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

Anthony C. Iannetti*
NASA Glenn Research Center
Aerospace Engineer
Anthony.C.Iannetti@nasa.gov

Daniel Thompson†
Ohio Wesleyan University
Computer Science
bluntwit@gmail.com

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 produce a three-dimensional meshed radial swirler with a 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. While this concept produces reduced emissions in flame tube tests, the mixing and combustion processes must be understood to optimize this concept.[1, 2, 3] CFD programs allow LDI design studies of on computers. However in past studies, the geometries were fixed because of the difficulties in creating a computational mesh.[4, 5] To better optimize the LDI concept using current CFD codes, parametric geometry manipulation is a necessary requirement.

While parametric geometries may be easily created use Computer Aided Design (CAD) tools like Pro Engineer, 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 is still a research problem. To solve this problem a Gridgen[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

* Aerospace Engineer, Combustion Branch, 21000 Brookpark Rd, MS 5-10.
† LERCIP Intern, May 2004 – August 2005, Combustion Branch.

perpendicular, swirler cup radius, and grid densities and produce 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 are: the use of unstructured grids[7], massively parallel computing – with almost perfectly linear scalability[8,9], a dynamic wall function with the effect of adverse pressure gradient[10], low Reynolds number wall treatment[11], a cubic non-linear k-epsilon turbulence model[12,13], and stiff laminar chemistry integration. Recently, viscous low-speed preconditioning[14,15] 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 computing turbulent, reacting flows.

Notes on Formatting and Naming

This document uses a couple of different fonts to indicate different 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 of the files with the extension `sh` into it. These files can be found in the BladeMaker root directory.

Parameter File Construction

```
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 The default Parameter File, `param.in`.

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.

The top section of this file 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 the meaning of each of these variables as well as the limits BladeMaker can handle can be found in Table 1 below. Because of "stretching" of the structured mesh, the variable blade_ang has a limitation of approximately -50 to 50 degrees to the perpendicular to keep mesh quality high.

Table 1 Descriptions & Limits in the Parameter File

Variable Name	Description	Lower Limit	Upper Limit	Data Type
ASW_TYPE	Analysis software package.	Any valid Gridgen analysis software type. See the Gridgen documentation for more information.		
ASW_DIM	Number of dimensions in the geometry.	3	3	positive integer
swirler_rad	Radius of the swirler cup.	1.0	less than bladevol_rad	positive real
bladevol_rad	The distance from the center of the swirler cup to the edge of the blade volume.	greater than swirler_rad	less than manifold_rad	positive real
manifold_rad	Radius of the entire radial swirler.	greater than bladevol_rad	infinite	positive real
num_blades	The number of blades in the 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 2 blades for unknown reasons.		
blade_rad	One-half of the blade width.			
blade_ang	The tilt of a single blade with respect to the perpendicular. Counterclockwise is positive and clockwise is negative.	-50.0	50.0	real
height_s	The height of the region of the swirler containing the blades.	0.0	infinite	positive real
height_cc	The height of the combustion chamber.	0.0	infinite	positive real
basic_con_dim	The number of points placed initially on a connector.	2	infinite	positive integer
blade_con_dim	The number of points placed on a connector near the 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	The file name for the analysis software file, like `patran.out`.	Any writable absolute file name.		
EOF	This is not really a variable, but signals the end of the file.			

The 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 needs to be modified in this script, as highlighted in Figure 2 below. `PARAM_FILE` must be set to the absolute file path of your parameter file instead of 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`

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.

Table 2 Shell Script Descriptions

Shell Script	Description	Output
`unix-unopt.sh`	Runs BladeMaker	An analysis software boundary conditions file, such as `patran.out`.

The shell script `unix-unopt.sh`, shown in Figure 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 Gridgen Version 15.07 or later available. If your system is not configured as such, you will need to use this section as a base to create your own shell script.

```
#!/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.

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 the BladeMaker Glyph Script

Previous Knowledge

This 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`. It is divided into a number of sections, indicated by comment headers. These sections are, in order: parameters, procedure definitions, and the main Glyph script. Each is covered in detail below.

