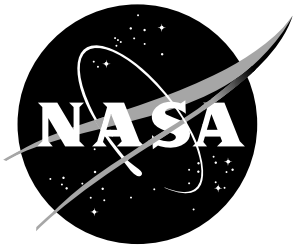
NASA/TM–20220008781



VULCAN-CFD User Manual: Ver. 7.2.0

Robert A. Baurle, Jeffery A. White, Matthew D. O'Connell,
Tomasz G. Drozda, and Andrew T. Norris

NASA Langley Research Center, Hampton, Virginia

November 2022

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 Report Server, thus providing one of the largest collection 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:

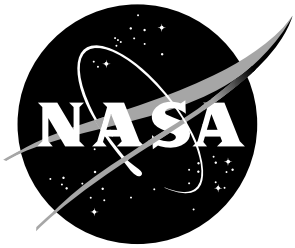
- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having 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 co-sponsored 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 organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

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 to `help@sti.nasa.gov`
- Phone the NASA STI Information Desk at 757-864-9658
- Write to:
  NASA STI Information Desk
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/TM–20220008781



VULCAN-CFD User Manual: Ver. 7.2.0

Robert A. Baurle, Jeffery A. White, Matthew D. O'Connell,
Tomasz G. Drozda, and Andrew T. Norris

NASA Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

November 2022

Acknowledgments

VULCAN-CFD development has been funded over the years by a variety of NASA and AFRL projects, but the new features described in this version of the User Manual were funded solely by the NASA Hypersonic Technology Project. The authors would also like to recognize others that made the latest software enhancements possible. Hiroaki Nishikawa is acknowledged for providing the ideas and fundamental research behind the node-centered gradient methodology that has been implemented in this release. A special thanks is extended to Mike Park for his assistance with the integration of *refine* with VULCAN-CFD, as well as Cameron Druyor and Kyle Thompson for their contributions to the T-infinity capabilities that have been made available to VULCAN-CFD. The authors would like to also acknowledge Mike Park and Ray Gomez for providing documentation for the *refine* grid adaptation process.

Available from:

NASA STI Program / Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199
Fax: 757-864-6500

Contents

1 Introduction

VULCAN-CFD offers a comprehensive set of capabilities to enable the simulation of continuum flowfields from subsonic to hypersonic conditions. The governing equations that are employed include allowances for both chemical and thermal nonequilibrium processes, coupled with a wide variety of turbulence models for both Reynolds-averaged and large eddy simulations. The software package can simulate two-dimensional, axisymmetric, or three-dimensional problems on structured multiblock meshes or fully unstructured meshes. A parabolic (i.e., space-marching) treatment can also be used for any subset of a structured mesh that can accommodate this solution strategy. The flow solver provides a significant level of geometric flexibility for structured grid simulations by allowing for arbitrary face-to-face C(0) continuous and non-C(0) continuous block interface connectivities. The unstructured grid paradigm allows for mixed element unstructured meshes that contain any combination of tetrahedral, prismatic, pyramidal, and hexahedral cell elements. The flow solver is also fully parallelized using MPI (Message Passing Interface) libraries in a data-parallel fashion, allowing for efficient simulations on modern High Performance Computing (HPC) systems. This document provides information related to the installation and execution of the VULCAN-CFD software package. A detailed description of the physical and numerical models available in the software are provided in the VULCAN-CFD Theory Manual.[1]

1.1 Getting Started

The user manual consists of 22 chapters organized in the following manner:

- Chapter 1 provides an introductory overview of VULCAN-CFD.
- Chapter 2 provides a description of hardware and software requirements for the installation and execution of VULCAN-CFD along with a description of the installation process.
- Chapter 3 describes the CFD process (as it pertains to VULCAN-CFD) and introduces some of the basic terminology used to characterize various aspects of the software.
- Chapters 4 through 7 walk through various example problems designed to familiarize the user with some of the high-level capabilities of the software.
- Chapter 8 provides a full description of all of the VULCAN-CFD input parameters.
- Chapter 9 provides a description of the available VULCAN-CFD boundary conditions, including auxiliary boundary specifications (such as those required for bleed modeling).
- Chapters 10 through 12 describe the setting of initial conditions, turbulence suppression, and ignition sources within user-specified bounding volumes.
- Chapter 13 describes the specifications to control time history output for unstructured grid simulations.

- Chapter 14 describes the mapping of boundary conditions to structured grid block boundaries.
- Chapters 15 and 16 describe the specification of C(0) and non-C(0) connectivity between structured grid blocks.
- Chapters 17, 18, and 19 describe the specification of spatial subblocks used to control turbulence suppression, ignition, and time history output for structured grid simulations.
- Chapter 20 describes the various output files that are generated for postprocessing purposes.
- Chapter 21 provides some general guidance to help with troubleshooting VULCAN-CFD simulations.
- Chapter 22 describes the database format(s) that provide the species thermodynamic and transport property information.
- Chapter 23 describes the database format that provides the additional species information required for simulations of flows in thermodynamic nonequilibrium.
- Chapter 24 describes the database format that provides the chemical kinetic rate information for flows in chemical nonequilibrium.
- Chapter 25 describes various utility codes included in the software suite that supplement the core CFD solver.

New users of VULCAN-CFD are encouraged to initially focus on the first 7 chapters, since this will provide the essential information needed to become familiar with the installation process, and how to perform a basic simulation setup. The remaining chapters can be regarded as more of a reference resource to be utilized as one becomes more familiar with the software.

2 Installation and Execution of VULCAN-CFD

The minimal installation requirements for VULCAN-CFD are a unix environment (e.g., linux, MacOS, or linux emulator on a Windows OS) and a FORTRAN 90 (or later) source code compiler. Additional third-party libraries are required for installations intended to support parallel simulations (e.g., OpenMPI or MPICH), graph-based grid partitioning (METIS[2]), and unstructured grid adaptation (Engineering Sketch Pad[3] and OpenCascade[4]). These third-party libraries must be installed prior to the installation of VULCAN-CFD to enable the capabilities that require them. VULCAN-CFD also must be installed and/or executed from within a C-shell, since C-scripts are used to control the installation and execution of VULCAN-CFD.

2.1 Installation Instructions

To begin the installation process, first make sure that you are in a linux C-shell. If you are uncertain, type the following in any open shell terminal:

```
echo  $shell
```

If something other than **/bin/tcsh** or **/bin/csh** is returned, then type the following command in the shell terminal:

```
tcsh
```

to open a C-shell. Also, be sure that your system does **not** have the "noclobber" shell variable set. If so (or if unsure whether this setting is active), disable it by adding the following line to your personal **.tcshrc** (or **.cshrc**) file located in your home directory:

```
unset  noclobber
```

After confirming that you are in a C-shell environment (and addressed the noclobber setting), extract the VULCAN-CFD tarball file in a directory (or folder) of your choosing, i.e.,

```
tar  -xvf  Ver_#.#.#.tar.gz
```

This will create the directory that houses VULCAN-CFD and all of its components. The next step is to enter the **Ver_#.#.#/Scripts** directory, which contains the various script files that control the VULCAN-CFD installation and execution processes. The file that requires editing to define the computational environment for VULCAN-CFD is the **vulcan_config** script. For the most part, this is the only script file that requires any modification by the user. However, under some circumstances, minor changes to the **mkvulcan.tcsh** file may also be required depending on the specific FORTRAN compiler version being utilized on your system. Using a text editor of your choice, edit the **vulcan_config** script file and select the appropriate settings required by each step of the installation process. An option is "selected" for each step by uncommenting the desired line (i.e., removing the leading # symbols from the line).

Step 1) Select the compilation environment.

(I) Select one of the following for installations on Linux systems

```
##setenv  PLATf  Linux_IFORT     (Intel FORTRAN Compiler)
##setenv  PLATf  Linux_GFC       (GNU FORTRAN Compiler)
##setenv  PLATf  Linux_NFC       (Nagware FORTRAN Compiler)
##setenv  PLATf  Linux_LFC       (Lahey FORTRAN Compiler)
```

(II) Otherwise, select one of the following for Unix/Linux systems that utilize their own proprietary FORTRAN compilers

```
##setenv  PLATf  HP-UX           (Hewlett Packard Unix System)
##setenv  PLATf  Sun             (Sun Solaris Unix System)
##setenv  PLATf  IBM             (IBM AIX Unix System)
##setenv  PLATf  Cray            (Cray Unix System)
```

Step 2) Select the execution mode (serial or specific parallel mode). A couple of points need to be clarified about this setting. The first point is that the parallel_mpi option is only meant to be used when the MPI libraries are explicitly linked through the FORTRAN compilation command line (rather than using a wrapper script like mpif90). This is a common practice for proprietary implementations of MPI. The other point of clarification involves systems that utilize the MPICH package. The parallel_mpich2 option should only be selected if the desire is to have VULCAN-CFD control the MPICH daemon environment (e.g., mpdboot or mpdallexit). If manual control of the MPICH daemons is desired (or if your batch system handles that for you), then choose the generic parallel_mpich option instead. If the parallel_mpich2 option is selected, then a remote shell protocol (rsh or ssh) must also be selected. VULCAN-CFD controls the MPICH daemon environment when this option is selected, which requires knowledge of the specific remote shell protocol used by MPICH. Note that many systems have rsh disabled due to security concerns, so the ssh protocol should be chosen under most circumstances.

```
##setenv  VULCAN_run_mode  serial            (serial install and execution)
##setenv  VULCAN_run_mode  parallel_mpi      (parallel via proprietary MPI)
##setenv  VULCAN_run_mode  parallel_mpich    (parallel via MPICH)
##setenv  VULCAN_run_mode  parallel_mpich2   (parallel via MPICH2)
##setenv  VULCAN_run_mode  parallel_openmpi  (parallel via OpenMPI)
##setenv  VULCAN_run_mode  parallel_mvapich  (parallel via MVAPICH)
```

Select the remote shell protocol if parallel_mpich2 was chosen above

```
##setenv  VULCAN_rmt_protocol  rsh
##setenv  VULCAN_rmt_protocol  ssh
```

Step 3) Define the VULCAN-CFD version number and installation path. The intent here is to encourage the inclusion of the version number as part of the installation path name,

but this is not required. The only requirement is that the "vulcanpath" environment variable points to the main directory of VULCAN-CFD.

##setenv vulcan_ver Ver_#.#.#

##setenv vulcanpath $HOME/Vulcan/$vulcan_ver

Step 4) Enable/Disable access to the following features of the T-infinity[5] framework for unstructured grid simulations:

- parallelized grid partitioning
- parallelized wall distance calculator
- export of volumetric plot files in Tecplot (.plt) and legacy VTK (.vtk) format mapped to the unpartitioned grid
- export of surface loads files in Tecplot (.plt) and legacy VTK (.vtk) format mapped to the unpartitioned grid
- export of time history data (for the full volume, boundary surfaces, or arbitrary planes) in Tecplot (.plt) format for unsteady simulations
- restart of a simulation with a different number of processors/cores than what was used to create the restart files
- access to certain flowfield initialization features
- access to anisotropic tetrahedral grid adaptation with *refine*[6]
- use of specialized extended gradient stencils

T-infinity **must** be enabled to perform unstructured grid simulations in a parallel fashion, and it requires that GCC 6.2 (or higher) and CMake 3.6 (or higher) be available on your system. The FORTRAN compiler selected in the first step above must also be 2003 compliant, since the iso_c_binding feature is used for FORTRAN / C interoperability. If an Intel compiler platform was chosen, then the Intel C and C++ compilers can optionally be selected instead of the GNU versions (which are used by default).

##setenv VULCAN_TINF disable

##setenv VULCAN_TINF enable

Step 5) If T-infinity is enabled and the anisotropic tetrahedral mesh adaptation feature is desired, then provide the paths to Engineering Sketch Pad (ESP)[3] and OpenCascade.[4]

##setenv OPENCASCADE_PATH <your_path_to_opencascade>

##setenv ESP_PATH <your_path_to_esp>

It is strongly recommended that you use a prebuilt ESP binary distribution from https://acdl.mit.edu/ESP/PreBuilts, which already contains a properly configured install of OpenCascade (see the anisotropic tetrahedral mesh adaptation section of Chapter 25 for further details).

Step 6) Enable/Disable the use of METIS[2] for graph-based partitioning. Enabling METIS is optional since native structured and unstructured (via T-infinity) grid partitioners are included with VULCAN-CFD. If METIS is enabled, set the desired version of METIS (or ParMETIS) and provide the path to the directory that houses the METIS (or ParMETIS) libraries, i.e.,

##setenv VULCAN_METIS disable

##setenv VULCAN_METIS enable

If enabled, then select either a serial version of the software,

##setenv VULCAN_METIS_VER METIS_4

##setenv VULCAN_METIS_VER METIS_5

or select a parallel version of the software,

##setenv VULCAN_METIS_VER PARMETIS_3

##setenv VULCAN_METIS_VER PARMETIS_4

and then define the path to the METIS (or ParMETIS) libraries:

##setenv VULCAN_METIS_LIB <your_path_to_metis_lib>

Any of the above METIS software packages can be selected for structured grid partitioning, but ParMETIS must be selected to enable graph-based partitioning for unstructured grids. If ParMETIS is selected, then the MPI package used to install ParMETIS must match the MPI package chosen when installing VULCAN-CFD.

Step 7) Specify if access to the Tecplot[7] binary I/O libraries is desired for the output of data files in either the .plt or .szplt format for structured grid simulations. If this feature is not desired, then simply select the VULCAN_TECIO disable line. Otherwise, select the option to either install the libraries as part of the VULCAN-CFD installation process, or link to an existing Tecplot installation to access them. To link to existing Tecplot I/O libraries, uncomment the setenv VULCAN_TECIO link line, and define the path to the top-level Tecplot installation directory. If parallel versions of the libraries are desired (allowing for faster I/O), then you must select the install option instead to ensure that the libraries are binary compatible with your specific MPI package.

##setenv VULCAN_TECIO disable

##setenv VULCAN_TECIO install

##setenv VULCAN_TECIO link

If linking to an existing Tecplot installation, define the path to the Tecplot directory:

##setenv VULCAN_TECIO_PATH <your_path_to_tecplot>

If installing the TecIO libraries included with VULCAN-CFD, choose whether to enable or disable use of the parallel TecIO libraries:

##setenv VULCAN_TECIO_MPI disable

##setenv VULCAN_TECIO_MPI enable

Step 8) Select (or define) the command line required for parallel execution of VULCAN-CFD. The command line structure for multicore execution of applications that utilize MPI libraries is not universal, but several common command line structures are provided and can be edited as required. This step can be skipped if the VULCAN-CFD execution mode was set to serial in Step 2) above. The command line for parallel execution must be expressed in terms of the shell variables that contain the information provided on the VULCAN-CFD command line during run-time execution. These shell variables are defined as follows:

$num_cpus is the number of processors (cores) specified on the command line

$vulcan_mpi_hst is the MPI host file specified on the command line

$vulcan_cmnd is the specific VULCAN-CFD executable file

Several representative scenarios commonly encountered on current HPC systems are described in the examples below:

Example 1: parallel execution command line specification for a shared computer resource that utilizes a batch scheduler (e.g., PBS or Slurm), or a local workstation that is not a shared resource. These scenarios are the most common ones encountered in practice, and do not require that the user specify a host list to be utilized by the simulation. In a batch environment, the batch scheduler assigns the host list for you, while on a personal workstation, the operating system controls the scheduling of the cores based on the system load.

set mpi_cmnd = 'mpiexec -n $num_cpus $vulcan_cmnd'

Example 2: parallel execution command line specification for a shared compute cluster that does not utilize a batch scheduler. This scenario requires that the user specify the specific host list for the simulation to avoid accessing cores that are already being used by other users on the system. Most MPI packages handle this scenario via the option -machinefile or -hostfile.

set mpi_cmnd = 'mpiexec -hostfile $vulcan_mpi_hst -n $num_cpus $vulcan_cmnd'

Note that there are many other command line arguments associated with the mpiexec command that may be beneficial (or required) on your particular system. If the above examples are not applicable to your system, then follow the suggested hierarchy below to determine how to define this command line for your system:

(1) If on a large shared resource, check the system FAQ or How-To pages.

