportional. The minimum number of blades possible for any blade_rad is 3. The macro will fail to run on two blades for unknown reasons.		
blade_rad	One-half of blade width			
blade_ang	Tilt of a single blade with respect to the perpendicular (Counterclockwise is positive and clockwise is negative.)	-50.0	50.0	Real
height_s	Height of region of swirler containing the blades	0.0	Infinite	Positive real
height_cc	Height of combustion chamber	0.0	Infinite	Positive real
basic_con_dim	Number of points placed initially on a connector	2	Infinite	Positive integer
blade_con_dim	Number of points placed on a connector near blades	Greater than basic_con_dim	Infinite	Positive integer
wall_ds	Average spacing for grid points near walls	Greater than 0.0	0.01	Positive real
fname	File name for analysis software file (i.e., patran.out)	Any writable absolute file name		
EOF	This is not really a variable, but it signals the end of the file.			

The top section of the file `param.in` serves as an identification of its contents and is ignored by BladeMaker. Each line after that is either a variable name or a value. If a variable name is encountered, the parser sets it to the value on the next line. So, for the file in figure 1, `ASW_TYPE` is set to `NCC`, `ASW_DIM` is set to 3, and so on. A description of each of these variables and the limits of BladeMaker are found in table 1. Because of “stretching” of the structured mesh, the variable `blade_ang` has a limitation of approximately -50 to 50° to the perpendicular to keep mesh quality high.

Glyph Script

Most users will only have to touch one line in the actual Glyph script, `blademaker.glf`. This is a plain text file and can be edited in any text editor. Just one line has to be modified in this script, as highlighted in figure 2. `PARAM_FILE` must be set to the absolute file path of your parameter file instead of to the default `/your/working/directory/param.in`.

```
#####
#           PARAMETERS           #
#####

# Set Pi - Tcl uses radians in trig functions.
set PI [expr {4.0 * atan (1.0)}]

set PARAM FILE "/your/working/directory/param.in"
set EOF "*EOF*"
```

Figure 2.—Changes in the glyph script `blademaker.glf`.

TABLE 2.—SHELL SCRIPT DESCRIPTIONS

Shell script	Description	Output
<code>unix-unopt.sh</code>	Runs BladeMaker	An analysis software boundary conditions file, such as <code>patran.out</code>

```
#!/bin/bash
# BladeMaker
# Gridgen Only Run Shell Script
# UNIX Version
# Created 19 July 2005

source /usr/local/etc/setup.sh
setup gridgenv157
gridgen blademaker.glf
```

Figure 3.—The `unix-unopt.sh` bash shell script.

Running BladeMaker

BladeMaker has been run as a batch script since version 1.0.2. We assume that Gridgen Version 15.07 or later is available on your computer. Simply instruct your shell to interpret the shell script appropriate for your case. Table 2 gives a description of the shell scripts.

The shell script `unix-unopt.sh` (fig. 3) will run BladeMaker alone. It does not link automatically to CFD codes or optimizers. This script assumes that you have access to the bash shell and have available Gridgen Version 15.07 or later. If your system is not configured as such, you will need to use this section as a basis for creating your own shell script.

The first line sources in the `setup.sh` shell script. On the second line, `setup.sh` is used to make Gridgen Version 15.07 available. The third and final line of code runs Gridgen in batch mode on the macro in `blademaker.glf`. A boundary condition file useable by an analysis software package will be created. This is usually `patran.out`, a PATRAN neutral file useable by the NCC.

Breakdown of BladeMaker Glyph Script

Previous Knowledge

The appendix assumes that the reader has basic knowledge of Glyph, or at least Tcl. Some sections may be confusing without such knowledge. More information on Tcl can be found at <http://www.tcl.tk/> and more information on Glyph can be found in Gridgen Version 15: Glyph Reference Manual.

General Overview

The BladeMaker script itself is stored in the ASCII text file `blademaker.glf`. Because the contents of `blademaker.glf` are referenced throughout the report, `blademaker.glf` is listed in the appendix. It is divided into a number of sections, indicated by comment headers. These sections appear in the following order: parameters, procedure definitions, and the main Glyph script. Each is covered in detail in the next sections.

Parameters Section

The purpose of the parameters section is twofold. First, three constants referred to by the macro are defined. Since Tcl does not provide a built-in π , the constant `PI` is set to $4.0 * \arctan(1.0)$, which is a good approximation of π . `RUN_MODE` defines whether or not the Gridgen will use the script interactively, set to 1, or in a batch mode, set to 0. `PARAM_FILE` once again defines the input parameters.

Procedure Definitions

In the procedure definitions section, all procedures used in the macro are defined. The section is subdivided into utility, connector, database, meshing, boundary condition, mesh refinement, and large-scale creation procedures.

Utility procedures.—Utility procedures are called in the procedure calls section but do not actually create any grid structure. Currently, the only procedure in this section is `ClearWorkspace`. This procedure takes no arguments and prepares the workspace for use by the macro by clearing any grid structure and setting all options to their default values. It also sets the analysis software type and dimension as defined in the parameters file.

Connector procedures.—Connector procedures bundle groups of primitive Glyph commands for certain functions. Without this section, these groups of commands would have to be repeatedly rewritten, which is tedious and leads to a large file size. These procedures are usually only called from within the large-scale creation procedures.

The first procedure in this section, `SplitConIntoNEqualParts`, was adapted from the Glyph procedure `ConSplitIntoN` found on the Pointwise Glyph exchange at <http://www.pointwise.com>. The procedure `SplitConIntoNEqualParts` takes a positive integer as the first parameter and a list of Glyph connector IDs as the second parameter and splits each connector in the list into the specified number of equal parts. It also returns the connector IDs of the resultant connectors.

The next connector procedure is `CreateEndptCon`. It takes a list of positive integers as its only parameter, which must have an even-numbered length. For each successive pair of positive integers in the argument, a connector is created between the endpoints of the corresponding elements in the list. This is returned by the Glyph command `conGetAll` in the namespace `gg`. For example, if we passed this procedure the list `0 3 10 63`, two connectors would be created. The first would be created between the endpoints of the connectors with Glyph IDs `[lindex [gg::conGetAll] 0]` and `[lindex [gg::conGetAll] 3]`, and the second connector would be formed between the endpoints of the

connectors with Glyph IDs [`lindex [gg::conGetAll] 10`] and [`lindex [gg::conGetAll] 63`].

The final connector procedure is `Create2PtCon`. This procedure takes as a parameter a list of points in three-dimensional space, which must have an even-valued length. A straight connector is drawn between the first point and the second point, then the third point and the fourth point, and so on. For example, if it were passed the list [`[0 4 7] [8 19 23] [7 5 8] [0 0 0]`], two connectors would be created. The first would start at (0,4,7) and end at (8,19,23). The second would begin at (7,5,8) and end at the origin.

Database procedures.—These procedures are much like the connector procedures, except that they are used to create database entities instead of connectors.

Currently there is only one database procedure called `Create2PtDB`, which takes as a parameter a list of indexes to the return value of `dbGetAll`. This parameter must have an even-numbered length, and these indexes must correspond to the Glyph IDs of points. The list is treated much like the list in `Create2PtCon`, except that straight database entities are created instead of straight connectors. For each pair in the parameter, a straight database entity is created beginning at the first element in the pair and ending at the second element. For example, if we passed this procedure the list [`0 1 34 68`], two straight database entities would be created. The first would be between the points with Glyph IDs 0 and 1, and the second between points with Glyph IDs 34 and 68.

Meshing procedures.—This collection of procedures serves to bundle the primitive commands used to create structured mesh entities such as structured domains into single procedures that can be easily called from the large-scale mesh creation procedures.

The first such procedure is called `CreateStrucDom` and is used to create structured domains. The parameter for this procedure is a list of indexes to the return value of `conGetAll`. The parameter must have a length that is evenly divisible by 4. Each set of four indexes is treated as a definition of the edges of a single structured domain. For example, if this procedure were passed the list [`0 4 13 55 0 8 16 3`] as input, two structured domains would be created. For the first domain, the first edge would be the connector with the Glyph ID found in the 0th element of the return value of `conGetAll`. The Glyph ID for the second edge would be found in the 4th element, the third in the 13th element, and the final edge in the 55th element. The second domain would be constructed in much the same fashion. This procedure eliminates the tedious copying and pasting of the same series of Glyph commands.

The next and final procedure is `CreateStrucBlk`. This is identical to `CreateStrucDom`, except that it creates structured blocks and thus must take a list that is evenly divisible by 6 as input. The list elements are treated as indexes to the return value of the command `domGetAll`.