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 * arc tan 1.0, which is a good approximation of π. `RUN_MODE` defines if 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

The procedure definitions section is where all procedures used in the macro are defined. It 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 re-written, which is both 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. 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 called 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 1st point and the 2nd point, then the 3rd point and the 4th point, and so on. For example, if it was 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 four. 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 1st 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 2nd edge would be found in the 4th element, the 3rd 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 six 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 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, that is, 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 clusters 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 potion of the swirler by changing the connector dimension from the default to blade_con_dim and clusters points near the blade walls by settings the average Δs value of the sub connectors 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 sub connectors to wall_ds.

Points must also be clustered along 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`. This is due to the fact 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, as well as other smaller procedures, in such an order as to create large sections of the final grid and geometry. While 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 blades 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 come up with 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 will fail if it has not.

Mesh Examples

Figure 1Figure 4 - Figure 7 show various meshes that have been generated by the BladeMaker script. The parameters used to generate these meshes are typical for a LDI CFD design study. All of these meshes are about 125,000 computational cells. In general, it takes about ten seconds to generate these meshes. Using meshes with higher cell counts can take up to two minutes to generate. These meshes were generated using a Linux workstation with 2GHz Intel Xeon 5100 series dual core processors.

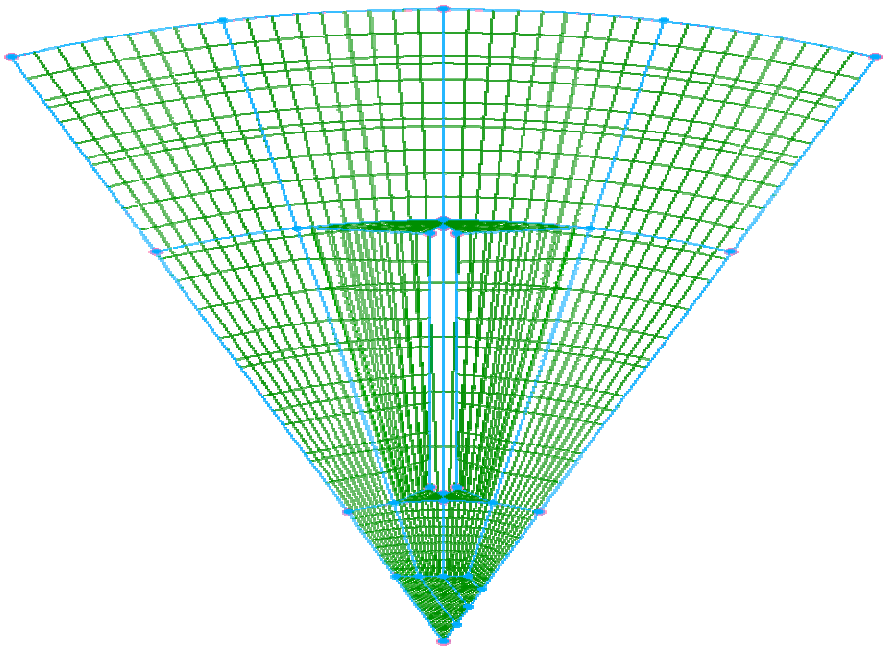

Figure 4 BladeMaker generated grid with no blade tilt, 8 blades, and a blade width of 0.2 - Top Down View.