(2) Check the man pages or website documentation for your specific MPI package.

(3) Contact the VULCAN-CFD development team for assistance.

All required system-specific installation options have now been selected, so save the file and exit the text editor. To test the validity of your settings, return to your home directory, edit your personal **.tcshrc** (or **.cshrc**) file, and add the following line to this file:

source <your_path_to_vulcan>/Scripts/vulcan_config

When completed, save the file and type the following command to activate the VULCAN-CFD settings in your C-shell environment:

source .tcshrc

Replace .tcshrc with .cshrc in the above instruction if you utilize a .cshrc to control your C-shell environment. If no errors are reported, then type:

cd $vulcanpath

This should take you back to the main directory of the VULCAN-CFD installation. If not, then this likely indicates that there is an inconsistency between the line entered in your .tcshrc (or .cshrc) file and the VULCAN-CFD installation path settings you provided in Step 3) above.

At this point, the VULCAN-CFD environment should be correctly defined for your particular system, and the software is ready to be installed. To perform a fresh install of VULCAN-CFD, type:

install_vulcan new

The install_vulcan script controls all aspects of the installation process, and can also be used to install individual components of VULCAN-CFD. However, the initial installation of VULCAN-CFD should always use the "new" option to ensure that all installation steps have been performed in the correct order. After the initial installation, the install_vulcan script can reliably be used to reinstall any particular aspect of VULCAN-CFD if required. A complete list of installation options is given below:

install_vulcan new → performs a clean install of everything
install_vulcan all → compiles everything but utility codes
install_vulcan pre → compiles the preprocessor routines
install_vulcan slv → compiles the flow solver routines
install_vulcan pst → compiles the postprocessor routines
install_vulcan pch → compiles the non-C(0) patch-processor codes
install_vulcan utl → compiles the utility codes
install_vulcan tinf → compiles the T-infinity software
install_vulcan sgld → compiles the structured grid partitioning software
install_vulcan cgns → builds the CGNS libraries
install_vulcan tecio → builds the TecIO libraries
install_vulcan cln → removes all object and executable files
install_vulcan dep → creates the Makefile dependency list

If the installation process is not successful, check the output stream for any compilation errors that are reported. The most common installation issue is the situation where a version of the FORTRAN compiler used on your system is different than the one used by the VULCAN-CFD development team (since vendors will often deprecate compilation options and/or rename them). If the error messages appear to be related to this scenario, then

the appropriate course of action is to edit the **Scripts/mkvulcan.tcsh** file, and search for the specific if-construct that defines the settings for your selected compilation environment. For instance, if the Linux_IFORT compiler was selected, then search for the following line:

if ("$PLATf" == "Linux_IFORT") then

to locate the section of the script that defines the various settings for the Intel compiler and modify (or remove) the deprecated option. The most likely culprit would be the V_CPU shell variable settings (which house the compilation options). If installation problems continue, or if other unresolved installation issues arise, then contact the VULCAN-CFD development team for assistance.

The implicit time advancement schemes in VULCAN-CFD make extensive use of operator overloading. This feature of FORTRAN often results in inefficient executable code, which prompted the implementation of a templating strategy for module files that dominate the overall simulation time. These templated files (denoted by a .FT extension) must be recompiled to build the executables specific to a simulation class if executables for that particular class have not yet been built. The triggering of this recompilation (when necessary) is handled in an automated fashion by the VULCAN-CFD command line execution script. However, if a partial recompilation at run-time is problematic on your system, then any number of executable libraries can be created a priori after the initial installation of VULCAN-CFD has completed. There are 2 install_vulcan command line options for this purpose. The first is

install_vulcan cse

which creates all of the executables required to simulate the cases present in the **Sample_cases** directory. The other option is

install_vulcan cce <installation_input_definitions>

which creates custom executables based on a user-supplied installation_input_definitions file. The contents of this file take the following form to define each simulation class to build:

#	Gas-Model-Option	Equation-Option	Species	Reactions	Turb Eqs
#	0=CP, 1=TP, 2=TNE	0=Euler, 1=N-S, 2=RAS/LES	>= 0	>= 0	>= 0
1	0	1	0	0	0
2	0	2	0	0	1
3	0	2	0	0	2
4	1	1	1	0	0
5	1	2	1	0	1
6	1	2	1	0	2
7	1	2	7	7	2
8	2	2	5	5	2

where the `Gas-Model-Option` column specifies whether the flowfield is calorically perfect, thermally perfect, or thermal nonequilibrium, the `Equation-Option` column specifies if the equation set is Euler, pure Navier-Stokes, or Navier-Stokes with modeled turbulence,

the `Species` column specifies the number of species to be solved, the `Reactions` column specifies the number of chemical kinetic steps in the reaction database used, and the `Turb Eqs` column specifies the number of turbulence transport equations to be solved. Note that the use of this option to build predefined executables allows a path forward for the building of loadable modules for VULCAN-CFD.

NOTE: If your computational environment requires an installation with prebuilt executables, then upon creation of all desired executable libraries, edit the **vulcan_config** script file and disable support for the on-the-fly FORTRAN templating:

setenv VULCAN_FT2F4OO_support disable

This configuration setting is just below the standard user-specified settings section.

As a final note, several additional customizable installation features are available below the standard user-specified setting section that can optionally be adjusted. These include control over the floating point precision, bit order for unformatted I/O, manual specification of the METIS integer bit size, and compilation in "debug mode" for help with the detection of potential coding issues. This latter option greatly increases simulation times and should only be enabled when trouble-shooting potential coding issues. In general, any adjustments to these options should be avoided by novice users of VULCAN-CFD with the possible exception of the METIS integer bit size, which may require a manual setting if the automatic detection logic fails. The user will be informed of this failure if it occurs when sourcing the **vulcan_config** file with a message stating that VULCAN-CFD assumes the integer bit size to be 32-bit. If METIS (or ParMETIS) was installed for 64-bit integer usage, then edit the vulcan_config file and search for this output message. Once found, set the **METIS_INT_BIT** setting to 64 instead of 32.

2.2 Postinstallation Instructions

VULCAN-CFD relies on commercial tools for the grid generation and boundary condition specification process. In order to enable seamless interoperability with CFD solvers, these tools typically provide some mechanism to interface CFD software specific information to their grid generation packages. VULCAN-CFD provides support for the Pointwise[8] (structured and mixed element unstructured grids) and GridPro[9] (structured grids) grid generation packages. Support in this context refers to the inclusion of auxiliary scripts and/or plugins that allow the export of the grid coordinates, boundary conditions, and connectivity information from these grid generation packages in a format recognized by the VULCAN-CFD flow solver.

The Pointwise grid generation package uses the Pointwise Plugin SDK (Software Development Kit) to interface with external CFD packages. VULCAN-CFD supplies interface plugins for both structured (VULCAN-STR) and unstructured (VULCAN-UNS) grid generation for users that plan to use this package to generate computational grids. These plugins have been supplied to Pointwise for redistribution, but have yet to be included with the Pointwise download. However, precompiled versions (for 64-bit linux systems) have

been included in the VULCAN-CFD tarball. If Pointwise is to be used on a 64-bit linux system, then simply copy these precompiled plugin libraries to the plugins directory associated with your installation of Pointwise (this may require root access privileges). More specifically stated, start from the main VULCAN-CFD directory,

cd $vulcanpath

and copy the VULCAN-CFD plugin libraries, i.e.,

Utilities/Grid_codes/Pointwise/Plugins/linux_x86_64/plugins/libCaeStrVULCAN.so

Utilities/Grid_codes/Pointwise/Plugins/linux_x86_64/plugins/libCaeUnsVULCAN.so

to your Pointwise plugin directory:

<your_path_to_pointwise>/linux_x86_64/plugins

This action should make the VULCAN-CFD plugins accessible in the Pointwise CAE solver list. To test whether the plugins are recognized and compatible with your Pointwise installation, launch Pointwise and select the CAE option from the top menu bar of the Pointwise window. If the following selections appear in the dialog window:

NASA/VULCAN-STR
NASA/VULCAN-UNS

then the libraries were recognized and loaded. If the libraries do not appear in the dialog window, or if the Pointwise installation resides on something other than a 64-bit linux system, then a manual build of the VULCAN-CFD plugins will be required. The instructions on building the VULCAN-CFD plugin libraries for linux or MacOS are described below (refer to the Pointwise website for build instructions on a Windows OS):

(1) Download the Pointwise SDK from http://www.pointwise.com/plugins (the current VULCAN-CFD plugins support PluginSDK 1.0 R8 and later).

(2) Extract (e.g., unzip) the Pointwise SDK and enter the PluginSDK directory.

(3) If using PluginSDK 1.0 R8, R9, or R10, type:
./mkplugin -str -c VULCAN
./mkplugin -uns -c VULCAN

If using PluginSDK 1.0 R11 or later, type:
./mkplugin -caes -c VULCAN
./mkplugin -caeu -c VULCAN

These commands will create the plugin subdirectories:
src/plugins/CaeStrVULCAN
src/plugins/CaeUnsVULCAN
and update the pluginRegistry.h and modulelocal.mk SDK files

(4) Edit the file: src/plugins/pluginRegistry.h
and change the ID numbers associated with the VULCAN lines to read as:
#define ID_CaeStrVULCAN 12210
#define ID_CaeUnsVULCAN 12211
This step assigns the ID used by each plugin to a value that hopefully ensures the introduction of unused plugin ID numbers.

(5) Copy the contents from the CaeStrVULCAN and CaeUnsVULCAN directories from <your_path_to_vulcan>/Utilities/Grid_codes/Pointwise/Plugins into the corresponding plugin subdirectories created in step (3) above:

cp <your_path_to_vulcan>/Utilities/Grid_codes/Pointwise/Plugins/CaeStr*/* src/plugins/CaeStrVULCAN
cp <your_path_to_vulcan>/Utilities/Grid_codes/Pointwise/Plugins/CaeUns*/* src/plugins/CaeUnsVULCAN

(6) Build the plugins, i.e., from the PluginSDK directory, type:

make machine=M BUILD=Release

where: M=linux_x86_64 (64-bit linux architecture) or M=macosx (Mac with OS X)

(7) Finally, copy the plugin libraries to the plugins directory of your Pointwise installation (this step may require root access). If the plugins were built on a linux system:

cd dist/linux_x86_64/plugins
cp libCaeStrVULCAN.so <your_path_to_pointwise>/linux_x86_64/plugins
cp libCaeUnsVULCAN.so <your_path_to_pointwise>/linux_x86_64/plugins

If the plugins were built from a MacOS:

cd dist/macosx/plugins
cp libCaeStrVULCAN.so <your_path_to_pointwise>/macosx/plugins
cp libCaeUnsVULCAN.so <your_path_to_pointwise>/macosx/plugins

Upon completion of these steps, the VULCAN-CFD plugins for Pointwise should appear in the CAE CFD solver list the next time Pointwise is launched.

The GridPro grid generation package uses ASCII text files for interfacing their grid generation process to a specific CFD flow solver. If GridPro is to be used to generate structured grids for VULCAN-CFD, and if you want to specify the VULCAN-CFD boundary conditions from within the GridPro Graphical User Interface (GUI), then the files specific to VULCAN-CFD must be accessible to GridPro. These files have been provided to the GridPro developers, and as of GridPro version 6.5, they have been included with the GridPro distribution. The VULCAN-CFD specific files are **ptymap.vulcan**, **ws_ptymap.vulcan**, **outE_vulcan.script**, and **outM_vulcan.script**; and the following files should reference these VULCAN-CFD specific files: **ptymap.menu**, **ws_ptymap.menu**, **gridfmt.menu**, and **ws_gridfmt.menu**. To check if your installation of GridPro includes these files, go to the directory:

<your_path_to_gridpro>/az_mngr

and type the following command:

ls *vulcan*

If the VULCAN-CFD specific files noted above appear in the listing, then no further action is required. If these files are not present, then follow the instructions below to properly interface GridPro with VULCAN-CFD (root access may be required):

(1) Copy the VULCAN-CFD specific GridPro files to <your_path_to_gridpro>/az_mngr:
cp <your_path_to_vulcan>/Utilities/Grid_codes/Gridpro/ptymap.vulcan
<your_path_to_gridpro>/az_mngr
cp <your_path_to_vulcan>/Utilities/Grid_codes/Gridpro/ws_ptymap.vulcan
<your_path_to_gridpro>/az_mngr
cp <your_path_to_vulcan>/Utilities/Grid_codes/Gridpro/outE_vulcan.script
<your_path_to_gridpro>/az_mngr
cp <your_path_to_vulcan>/Utilities/Grid_codes/Gridpro/outM_vulcan.script
<your_path_to_gridpro>/az_mngr

(2) Edit the <your_path_to_gridpro>/az_mngr/ptymap.menu file and add the line:
VULCAN & pty=VULCAN & ptymap.vulcan

(3) Edit the <your_path_to_gridpro>/az_mngr/ws_ptymap.menu file and add the line:
VULCAN & pty=VULCAN & ws_ptymap.vulcan

(4) Edit the <your_path_to_gridpro>/az_mngr/gridfmt.menu file and add the line:
VULCAN & VULCAN & outE_vulcan.script & outM_vulcan.script

(5) Edit the <your_path_to_gridpro>/az_mngr/ws_gridfmt.menu file and add the line:
VULCAN & Elementary & outE_vulcan.script & Merged & outM_vulcan.script

Upon completion of these steps, VULCAN-CFD should appear in the CFD solver list the next time that the GridPro GUI is launched.

2.3 VULCAN-CFD Command Line Execution

VULCAN-CFD command line execution is controlled by the C-script **vulcan**, so if your default shell is not tcsh (or csh), then you must enter a C-shell by typing:

tcsh

prior to using VULCAN-CFD. Scripting the execution environment for VULCAN-CFD allows a single command line interface to control most (usually all) of the steps required by the simulation process. In a parallel simulation environment, this script also allows for the implementation of a uniform command line execution structure that is agnostic to the specific MPI library used to build the software (knowledge of the parallel execution details for your particular system are only required during the installation process). The specifics of the VULCAN-CFD command line structure depend on whether the software was installed for a parallel or a serial execution environment. The VULCAN-CFD command line for parallel execution is the following:

vulcan <num_cpus> <host_file> <options> <vulcan_input_file> [<vulcan_output_file>]

where:

num_cpus denotes the number of processors (cores) to be utilized
host_file denotes the host file containing the list of MPI hosts
options defines the VULCAN-CFD execution options:

p execute VULCAN-CFD preprocessor steps
s execute VULCAN-CFD flow solver
g execute VULCAN-CFD postprocessor
r execute recomposition utilities to map structured grid data back to the unpartitioned state

vulcan_input_file specifies the name of the VULCAN-CFD input file
vulcan_output_file specifies the name of the VULCAN-CFD screen output file [optional]

Multiple VULCAN-CFD execution options can be specified on the command line (e.g., -psgr, -sg, -sr, or -gr), and the MPI host file (**host_file**) does not have to exist if the parallel execution command line that you provided in the **vulcan_config** file does not require one (e.g., when using a batch scheduler or a local workstation). However, a dummy file name must still be included on the VULCAN-CFD command line. If a serial installation of VULCAN-CFD was performed, then the **num_cpus** and **host_file** specifications are omitted from the VULCAN-CFD command line, resulting in the following command line syntax for a serial execution environment:

vulcan <options> <vulcan_input_file> [<vulcan_output_file>]

An example PBS batch script that illustrates the use of this command line structure is provided in Appendix A.