Boundary condition procedures.—These procedures are for tagging boundary conditions as well as for generating the analysis software file, like `patran.out`.

The first such procedure, called `TagBoundaryConditions`, tags certain domains as being boundary conditions. These boundary conditions are very specific to this particular test case. The bottom of the swirler (i.e., the lower domain in the blade region) is marked as having an “inlet-uvw” boundary. The top most domain on the combustion chamber that runs parallel to this is tagged as the “exit” BC. From a top-down view, the leftmost domain in the swirler cup is tagged as a “periodic-a” BC and the rightmost is tagged as a “periodic-b” BC. The leftmost blade volume domain is tagged as a Gridgen generic boundary condition “BndCond 20,” the leftmost combustion chamber domain as “BndCond 21,” the rightmost blade volume domain as “BndCond 22,” and finally the rightmost combustion chamber domain as “BndCond 23.”

The final boundary condition procedure, called `ExportASW`, takes a file name as a parameter and creates the analysis software boundary condition file. If our analysis software is set to NCC, this would be the `patran.out` file. The file name tells where to create the file.

Mesh refinement procedures.—The procedures are for refining the mesh that has been previously generated. The first such procedure, called `ClusterPts`, is called from within other mesh refinement procedures. It takes three parameters, the average Δs value near the walls (`ds`), a list of indexes to the return value of `conGetAll` (`index`), and an end of the connector to cluster points near (`loc`). For each element of the return value of `conGetAll` that is listed in `index`, the procedure cluster points to a value of the average Δs value `ds` near the end specified in `loc`—which is either “start” or “end.”

`RefineSwirlerGrid` takes two parameters, the average Δs value near the walls (`wall_ds`) and a connector dimension (`blade_con_dim`). This procedure increases the grid density near the blades in the lower portion of the swirler by changing the connector dimension from the default to `blade_con_dim` and cluster points near the blade walls by setting the average Δs value of the subconnectors pointing to the blade walls to `wall_ds`.

`RefineCombustionChamberGrid` takes a single parameter, an average Δs value (`wall_ds`) and increases the density of the combustion chamber grid by a factor of one-quarter times the ratio of the height of the combustion chamber to the height of the swirler cup. This is done since the combustion chamber is usually much taller than the swirler cup, and the grid would otherwise become less dense in the combustion chamber. Next, points are clustered near the combustion chamber outside wall and the wall of the combustion chamber facing the swirler cup by setting the average Δs value of the appropriate subconnectors to `wall_ds`.

Points must also be clustered along the wall in the buffer region between the swirler cup and the combustion chamber. There is one such wall facing the combustion chamber and another facing the swirler cup. The reader may inquire as to why there is no such procedure in `blademaker.glf`. The reason is that the procedure `CreateBuffer` assumes that both `CreateSwirler` and `RefineSwirlerGrid` have already been called. Thus, `CreateBuffer` creates a buffer volume that already has a refined mesh.

Large-scale creation procedures.—These procedures call the database, connector, mesh creation, and mesh refinement, in addition to other smaller procedures, in such an order as to create large sections of the final grid and geometry. Although many of the previous procedures could easily be transplanted to another Glyph mesh generation script for a completely different case, these are specific to this geometry and mesh.

The first of these procedures, `MakeWhole`, is an optional procedure in place for demonstration purposes only. As distributed, `blademaker.glf` never actually calls this procedure. However, the user could easily provide such a call by adding the line `MakeWhole` to the procedure calls section. This procedure simply creates the whole radial swirler from the slice via copying by rotation.

The next procedure, `CreateSwirler`, is responsible for creating the lower portion of the radial swirler slice that contains the swirler cup and blades. It takes as parameters the radius of the swirler cup, the distance from the center of the swirler cup to a point on the edge of the blade volume, the radius of the whole radial swirler, the number of blades in the swirler, the angle a single blade makes with respect to the perpendicular, half the width of a single blade, the height of the swirler cup, and the number of points to initially place on all connectors in this region. The blade angle is considered positive in the counterclockwise direction and negative in the clockwise direction.

The third procedure, `CreateBuffer`, takes just one parameter, the height of the swirler cup. It creates a slice of the buffer region between the combustion chamber and the swirler cup by copying the swirler cup via a translation of height equal to the parameter.

The final large-scale creation procedure, `CreateCombustionChamber`, is responsible for the creation of the combustion chamber slice. It takes three parameters: the height of the swirler cup, the desired height of the combustion chamber, and the number of points to be placed by default on all connectors in the combustion chamber. This last parameter should be set equal to the number of points placed by default on connectors in the bottom of the swirler.

Procedure Calls

This section calls the utility, large-scale creation, mesh creation, mesh refinement, and boundary condition procedures in such an order as to create a complete slice of the radial swirler. A slice is created to lessen the time needed to devise a solution for this case. One slice follows a single blade. The order of these procedure calls should *not* be modified. Some procedures rely upon other procedures already having been called. For example, the procedures `CreateBuffer` and `CreateCombustionChamber` assume that the procedure `CreateSwirler` has already been successfully called, and they will fail if it has not.

Mesh Examples

Figures 4 to 7 show various meshes that have been generated by the `BladeMaker` script. The parameters used to generate the meshes are typical for a LDI CFD design study. All the meshes comprise about 125 000 computational cells and, in general, take about 10 sec to generate. Using meshes with higher cell counts can take up to 2 min to generate. A Linux workstation with 2-GHz Intel Xeon 5100 series dual-core processors was used for mesh generation.

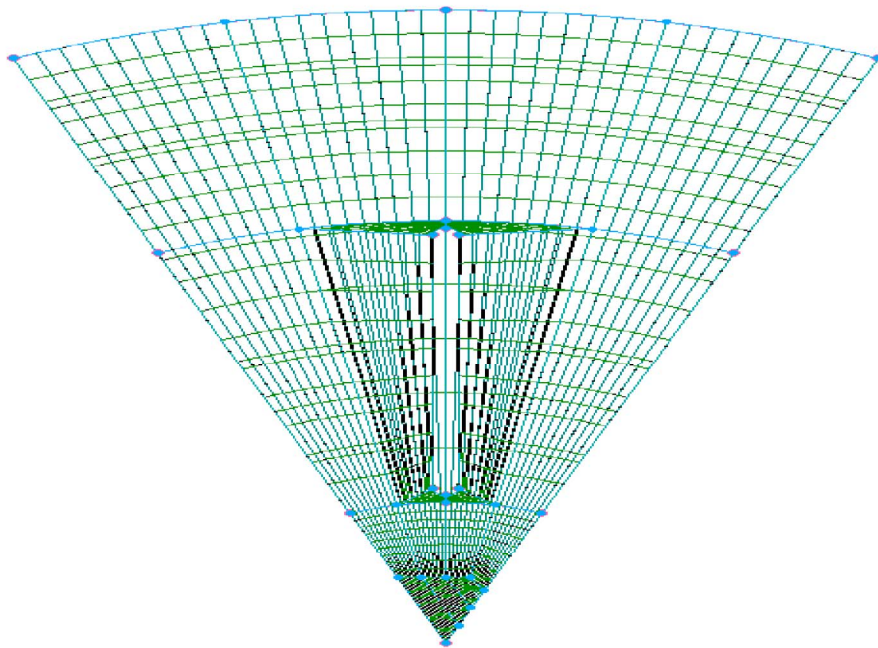


Figure 4.—Top-down view of `BladeMaker`-generated grid with no blade tilt.
Number of blades, 8; blade width, 0.2.

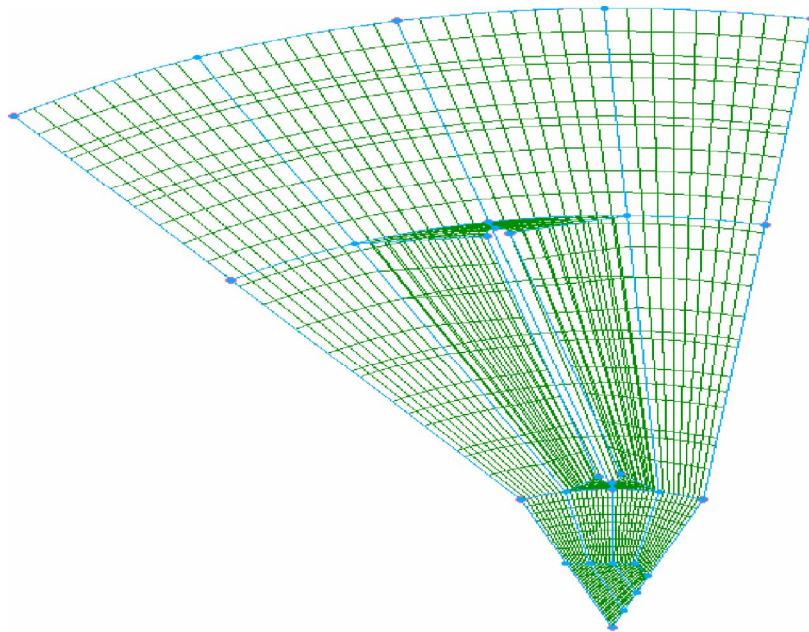


Figure 5.—Top-down view of BladeMaker-generated grid with blade tilt of 15.0°. Number of blades, 8; blade width, 0.2.