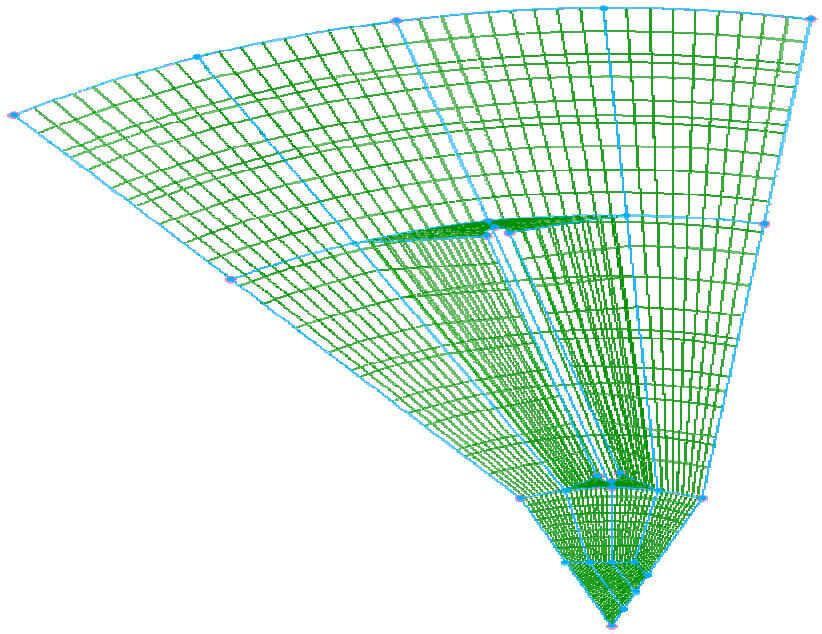

Figure 5 BladeMaker generated grid with a blade tilt of 15.0 degrees, 8 blades, and a blade width of 0.2 - Top Down View.

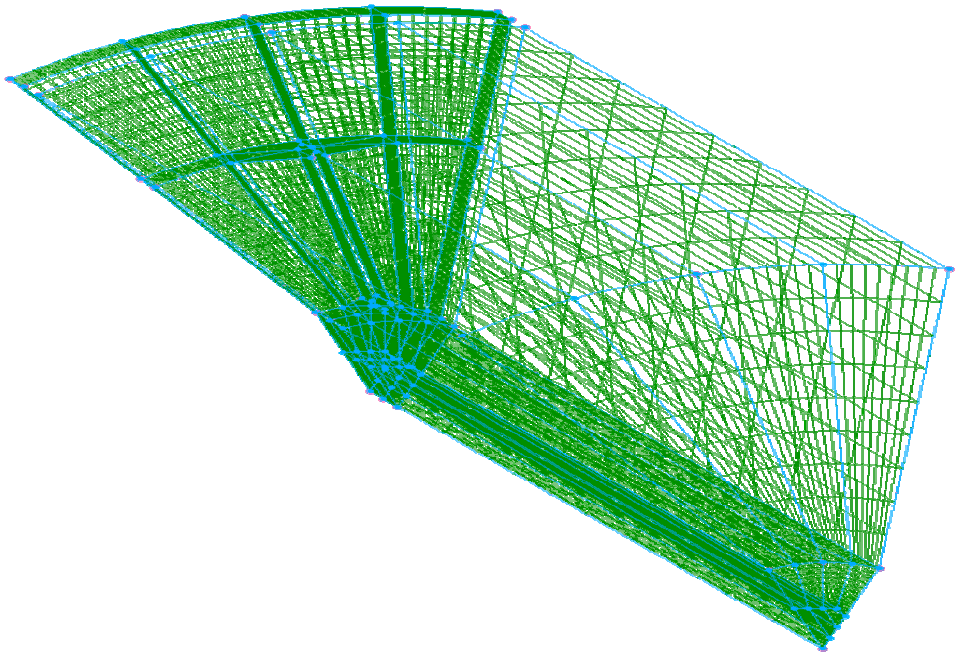

Figure 6 BladeMaker generated grid with a blade tilt of 15.0 degrees, 8 blades, a blade width of 0.2, swirler height of 1.0, and the combustion chamber height of 30.0 - Perspective View.

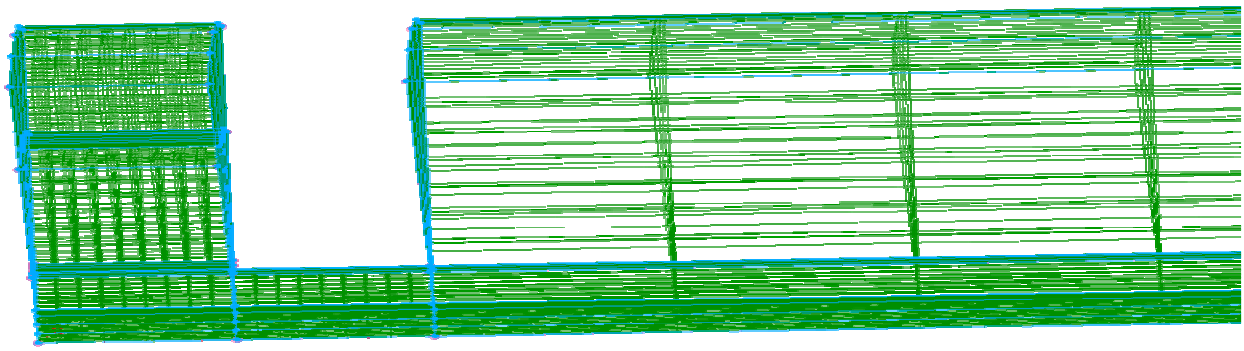

Figure 7 BladeMaker generated grid with a blade tilt of 15.0 degrees, 8 blades, a blade width of 0.2, swirler height of 3.0, and the combustion chamber height of 30.0 - SideView.