Under normal circumstances, execution of VULCAN-CFD is halted when either the number of iterations specified in the input file has been reached, or if the supplied convergence criteria has been met. However, there are many instances where the user may want to end the simulation in an interactive manner. With the introduction of multiprocessor / parallel capabilities, stopping an active simulation became more tedious than simply issuing a <Ctrl> C or kill -9 #### command. MPI can spawn processes on multiple nodes, which must all be halted to completely stop an active execution. Moreover, there may be instances where the user desires to halt the execution of VULCAN-CFD while also dumping restart files so that the current state of the simulation can be recovered. To accommodate these scenarios, an active VULCAN-CFD simulation can be terminated by placing one of the following files in the working directory of the simulation:

STOP_VULCAN execution is halted
STOP_VULCAN_WR execution is halted and simulation restart files are written
STOP_VULCAN_PP execution is halted, simulation restart files are written, and postprocessing (if requested on the command line) is performed

If nothing is specified in these files, then the manner in which the simulation halts is dependent on the type of region currently being solved. If the region being solved is an elliptic region, then execution will terminate within 5-15 iterations of the current iteration (the iteration padding is an attempt to account for any network lag that may be present on network mounted disks). If the region being solved is a parabolic region, then execution will terminate as soon as the simulation finishes on the current marching plane. Additional control of the conditions under which the VULCAN-CFD simulation will be halted is provided by adding a single line of information to the above files:

#
The specification of only an integer value will terminate the execution at a specified iteration number (elliptic regions) or plane number (parabolic regions). This is the only entry that has any effect for regions that are space-marched. Note that any specified iteration (or plane) number provided must not have yet been reached during the execution of VULCAN-CFD, otherwise the simulation will continue as if no termination request was made.

REL RESIDUAL EXIT CRITERIA #.#
This line specifies the relative residual error L2-norm criteria for halting the simulation. The numerical value specified here is a relative L2-norm criteria in terms of orders of magnitude (e.g., a value of 6 implies that the execution will be halted after the L2-norm of the residual error has been reduced by at least 6 orders of magnitude relative to the value at the first iteration). The number can be entered as a positive or negative value (the sign is ignored). This entry is only intended for elliptic regions, and the value entered here will override the value specified in the VULCAN-CFD input file.

ABS RESIDUAL EXIT CRITERIA #.#
This line specifies the absolute residual error L2-norm criteria for halting the simulation. The numerical value specified here is the actual L2-norm criteria in terms of orders of magnitude (e.g., a value of 6 implies that the execution will be halted after the L2-norm of the residual error has dropped to a value of 10^{-6}). The number can be entered as a positive or negative value (the sign is ignored). This entry is only intended for elliptic regions, and the value entered here will override the value provided in the VULCAN-CFD input file.

MASS FLOW ERROR EXIT CRITERIA #.#
This line defines the relative mass flow error (difference in mass flow rates exiting and entering the computational domain divided by the flow rate entering the domain) used to halt the simulation. The value specified here should be entered as a percentage with valid values ranging between 0.0 and 100.0. Note that this criteria is a localized convergence metric, so it should not be used as an exit criteria until the simulation is close to a satisfactory level of convergence. This entry is only intended for elliptic regions.

NOTE: The specification of an iteration (elliptic regions) or plane (parabolic regions) number will always halt the simulation regardless of the simulation state. The behavior when specifying a convergence criteria (i.e., residual L2-norm or mass flow error) is dependent on the current simulation state:

- If the simulation is currently iterating on a coarse grid level, then any exit criteria specified will remain in effect until the finest grid level is reached. At that point, the **STOP_VULCAN_PP** (or **STOP_VULCAN_WR**) file will be removed and the simulation will proceed as dictated by the VULCAN-CFD input file.

- If the simulation is currently using a temporary 1^{st}-order scheme (as specified by the **1ST-ORD SWITCH** setting in the region control specification section of the input file), then any exit criteria specified will remain in effect until the scheme reverts back to the specified accuracy order. At that point, the **STOP_VULCAN_PP**

(or **STOP_VULCAN_WR**) file will be removed and the simulation will proceed as dictated by the VULCAN-CFD input file.

- If the simulation involves more than one computational region, then any exit criteria specified for the current region will cease to be active (i.e., the file will be removed) once the simulation for the current region is completed.

NOTE: The convergence criteria specifications are not applicable for the **STOP_VULCAN** file, since there is no reason to halt a simulation based on a convergence metric if restart files are not exported. Common situations where interactive simulation control may be desired are:

(1) Allowing the simulation to transition to the next grid level earlier (or later) than the number of specified coarse grid iterations based on the observed convergence rate.

(2) Allowing the simulation to transition to higher-order earlier (or later) than the specified number of **1ST-ORD SWITCH** iterations based on the observed convergence rate.

(3) Allowing an override of the convergence criteria specified in the VULCAN-CFD input file (either to allow for deeper convergence or to continue on to the next stage of the simulation if the L2-norm of the residual error has stalled).

(4) Allowing an exit criteria based on mass flow error to be introduced late in the convergence process for simulations that are hard to converge globally based on L2-norm assessments of the residual error.

As a final note, the user can also force a restart file dump without halting execution for elliptic regions by placing a file named **FORCE_VULCAN_WR** in the working directory of the VULCAN-CFD simulation. This feature allows the user to manually execute the postprocessor (given the current simulation state) without interrupting the solver execution. An iteration number can also be added to this file to control precisely when the restart files are output. Finally, any existing interactive solver control files (**STOP_VULCAN**, **STOP_VULCAN_WR**, **STOP_VULCAN_PP**, **FORCE_VULCAN_WR**) are removed when a new execution of VULCAN-CFD is initiated to avoid an inadvertent stop of the execution process.

3 VULCAN-CFD Analysis Process

At a high level, the VULCAN-CFD analysis process can be described by the following steps:

(1) Definition of the computational domain, typically based on a provided CAD (Computer Aided Design) representation of the geometry together with some knowledge of flowfield properties at the inflow, outflow, and/or farfield boundaries of this domain

(2) Generation of an appropriate grid (structured or unstructured) for the computational domain

(3) Specification of appropriate boundary conditions for the computational domain

(4) Creation of an input file for the flow solver that defines the physical and numerical models to be employed

(5) Load balancing of the computational domain for efficient parallel execution

(6) Execution of the flow solver and monitoring of convergence

(7) Flowfield postprocessing to obtain the desired quantities of interest (e.g., flow visualization, extraction of forces and moments, or extraction of performance metrics)

The proper definition of the computational domain is a critical step that can greatly affect the accuracy and/or robustness of the CFD analysis. Decisions on the placement of the inflow, outflow, and farfield boundaries should be made based on the well-posedness of the particular boundary condition being applied, and to the extent possible, these boundaries should be placed in regions with minimal flowfield gradients in the direction normal to the boundary. When generating the grid, one must also consider the resolution requirements for the chosen simulation strategy (Euler or Navier-Stokes, Reynolds-averaged or scale-resolving, etc.), in addition to the resolution of specific geometric features, and any anticipated flow physics that the simulation is intended to capture. The specification of boundary conditions is usually handled as part of the grid generation process, since knowledge of the specific boundary treatment is often an important consideration when generating a grid. For example, the grid spacing requirements for a no-slip surface with properly resolved sublayer and log layer regions are much more stringent than those required for simulations that model the sublayer and some portion of the log layer region with a wall function strategy. Another item to consider when setting boundary conditions is the force accounting for key components of the system. The lowest level of surface load extraction performed by VULCAN-CFD is the boundary condition level, so the grouping of boundary conditions for the components where isolated surface load output is desired should also be thought through when setting the boundary conditions. The creation of the input file for the VULCAN-CFD simulation is typically best carried out by editing an existing input file for a similar case, e.g., one of the sample cases included with the distribution. Utilities for load balancing the simulation for efficient parallel execution is included with the VULCAN-CFD distribution. Information to monitor simulation convergence (e.g., L2 norm of the equation set residual error, mass flow rate difference between inflow and outflow boundaries, and integrated surface loads) is

output in real-time to the screen and to column separated data files to allow a visual assessment of convergence. Postprocessing files that are output by VULCAN-CFD entail both volumetric flowfield data files and surface data files. In addition, utility codes are provided to aid with the extraction of auxiliary data derived from the CFD data (e.g., boundary layer properties and propulsion system performance metrics).

VULCAN-CFD relies on commercial tools for the grid generation and boundary condition specification process. In particular, VULCAN-CFD provides full support for Pointwise[8] (structured and mixed element unstructured grids) and GridPro[9] (structured grids). Full support in this context refers to the ability to output the grid coordinates, boundary conditions, and connectivity information in a format that is accessible to the VULCAN-CFD flow solver. VULCAN-CFD has adopted the NASA PLOT3D[10] format standard for structured grid files, while the AFLR3 format[11] is used for mixed element unstructured grids. Examples of how to use these grid generation packages to output the computational grid, boundary conditions, and connectivity data for both structured and unstructured grid simulations are provided in the chapters that follow.

VULCAN-CFD does not include software for visualization of simulation results, and instead relies on either commercial (e.g., Tecplot[7], FieldView[12], Ensight[13]) or open source (e.g., ParaView[14]) scientific visualization software packages. Several output file formats are available for the export of volumetric flowfield data that can be loaded into these (and other) visualization software packages. In particular, VULCAN-CFD can export volumetric data files in PLOT3D (structured grids only), VTK (unstructured grids only), CGNS, or binary Tecplot formats. A general purpose performance extraction tool geared toward air-breathing propulsion system applications is included with VULCAN-CFD to provide further postprocessing capabilities for engine flowpath simulations. A selection of Tecplot[7] utilities and MACRO files are also provided to further manipulate the volumetric and/or surface data for a variety of purposes.

4 Structured Grid Simulation Example

The first example problem chosen to describe the VULCAN-CFD structured grid simulation process is hypersonic flow over a cylinder. This case is a simple 2-D model problem for reentry where a blunt vehicle is using atmospheric drag to decelerate on entry into the atmosphere. The focus of this exercise is on the simulation process rather than physical modeling fidelity, so this example uses a calorically perfect gas model. This is one of the cases present in the **Sample_cases** directory:

<your_path_to_vulcan>/Sample_cases/Input_files/cyl_hyp.dat

and the PLOT2D format grid file developed for this simulation can be found in the directory:

<your_path_to_vulcan>/Sample_cases/Grids/cyl_hyp.grd

Even though an input file has already been developed for this case, it is informative to walk through the process of developing an input file once the grid has been generated. If Pointwise is available on your system, launch Pointwise and import the cyl_hyp.grd file (if Pointwise is not available to you, then skip the rest of this paragraph and proceed to the next one that describes the VULCAN-CFD input requirements for this case). To load the grid file into Pointwise, select **Import** from the **File** tab, and then select **Grid**. From there, navigate to the **Sample_cases/Grids** directory and select the **cyl_hyp.grd** file. This particular grid file does not have a standard PLOT3D format extension, so select PLOT3D from the list provided in the **Files of type** box and then open the file. Pointwise should have detected that this is an ASCII, 2-D, multiblock file without I-Blanking, and if so, select **OK**. At this point the 2-D grid file should be loaded into Pointwise. The next step is to set the problem dimension (2-D) and the solver (NASA/VULCAN-STR). Both of these settings are housed under the **CAE** tab of Pointwise. The boundary conditions required for the simulation can now be specified by selecting **Set Boundary Conditions** from the **CAE** tab:

- Select the inflow boundary and provide it with the name FARFIELD and choose INFLOW for the CAE Type
- Select the 2 outflow boundaries and provide them with the name OUTFLOW and choose OUTFLOW for the CAE Type
- Select the cylinder surface and provide it with the name WALL and choose WALL for the CAE Type
- Close the **Set BC** window

Note that the boundary names were selected to match what has already been provided in the existing input file for this case, but in general, these names can be any uniquely defined string containing up to 12 characters. Note also that the list of CAE types available in the Pointwise plugin represent generic classes of boundary conditions. The specific boundary conditions to be imposed are selected later when finalizing the VULCAN-CFD input file. VULCAN-CFD supports many different boundary conditions, so limiting the selection of CAE types to generic classes allows the Pointwise plugin to remain unchanged as new boundary conditions are added to the flow solver. Moreover, many of the boundary conditions available in VULCAN-CFD require auxiliary data (e.g., wall temperature or inflow

properties), which are details that are often more easily supplied during the setup of the input file for the simulation rather than during the grid generation process. The CAE file is now ready to be exported. Select the blocks (domains) to be output and then click **File → Export → CAE**. The CAE file will have a .bcs extension (by default), and is essentially a partial VULCAN-CFD input file that serves as a starting point for the input file creation process.

The partial VULCAN-CFD input file exported by Pointwise is given below:

```
$---------------- Boundary and connectivity control -----------------$
BLOCKS               1.0  (no. of blocks)
BCGROUPS             3.0  (no. of boundary condition groups)
BCOBJECTS            0.0  (no. of boundary condition objects)
FLOWBCS              4.0  (no. of boundary conditions)
CUTBCS               0.0  (no. of C(0) connectivity conditions)
PATCHBCS             0.0  (no. of non-C(0) connectivity conditions)
$-------------------- Sample Block Configuration --------------------$
BLOCK CONFIG         2.0   (no. of block configuration lines)
BLK  VISC (N, T, or F)  TURB  REAC  REGION
  1    F                  Y           1
  2    F                  Y           2
$---------------------- Sample Elliptic Region ----------------------$
SOLVER/STATUS
     E/A
KAPPA  LIMITER  LIM COEF  FLUX SCHEME  ENT FIX (U)  ENT FIX (U+a)
  3       2        0.0       LDFSS         0.0          0.0
FMG-LVLS   NITSC  NITSF   1ST-ORD SWITCH   REL RES   ABS RES   DQ COEF
    2       100    900          -1          -6.0      -8.0       0.25
TURB CONV   DT RATIO   NON EQUIL   POINT IMP  COMP MODEL  CG WALL BC
   2ND         1.0        20.0         Y           N         WMF
SCHEME  TIME-STEP  STATS  MIN-CFL VAR-CFL # CFL VIS-DT IMP-BC REG-RES
  ILU     LOCAL      10     1.0      Y      4      N      Y      N
SWEEP-DIR  JAC-UPDATE  START  NUM-SLVS  RELAX
    0           0        1        5      0.5
   1    100   101   200
  0.5  5.0e2  1.0  1.0e2
$---------------------- Sample Parabolic Region ----------------------$
SOLVER/STATUS
     M/A
KAPPA  LIMITER  LIM COEF  FLUX SCHEME  ENT FIX (U)  ENT FIX (U+a)
  3       2        0.0       LDFSS         0.0          0.0
SM-ORD  BEG:END  FMG BEG:END  VIG COEF  MEAN-LIM  TURB-LIM  SUB-STEP
  2     001 999      001 005    0.95       1.0       1.0        M
FMG-LVLS   NITSF   1ST-ORD SWITCH   REL RES   ABS RES   DQ COEF
    1       500           0          -6.0      -8.0       0.5
TURB CONV   DT RATIO   NON EQUIL   POINT IMP  COMP MODEL  CG WALL BC
   2ND         1.0        20.0         Y           N         WMF
```

```
SCHEME  TIME-STEP  STATS  MIN-CFL VAR-CFL # CFL VIS-DT IMP-BC REG-RES
  DAF     LOCAL      10     1.0      Y      2      Y      N      N
   1       5
   1       1
  0.1     1.0
  5.0     5.0
!************** End of VULCAN-CFD solver control data ***************!

BCGROUPS:   NAME       TYPE       OPTION
            WALL       WALL       PHYSICAL
            OUTFLOW    OUTFLOW    PHYSICAL
            FARFIELD   INFLOW     PHYSICAL

BC NAME    BLK  FACE  PLACE  IND1  BEG   END  IND2  BEG   END  IN-ORD
WALL         1     J    MIN     K   MIN   MAX     I   MIN   MAX     0
OUTFLOW      1     I    MAX     J   MIN   MAX     K   MIN   MAX     0
FARFIELD     1     J    MAX     K   MIN   MAX     I   MIN   MAX     0
OUTFLOW      1     I    MIN     J   MIN   MAX     K   MIN   MAX     0
```

This file contains the boundary condition lines, block-to-block connectivity lines (if applicable), a skeletal boundary condition groups section, skeletal region configuration sections for both elliptic and space-marching regions, and a skeletal block configuration section. The following steps are required to complete the creation of the VULCAN-CFD input file:

- Define the problem dimension, grid file name and format, and restart file information (see Chapter 8 for further details).
- Define the desired format (PLOT3D, CGNS, or TECPLOT) for the volumetric plot files and the desired flow variables to be output (see Chapter 8).
- Define the equation set to be solved (Navier-Stokes or Euler), the thermodynamic and transport models and, if applicable, the turbulence model and/or chemical kinetic model (see Chapter 8).
- Define the reference (e.g., freestream) conditions which are to be used for nondimensionalization purposes along with the specification of Prandtl number(s) for heat diffusion (see Chapter 8). If the simulation involves multiple chemical constituents, then the Schmidt number(s) should also be provided to fully define the species mass diffusion processes.
- Modify the block configuration line(s) as appropriate for the simulation (see Chapter 8).
- Modify the region configuration section(s) as appropriate for the simulation (see the region control specification section of Chapter 8 for further details). For this case, the space-marching region section should be removed.
- Replace the generic boundary condition class in the **TYPE** column with a specific boundary condition for each boundary condition group name present in the boundary

condition groups section. The available boundary conditions (and descriptions of any required auxiliary data) are provided in Chapter 9.