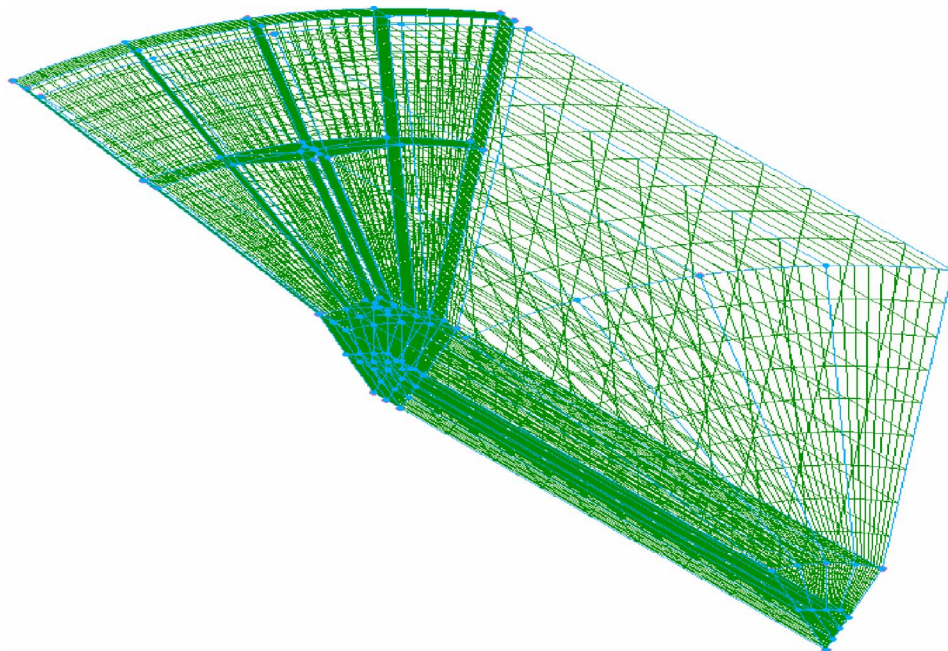


Figure 6.—Perspective view of BladeMaker-generated grid with blade tilt of 15.0°. Number of blades, 8; blade width, 0.2; swirler height, 1.0; combustion chamber height, 30.0.

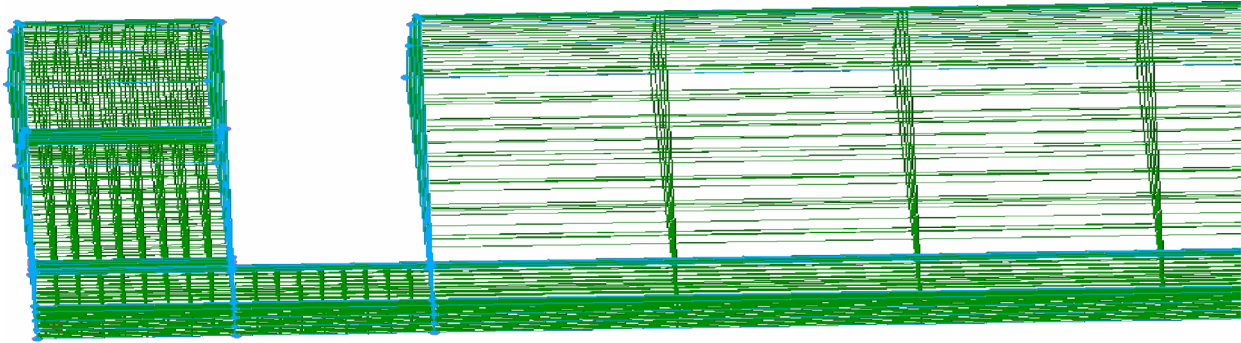


Figure 7.—Side view of BladeMaker generated-grid with blade tilt of 15.0°. Number of blades, 8; blade width, 0.2; swirler height, 3.0; combustion chamber height, 30.0.

Concluding Remarks

The BladeMaker script removes a major time bottleneck for studying lean-direct-injection (LDI) concepts with computational fluid dynamics (CFD) codes, like the National Combustion Code (NCC). Although the geometry produced is a simple can-annular (canned) combustor, it is suitable for studying important combustor design parameters, such as the swirler blade angle. The canned combustor may be studied as a periodic slice or as a full can-annular combustor. The mesh density may also be varied, which is also important for determining whether or not the resulting CFD solution is mesh independent.

We believe that this script is mostly error free in its current format, but 100-percent reliability cannot be guaranteed. Do not use this script in its current form for mission-critical applications. The script is being released in an open source licensing model, as documented in NASA OPEN SOURCE AGREEMENT VERSION 1.3.2 (ref. 16). By releasing BladeMaker in an open source model, we *strongly* believe that this will increase software quality via community peer review, accelerate BladeMaker development via community contributions, maximize the awareness and impact of NASA research, and increase dissemination of this software in support of NASA's education mission.

Appendix—BladeMaker Code Listing

```
#####
#####
####          B L A D E M A K E R          ####
####          -----                      ####
####                                               ####
#### BladeMaker is a Glyph macro for Gridgen Ver 15.07 to be ####
#### used for the automatic creation of radial swirlers.    ####
####                                               ####
#### Scripting By: Daniel Thompson                    ####
#### Grid & Geometry By: Anthony C. Iannetti          ####
#### Build Info: Release Version 1.0.2                ####
#### Created: 18 July 2005                            ####
####                                               ####
####                               See manual for contact info. ####
#####
#####
# Require the appropriate Glyph version.
# Removal of this line may allow BladeMaker to run on Gridgen v15.05 - 15.06.
package require PWI_Glyph 1.6.7

#####
#          PARAMETERS          #
#####

# Set Pi - Tcl uses radians in trig functions.
set PI [expr {4.0 * atan (1.0)}]

# POSSIBLE VALUES FOR RUN_MODE
#
# 0: BladeMaker takes parameter values from params.in. Be sure
#    to use this mode if running this macro from unix-unopt.sh
# 1: BladeMaker uses a crude GUI to get user input.
set RUN_MODE 0

set PARAM_FILE "/your/working/directory/param.in"
set EOF "*EOF*"

# Read the parameter file.
set cin [open $PARAM_FILE]

gets $cin line

while {$line != $EOF}\
{
  switch $line\
  {
    "ASW_TYPE" {set $line [gets $cin]}
    "ASW_DIM"  {set $line [gets $cin]}
    "swirler_rad" {set $line [gets $cin]}
    "bladevol_rad" {set $line [gets $cin]}
    "manifold_rad" {set $line [gets $cin]}
    "num_blades" {set $line [gets $cin]}
  }
}
```

```

"blade_ang" {set $line [gets $cin]}
"blade_rad" {set $line [gets $cin]}
"height_s" {set $line [gets $cin]}
"basic_con_dim" {set $line [gets $cin]}
"blade_con_dim" {set $line [gets $cin]}
"wall_ds" {set $line [gets $cin]}
"height_cc" {set $line [gets $cin]}
"fname" {set $line [gets $cin]}
}

gets $cin line
}

#####
#          UTILITY PROCEDURES          #
#####

proc ClearWorkspace {}\
{
#####
# Procedure: ClearWorkspace           #
# Purpose: Prepares the workspace for our #
#          use.                       #
#####

global ASW_TYPE
global ASW_DIM

gg::memClear
gg::aswDeleteBC -glob "*"
gg::aswDeleteVC -glob "*"
gg::aswSet $ASW_TYPE -dim $ASW_DIM
gg::defReset
gg::tolReset
gg::updatePolicy DISPLAY_AND_INPUT
}

#####
#          CONNECTOR PROCEDURES        #
#####

proc SplitConIntoNEqualParts {num_pieces con_list}\
{
#####
# Procedure: SplitConIntoNEqualParts   #
# Purpose: Splits up connectors into equal #
#          parts.                       #
# Note: This procedure adapted from     #
#          ConSplitIntoN on the Pointwise Glyph #
#          exchange @ http://www.pointwise.com. #
#####

set return_con_list $con_list

foreach con $con_list\
{
for {set split 1} {$split < $num_pieces} {incr split}\