Conclusions

The BladeMaker script removes a major time bottleneck for studying LDI concepts with CFD codes, like the NCC. While the geometry produced is a simple can-annular (canned) combustor, it is suitable for studying important combustor design parameters, like 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. This is also important for determining if the resulting CFD solution is mesh independent.

We believe this script is mostly error free in its current format, but 100% 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.[16] By releasing BladeMaker in an open source model, we *strongly* believe 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 this software in support of NASA's education mission.

Acknowledgements

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

Appendix A: 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, R., Mansour, A., Partelow, L., Wey, C., (2004), "Experimental Sector and Flame-Tube Evaluations of a Multipoint Integrated Module Concept for Low Emission Combustors" GT-2004-53263, ASME Turbo Expo 2004, Vienna, Austria.

[2] Tacina, R., Wey, C., Laing, P., and Mansour, A., (2002), "Sector Tests of a Low-NOx, Lean-Direct-Injection, MultiPoint Integrated Module Combustor Concept," GT-2002-30089.

[3] Tacina, R., Wey, C., Laing, P., and Mansour, A., (2002), "A Low NOx Lean-Direct Injection, MultiPoint Integrated Module Combustor Concept for Advanced Aircraft Gas Turbines," July 9-12, 2001, Porto, Portugal. NASA TM-2002-211347.

[4] Iannetti, A. C., Kundu, K., Foster, L., and Redding, C., (2007), "National Combustion Code Calculations of a NASA Low-NOx Hydrogen Injector Concept", AIAA-2007-838, 45th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

[5] Iannetti, A. C., Moder, J. P., (2005), "Analysis of MEMS-LDI Technology Using the National Combustion Code", AIAA-2005-0169, 43rd AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

[6] Gridgen, A Meshing Toolkit, Pointwise, Inc.,213 S. Jennings Ave.,Fort Worth, Texas 76104-1107,USA, http://www.pointwise.com/ .

[7] Stubbs, R., M., and Liu, N.-S., (1997), "Preview of the National Combustion Code", AIAA 97-3114, 33rd AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, Seattle, WA, USA, July 6-9.

[8] Quealy, A., Ryder, R., Norris, A., and Liu, N.-S., (2000), "National Combustion Code: Parallel Implementation and Performance", NASA TM-2000-209801.

[9] Quealy, A., (2000), "National Combustion Code Parallel Performance Enhancements", NASA/CR-2002-211340.

[10] Shih, T.-H., Povinelli, L. A., Liu, N.-S and Chen, K.-H., (2000), "Generalized Wall Function for Complex Turbulent Flows", NASA TM-2000-209936.

[11] Chien, K. Y., "Prediction of Boundary Layer Flows with a Low – Reynolds Number Turbulence Model", (1982), AIAA J., Vol. 20, No. 1, pp 33-38.

[12] Shih, T.-H., Chen, K.-H., Liu, N.-S., (1998), Lumley, J. L., "Modeling of Turbulent Swirling Flows", NASA-TM –113112.

[13] Shih, T.-H., Chen, K.-H., and Liu, N.-S., (1998), "A Non-Linear k-epsilon Model for Turbulent Shear Flows", AIAA Paper 98-3983.

[14] S. Venkateswaran, J.M. Weiss, C.L. Merkle, Y.-H. Choi, (1992), "Propulsion-Related Flowfields Using Preconditioned Navier-Stokes Equations", AIAA-92-3437.

[15] S. Venkateswaran, C.L. Merkle, (2002), "Efficiency and Accuracy Issues in Contemporary CFD Algorithms", AIAA-2002-2251.

[16] "NASA Open Source Software", http://opensource.arc.nasa.gov/ .