The complete input file provided for this example case is given below:

```
$************ Beginning of VULCAN-CFD solver control data ************$
$-------------------- Dimensionality of problem --------------------$
TWOD                        (TWOD=2D, AXISYM=Axisymmetric, THREED=3D)
$----------------------- Input control data -----------------------$
GRID                 0.0    (0=plot3d->3d ; plot2d->2d/axi, 1=plot3d->all)
  ../Grids/cyl_hyp.grd
GRID FORMAT          3.0    (1=sb fmt, 2=sb unf, 3=mb fmt, 4=mb unf)
RESTART IN                  (input restart file name to follow)
  cyl_hyp.restart
RESTART OUT          0.0    (output restart file name to follow)
  cyl_hyp.restart
$----------------------- Output control data ----------------------$
PLOT3D OUTPUT        2.0    (0=off, 1=fmt, 2=unf/seq, -2=unf/sio)
32 BIT BINARY               (write plot files as 32 or 64 bit binary)
PLOT FUNCTION        5.0    (no. of plot function names)
  DENSITY
  VELOCITY
  MACH NO.
  PRESSURE
  TEMPERATURE
$------------------------ Equation set data -----------------------$
GLOBAL VISCOUS              (solve N-S equations using global algorithm)
DDT ROE INVISCID JACOBIAN  0.3  (Roe Jacobian via operator overloading)
$-------------------- Gas and thermodynamic data --------------------$
GAS/THERMO MODEL     0.0    (0=calorically perfect, 1=thermally perfect)
$----------------------- Transport model data ---------------------$
VISCOSITY MODEL      1.0    (0=Power, 1=Sutherland, 2=Suth/lin, 4=McBride)
CONDUCTIVITY MODEL   0.0    (0=Prandtl no.)
$-------------------- Reference condition data ---------------------$
ANGLE REF. FRAME     0.0    (0=AOA in xy plane, 1=AOA in xz plane)
ALPHA                0.0    (angle of attack measured C.C.W in degrees)
NONDIM               1.0    (1=static conditions, 2=total conditions)
GAMMA                1.4
MACH NO.             17.605
GAS CONSTANT         287.04667
STATIC TEMPERATURE   200.00000
STATIC PRESSURE      57.598000
UNIT REYNOLDS NO.    376930.00
SUTHERLANDS LAW S0   110.55556
PRANDTL NO.          0.72
$--------------------- Boundary and cut control --------------------$
```

```
BLOCKS              1.0  (no. of blocks in input grid)
BCGROUPS            3.0  (no. of boundary condition groups)
BCOBJECTS           1.0  (no. of boundary condition objects)
FLOWBCS             4.0  (no. of boundary conditions)
CUTBCS              0.0  (no. of cut connectivity conditions)
BLOCK CONFIG        1.0  (no. of block configurations lines)
BLK  VISC (N, T, or F)  TURB  REAC  REGION
 1     F                              1
$---------------- Solution Methodology for Region 1 ----------------$
SOLVER/STATUS
     E/A
KAPPA  LIMITER  LIM COEF  FLUX SCHEME  ENT FIX (U)  ENT FIX (U+a)
  3       3        0.0       LDFSS         1.0          0.0
FMG-LVLS   NITSC  NITSF   1ST-ORD SWITCH   REL RES   ABS RES   DQ COEF
   1              1000          500          -8.0     -12.0      0.5
TURB CONV   DT RATIO   NON EQUIL   POINT IMP  COMP MODEL  CG WALL BC
   2ND         1.0        10.0         Y           N         STW
SCHEME  TIME-STEP  STATS  MIN-CFL VAR-CFL # CFL VIS-DT IMP-BC REG-RES
  ILU     LOCAL      10     1.0      Y      6      N      Y      N
SWEEP-DIR  JAC-UPDATE  START  NUM-SLVS  RELAX
    0           0        1        3      0.9
   1    100   250  500   501   600
  0.1  10.0  50.0  50.0  1.0  150.0
!************** End of VULCAN-CFD solver control data **************!

BCGROUPS:   NAME       TYPE       OPTION
            FARFIELD   REF_IN     PHYSICAL
            OUTFLOW    EXTRAP     PHYSICAL
            WALL       IWALL      PHYSICAL
  Wall Temp  Rlx
    500.0    1.0

BCOBJECTS:  NAME                   NO_OF_MEMBERS
            IN_AND_OUT             2
            FARFIELD   OUTFLOW

BC  NAME  BLK  FACE  PLACE  IND1  BEG   END  IND2  BEG   END  IN-ORD
OUTFLOW     1    I    MIN     J   MIN   MAX    K   MIN   MAX    0
OUTFLOW     1    I    MAX     J   MIN   MAX    K   MIN   MAX    0
WALL        1    J    MIN     I   MIN   MAX    K   MIN   MAX    0
FARFIELD    1    J    MAX     I   MIN   MAX    K   MIN   MAX    0
```

To execute VULCAN-CFD for this example problem, make sure that you are in a C-shell window and navigate to the **Sample_cases/Input_files** directory. If VULCAN-CFD was installed as a parallel application, then type the following command:

vulcan 1 host_file -psg cyl_hyp.dat

If instead VULCAN-CFD was installed as a serial application, type:

vulcan -psg cyl_hyp.dat

The simulation should take less than a minute to complete and exhibit the convergence history shown in the left image of Fig. 1. The convergence history is written to the **vulcan.ifam_his_1.tec** file, which is a native Tecplot ASCII format data file. This file contains the Integrated Force And Moment (IFAM) history for each cycle (iteration) performed during the simulation, and the number appended to the root name of this file corresponds to the region number. The specific quantities written at each cycle are the Courant-Friedrichs-Lewy (CFL) number, L2-norm of the residual error (sum of mass, momentum, and energy equations), mass flow conservation error, Cartesian components of the total force and moment vectors acting on all surfaces, total heat load through all no-slip surfaces, and the Cartesian components of the viscous force vector acting on all no-slip surfaces. Integral convergence behavior specific to each inflow, outflow, and surface boundary are also exported to files in the **BC_files** directory. The integrated surface loads are exported at the surface boundaries, while integrated mass flux, momentum component fluxes, and total enthalpy flux are exported at each inflow and outflow boundary. The volumetric plot data files (PLOT3D format) and the discrete surface loads file (**vulcan.loads.tec**) are located in the **Plot_files** directory. Images obtained by postprocessing these files are shown in Fig. 1 that illustrate the bow shock that forms as the flow navigates over the cylindrical body.

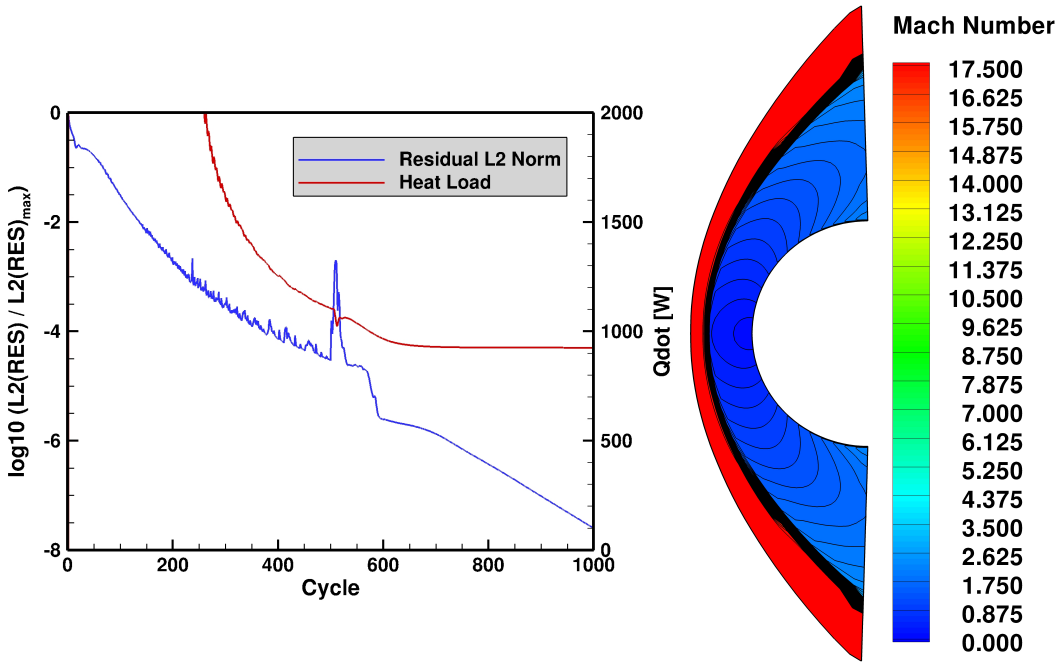


Figure 1: Convergence history and Mach number contours for the reentry model problem.

For users interested in hypersonic applications, this case is a good choice to experiment with the high enthalpy gas models available in the software. Modifications to the input file are required for the thermodynamic model, transport model, and reference condition data sections to perform either a thermal equilibrium or thermal nonequilibrium simulation. Chapter 8 provides the details on the modifications that are required, and the user is also encouraged to examine other example cases that have been set up for either thermal equilibrium (e.g., temp_mix_layer_tvd.dat) or thermal nonequilibrium (e.g., nozzle_axi_tne.dat) for further guidance.

5 Structured Grid Simulation Example with Partitioning

This chapter walks through the VULCAN-CFD structured grid simulation process when grid partitioning is desired to obtain adequate load balancing for parallel processing. The case chosen to illustrate this process is a 2-D problem for computational efficiency, but the process is identical when applied to a 3-D case. The model problem chosen for this activity is a simplified inlet geometry for a hypersonic propulsion system that contains a perforated plate bleed system to mitigate the flow separation from shock / boundary layer interactions. The generation of grids for perforated plate bleed systems can be a tedious process, so empirical bleed/effusion models have been developed to model these devices. A description of the bleed models available in VULCAN-CFD (and the input options that invoke them) are provided in Chapter 9. This example problem also invokes the topology analysis option of VULCAN-CFD to properly handle the topological singularities that are present in the structured grid created for this geometry. The VULCAN-CFD input file for this example (bleed_shock_sing.dat) is one of the cases present in the **Sample_cases** directory.

Partitioning structured grids can be a more challenging task than unstructured grids due to the need to retain structured grid blocks during the partitioning process. To maximize computational efficiency, the partitioning process should result in an adequate level of load balancing, while minimizing the number of blocks the grid is split into. This has proven to be difficult to automate, so the partitioning of structured grids must currently be performed as a stand-alone preprocessing step. However, the input required for the partitioning process can be generated for you by simply entering the command line execution syntax for the VULCAN-CFD preprocessor with the number of processors (cores) that are desired. To illustrate the process, navigate to the **Sample_cases/Input_files** directory, remove any existing **grid_split.inp** file (if present), and then type:

vulcan 6 host_file -p bleed_shock_sing.dat

to create an input file (**grid_split.inp**) to partition the structured grid for 6 processors. The output that is returned is the following:

```
Evaluating the input file for auto-partitioning support

Auto-partitioning is not possible for this simulation!
Checking for evidence of manual partitioning/splitting

Found a PROCESSORS line in the VULCAN input file
that is inconsistent with the command line request!
A manual partition is required using the grid_split utility!

Creating an example grid_split.inp file to assist you!

Grid partitioning input file documentation can be found in:
<your_path_to_vulcan>/Doc_manual/manual_user.pdf
```

The **grid_split.inp** file generated for you has the following contents:

```
START GLOBAL
  SPLIT_OPT          1  # Structured grid splitting algorithm (0 - 2)
  NUM_PROCS          6  # Target number of processors
  LOAD_PERC         95  # Structured grid load balance efficiency (%)
  BLK_MIN_DIM        8  # Minimum structured block dimension
  NUM_LEVELS         2  # Number of grid levels
  OLD_VULCAN_INPUT  bleed_shock_sing.dat
  NEW_VULCAN_INPUT  bleed_shock_sing_6_procs.dat
  NEW_GRID          bleed_shock_sing_6_procs.grd
END
#
# STRUCTURED GRID BLOCK SPLITTING CONSTRAINTS
#
START BLOCK
#BLOCK   KEY WORD   INDEX  BEG   END
#  0     NO_SPLIT     I     1     0
#  0     NO_SPLIT     J     1     0
#  0     NO_SPLIT     K     1     0
END
#
# STRUCTURED GRID BOUNDARY CONDITION SPLITTING CONSTRAINTS
#
START BC
#BC NAME    KEY WORD
#SINGULAR   NOT_ALLOWED
#NOSLPSNG   NOT_ALLOWED
END
#
EOF
```

This input file directs all aspects of the grid partitioning process, which involves creating a new partitioned grid file and a new VULCAN-CFD input file that is consistent with the partitioned grid. The **GLOBAL** section of the input file contains all of the main input parameters that govern how the grid is to be partitioned. The remaining sections (**BLOCK** and **BC**) provide for control of the partitioning process at the block and boundary condition level, respectively. A complete description of the options available to control the partitioning process is provided in the structured grid partitioning section of Chapter 25. However, for the purposes of this exercise, no modifications are required to the default input specification.