```



```

{
set ds [expr 1.0 / [expr {$num_pieces - $split + 1}]]
set pt [gg::conGetPt $con -arc $ds]
set new_con [gg::conSplit $con $pt]
set new_dim [gg::conDim $new_con]
set needed_dim [expr {$num_pieces - $split + 1}]

if {$new_dim > 0 && $new_dim < $needed_dim}\
{
set delta [expr {$needed_dim - $new_dim}]
set old_dim [gg::conDim $con]

gg::conRedimBegin
gg::conRedim $new_con [expr {$new_dim + $delta}]
gg::conRedim $con [expr {$old_dim - $delta}]
gg::conRedimEnd
}

set con $new_con
set return_con_list [lappend return_con_list $con]
}

return $return_con_list
}

proc CreateEndptCon {index}\
{
#####
# Procedure: CreateEndptCon #
# Purpose: Used to create a connector #
# between the endpoints of two #
# other connectors. #
#####

for {set i 0} {$i < [llength $index]} {incr i 2}\
{
gg::conBegin
gg::segBegin
gg::segAddControlPt [gg::conGetPt [lindex [gg::conGetAll] [lindex $index $i]]
-arc 1.0]
gg::segAddControlPt [gg::conGetPt [lindex [gg::conGetAll] [lindex $index
[expr {$i + 1}]]] -arc 1.0]
gg::segEnd
lappend r [gg::conEnd]
}

return $r
}

proc Create2PtCon {points}\
{
#####
# Procedure: Create2PtCon #
# Purpose: Used to create a connector #
# between any 2 given points. #
#####

```

```

for {set i 0} {$i < [llength $points]} {incr i 2}\
{
gg::conBegin
gg::segBegin
gg::segAddControlPt [lindex $points $i]
gg::segAddControlPt [lindex $points [expr {$i + 1}]]
gg::segEnd
lappend r [gg::conEnd]
}

return $r
}

proc SplitConAtPer {percent index}\
{
#####
# Procedure: SplitConAtPer #
# Purpose: Used to split a connector in half #
# at a given percentage of its #
# total length. #
#####

gg::conSplit [lindex [gg::conGetAll] $index] [gg::conGetPt [lindex
[gg::conGetAll] $index] -arc $percent]
}

#####
# DATABASE PROCEDURES #
#####

proc Create2PtDB {index}\
{
#####
# Procedure: Create2PtDB #
# Purpose: Used to create a 2 point, #
# straight, 3D line as a database #
# entity. #
#####

for {set i 0} {$i < [llength $index]} {incr i 2}\
{
gg::dbCurveBegin -type AKIMA
gg::dbCurveAddPt -db [list 0 0 [lindex [gg::dbGetAll] [lindex $index $i]]]
gg::dbCurveAddPt -db [list 0 0 [lindex [gg::dbGetAll] [lindex $index [expr
{$i + 1}]]]]
lappend r [gg::dbCurveEnd]
}

return $r
}

#####
# MESHING PROCEDURES #
#####

proc CreateStrucDom {index}\

```

```

{
#####
# Procedure: CreateStrucDom #
# Purpose: Used to create a structured #
# domain. #
#####

for {set i 0} {$i < [llength $index]} {incr i 4}\
{
gg::domBegin -type STRUCTURED
gg::edgeBegin
gg::edgeAddCon [lindex [gg::conGetAll] [lindex $index $i]]
gg::edgeEnd
gg::edgeBegin
gg::edgeAddCon [lindex [gg::conGetAll] [lindex $index [expr {$i + 1}]]]
gg::edgeEnd
gg::edgeBegin
gg::edgeAddCon [lindex [gg::conGetAll] [lindex $index [expr {$i + 2}]]]
gg::edgeEnd
gg::edgeBegin
gg::edgeAddCon [lindex [gg::conGetAll] [lindex $index [expr {$i + 3}]]]
gg::edgeEnd
lappend r [gg::domEnd]
}

return $r
}

proc CreateStrucBlk {index}\
{
#####
# Procedure: CreateStrucBlk #
# Purpose: Used to create a structured block.#
#####

for {set i 0} {$i < [llength $index]} {incr i 6}\
{
gg::blkBegin -type STRUCTURED
gg::faceBegin
gg::faceAddDom [lindex [gg::domGetAll] [lindex $index $i]]
gg::faceEnd
gg::faceBegin
gg::faceAddDom [lindex [gg::domGetAll] [lindex $index [expr {$i + 1}]]]
gg::faceEnd
gg::faceBegin
gg::faceAddDom [lindex [gg::domGetAll] [lindex $index [expr {$i + 2}]]]
gg::faceEnd
gg::faceBegin
gg::faceAddDom [lindex [gg::domGetAll] [lindex $index [expr {$i + 3}]]]
gg::faceEnd
gg::faceBegin
gg::faceAddDom [lindex [gg::domGetAll] [lindex $index [expr {$i + 4}]]]
gg::faceEnd
gg::faceBegin
gg::faceAddDom [lindex [gg::domGetAll] [lindex $index [expr {$i + 5}]]]
gg::faceEnd
lappend r [gg::blkEnd]
}

```

```

}

return $r
}

#####
#   BOUNDARY CONDITION PROCEDURES   #
#####

proc TagBoundaryConditions {}\
{
#####
# Procedure: TagBoundaryConditions   #
# Purpose: Tags the boundary conditions. #
#####

###
### Inlet-UVW (Bottom of Blades)
###

set doms [lrange [gg::domGetAll] 0 16]
lappend doms [lindex [gg::domGetAll] 81]
lappend doms [lindex [gg::domGetAll] 82]
gg::aswSetBC $doms "inlet-uvw"
unset doms

###
### Exit (Top of Combustion Chamber)
###

set doms [lrange [gg::domGetAll] 144 147]
lappend doms [lindex [gg::domGetAll] 110]
lappend doms [lindex [gg::domGetAll] 115]
lappend doms [lindex [gg::domGetAll] 119]
lappend doms [lindex [gg::domGetAll] 122]
lappend doms [lindex [gg::domGetAll] 126]
lappend doms [lindex [gg::domGetAll] 129]
lappend doms [lindex [gg::domGetAll] 132]
gg::aswSetBC $doms "exit"
unset doms

###
### Periodic-A (Left Hand Swirler Cup Face)
###

lappend doms [lindex [gg::domGetAll] 34]
lappend doms [lindex [gg::domGetAll] 78]
lappend doms [lindex [gg::domGetAll] 91]
lappend doms [lindex [gg::domGetAll] 105]
lappend doms [lindex [gg::domGetAll] 116]
lappend doms [lindex [gg::domGetAll] 133]
gg::aswSetBC $doms "periodic-a"
unset doms

###
### Periodic-B (Right Hand Swirler Cup Face)
###

```

```

lappend doms [lindex [gg::domGetAll] 35]
lappend doms [lindex [gg::domGetAll] 80]
lappend doms [lindex [gg::domGetAll] 102]
lappend doms [lindex [gg::domGetAll] 108]
lappend doms [lindex [gg::domGetAll] 112]
lappend doms [lindex [gg::domGetAll] 124]
gg::aswSetBC $doms "periodic-b"
unset doms

###
### Periodic-C (Blade Reigon Left Face)
###

gg::aswCreateBC "BndCond 20" -solid 1 -id 20
lappend doms [lindex [gg::domGetAll] 46]
lappend doms [lindex [gg::domGetAll] 64]
gg::aswSetBC $doms "BndCond 20"
unset doms

###
### Periodic-D (Combustion Chamber Left Face)
###

gg::aswCreateBC "BndCond 21" -solid 1 -id 21
gg::aswSetBC [lindex [gg::domGetAll] 135] "BndCond 21"

###
### Periodic-E (Blade Reigon Right Face)
###

gg::aswCreateBC "BndCond 22" -solid 1 -id 22
lappend doms [lindex [gg::domGetAll] 51]
lappend doms [lindex [gg::domGetAll] 68]
gg::aswSetBC $doms "BndCond 22"

###
### Periodic-F (Combustion Chamber Right Face)
###

gg::aswCreateBC "BndCond 23" -solid 1 -id 23
gg::aswSetBC [lindex [gg::domGetAll] 139] "BndCond 23"
}

proc ExportASW {fname}\
{
#####
# Procedure: ExportASW #
# Purpose: Creates an analysis software #
# file (like patran.out). #
#####

gg::aswExport $fname
}

#####