The command line syntax for executing the partitioner depends on whether ParMETIS was enabled in the **vulcan_config** file during the installation process. If ParMETIS was enabled, then the following command line should be used to execute the **grid_split** utility:

grid_split <num_cpus> <host_file>

where:

num_cpus denotes the number of processors (cores) to be used by ParMETIS
host_file denotes the host file containing the list of MPI hosts

As was the case for parallel execution of VULCAN-CFD, the MPI host file (**host_file**) does not have to be present if the parallel execution command line that you provided in the **vulcan_config** file does not require one (e.g., when using a batch scheduler or a local workstation). However, a dummy file name must still be included on the **grid_split** command line. If ParMETIS was not enabled in the **vulcan_config** file, then the proper command line syntax for the **grid_split** utility is simply:

grid_split

Upon successful execution of the partitioner, a message stating that the load balance tolerance was met should be displayed, along with the following output describing how the grid has been partitioned:

```
Parent Block   Child Blocks
---------------------------
      1              1
      2              4
      3              1
      4              4
      5              1
      6              1
      7              1
      8              1
      9              2
     10              1
     11              1
     12              1
     13              1
     14              1
     15              1
     16              2
---------------------------
# of blocks  =      24
# of bcs     =      20
# of cuts    =     142
# of patches =       0
---------------------------
```

Even though the output indicates that the partitioning process met the desired load balance tolerance, it is prudent to check the load_bal_his.txt file for any alternative partitions that

achieved a similar load balance efficiency with fewer "Child" blocks. For this case, there was no obvious improvement to be had by reducing the load balance tolerance. If there was, then the **LOAD_PERC** line in the **grid_split.inp** file should be changed to the value that was deemed acceptable, and the partitioner executed again with this setting. A description of the files created by the partitioning utility that are used in the VULCAN-CFD simulation process are given below:

bleed_shock_sing_6_procs.dat	partitioned VULCAN-CFD input file
bleed_shock_sing_6_procs.grd	partitioned structured grid file
MERGE_MAP.DAT	file housing the mapping of child blocks to parent blocks
MPI_MAP.TMP	file housing the mapping of child blocks to processors

The file names of the partitioned VULCAN-CFD input file and grid file are the user-specified names provided in the **grid_split.inp** file. The **MERGE_MAP.DAT** file is utilized if the "-r" option is included in the VULCAN-CFD command line to recompose the volumetric plot files back to the unpartitioned state (for easier postprocessing). The **MPI_MAP.TMP** contains the mapping of child blocks to processors based solely on the number of cells assigned to each processor. This file can be manually modified to swap child blocks of similar size across partitions with the goal of reducing the communication overhead, but this is rarely practical for large real-world simulations. If this file is not present in the working directory during VULCAN-CFD execution, the preprocessor will simply map the blocks to the processors using the same algorithm (that ignores inter-processor communication) utilized by the partitioner.

After the successful execution of the partitioner, VULCAN-CFD can efficiently be executed on 6 computational cores, i.e.,

```
vulcan 6 host_file -psgr bleed_shock_sing_6_procs.dat
```

The simulation should take less than 10 minutes to complete and exhibit the convergence history shown in Fig. 2. The use of a limiter has caused the residual error to stall after dropping roughly 4.5 orders of magnitude. Disabling the limiter and utilizing a fully upwind scheme (**KAPPA** value of 2 instead of 3) will allow the residual error to continue to decrease. However, as indicated by the temporal history of the surface shear force and the temporal history of the bleed specification, the simulation is adequately converged. The inclusion of the "-r" option forced the creation of volumetric plot files that have been mapped back to their original unpartitioned state (based on the mapping provided in the **MERGE_MAP.DAT** file). These recomposed files are located in the **Recomp_files/Plot_files** directory. At this time, the surface loads file is not mapped back to its unpartitioned state, but the partitioned version of this file is located in the **Plot_files** directory along with the partitioned volumetric plot files. If desired, an unpartitioned surface loads file can be created by a 2-step process. The first step involves recomposing the restart files using the **restart_merge** utility (refer to the structured grid partitioning section of Chapter 25 for instructions on using this utility). Once the restart files have been recomposed, the second step involves execution of the postprocessor using the original unpartitioned VULCAN-CFD input file without any volumetric plot file input lines present,

since these plot files have already been recomposed.

Images that illustrate the effect that boundary layer bleed has on the shock / boundary layer interaction region is shown in Fig. 3. Mach number contours are shown in the top image to illustrate both the shock and expansion wave structure as well as the boundary layer response to the impinging shock, while the ratio of the eddy viscosity to molecular viscosity contours are shown in the lower image to illustrate the effect of the shock interaction on the turbulent nature of the boundary layer. The bleed region extends from $x = 0.0$ to $x = 3.75$ inches, and both contour images show a thinning of the boundary layer as the low momentum fluid near the wall is bled off leaving a more robust boundary layer resistant to flow separation. In fact, this bleed system completely removed the boundary layer separation that would have otherwise resulted from this shock boundary layer interaction.

For users interested in bleed modeling for boundary layer control, this case is a good choice to experiment with the various bleed models that are available. The current simulation was set up to use the Doerffer-Bohning bleed model, but minimal modifications are required to swap out the bleed model (see Chapter 9 for the modifications required). The bleed model can also be deactivated by simply replacing the bleed specification with a standard no-slip adiabatic wall setting.

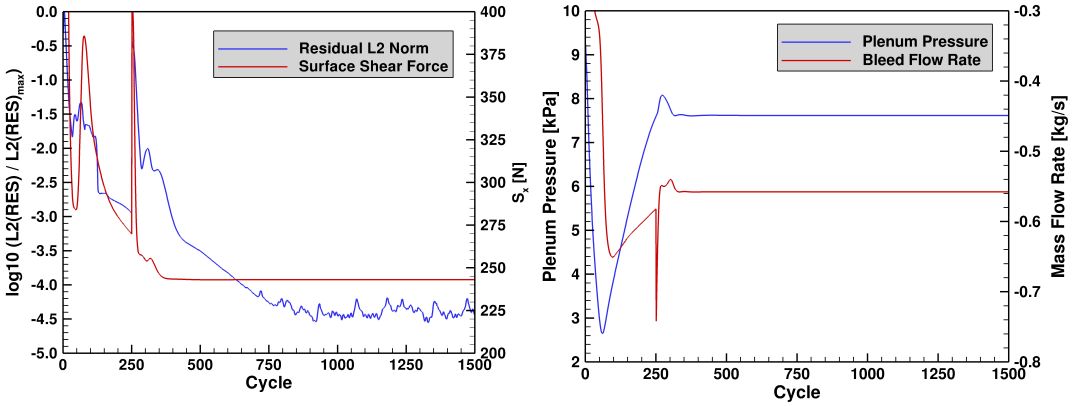


Figure 2: Convergence history for the inlet with bleed model problem.

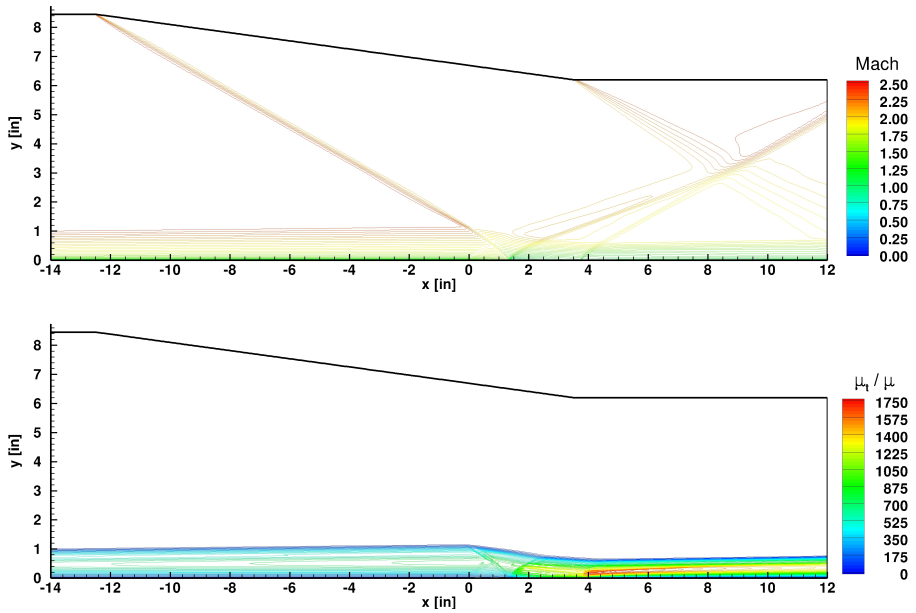


Figure 3: Mach number (top) and turbulent viscosity (bottom) contours for the inlet with bleed model problem.

6 Structured Grid Multiregion Simulation Example

This chapter walks through the VULCAN-CFD simulation process for predominantly supersonic flow applications that permit the decomposition of the computational domain into separate regions that can be solved sequentially. This simulation strategy can greatly reduce the time required to obtain solutions, particularly when computational resources are limited. The decomposition of the domain into regions is made possible by the fact that a supersonic flowfield is not affected by downstream flow conditions. Similarly, the boundary layers (which may contain subsonic flow very close to the surface) are well approximated by ignoring downstream conditions except when strong adverse streamwise pressure gradients are present. A relevant scenario that could benefit from a multiregion analysis approach is the forebody / inlet system of a scramjet engine as illustrated in Fig. 4. The scenario shown here can be decomposed into at least 3 regions:

(1) A region encompassing the freestream and a small portion of the forebody (which may contain blunt leading edges)

(2) A region that includes most of the remaining forebody length

(3) A region that includes the inlet

The first region would require a full elliptic simulation algorithm due to the presence of a bow shock that forms as the flow maneuvers around the blunt leading edge of the forebody. The second region could be simulated using the parabolized Navier-Stokes equations (i.e., space-marching) provided that the flow remains supersonic outside of the boundary layer, and any shock / boundary layer interactions that may be present are weak enough to avoid flow separation. The third region would require an elliptic simulation strategy to allow for the presence of a bow shock at the cowl leading edge, and the possibility of boundary layer separation inside of the inlet from shock waves formed during the inlet compression process.

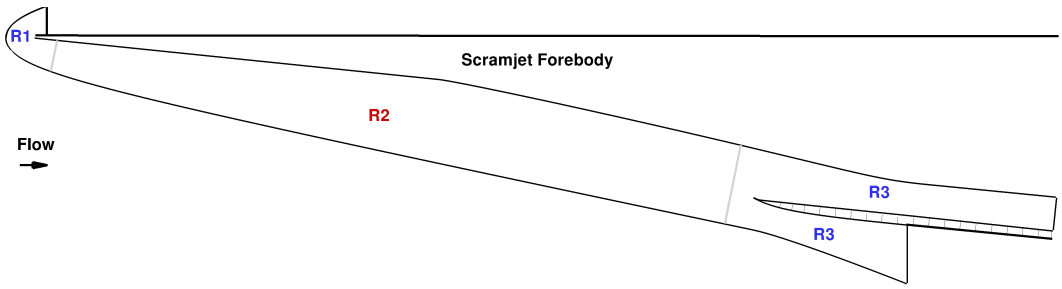


Figure 4: Sample region configuration for the forebody / inlet portion of a scramjet engine (R → elliptic region, R → parabolic region).

Another common scenario that could benefit from a multiregion analysis is the simulation of a direct-connect facility flowpath for scramjet combustor testing (see Fig. 5). This scenario can be broken up into at least 3 regions as well:

(1) An elliptic region encompassing the facility nozzle plenum to some streamwise station downstream of the nozzle throat

(2) A parabolic region that includes the remainder of the facility nozzle and some portion of the isolator section (upstream of the expected combustion-induced shock train)

(3) An elliptic region to cover the remaining portion of the isolator and the combustor, since both are expected to contain regions of flow separation

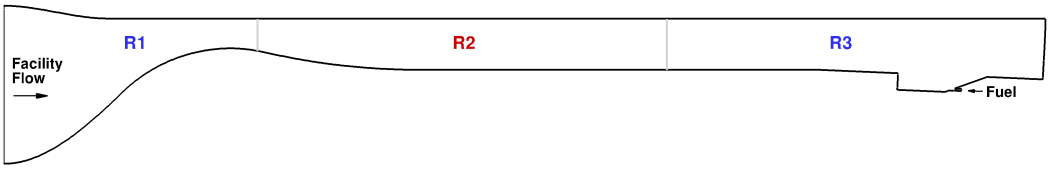


Figure 5: Sample region configuration for a scramjet combustor direct-connect rig flowpath (R → elliptic region, R → parabolic region).

A simple example that utilizes the multiregion capabilities of VULCAN-CFD is the 2-D reacting mixing layer case (splitter_fr.dat) located in the **Sample_cases/Input_files** directory. The geometry consists of a splitter plate that initially separates a fuel stream from an oxidizer stream, with mixing and combustion taking place downstream of the plate. This example also introduces the user to the reacting flow capabilities of VULCAN-CFD. This case has been decomposed into three space marching regions, so it executes relatively rapidly without partitioning the problem across multiple processors. However, it is beneficial to walk through the partitioning process for this multiregion scenario. When choosing the number of processors to use for a space-marching simulation, one must keep in mind that the grid can only be partitioned across streamwise planes, since the space-marching planes must be solved sequentially (by design). Given that this problem is comprised of only 2-D space-marching regions, each space-marching plane consists of only a single column of grid cells. As a result, the number of processors must be kept low to ensure that the communication overhead does not overwhelm the computational efficiency gained by partitioning. For this case, utilizing more than 2 processors will likely not provide any appreciable speedup. As was discussed in Chapter 5, the easiest way to proceed with the partitioning process is to first remove any **grid_split.inp** files that may be present, and then execute the VULCAN-CFD preprocessor with the splitter_fr.dat file and the desired number of processors provided on the command line, i.e.,

```
vulcan 2 host_file -p splitter_fr.dat
```

The execution of this command will fail by design, but an input file (**grid_split.inp**) for the partitioner will be created for you. The output that is returned is the following:

```
Evaluating the input file for auto-partitioning support

Multiple instances of the key word FMG were found
This implies that more than one region is present

Auto-partitioning is not possible for this simulation!
```

```
Checking for evidence of manual partitioning/splitting

Unable to find a PROCESSORS line in the VULCAN input file!
A manual partition is required using the grid_split utility!

Creating an example grid_split.inp file to assist you!

Grid partitioning input file documentation can be found in:
<your_path_to_vulcan>/Doc_manual/manual_user.pdf
```

The **grid_split.inp** file that was generated can be directly utilized for this particular multiregion problem because each region can efficiently be partitioned for the same number of processors. However, more complicated multiregion simulations may require that each region be load balanced for a different number of processors (this is often necessary for multiregion simulations that contain both elliptic and parabolic regions). Control of the partitioning process on a region-by-region basis is accommodated by the **REGION** section of the **grid_split.inp** file (see the structured grid partitioning section of Chapter 25 for further details). To execute the partitioner type:

grid_split <num_cpus> <host_file>

if ParMETIS was enabled during the VULCAN-CFD installation process, or simply type:

grid_split

if ParMETIS was not enabled. Upon successful execution of the partitioner, the following output should be displayed describing how the grid has been partitioned:

```
Parent Block   Child Blocks
----------------------------
      1              2
      2              2
      3              2
----------------------------
# of blocks  =       6
# of bcs     =      16
# of cuts    =      14
# of patches =       0
----------------------------
```

The load balance efficiency achieved for each region is written to the load_bal_his_#.txt files (where "#" represents the region number), and an examination of these files indicates that each region has been optimally load balanced (i.e., 100% efficiency). VULCAN-CFD can now efficiently simulate each of the 3 regions using 2 computational cores, i.e.,

vulcan 2 host_file -psgr splitter_fr_2_procs.dat

The simulation should take about 5 minutes to complete and produce the convergence histories shown in Fig. 6 for each region. The **vulcan.ifam_his_#.tec** files, where "#" is the region number, are not created for parabolic regions (nor are any boundary specific history files), so the contents of the **vulcan.res.tec** file must be postprocessed to observe the convergence history of each region. The convergence history for each plane is output to this file in separate Tecplot zones for each computational region. A close examination of these images shows that each marching plane met the desired convergence criteria (6 orders of magnitude drop in the L2-norm of the residual error).