```

```

#           MESH REFINEMENT PROCEDURES           #
#####

proc ClusterPts {ds index loc}\
{
#####
# Procedure: ClusterPts                               #
# Purpose: Used to cluster points on the             #
#           start or end of given connectors.        #
#####

foreach i $index\
{
lappend pts [lindex [gg::conGetAll] $i]
}

if {$loc == "start"}\
{
gg::conBeginSpacing $pts $ds
}

if {$loc == "end"}\
{
gg::conEndSpacing $pts $ds
}
}

proc RefineSwirlerGrid {wall_ds blade_con_dim}\
{
#####
# Procedure: RefineSwirlerGrid                       #
# Purpose: Used to refine the grid of the           #
#           swirler portion of the radial           #
#           swirler.                                 #
#####

# Redimension the reigon around the blades.
gg::conRedimBegin
gg::conRedim [lindex [gg::conGetAll] 0] $blade_con_dim
gg::conRedim [lindex [gg::conGetAll] 2] $blade_con_dim
gg::conRedimEnd

# Cluster points around the blade wall.
ClusterPts $wall_ds [list 31 33 34 64 68 73] "start"
ClusterPts $wall_ds [list 30 35 36 62 65 70] "end"

# Cluster points around the rear face.
for {set i 88} {$i <= 115} {incr i}\
{
lappend index $i
}

ClusterPts $wall_ds $index "start"

# Cluster points around the forward face.
ClusterPts $wall_ds $index "end"
unset index

```

```

}

proc RefineCombustionChamberGrid {wall_ds}\
{
#####
# Procedure: RefineCombustionChamberGrid      #
# Purpose: Used to refine the grid of the     #
#         combustion chamber portion of the   #
#         radial swirler.                     #
#####

# Cluster points around the wall facing the blades.
for {set i 156} {$i <= 159} {incr i}\
{
lappend subcon_index $i
}

for {set i 163} {$i <= 164} {incr i}\
{
lappend subcon_index $i
}

for {set i 171} {$i <= 172} {incr i}\
{
lappend subcon_index $i
}

for {set i 181} {$i <= 184} {incr i}\
{
lappend subcon_index $i
}

for {set i 193} {$i <= 194} {incr i}\
{
lappend subcon_index $i
}

lappend subcon_index 175
lappend subcon_index 178
lappend subcon_index 167

ClusterPts $wall_ds $subcon_index "start"

unset subcon_index

# Cluster points around the wall near the outside of the chamber.
for {set i 195} {$i <= 204} {incr i}\
{
lappend subcon_index $i
}

ClusterPts $wall_ds $subcon_index "end"

unset subcon_index
}

# RefineBufferGrid taken out - 13 July 2005

```

```

# CreateBufferGrid now assumes that RefineSwirlerGrid
# has already been called, thus CreateBufferGrid
# will copy the refinement.

#####
#   LARGE-SCALE CREATION PROCEDURES   #
#####

proc MakeWhole {num_blades}\
{
#####
# Procedure: MakeWhole                #
# Purpose: Copies the geometry and grid via #
#           rotation about the z-axis to   #
#           generate the whole radial swirler.#
#####

set db_xform [gg::dbGetAll]
set blk_xform [gg::blkGetAll]
set wedge_ang [expr {360.0 / $num_blades}]

for {set i 1} {$i < $num_blades} {incr i}\
{
gg::dbCopyBegin $db_xform
gg::xformRotate [list 0 0 0] [list 0 0 1] [expr {$wedge_ang * $i}]
gg::dbCopyEnd

gg::blkCopyBegin $blk_xform
gg::xformRotate [list 0 0 0] [list 0 0 1] [expr {$wedge_ang * $i}]
gg::blkCopyEnd
}
}

proc CreateSwirler {swirler_rad bladevol_rad manifold_rad num_blades
blade_ang blade_rad height basic_con_dim}\
{
#####
# Procedure: CreateSwirler            #
# Purpose: Creates the swirler cup-blade #
#           region of the radial swirler. #
#####

# VARIABLES

# Miscellaneous
global PI
global VERBOSE_MODE
global RUN_MODE
set wedge_ang [expr {(2.0 * $PI) / $num_blades}]

# Lower Plane Points
set origin [list 0 0 0]
set swirler_lp [list [expr {- $swirler_rad * (sin ($wedge_ang / 2.0))}] [expr
{$swirler_rad * (cos ($wedge_ang / 2.0))}] 0]
set swirler_cp [list 0 $swirler_rad 0]
set swirler_rp [list [expr {$swirler_rad * (sin ($wedge_ang / 2.0))}] [expr
{$swirler_rad * (cos ($wedge_ang / 2.0))}] 0]

```



```

set bladevol_lp [list [expr {-bladevol_rad * (sin (wedge_ang / 2.0))}]
[expr {bladevol_rad * (cos (wedge_ang / 2.0))}] 0]
set bladevol_cp [list 0 bladevol_rad 0]
set bladevol_rp [list [expr {bladevol_rad * (sin (wedge_ang / 2.0))}] [expr
{bladevol_rad * (cos (wedge_ang / 2.0))}] 0]
set manifold_lp [list [expr {-manifold_rad * (sin (wedge_ang / 2.0))}]
[expr {manifold_rad * (cos (wedge_ang / 2.0))}] 0]
set manifold_cp [list 0 manifold_rad 0]
set manifold_rp [list [expr {manifold_rad * (sin (wedge_ang / 2.0))}] [expr
{manifold_rad * (cos (wedge_ang / 2.0))}] 0]
set inner_blade_lp [list -blade_rad [expr {swirler_rad + 2 * blade_rad}]
0]
set inner_blade_cp [list 0 [expr {swirler_rad + blade_rad}] 0]
set inner_blade_rp [list blade_rad [lindex $inner_blade_lp 1] 0]
set outer_blade_lp [list [lindex $inner_blade_lp 0] [expr {bladevol_rad - 2
* blade_rad}] 0]
set outer_blade_cp [list 0 [expr {bladevol_rad - blade_rad}] 0]
set outer_blade_rp [list [lindex $inner_blade_rp 0] [lindex $outer_blade_lp
1] 0]

# LOWER PLANE

# Define Database Entities

# Add Points to Database
gg::dbPtsBegin
gg::dbPtsAddPt $origin
gg::dbPtsAddPt $swirler_lp
gg::dbPtsAddPt $swirler_cp
gg::dbPtsAddPt $swirler_rp
gg::dbPtsAddPt $bladevol_lp
gg::dbPtsAddPt $bladevol_cp
gg::dbPtsAddPt $bladevol_rp
gg::dbPtsAddPt $manifold_lp
gg::dbPtsAddPt $manifold_cp
gg::dbPtsAddPt $manifold_rp
gg::dbPtsAddPt $inner_blade_lp
gg::dbPtsAddPt $inner_blade_cp
gg::dbPtsAddPt $inner_blade_rp
gg::dbPtsAddPt $outer_blade_lp
gg::dbPtsAddPt $outer_blade_cp
gg::dbPtsAddPt $outer_blade_rp
gg::dbPtsEnd

Create2PtDB [list 0 1\
1 4\
4 7\
0 3\
3 6\
6 9]

# Swirler Domain - Arc
gg::dbCurveBegin -type CIRCULAR_ARC
gg::dbCurveAddPt -db [list 0 0 [lindex [gg::dbGetAll] 1]]
gg::dbCurveAddPt -db [list 0 0 [lindex [gg::dbGetAll] 3]]
gg::dbCurveAddPt -db [list 0 0 [lindex [gg::dbGetAll] 2]]
gg::dbCurveEnd