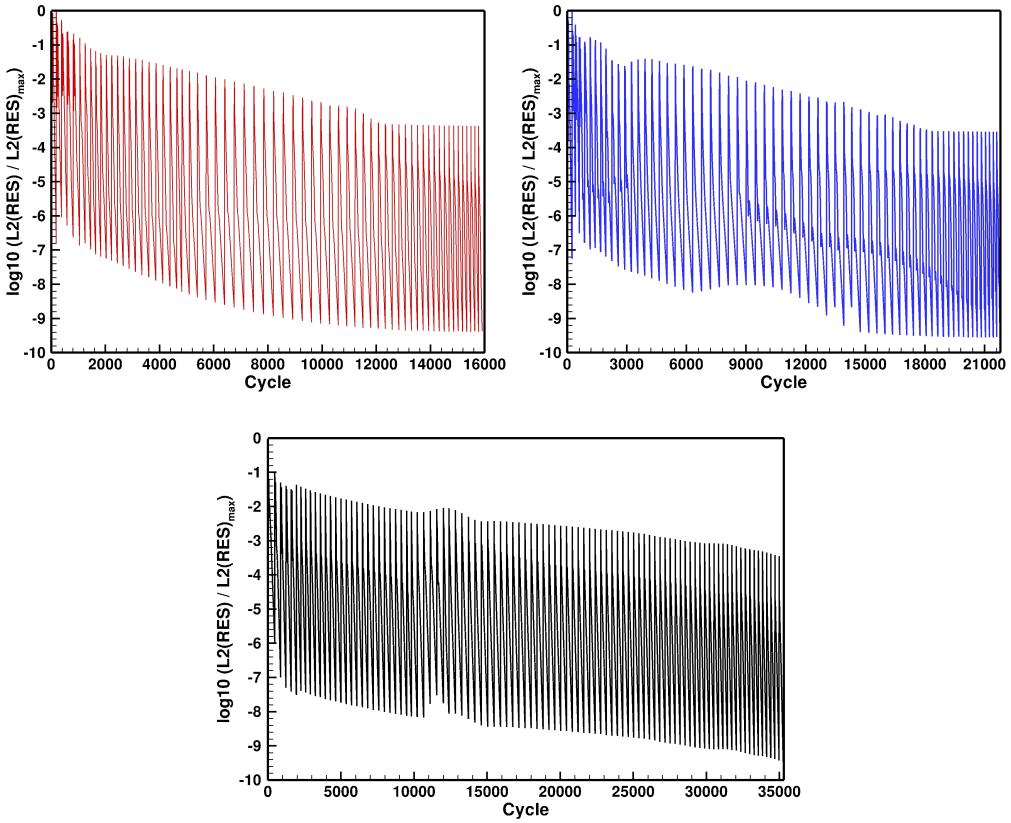


Figure 6: Convergence history for each of the 3 computational regions: Region 1 (top left), Region 2 (top right), Region 3 (bottom).

A visualization of the mixing and combustion processes that occur downstream of the plate are shown in Fig. 7 in the form of temperature and steam mass fraction contours. The user is encouraged to modify the input settings for this case to examine the computational savings (both time and resource) offered by a multiregion simulation strategy, as well as the savings offered by using a parabolic solution procedure in lieu of a full elliptic approach. For example, this problem could be set up as a single elliptic region with the computational domain partitioned more aggressively to match the computational time required by the pure space-marching approach. However, this would require the use of more processors, which

for large problems, may strain the available resources for users that have a limited computational capability.

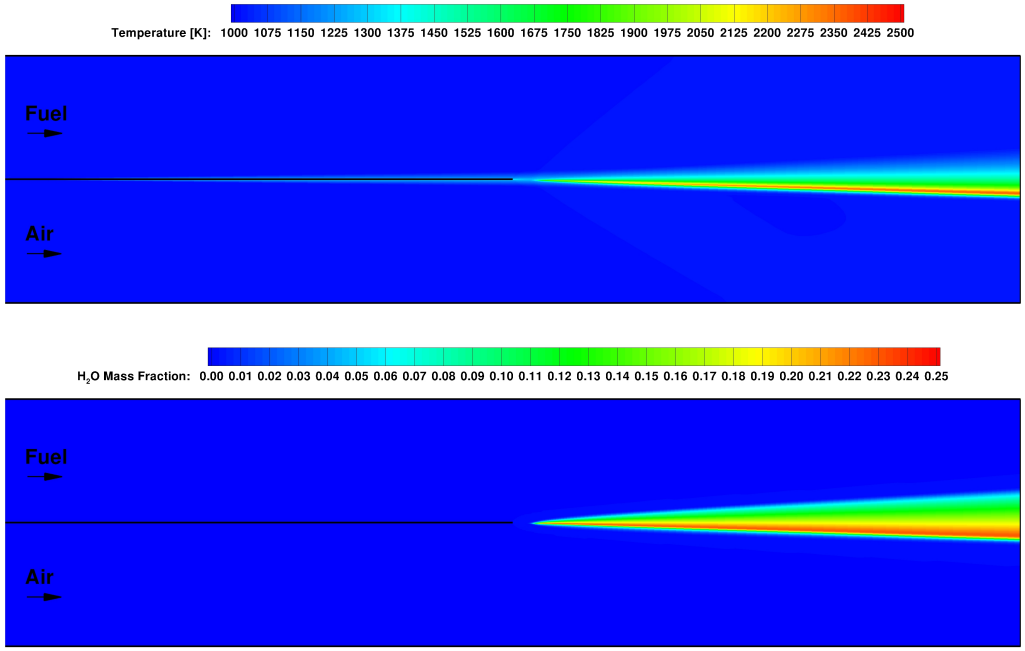


Figure 7: Temperature (top) and H_2O mass fraction (bottom) contours for the reacting mixing layer.

7 Unstructured Grid Simulation Example

The previous chapters walked through several examples on the use of VULCAN-CFD for simulations with structured (or i,j,k ordered) grids. VULCAN-CFD has a rich heritage as an accurate and robust flow solver for structured grids, but the generation of structured grids can be an arduous and time-consuming process for complex geometries. Moreover, the proliferation of CFD into the engineering workflow has in many instances placed a higher priority on rapid turn-around time than on simulation accuracy. VULCAN-CFD can accommodate both structured and unstructured grids, so the unstructured path can be chosen for applications that do not allow for the time required to develop high quality structured grids, or when the geometries are too complex for the generation of multiblock structured grids. The contents of the VULCAN-CFD input file for an unstructured grid simulation mirror closely to what is required for structured grid simulations. The primary difference is that the sections for boundary condition and cut connectivity mappings are no longer present, since the lack of i,j,k order forces this information to reside within the grid file. The space-marching features of VULCAN-CFD are also not available for unstructured grids, since the lack of i,j,k order prevents the specification of readily available parabolic marching planes.

At the current time, Pointwise is the only grid generation package that is fully supported by VULCAN-CFD for unstructured grid simulations. However, other grid generation packages (e.g., Heldenmesh[15]) can be used to generate mixed element unstructured grids for VULCAN-CFD provided that allowances are present to export the grid file in AFLR3 format along with an ASCII text file that maps user-defined boundary condition names to the integer values assigned by the grid generator. If the grid has been generated from within Pointwise, the process for setting the boundary conditions is analogous to that required for structured grids (see Chapter 4). However, the unstructured CAE plugin (NASA/VULCAN-UNS) must be chosen from the **CAE** tab of Pointwise instead of the structured version (NASA/VULCAN-STR). Also note that the problem dimension for unstructured grid simulations is currently limited to 3-D. Therefore, a single row of cells must be created by extruding (2-D) or rotating (axisymmetric) any 2-D grid created within Pointwise, with symmetry or periodic conditions applied to the spanwise boundaries to produce a valid unstructured grid for simulations intended to be 2-D or axisymmetric. The boundary condition mapping file that is output has the same root name as the grid file, but with a .mapbc extension. This file has the format shown by the example below:

```
5
1               101             INFLOW
2               102             OUTFLOW
4               103             SYMMETRY
3               102             OUTFLOW
5               100             WALL
```

where the first integer is the number of boundary conditions that were set, followed by 3 columns of information for each boundary condition. The first column contains the integer supplied to the grid file that tags each particular boundary condition defined. These integer values must be a sequentially ordered list starting at 1. The last column is a generic

character string denoting the class of boundary condition present at the surface. The middle column of integers is currently ignored, but is reserved for potential future use.

The problem chosen to introduce the user to the unstructured features of VULCAN-CFD is a case involving supersonic flow past a backward facing step. The input file for this case (back_step_uns.dat) is located in the **Sample_cases/Input_files** directory, and the corresponding AFLR3 format grid file (back_step.b8.ugrid) is in the **Sample_cases/Grid_files** directory. Note that the boundary condition mapping file, back_step.mapbc, is only used to provide the information required to set up the boundary condition groups section of the VULCAN-CFD input file. The boundary condition order used to define these groups **must** match the order in the mapping file.

The partitioning of unstructured grids is not constrained by any existing grid structure, so the steps required to partition the grid are performed for you as an automated part of the unstructured grid simulation process. As a result, unstructured simulations can be executed in parallel by simply providing the desired number of cores to be utilized on the VULCAN-CFD command line, e.g.,

vulcan 8 host_file -psg back_step_uns.dat

Note that even though a partitioned VULCAN-CFD input file is created as part of the simulation process (with _#_procs appended to the root name of the input file), this file should **not** be directly used on the VULCAN-CFD command line (as is required for structured grid simulations). This case, if executed with 8 processors, should require less than 10 minutes to complete and exhibit the convergence history shown in Fig. 8. Contours of the Mach number field are provided in Fig. 9. The shape and extent of the flow recirculation region is sensitive to the particular turbulence model chosen, as well as any compressibility corrections that are utilized. The user is encouraged to use this example problem to investigate the impact of these sensitivities.

As a final note, the unstructured grid capabilities in VULCAN-CFD are not as mature as those for structured grids, so a few of the structured grid VULCAN-CFD capabilities and utility codes have yet to be fully extended for use with unstructured grids. Structured grid simulation features that have yet to be implemented for unstructured grids, as well as notable differences between structured and unstructured grid simulations, are summarized below:

- There is currently no 2-D (or axisymmetric) simulation option for unstructured grid simulations. In order to simulate a 2-D problem, the user must generate a grid using at least a single row of grid cells with symmetry (or periodic) conditions applied at the bounding faces in the third dimension. Similarly, the simulation of an axisymmetric problem requires at least a single azimuthal strip of grid cells (rotated about the axis of symmetry) with symmetry (or periodic) conditions applied at the bounding azimuthal faces.
- The multiregion capability is not currently available for unstructured grid simulations.

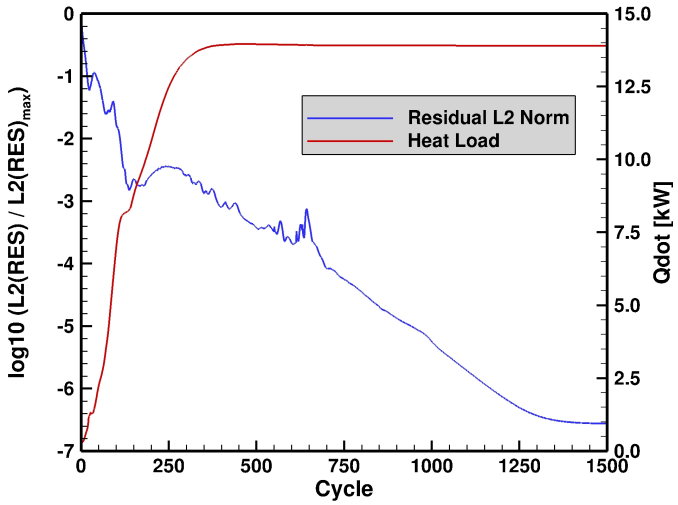


Figure 8: Convergence history for the backward facing step problem.

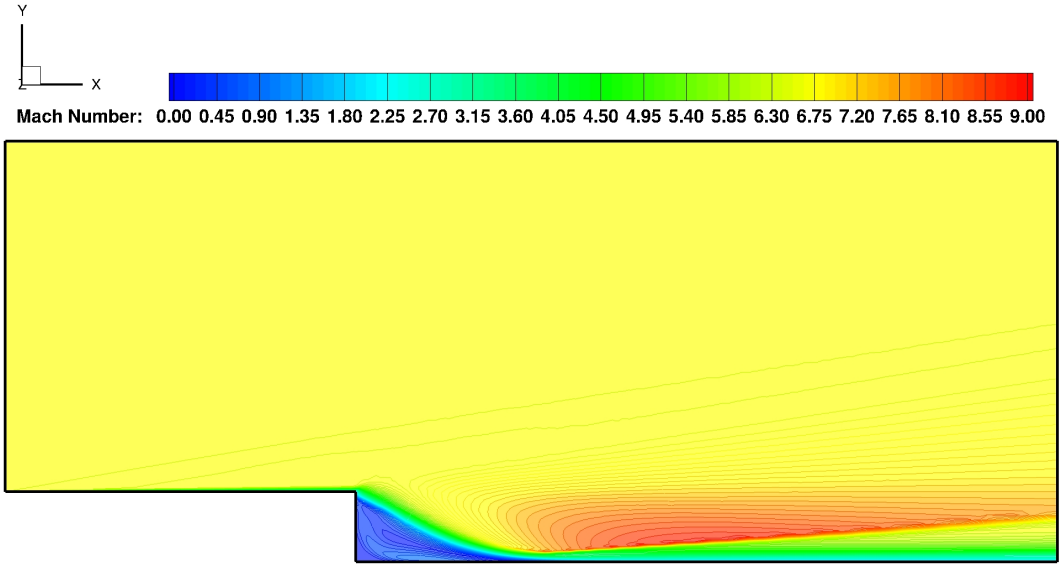


Figure 9: Mach number contours for the backward facing step flowfield.

- Non-C(0) block (partition) interfacing is not currently available for unstructured grid simulations.

- Reduced dissipation numerical schemes have yet to been introduced for unstructured grids, so the LES and/or hybrid RAS/LES closures are not yet available for unstructured grid simulations.

- Unstructured volumetric plot data files are output in the original unpartitioned state if **TECPLOT OUTPUT** or **VTK OUTPUT** is specified in your VULCAN-CFD input file when T-infinity has been enabled. CGNS format data files are written in the partitioned state, so **CGNS OUTPUT** should typically not be selected except for serial installs where the T-infinity output formats are not accessible.

- Unstructured surface loads data are output in the unpartitioned state as a binary Tecplot format file (vulcan.loads.plt) and a legacy VTK format file (vulcan.loads.vtk) when T-infinity has been enabled. For serial installs, the surface loads data are written as an ASCII Tecplot format file (vulcan.loads.tec) as is the case for structured grid simulations.

- The native VULCAN-CFD restart files are output to separate files for each unstructured grid partition, implying the number of partitions must remain unchanged when the simulation is restarted. A simulation restart is possible with a different number of cores (implying a different number of partitions) than used when the restart files were created by adding the line **USE TINF RESTART FILE** to the VULCAN-CFD input file. This option should not be used as a general replacement for VULCAN-CFD restart files, however, because this file only contains the minimal information required to initialize a simulation.

8 VULCAN-CFD Input Parameter Description

The VULCAN-CFD input file provides access to an abundance of options for defining the computational domain, setting boundary conditions, initializing the flow, controlling the solution process, and selecting the information to be postprocessed for plotting. This chapter focuses on the input parameters associated with the general input parameters of the VULCAN-CFD input file. Additional input descriptions that are specific to structured grid simulations (e.g., boundary condition assignments, block-to-block connectivity conditions, turbulence suppression regions, ignition regions) are described in subsequent chapters. With the notable exception of the Region Control Specification inputs, the input lines for VULCAN-CFD are mostly expressed as either a keyword, or a keyword followed by a real-valued number. For the latter scenario, at least 2 blank spaces must be present between the keyword and the numerical value to allow for the presence of a single blank space in the keyword. An optional trailing descriptor is also allowed, e.g.,

```
TWOD                (TWOD=2D, AXISYM=Axisymmetric, THREED=3D)
PROCESSORS   2.0    (no. of processors to invoke)
```

Any number of comment (or blank) lines may also be included. Comment lines are designated by specifying "$" as the first character in the line, e.g.,

```
$ - - - - - - - - - Reference condition data - - - - - - - - - $
```

Finally, unless specifically stated otherwise, the input lines can be specified in any order.

NOTE: All numerical input values that are intended to be integers will be converted to an integer representation using the FORTRAN **nint** function.