```

```

Create2PtDB [list 10 13\
 12 15]

# Blade - Innermost Arc
gg::dbCurveBegin -type CIRCULAR_ARC
gg::dbCurveAddPt -db [list 0 0 [lindex [gg::dbGetAll] 10]]
gg::dbCurveAddPt -db [list 0 0 [lindex [gg::dbGetAll] 12]]
gg::dbCurveAddPt -db [list 0 0 [lindex [gg::dbGetAll] 11]]
gg::dbCurveEnd

# Blade - Outermost Arc
gg::dbCurveBegin -type CIRCULAR_ARC
gg::dbCurveAddPt -db [list 0 0 [lindex [gg::dbGetAll] 13]]
gg::dbCurveAddPt -db [list 0 0 [lindex [gg::dbGetAll] 15]]
gg::dbCurveAddPt -db [list 0 0 [lindex [gg::dbGetAll] 14]]
gg::dbCurveEnd

# Rotate Left Blade Volume and Left Manifold Volume Akimas
gg::dbTransformBegin [list [lindex [gg::dbGetAll] 17] [lindex [gg::dbGetAll]
18] [lindex [gg::dbGetAll] 4] [lindex [gg::dbGetAll] 7]] -maintain_linkage
gg::xformRotate $swirler_lp [ggu::vec3Add $swirler_lp [list 0 0 1]]
$blade_ang
gg::dbTransformEnd

# Rotate Right Blade Volume and Right Manifold Volume Akimas
gg::dbTransformBegin [list [lindex [gg::dbGetAll] 20] [lindex [gg::dbGetAll]
21] [lindex [gg::dbGetAll] 6] [lindex [gg::dbGetAll] 9]] -maintain_linkage
gg::xformRotate $swirler_rp [ggu::vec3Add $swirler_rp [list 0 0 1]]
$blade_ang
gg::dbTransformEnd

# Rotate Blade Volume Center Point
gg::dbTransformBegin [list [lindex [gg::dbGetAll] 5] [lindex [gg::dbGetAll]
8]] -maintain_linkage
gg::xformRotate $swirler_cp [ggu::vec3Add $swirler_cp [list 0 0 1]]
$blade_ang
gg::dbTransformEnd

# Rotate Blade
gg::dbTransformBegin [list [lindex [gg::dbGetAll] 23] [lindex [gg::dbGetAll]
24] [lindex [gg::dbGetAll] 26] [lindex [gg::dbGetAll] 25] [lindex
[gg::dbGetAll] 13] [lindex [gg::dbGetAll] 14] [lindex [gg::dbGetAll] 15]
[lindex [gg::dbGetAll] 10] [lindex [gg::dbGetAll] 12]] -maintain_linkage
gg::xformRotate $inner_blade_cp [ggu::vec3Add $inner_blade_cp [list 0 0 1]]
$blade_ang
gg::dbTransformEnd

# Blade Volume - Arc
gg::dbCurveBegin -type CIRCULAR_ARC
gg::dbCurveAddPt -db [list 0 0 [lindex [gg::dbGetAll] 4]]
gg::dbCurveAddPt -db [list 0 0 [lindex [gg::dbGetAll] 6]]
gg::dbCurveAddPt -db [list 0 0 [lindex [gg::dbGetAll] 5]]
gg::dbCurveEnd

# Manifold Volume - Arc
gg::dbCurveBegin -type CIRCULAR_ARC

```

```

gg::dbCurveAddPt -db [list 0 0 [lindex [gg::dbGetAll] 7]]
gg::dbCurveAddPt -db [list 0 0 [lindex [gg::dbGetAll] 9]]
gg::dbCurveAddPt -db [list 0 0 [lindex [gg::dbGetAll] 8]]
gg::dbCurveEnd

# Define Connectors

# Create the connectors on the database entities.
gg::conOnDBEnt ALL

# Swirler Cup
SplitConIntoNEqualParts 4 [list [lindex [gg::conGetAll] 6]]
SplitConAtPer 0.25 0
SplitConAtPer 0.25 2
Create2PtCon [list $origin $swirler_cp]
SplitConAtPer 0.4 18
CreateEndptCon [list 14 18\
 16 18]
gg::conDelete [lindex [gg::conGetAll] 18]
SplitConAtPer 0.5 19
SplitConAtPer 0.5 19
SplitConAtPer 0.5 14
SplitConAtPer 0.5 15
CreateEndptCon [list 21 19\
 23 17]
SplitConAtPer 0.5 25
SplitConAtPer 0.5 25
CreateEndptCon [list 17 10\
 19 12]

# Blade Volume and Blade
SplitConIntoNEqualParts 4 [list [lindex [gg::conGetAll] 8]]
SplitConIntoNEqualParts 2 [lrange [gg::conGetAll] 6 7]
CreateEndptCon [list 7 28\
 8 32\
 34 29\
 9 30\
 4 36\
 5 39\
 9 33]
Create2PtCon [list [gg::conGetPt [lindex [gg::conGetAll] 8] -arc 0.0]
[gg::conGetPt [lindex [gg::conGetAll] 32] -arc 0.0]]

# Manifold Volume
SplitConIntoNEqualParts 4 [lindex [gg::conGetAll] 6]

for {set i 27} {$i <= 29} {incr i}\
{
CreateEndptCon [list $i [expr {16 + $i}]]
}

gg::conDim ALL $basic_con_dim

# Create Domains

# Swirler Cup

```

```

CreateStrucDom [list 19 23 21 17\
 23 20 15 22\
 22 16 14 24\
 21 24 13 18\
 13 25 6 10\
 14 12 7 25\
 16 26 8 12\
 15 11 9 26]

# Blade Volume and Blade
CreateStrucDom [list 6 35 27 0\
 42 4 39 35\
 7 36 31 42\
 39 33 37 28\
 8 41 32 36\
 34 40 29 37\
 41 38 40 5\
 9 2 30 38]

# Manifold
CreateStrucDom [list 27 47 43 1\
 28 48 44 47\
 29 49 45 48\
 30 3 46 49]

# UPPER PLANE

# Create Database Entities via Copy
gg::dbCopyBegin ALL
gg::xformTranslate "0 0 $height"
gg::dbCopyEnd

# Create Connectors and Domains via Copy
gg::domCopyBegin ALL
gg::xformTranslate "0 0 $height"
gg::domCopyEnd

# MIDDLE PLANE

# Create Database Entities and Connectors
gg::conDim [gg::conOnDBEnt [Create2PtDB [list 0 29\
 1 30\
 2 31\
 3 32\
 4 33\
 5 34\
 6 35\
 7 36\
 8 37\
 9 38]]] $basic_con_dim

gg::conDim [CreateEndptCon [list 17 53\
 19 50\
 21 52\
 18 61\
 20 54\
 13 60\

```

```
22 56\  
14 58\  
26 67\  
25 62\  
35 71\  
38 87\  
42 74\  
32 83\  
31 78\  
4 75\  
5 88\  
33 79\  
43 92\  
45 97]] $basic_con_dim
```

```
# Create Domains
```

```
CreateStrucDom [list 17 100 53 110\  
18 110 61 113\  
19 111 50 100\  
10 113 64 101\  
114 69 103 11\  
6 119 63 101\  
7 102 66 119\  
8 118 68 102\  
9 103 70 118\  
23 111 51 112\  
24 112 59 115\  
15 114 55 116\  
116 57 117 16\  
25 115 62 119\  
12 117 65 102\  
26 116 67 118\  
0 101 73 104\  
119 71 120 35\  
4 122 75 125\  
5 123 88 126\  
38 118 87 121\  
103 89 106 2\  
42 119 74 122\  
31 124 78 122\  
124 83 123 32\  
118 82 123 41\  
27 104 72 120\  
28 120 81 105\  
105 86 121 29\  
120 76 125 39\  
125 79 127 33\  
127 34 126 84\  
85 121 40 126\  
121 90 106 30\  
104 93 107 1\  
120 91 128 47\  
105 94 108 48\  
121 96 129 49\  
106 98 109 3\  
107 92 128 43\  
128 95 108 44\  
]]
```

```
108 97 129 45\  
129 99 109 46\  
127 80 105 37\  
102 77 124 36\  
21 110 52 112\  
112 56 116 22\  
20 111 54 114\  
13 113 60 115\  
115 58 117 14]
```

```
# CREATE ALL STRUCTURED BLOCKS
```

```
CreateStrucBlk [list 0 20 40 42 49 85\  
3 23 41 50 85 88\  
1 21 49 51 86 87\  
2 22 50 52 86 89\  
4 24 43 45 53 88\  
5 25 46 53 54 89\  
6 26 47 52 54 55\  
7 27 44 48 51 55\  
8 28 45 56 57 66\  
9 29 57 58 62 69\  
10 30 46 62 63 84\  
11 31 67 69 70 83\  
13 33 68 71 72 83\  
12 32 47 64 65 84\  
14 34 59 60 65 72\  
15 35 48 60 61 73\  
16 36 66 74 75 79\  
17 37 67 75 76 80\  
18 38 68 76 77 81\  
19 39 73 77 78 82]
```

```
###  
### BEGIN PATCH - ADDED 13 JULY 2005 TO AVOID HAVING A CELL  
### AGAINST 2 PERIOD BOUNDARY CONDITIONS  
### PATCH IS HACKY - REMOVE IN FUTURE  
###
```

```
gg::blkDelete [lrange [gg::blkGetAll] 0 2] -doms -cons  
gg::conOnDBEnt [lindex [gg::dbGetAll] 58]
```

```
gg::conBegin  
gg::segBegin  
gg::segAddControlPt $origin  
gg::segAddControlPt [gg::conGetPt [lindex [gg::conGetAll] 98] -arc 0.0]  
gg::segEnd  
gg::conEnd
```

```
gg::conBegin  
gg::segBegin  
gg::segAddControlPt $origin  
gg::segAddControlPt [gg::conGetPt [lindex [gg::conGetAll] 99] -arc 0.0]  
gg::segEnd  
gg::conEnd
```

```
gg::conBegin  
gg::segBegin
```

```

gg::segAddControlPt $origin
gg::segAddControlPt [gg::conGetPt [lindex [gg::conGetAll] 18] -arc 0.0]
gg::segEnd
gg::conEnd

CreateEndptCon [list 115 98 \
    115 97 \
    115 99 ]

gg::conDim [lrange [gg::conGetAll] 115 121] $basic_con_dim

CreateStrucDom [list 115 119 98 116\
    115 120 97 118\
    115 121 99 117\
    116 118 18 13 \
    118 117 15 17 \
    119 49 48 120\
    120 121 44 45 ]

CreateStrucBlk [list 40 76 78 79 81 83\
    41 75 79 80 82 84]

###
### END SWIRLER GRID PATCH
###

}

proc CreateBuffer {height_of_swirler_cup}\
{
#####
# Procedure: CreateBuffer #
# Purpose: Creates the buffer between the #
# combustion chamber and the #
# swirler cup. #
#####

gg::dbCopyBegin [concat\
[lrange [gg::dbGetAll] 29 32]\
[lindex [gg::dbGetAll] 45]\
[lindex [gg::dbGetAll] 48]\
[lindex [gg::dbGetAll] 51]\
[lrange [gg::dbGetAll] 58 61]]
gg::xformTranslate "0 0 $height_of_swirler_cup"
gg::dbCopyEnd

gg::blkCopyBegin [concat [lrange [gg::blkGetAll] 0 4] [lrange [gg::blkGetAll]
17 18]]
gg::xformTranslate "0 0 $height_of_swirler_cup"
gg::blkCopyEnd
}

proc CreateCombustionChamber {height_of_swirler_cup
height_of_combustion_chamber basic_con_dim}\
{
#####
# Procedure: CreateCombustionChamber #

```

```

# Purpose: Creates the combustion chamber.      #
# Note: Assumes CreateSwirler has been         #
#        called.                               #
#####

set twice_height_of_swirler_cup [expr {2 * $height_of_swirler_cup}]

# Figure out the factor to multiply basic_con_dim
# by to maintain grid density.
if {$height_of_swirler_cup < $height_of_combustion_chamber}\
{
  set chamber_con_dim [expr {int(($height_of_combustion_chamber /
$height_of_swirler_cup) * $basic_con_dim * 0.15)}]
}\
\
elseif {$height_of_swirler_cup == $height_of_combustion_chamber}\
{
  set chamber_con_dim $basic_con_dim
}\
\
else\
{
  set chamber_con_dim [expr {int($height_of_swirler_cup /
$height_of_combustion_chamber) * $basic_con_dim * 0.15}]
}

# INNER COMBUSTION CHAMBER

# Copy what we can from the buffer.
gg::dbCopyBegin [lrange [gg::dbGetAll] 68 78]
gg::xformTranslate "0 0 $height_of_swirler_cup"
gg::dbCopyEnd

gg::domCopyBegin [concat [lindex [gg::domGetAll] 85] [lindex [gg::domGetAll]
90] [lindex [gg::domGetAll] 95]\
[lindex [gg::domGetAll] 98] [lindex [gg::domGetAll] 101] [lindex
[gg::domGetAll] 107]\
[lindex [gg::domGetAll] 109]]
gg::xformTranslate "0 0 $height_of_swirler_cup"
gg::domCopyEnd

# Create connectors on database entities.
gg::conOnDBEnt [lrange [gg::dbGetAll] 86 89]

# Create extra connectors.
CreateEndptCon [list 132 158\
141 163\
149 167\
150 168\
151 169\
130 156\
122 152\
124 154]

gg::conDim [lrange [gg::conGetAll] 170 181] $basic_con_dim

# Create Domains

```



```

CreateStrucDom [list 170 167 176 149\
 170 168 177 150\
 170 169 178 151\
 176 156 179 130\
 177 155 179 125\
 177 152 180 122\
 178 164 180 143\
 179 154 181 124\
 180 153 181 123\
 176 159 171 133\
 179 157 174 131\
 181 160 172 137\
 180 162 175 140\
 178 165 173 144\
 158 174 132 171\
 161 172 138 174\
 172 163 175 141\
 175 166 173 145]

# Create Blocks
CreateStrucBlk [list 109 116 118 119 122 123\
 107 115 117 118 120 121\
 85 110 121 122 124 125\
 101 114 123 129 130 134\
 98 113 125 128 129 133\
 95 112 124 127 128 132\
 90 111 120 126 127 131]

# Stretch the Database to Size
gg::dbTransformBegin [lrange [gg::dbGetAll] 79 89]
gg::xformStretch [list 0 0 [expr {2 * $height_of_swirler_cup}]]\
 [list 0 0 [expr {3 * $height_of_swirler_cup}]]\
 [list 0 0 [expr {(3 * $height_of_swirler_cup) +
 $height_of_combustion_chamber}]]
gg::dbTransformEnd

# Stretch the Blocks to Size
gg::blkTransformBegin [lrange [gg::blkGetAll] 26 32]
gg::xformStretch [list 0 0 [expr {2 * $height_of_swirler_cup}]]\
 [list 0 0 [expr {3 * $height_of_swirler_cup}]]\
 [list 0 0 [expr {(3 * $height_of_swirler_cup) +
 $height_of_combustion_chamber}]]
gg::blkTransformEnd

# OUTER CHAMBER

# Copy some DB entities from the blade reigon and create connectors on
# them.

gg::dbCopyBegin [concat\
[lrange [gg::dbGetAll] 7 9]\
[lindex [gg::dbGetAll] 28]\
[lrange [gg::dbGetAll] 36 38]\
[lindex [gg::dbGetAll] 57]\
[lrange [gg::dbGetAll] 65 67]]
gg::xformTranslate "0 0 $twice_height_of_swirler_cup"
gg::conOnDBEnt [gg::dbCopyEnd]

```

```

SplitConIntoNEqualParts 2 [lrange [gg::conGetAll] 185 188]

gg::conDim [lrange [gg::conGetAll] 182 192] $basic_con_dim

gg::conDim [CreateEndptCon [list 185 189\
  187 191]] $basic_con_dim

# Stretch to fit.
gg::dbTransformBegin [lrange [gg::dbGetAll] 90 100]
gg::xformStretch [list 0 0 [expr {2 * $height_of_swirler_cup}]]\
  [list 0 0 [expr {3 * $height_of_swirler_cup}]]\
  [list 0 0 [expr {(3 * $height_of_swirler_cup) +
$height_of_combustion_chamber}]]
gg::dbTransformEnd

gg::conTransformBegin [lrange [gg::conGetAll] 182 194]
gg::xformStretch [list 0 0 [expr {2 * $height_of_swirler_cup}]]\
  [list 0 0 [expr {3 * $height_of_swirler_cup}]]\
  [list 0 0 [expr {(3 * $height_of_swirler_cup) +
$height_of_combustion_chamber}]]
gg::conTransformEnd

# Create some more DB entities, and create dimensioned connectors on them.
gg::conDim [gg::conOnDBEnt [Create2PtDB [list 80 94\
  81 95\
  82 96\
  69 90\
  70 91\
  71 92]]] $basic_con_dim

# Create extra dimensioned connectors.
gg::conDim [CreateEndptCon [list 179 189\
  173 191\
  132 185\
  141 187]] $basic_con_dim

# Create Domains
CreateStrucDom [list 181 195 182 198\
  178 201 193 203\
  175 196 183 199\
  171 202 194 204\
  172 197 184 200\
  182 189 193 185\
  193 190 183 186\
  183 191 194 187\
  194 192 184 188\
  179 195 189 201\
  176 201 190 196\
  173 196 191 202\
  169 202 192 197\
  132 203 185 198\
  138 199 186 203\
  141 204 187 199\
  145 200 188 204]

# Create Blocks