8.1 Parallel Processing Control Data

PROCESSORS #.#
This input specifies the number of processors to be used in the simulation. VULCAN-CFD is parallelized by assigning grid blocks (or grid partitions) to individual processors, and communication between processors is accomplished using the Message Passing Interface (MPI) standard. Hence, for VULCAN-CFD to function in a parallel fashion, the computational domain must be broken up into multiple grid blocks (or partitions). Grid partitioning is performed on-the-fly for unstructured grid simulations, and a structured grid partitioning utility (see Chapter 25) is included with VULCAN-CFD to help with load balancing the computational domain to any desired number of processors. Note that it is permissible to have more than one block assigned to a processor (i.e., fewer processors than blocks), which is a common occurrence for structured grid simulations.

NOTE: The number of processors specified on this line must match the value that the solver will access via the VULCAN-CFD command line for structured grid simulations. This is required because the partitioning step is not automated in the VULCAN-CFD simulation process for structured grid simulations, and this numerical value is used by the preprocessor

to determine the memory requirements for parallel processing. For unstructured grid simulations, the number of processors must be set to 1.0 (or the **PROCESSORS** line should simply be omitted) to invoke the automated partitioning step. Otherwise, VULCAN-CFD assumes that the simulation was already manually partitioned, and the number of processors specified on this line must match the number of processors specified on the VULCAN-CFD command line.

MESSAGE MODE #.#
This input defines the MPI message passing mode to be utilized. A value of 0.0 will invoke the standard blocking MPI sends and receives, while a value of 1.0 invokes a nonblocking strategy. The nonblocking strategy for structured grids involves two passes through the block interface connectivity conditions. The first pass uses the MPI buffered send to write all messages to a buffer, and the second pass receives all the messages sent to the buffer with a standard blocking receive. The nonblocking strategy used for unstructured blocks is the direct use of nonblocking sends and nonblocking receives. The nonblocking options provide the most efficient means of exchanging data, since it minimizes any latency when two processors that are exchanging data are not well synced. Both modes perform comparably if the computational load significantly outweighs the communication costs. However, the nonblocking communication will extend the linear speedup as the number of processors is increased for a given problem size. If this input line is not present, then the nonblocking MPI strategy will be used.

8.2 Computational Domain Dimension Specification

THREED
Solves the governing equations in a 3-D Cartesian coordinate system (x,y,z).

AXISYM
Solves the governing equations for axisymmetric flow without swirl (x,r).

TWOD
Solves the governing equations for Cartesian 2-D flow (x,y).

ONED
Solves the governing equations for Cartesian 1-D flow (x).

NOTE: **THREED** is currently the only allowable option for unstructured grid simulations, so a three-dimensional grid (with at least 1 cell in the third dimension) must always be supplied with symmetry conditions supplied as the boundary condition at the boundaries of the third dimension.

SUPPRESS U-MOMENTUM
Disable the x-momentum equation and force the x component of velocity to zero.

SUPPRESS V-MOMENTUM
Disable the y-momentum equation and force the y component of velocity to zero.

SUPPRESS W-MOMENTUM
Disable the *y*-momentum equation and force the *y* component of velocity to zero.

NOTE: The momentum suppression options are intended to force Cartesian 2-D (or 1-D) flow for unstructured simulations, which currently always integrates the full 3-D equation set. These options should not be employed for azimuthally symmetric bodies of revolution.

8.3 Computational Grid Input Data

STR GRID #.#
This input specifies that a PLOT3D (or PLOT2D) format structured grid file is to be read, with the name of the grid file appearing on the next line. A full or relative path to the grid file may be included as part of the grid file name. The value specified determines whether the grid file is a PLOT3D or PLOT2D format file for problem dimensions that are not 3-D. A value of 0.0 specifies that the grid file is supplied in PLOT2D format, and a value of 1.0 denotes that the grid file has a PLOT3D format. For 3-D problems the numerical value is ignored.

NOTE: Given the legacy of VULCAN-CFD as a structured grid code, this input can also be invoked by using the string **GRID** for backward compatibility.

STR GRID FORMAT #.#
This input specifies the attributes of the PLOT3D (or PLOT2D) structured grid file:

1.0 = single block ASCII formatted PLOT3D format file
2.0 = single block FORTRAN unformatted PLOT3D format file
3.0 = multiblock ASCII formatted PLOT3D format file
4.0 = multiblock FORTRAN unformatted PLOT3D format file

NOTE: Given the legacy of VULCAN-CFD as a structured grid code, this input can also be invoked by using the string **GRID FORMAT** for backward compatibility.

UNS GRID #.#
This input specifies the type of unstructured grid file that is to be read, with the name of the grid file appearing on the next line. A full or relative path to the grid file may be included as part of the grid file name.

NOTE: VULCAN-CFD currently only supports the AFLR3 format for unstructured grid files, so the numerical value is not currently required. However, other formats are expected to be added in the future, so a numerical value of 1.0 should be entered. Moreover, the file attributes of an AFLR3 grid file are dependent on the specific extensions provided to the AFLR3 grid file name, e.g.,

.r4.ugrid → specifies a sequential, unformatted, 4-byte real, big endian file
.r8.ugrid → specifies a sequential, unformatted, 8-byte real, big endian file

.b4.ugrid → specifies a streamIO (i.e., C-binary), 4-byte real, big endian file
.b8.ugrid → specifies a streamIO (i.e., C-binary), 8-byte real, big endian file
.lr4.ugrid → specifies a sequential, unformatted, 4-byte real, little endian file
.lr8.ugrid → specifies a sequential, unformatted, 8-byte real, little endian file
.lb4.ugrid → specifies a streamIO (i.e., C-binary), 4-byte real, little endian file
.lb8.ugrid → specifies a streamIO (i.e., C-binary), 8-byte real, little endian file
.ugrid → specifies an ASCII formatted file

so this aspect of the grid file name is not arbitrary.

USE TINF GRAPH PARTITIONER
This input invokes ParMETIS to partition the unstructured grid based on a connectivity graph instead of the default native bisection partitioning algorithm. The use of this input option for unstructured grid partitioning requires that T-infinity be enabled with ParMETIS support.

USE TINF CACHE REORDER
Reorder the cells using a reverse Cuthill-McKee reordering scheme with the goal of improving the cache efficiency of the solver. The default is to retain the ordering provided by the grid file.

USE TINF RANDOM REORDER
Reorder the cells using a random reordering scheme to potentially improve the convergence behavior of the point Gauss-Seidel implicit scheme. The choice of matrix ordering has been shown to impact the convergence behavior of implicit solvers,[16] and random reordering has been shown to be a more robust option than reverse Cuthill-Mckee in this regard. The default is to retain the ordering provided by the grid file.

USE TINF Q REORDER #.#
Reorder the cells using a Q reordering scheme to potentially improve the convergence behavior of the point Gauss-Seidel implicit scheme. The choice of matrix ordering has been shown to impact the convergence behavior of implicit solvers,[16] and Q reordering has been shown to be a more robust option than reverse Cuthill-Mckee with less of a performance penalty as compared to full random ordering. This option requires the specification of a "prune" width, which balances randomization (low prune width) with smaller bandwidth ordering (high prune width) like reverse Cuthill-Mckee. A value of 8 is often a good balance between performance and robustness.

GRID SCALING FACTOR #.#
This input specifies the multiplication factor (positive values only) required to convert grid coordinates to units of meters. For example, the scaling factor required for a grid that is provided in inches would be 0.0254. Note that the grid coordinates will be scaled back to their original units prior to writing out any files used for plotting purposes.

C(0) TOLERANCE #.#
This input (positive values only) specifies the allowable tolerance for grid connectivity

checks that are performed at structured grid block interfaces by the flow solver. In some circumstances (usually due to some loss in precision), the default value of 0.0001 may need to be increased. This input line should only be added to the input file if VULCAN-CFD erroneously reports an error in the C(0) connectivity information due to precision issues in the grid file.

SYMMETRY TOLERANCE #.#
This input (positive values only) specifies the allowable symmetry plane angle deviation tolerance (in degrees) for checks performed by the VULCAN-CFD flow solver to ensure symmetry boundary conditions are only specified on planar surfaces. In some circumstances (usually due to some loss in precision), the default value of 0.1 may need to be increased. This input line should only be added to the input file if VULCAN-CFD erroneously reports that a symmetry condition is being applied to a nonplanar boundary due to precision issues in the grid file.

PROFILE XYZ TOLERANCE #.#
This input (positive values only) specifies the allowable deviation when matching coordinate values between any provided "profile" files that are used for boundary condition specifications and the grid boundaries that utilize the profile data. This input is intended to provide the user with control of the tolerance when the default value (machine epsilon)$^{\frac{3}{4}}$ is not sufficient. This input line should only be added to the input file if VULCAN-CFD erroneously reports that the coordinates could not be matched due to precision issues in the grid and/or profile files.

AXISYM ANGLE #.#
VULCAN-CFD performs axisymmetric structured grid simulations on a polar sector grid with the velocity and gradients nulled in the azimuthal direction. The value specified here defines the sector angle (in degrees) to use when forming the 3-D sector mesh when a 2-D grid is supplied. The default value is 5 degrees if a 2-D grid is supplied.

NOTE: If a 3-D polar sector mesh is supplied, then the axisymmetric sector angle used for the simulation is the value extracted from the supplied 3-D grid (i.e., any value entered on this line will be ignored).

K-AXIS ORIENTATION #.#
This input converts a k-axis orientation for a left-hand-rule grid to one for a right-hand-rule grid. By default, VULCAN-CFD assumes that the k-direction for a 2-D or axisymmetric simulation is aligned with the positive z-direction (corresponding to a default value of 1.0). If this is not the case, set the **K-AXIS ORIENTATION** value to -1.0 to swap the direction.

8.4 Simulation Initialization and Restart Control Data

RESTART IN
This input specifies the restart file root name for any restart files used to initialize one or more regions. Whether or not the restart files are actually to be read in and used is controlled on a region-by-region basis in the region control specification section of the input

file, which is described at the end of this chapter. The root name of the input restart files (including the full path if desired) must be provided on the next line.

RESTART OUT #.#
This input specifies that a solution restart file is to be written out every "N" cycles of the simulation, where "N" is the value specified. If a value of 0.0 is entered, the restart file will only be written when the simulation of each region has completed. The root name of the output restart files (including the full path if desired) must be provided on the next line.

NOTE: Any subdirectories included as part of the path to the output restart files that do not exist will be created for you.

NOTE: The restart file name structure consists of a root name that the user provides followed by 1 or more extensions to denote the type of information the file contains:

<root_file_name>_M	information that maps grid partitions (blocks) to regions and boundary condition data
<root_file_name>_#_S	region specifications for region "#" (e.g., region solver settings, number of grid partitions in the region, region convergence history)
<root_file_name>_#_F	flowfield data for structured grid partition number "#"
<root_file_name>_#_U	flowfield data for unstructured grid partition number "#"
<root_file_name>_T	structured grid block topology data at topological singularities (i.e., 3-point and 5-point block to block connectivities)

NOTE: Given that individual restart files are output for each grid partition, it is recommended that a subdirectory be provided as part of any restart file name to limit the number of files that are written to the working directory of the VULCAN-CFD simulation.

USE TINF RESTART FILE #.#
This input (if present) overrides the standard restart files supplied by the **RESTART IN** line for unstructured simulations. This override should only be used in the following scenarios:

(a) to restart a simulation with a different number of processors than used previously
(b) to initialize a simulation based on interpolated restart file data from a different grid

The use of this input option requires that T-infinity be enabled, and the name of this restart file is the root file name provided by the **RESTART IN** line with a .snap extension. This file must reside in the same directory used for the **RESTART IN** restart files. The numerical value (if provided) determines whether to reset the iteration counter. A nonzero value will extract the residual error norm history from the standard VULCAN-CFD restart files, while a value of zero (or the absence of a numerical value) will ignore any existing residual history and treat the restart data as an initialization.

NOTE: This option should not be used as a general replacement for the standard VULCAN-CFD restart files because the data contained in this unified (unpartitioned) file only contains the minimal information required to initialize a simulation.

RESTART ON GRID LEVEL #.#
This input allows a structured grid simulation to be restarted from the grid level indicated. This option overrides the default that forces the flow solver to simply pick up where it left off when the previous simulation ended. The grid levels are numbered in ascending order from the coarsest to the finest level (e.g., if 3 multigrid levels are utilized, then 1.0 → coarsest grid, 2.0 → medium grid, and 3.0 → fine grid).

IGNORE 1ST-ORDER RESTART CYCLES
VULCAN-CFD allows a simulation to utilize a 1^{st}-order advection scheme for a specified number of iteration cycles to aid with the flushing out of initial condition transients through the use of a more robust dissipative numerical scheme. Information related to the number of cycles that were solved 1^{st} order is written to the restart files so that the value specified in the input file can be decremented appropriately when a simulation is restarted. This input permits an override of the default behavior by ignoring the number of 1^{st}-order cycles that were previously performed.

NOTE: The use of this default override option can be avoided by adhering to the following best practice guideline. If the actual number of initial cycles to be performed 1^{st} order is not known when setting up a new simulation, then set the 1^{st}-order switch flag in the region control specification section to a very large value. Once the decision is made to switch to the specified higher-order scheme (based on monitoring the solution convergence history), the 1^{st}-order switch flag can then be reset to zero prior to restarting the simulation. VULCAN-CFD will recognize this as a desire to immediately switch to the higher-order scheme.

RE-INITIALIZE BLOCKS #.#
This input allows the data in selected structured grid blocks (or unstructured partitions) to be reinitialized after reading the restart files. The actual block/partition numbers to be reinitialized are specified on the next line as a comma (or blank space) separated integer list, e.g.,

RE-INITIALIZE BLOCKS 3.0 (no. of blocks to reinitialize)
3, 5, 7

NOTE: The flowfield data will only be reinitialized for blocks with special initialization options, e.g., structured grid boundary condition propagation (see the **IN-ORD** column description in Chapter 14), initialization via bounding volume geometries (see Chapter 10), or the boundary condition blending **BLEND** and boundary layer thickness initialization **IBL** options described in the available boundary condition options section of Chapter 9.

INITIAL VELOCITY SCALE FACTOR #.#
This input line defines the scaling factor used to scale the velocity values when initializing the flowfield for the blocks provided in the **BLOCKS TO SCALE VELOCITY** input specification. This input is useful when initializing regions of the flowfield where the velocity is expected to be significantly different than the reference condition velocity.

BLOCKS TO SCALE VELOCITY #.#
This input allows the initial condition for the velocity values to be scaled for a specified

structured grid block list. The number of blocks to have their velocity values scaled is determined by the value specified, and the actual block numbers associated with the scaling are provided on the next line as a comma (or blank space) separated integer list, e.g.,

```
INITIAL VELOCITY SCALE FACTOR   0.1   (velocity scaling factor)
BLOCKS TO SCALE VELOCITY        5.0   (no. of blocks to scale velocity)
1, 2, 3, 6, 7
```

NOTE: If this input line is not accompanied by an **INITIAL VELOCITY SCALE FACTOR** input line, then the velocity scaling factor will be set to zero.

INIT. BLENDING FACTOR #.#
This input parameter controls how far the boundary condition state will be blended into the interior of structured grid simulations for flowfield initialization purposes. The value entered is the fraction of cells within each block with boundary faces that use the **BLEND** option (see the available boundary condition options section of Chapter 9). For example, a value of 0.5 blends the boundary condition into the interior for 50% of the cells in the boundary face computational direction (i, j, or k) direction, while a value of 0.0 disables the blending. This input only pertains to structured grid simulations, and the default value is 0.25.

INIT. BOUNDARY LAYER THICKNESS #.#
This input parameter forces the no-slip surface boundary conditions to be linearly blended with the flowfield initialization over the distance provided by the value specified on this line. The purpose of this input is to mitigate the discontinuous jump condition formed at the intersection of the initialized flowfield and the no-slip surfaces.