```

```

CreateStrucBlk [list 134 135 136 140 144 148\
 131 136 137 141 145 149\
 128 137 138 142 146 150\
 125 138 139 143 147 151]

# Redimension grid to maintain grid density.
gg::conRedimBegin
gg::conRedim [lindex [gg::conGetAll] 195] $chamber_con_dim
gg::conRedim [lindex [gg::conGetAll] 10] $chamber_con_dim
gg::conRedim [lindex [gg::conGetAll] 6] $chamber_con_dim
gg::conRedim [lindex [gg::conGetAll] 7] $chamber_con_dim
gg::conRedim [lindex [gg::conGetAll] 8] $chamber_con_dim
gg::conRedim [lindex [gg::conGetAll] 156] $chamber_con_dim
gg::conRedimEnd
}

#####
# GRAPHICAL INTERFACE PROCEDURES #
#####

proc DrawButtonFireEvent {databaseAttribInputSwirlerCupRadius
databaseAttribInputBladeVolumeRadius databaseAttribInputManifoldRadius
databaseAttribInputNumberOfBlades databaseAttribInputBladeAngle
databaseAttribInputBladeRadius databaseAttribInputHeightS
databaseAttribInputHeightCC meshAttribInputDefaultConnectorDim}\
{
#####
# Procedure: DrawButtonFireEvent #
# Purpose: Handle the pressing of the DRAW #
# button in the GUI. #
#####

ClearWorkspace
CreateSwirler [$databaseAttribInputSwirlerCupRadius get]
[$databaseAttribInputBladeVolumeRadius get]
[$databaseAttribInputManifoldRadius get] [$databaseAttribInputNumberOfBlades
get] [$databaseAttribInputBladeAngle get] [$databaseAttribInputBladeRadius
get] [$databaseAttribInputHeightS get] [$meshAttribInputDefaultConnectorDim
get]
CreateBuffer [$databaseAttribInputHeightS get]
CreateCombustionChamber [$databaseAttribInputHeightS get]
[$databaseAttribInputHeightCC get] [$meshAttribInputDefaultConnectorDim get]
MakeWhole [$databaseAttribInputNumberOfBlades get]
}

proc CreateGUI {}\
{
#####
# Procedure: CreateGUI #
# Purpose: Creates a very crude GUI for #
# BladeMaker. #
# Note: This procedure is for TESTING #
# PURPOSES ONLY. The GUI is cryptic #
# and probably won't even exist in #
# BladeMaker 1.0 #
#####

```

```

gg::tkLoad

wm title . "BladeMaker Testing GUI"

set row 0

foreach i {SwirlerCupRadius BladeVolumeRadius ManifoldRadius NumberOfBlades
BladeAngle BladeRadius HeightS HeightCC}\
{
label .databaseAttribLabel$i -text $i
entry .databaseAttribInput$i
lappend entrylist .databaseAttribInput$i

grid .databaseAttribLabel$i -row $row -column 0 -sticky w
grid .databaseAttribInput$i -row $row -column 1

incr row
}

set row 0

foreach i {DefaultConnectorDim}\
{
label .meshAttribLabel$i -text $i
entry .meshAttribInput$i
lappend entrylist .meshAttribInput$i

grid .meshAttribLabel$i -row $row -column 2 -sticky w
grid .meshAttribInput$i -row $row -column 3

incr row
}

set row 8
set col 0

frame .buttons

button .buttons.draw -text "Draw" -command "DrawButtonFireEvent $entrylist"
button .buttons.close -text "Close" -command "exit"

pack .buttons.draw -side left
pack .buttons.close -side right

grid .buttons -row $row -column $col

gg::tkLoop
}

#####
# BEGIN MAIN GLYPH SCRIPT #
#####

###
### DO ***NOT*** MODIFY THIS SECTION UNLESS
### YOU KNOW WHAT YOU ARE DOING
### ORDER OF FUNCTION CALLS ***DOES*** MATTER

```

```

###

ClearWorkspace

switch $RUN_MODE\
{
0\
{
CreateSwirler $swirler_rad $bladevol_rad $manifold_rad $num_blades $blade_ang
$blade_rad $height_s $basic_con_dim
RefineSwirlerGrid $wall_ds $blade_con_dim
CreateBuffer $height_s
CreateCombustionChamber $height_s $height_cc $basic_con_dim
RefineCombustionChamberGrid $wall_ds
TagBoundaryConditions
ExportASW $fname
}
\
1\
{
CreateGUI
}
}

###
### END DO NOT MODIFY SECTION
###

```


References

1. Tacina, Robert, et al.: Experimental Sector and Flame-Tube Evaluations of a Multipoint Integrated Module Concept for Low Emission Combustors. ASME Paper GT2004-53263, 2004.
2. Tacina, Robert, et al.: Sector Tests of Low-NO_x, Lean-Direct-Injection, Multipoint Integrated Module Combustor Concept. ASME Paper GT2002-30089, 2002.
3. Tacina, Robert, et al.: A Low NO_x Lean-Direct Injection, Multipoint Integrated Module Combustor Concept for Advanced Aircraft Gas Turbines. NASA/TM-2002-211347, 2002.
4. Iannetti, A., et al.: National Combustion Code Calculations of a NASA Low-NO_x Hydrogen Injector Concept. AIAA-2007-838, 2007.
5. Iannetti, A.; and Moder, J.: Analysis of MEMS-LDI Technology Using the National Combustion Code. AIAA-2005-169, 2005.
6. Gridgen CFD Meshing Software. Pointwise, Inc., Fort Worth, TX. <http://www.pointwise.com/> Accessed Jan. 30, 2008.
7. Stubbs, Robert M.; and Liu, Nan-Suey: Preview of the National Combustion Code. AIAA-1997-3114, 1997.
8. Quealy, A., et al.: National Combustion Code: Parallel Implementation and Performance. NASA/TM-2000-209801 (AIAA Paper 2000-0336), 2000.
9. Quealy, Angela; and Benyo, Theresa: National Combustion Code Parallel Performance Enhancements; Final Report. NASA/CR-2002-211340 (AIAA Paper 2002-3706), 2002.
10. Shih, Tsan-Hsing, et al.: Generalized Wall Function for Complex Turbulent Flows. NASA/TM-2000-209936 (ICOMP-2000-04), 2000.
11. Chien, Kuei-Yuan: Predictions of Channel and Boundary-Layer Flows With a Low-Reynolds-Number Turbulence Model. AIAA J., vol. 20, no. 1, 1982, pp. 33-38.
12. Shih, Tsan-Hsing, et al.: Modeling of Turbulent Swirling Flows. NASA TM-113112 (ICOMP-97-08), 1997.
13. Shih, Tsan-Hsing; and Liu, Nan-Suey: A Non-Linear k-Epsilon Model for Turbulent Shear Flows. AIAA-1998-3983, 1998.
14. Venkateswaran, S.; Weiss, J.M.; and Merkle, C.L.: Propulsion-Related Flowfields Using the Preconditioned Navier-Stokes Equations. AIAA-1992-3437, 1992.
15. Venkateswaran, S.; and Merkle, C.L.: Efficiency and Accuracy Issues in Contemporary CFD Algorithms. AIAA-2000-2251, 2000.
16. NASA Open Source Software. <http://opensource.arc.nasa.gov/> Accessed Jan. 30, 2008.

REPORT DOCUMENTATION PAGE

*Form Approved
OMB No. 0704-0188*

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE (DD-MM-YYYY) 01-07-2008		2. REPORT TYPE Technical Memorandum		3. DATES COVERED (From - To)	
4. TITLE AND SUBTITLE An Idealized, Single Radial Swirler, Lean-Direct-Injection (LDI) Concept Meshing Script				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Iannetti, Anthony, C.; Thompson, Daniel				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER WBS 984754.02.07.19.02	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration John H. Glenn Research Center at Lewis Field Cleveland, Ohio 44135-3191				8. PERFORMING ORGANIZATION REPORT NUMBER E-16261	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001				10. SPONSORING/MONITORS ACRONYM(S) NASA	
				11. SPONSORING/MONITORING REPORT NUMBER NASA/TM-2008-215048	
12. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified-Unlimited Subject Category: 07 Available electronically at http://gltrs.grc.nasa.gov This publication is available from the NASA Center for AeroSpace Information, 301-621-0390					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT To easily study combustor design parameters using computational fluid dynamics codes (CFD), a Gridgen Glyph-based macro (based on the Tcl scripting language) dubbed BladeMaker has been developed for the meshing of an idealized, single radial swirler, lean-direct-injection (LDI) combustor. BladeMaker is capable of taking in a number of parameters, such as blade width, blade tilt with respect to the perpendicular, swirler cup radius, and grid densities, and producing a three-dimensional meshed radial swirler with a can-annular (canned) combustor. This complex script produces a data format suitable for but not specific to the National Combustion Code (NCC), a state-of-the-art CFD code developed for reacting flow processes.					
15. SUBJECT TERMS Computational fluid dynamics; Optimization; Emissions; Combustion					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (include area code)
U	U	U	UU	40	STI Help Desk (email:help@sti.nasa.gov) 301-621-0390