NOTE: The boundary layer thickness value must be provided with units that are consistent with the value specified for the **GRID SCALING FACTOR**. For instance, if the grid coordinates are supplied in units of inches, then the value entered here must also be provided in inches.

NOTE: This input will initialize the same boundary layer thickness to all of the no-slip surfaces. The same feature can also be enabled for each no-slip surface independently, providing more fine-grained control over the boundary layer initialization process (see the boundary condition options section of Chapter 9 for further details).

8.5 Output Control Data

CGNS OUTPUT #.#
This input line designates the output of volumetric plot data to a CGNS format binary file. The file generated (**vulcan_solution.cgns**) resides in the **Plot_files** directory.

0.0 = disables CGNS file format output (default)
1.0 = outputs flowfield data in the older ADF2 format
2.0 = outputs flowfield data in the newer ADF3 format

NOTE: The CGNS format is the only file format that is compatible with all of VULCAN-CFD's features. However, the data that are output using this file format will have the partitioned grid structure (i.e., the data are not currently recomposed back to the original unpartitioned state). Also, due to an inherent limitation in the CGNS data structure, the ASCII character "/" is not permitted to be part of a flow variable name. As a result, the units that are included as part of the plot variable names are removed for CGNS format data files.

TECPLOT OUTPUT #.#
This input line designates the output of volumetric plot data to a Tecplot format binary file. The file generated (**vulcan_solution.plt** or **vulcan_solution.szplt**) resides in the **Plot_files** directory. The use of this input option for unstructured grid simulations requires that T-infinity be enabled. The use of this input option for structured grid simulations requires that TecIO be enabled.

0.0 = disables Tecplot file format output (default)
1.0 = outputs flowfield data to the .plt file format
2.0 = outputs flowfield data to the .szplt file format (3-D structured grid simulations only)

NOTE: The Tecplot format is currently only supported for fully elliptic simulations.

NOTE: Tecplot format data files are always written out in the unpartitioned state for unstructured grid simulations. For structured grid simulations, the partitioned data will only be recomposed back to the original unpartitioned state provided that the **MERGE_MAP.DAT** file is present in the working directory of the VULCAN-CFD simulation.

VTK OUTPUT #.#
This input line designates the output of volumetric plot data to a legacy VTK format binary file for unstructured grid simulations if T-infinity has been enabled. The file generated (**vulcan_solution.vtk** resides in the **Plot_files** directory.

0.0 = disables legacy VTK file format output (default)
1.0 = outputs flowfield data to the legacy .vtk file format

NOTE: VTK format data files are always written out in the unpartitioned state.

PLOT3D OUTPUT #.#
This input line designates the output of structured grid volumetric plot data to a PLOT3D format file. The grid coordinates are written to the **plot3d.g** file, the flow variables are written to the **plot3d.f** file, the variable names are written to the **plot3d.nam** file, and information related to the boundary data is written to the **plot3d.g.fvbnd** file.

0.0 = disables PLOT3D file format output (default)
1.0 = outputs flowfield data to ASCII PLOT3D format files
2.0 = outputs flowfield data to unformatted sequential PLOT3D format files
-2.0 = outputs flowfield data to unformatted stream I/O PLOT3D format files

NOTE: The PLOT3D format is only relevant to structured grid simulations. Data files written to this file format can be recomposed (albeit non-natively) back to the original unpartitioned state provided that the **MERGE_MAP.DAT** file is present in the working directory of the VULCAN-CFD simulation, and the "-r" option is included on the VULCAN-CFD command line. The partitioned plot data files are written to the **Plot_files** directory, and the unpartitioned plot data files (if recomposed) are written to the **Recomp_files/Plot_files** directory.

PLOT PLANAR #.#
This input line allows for the output of PLOT3D volumetric plot data to a PLOT3D planar format file.

0.0 = PLOT3D whole (default)
1.0 = PLOT3D I-planes
2.0 = PLOT3D J-planes
3.0 = PLOT3D K-planes

NOTE: The planar PLOT3D file formats are only available when the unformatted sequential PLOT3D format is chosen. In general, the "whole" format is preferred unless the number of data entries written for one or more of the structured grid blocks is too large to support a 4-bit integer end-of-record indicator. An alternative option to overcome the 4-bit integer end-of-record indicator limit is to utilize the unformatted stream I/O PLOT3D format option if the postprocessor being utilized supports this feature.

PLOT FUNCTION #.#
This input specifies the specific variables to be output to the volumetric plot files. The value entered here indicates the number of plot functions that are listed immediately below this line. The list of available plot function names is given below:

MACH NO.	→	Mach number
VELOCITY	→	Cartesian velocity components
PRESSURE	→	static pressure
GAUGE PRESSURE	→	static pressure - reference pressure
TOTAL PRESSURE	→	stagnation pressure
PITOT PRESSURE	→	Pitot pressure
TEMPERATURE	→	static (or translational/rotational) temperature
TOTAL TEMPERATURE	→	stagnation temperature
DENSITY	→	static density
TOTAL DENSITY	→	stagnation density
ENTHALPY	→	static enthalpy
SENSIBLE ENTHALPY	→	static enthalpy referenced to 0° K
TOTAL ENTHALPY	→	stagnation enthalpy
ENTROPY	→	mixture entropy
SPECIFIC HEAT RATIO	→	ratio of specific heats (γ)
VIBRATIONAL TEMPERATURE	→	vibrational/electronic temperature
VIBRATIONAL ENERGY	→	vibrational/electronic energy

VISCOSITY	→	mixture molecular viscosity
CONDUCTIVITY	→	mixture molecular thermal conductivity
CONDUCTIVITY (TRN/ROT)	→	translational/rotational thermal conductivity
CONDUCTIVITY (VIB/ELE)	→	vibrational/electronic thermal conductivity
PRANDTL NO.	→	mixture Prandtl number
TKE	→	turbulence kinetic energy
OMEGA	→	specific turbulent dissipation rate (ω)
SPALART	→	Spalart transported viscosity variable ($\tilde{\nu}$)
EDDY VISCOSITY RATIO	→	turbulent to molecular viscosity ratio (μ_t/μ)
MENTER FUNCTION	→	Menter baseline blending function
HYBRID RAS/LES FUNCTION	→	hybrid RAS/LES blending function
THIVET FUNCTION	→	Thivet turbulence realizability function
DURBIN FUNCTION	→	Durbin turbulence realizability function
WALL DISTANCE	→	distance to nearest wall
FILTER WIDTH	→	LES subgrid filter width
CMU*	→	variable EAS viscosity coefficient
DYNAMIC CMU	→	dynamic subgrid viscosity coefficient
REYNOLDS STRESS	→	Reynolds stress tensor
REYNOLDS HEAT FLUX	→	Reynolds heat flux
REYNOLDS MASS FLUX	→	Reynolds mass flux (auxiliary data required)
MASS FRACTION	→	species mass fractions (auxiliary data required)
PRODUCTION RATE	→	species production rates (auxiliary data required)
HEAT RELEASE	→	chemical heat release term $\left(-\sum_{m=1}^{ns} \Delta h_{f_m}\ \dot{w}_m\right)$
VELOCITY DIVERGENCE	→	divergence of the velocity field
Q-CRITERION	→	Q-criterion for vortical structure visualization
|VORTICITY|	→	vorticity magnitude
|STRAIN RATE|	→	strain rate magnitude
|GRAD(PRESSURE)|	→	pressure gradient magnitude
|GRAD(TEMPERATURE)|	→	temperature gradient magnitude
|GRAD(DENSITY)|	→	density gradient magnitude
HYBRID ADVECTION SENSOR	→	hybrid advection scheme sensor
LIMITER COEFFICIENT	→	stencil-based limiter coefficient

NOTE: The stencil-based limiter coefficient is a vector with components that correspond to each reconstructed variable. The scalar value that is output is the smallest vector component at each cell, so this plot function provides a visual of the maximum level of limiting used during the inviscid flux reconstruction when a stencil-based limiter is chosen for unstructured grid simulations.

NOTE: The species-dependent plot functions (MASS FRACTION, PRODUCTION RATE, REYNOLDS MASS FLUX) require 2 additional lines of information. The first line is an integer that specifies the number of chemical constituents to consider. The second line lists the particular species names to be output, e.g.,

```
PLOT FUNCTION   6.0    (no. of plot function names)
DENSITY
```

VELOCITY
MACH NO.
PRESSURE
TEMPERATURE
MASS FRACTION
3
H2 H2O OH

32 BIT BINARY
This input line specifies the precision of unformatted volumetric plot files to be 32-bit (this is the default). The use of 32-bit precision produces smaller data files, which is typically acceptable for plotting purposes.

64 BIT BINARY
This input line specifies the precision of unformatted volumetric plot files to be 64-bit. The use of 64-bit precision is advantageous if the performance extraction utility (see Chapter 25) is to be utilized. The extraction of performance metrics often requires the use of iterative methods to solve nonlinear algebraic relationships, and the added precision allows a tighter convergence criteria to be applied.

EXCLUDE SLIP WALLS
This input line forces the exclusion of slip surfaces from the VULCAN-CFD force and moment accounting (slip surfaces are included by default).

OUTPUT FLUXES
This input line forces the inviscid fluxes for all i, j, and k faces of structured blocks to be output as a postprocessing step for use with the performance extraction utility (see Chapter 25). The fluxes extracted with this option are fully consistent with the inviscid flux scheme used by the solver, so any errors introduced by reconstructing the fluxes from flowfield data extracted from the plot data files are avoided.

NOTE: There is no recomposition utility for the flux files, so if the plot files have been recomposed back to their unpartitioned state, the restart files must first be recomposed using the **restart_merge** utility (refer to Chapter 25 for instructions on using this utility). The postprocessor can then be executed using the original unpartitioned VULCAN-CFD input file to create the topologically consistent flux files.

UPDATE TIME AVERAGE #.#
VULCAN-CFD allows for on-the-fly ensemble averaging of unsteady flowfield data for the purpose of generating a set of restart files with statistically converged mean flow properties. This input option determines the iteration cycle interval for updating the ensemble average. If this input line is present with a nonzero value, then restart files that contain the ensemble-averaged flowfield data are written to the directory **Time_files** whenever the standard restart files are written. If a full path is provided as part of the **RESTART OUT** restart file name, then the path will be removed for the time-averaged restart files to force their placement in the **Time_files** directory. If restart files are present in this directory, and if a nonzero value

is specified for this input line, then the postprocessor will replace the standard restart file data with the time-averaged restart file data. To override this default behavior, simply set this input value to 0.0 (or comment this line out) and execute the postprocessor to create plot files with the time-accurate restart file data.

NOTE: This input option should not be activated until the solution has reached a statistically stationary state. Otherwise, much larger ensembles may be required to effectively eliminate any biasing introduced into the ensemble average if startup transients were still present. A value of 0.0 can be specified to disable the averaging process during the early stages of the simulation.

OUTPUT H.O.T. #.#
This input line determines whether any higher-order statistics are to be formed and added to the time-averaged restart files.

0.0 = skip the output of higher-order terms (default)
1.0 = output the Reynolds stress tensor and scalar flux vectors
2.0 = output scalar variances (pressure, temperature, mass fraction) and covariances
3.0 = output all of the above 2^{nd}-order terms

OUTPUT S.G.T. #.#
This input line determines whether the subgrid (or modeled) stresses and flux vectors are to be formed and added to the time-averaged restart files.

0.0 = skip the output of modeled subgrid terms (default)
1.0 = output the modeled stress tensor and scalar flux vectors

OUTPUT TIME HISTORY #.#
When performing time-accurate simulations, this input controls the output frequency (cycle interval) of the flow variables that are present in the **PLOT FUNCTION** list. This input requires the specification of time history I/O (see Chapter 13 for unstructured grid time history specification details and Chapter 19 for the details related to structured grid time history specifications).

NOTE: For backward compatibility, the root name for the time history files can be provided on the next line for structured grid simulations. If a time history file name is detected, then all the file name strings provided in the structured grid time history subblock specification will be ignored.

NOTE: Structured grid time history files are currently only output to PLOT3D format files (i.e., **PLOT3D OUTPUT** must be enabled), and individual PLOT3D format files are written at each time interval for every subdomain specified in the structured grid time history subblock specification section of the VULCAN-CFD input file. For ease of postprocessing, these files can be merged using the **time_merge** utility described in Chapter 25. This utility will either merge all of the files into a single ASCII Tecplot file (as separate Tecplot zones), or alternatively merge all of the subdomains into separate PLOT3D function files for each

time interval. The first option is convenient for users of Tecplot, while the latter is better suited for users of Fieldview.

NOTE: Unstructured grid time history files are currently only output to Tecplot binary (.plt) files (i.e., **TECPLOT OUTPUT** must be enabled), and requires that T-infinity be enabled as well.

COMPUTE TOTAL PRESSURE LOSS
This input line determines the total pressure loss through the system by mass flux weighting the total pressure at all outflow and inflow boundaries. The total pressure loss is then output by the postprocessor using the relationship below.

$$\left(1.0 - \frac{P_{\circ}^{out}}{P_{\circ}^{in}}\right) \times 100$$

NOTE: The calculation of total pressure loss assumes all inflow boundaries contribute to the mass flux weighted inflow total pressure value, while all outflow boundaries contribute to the mass flux weighted outflow total pressure value. If some subset of these boundaries should be ignored, then the VULCAN-CFD performance extraction utility (**perf_ext**) must be used instead.

COMPUTE COMBUSTION EFFICIENCY
This input line outputs the combustion efficiency based on the consumption of the least available reactant. If chemical reactions are not enabled, then this option outputs the mixing efficiency (in this scenario the string **COMPUTE MIXING EFFICIENCY** can be used instead). At least 3 additional input lines (possibly as many as 5) must accompany this option as described below:

OXIDIZER SPECIES INDEX #.#
This input specifies the index of the oxidizer species. If this species is not a pure oxidizer (e.g., air), then the fraction of oxidizer present in this chemical constituent must also be provided on the next line.

PURE OXIDIZER FRACTION #.#
This input specifies the fraction of the oxidizer species that is truly the oxidizer (default is 1.0).

FUEL SPECIES INDEX #.#
This input specifies the index of the fuel species. If this species is not a pure fuel, then the fraction of fuel present in this chemical constituent must also be provided on the next line.

PURE FUEL FRACTION #.#
This input specifies the fraction of the fuel species that is truly the fuel (default is 1.0).

STOICHIOMETRIC F/O MASS RATIO #.#
This input specifies the stoichiometric fuel to oxidizer ratio.

NOTE: The fuel and oxidizer species indices are based on the species ordering in the chemical constituent data section of the VULCAN-CFD input file.

NOTE: VULCAN-CFD only allows the specification of a single fuel species for the evaluation of the combustion (or mixing) efficiency. If the fuel is a blend of multiple fuel species, then they must be lumped (if possible) and recast as a single fuel species in order to use this feature. The **mix_fit** utility can be used for this purpose. If lumping the fuel species is not possible (e.g., the fuel species appear independently in the kinetic model), then the VULCAN-CFD performance extraction utility (**perf_ext**) must be used instead.

NOTE: The calculation of the combustion (or mixing) efficiency assumes all inflow boundaries contribute to the inflow fuel and oxidizer flow rates, while all outflow boundaries contribute to the outflow fuel and oxidizer flow rates. If some subset of these boundaries should be ignored, then the VULCAN-CFD performance extraction utility (**perf_ext**) must be used instead.

COMPUTE FUEL PENETRATION #.#
This input line outputs the fuel penetration height at the outflow boundary as the maximum distance from any wall where the local equivalence ratio is greater than the threshold value specified on this line. This option must be used in conjunction with the **COMPUTE COMBUSTION EFFICIENCY** (or **COMPUTE MIXING EFFICIENCY** option), and is currently only applicable to nonreacting simulations.

NOTE: The calculation of the fuel penetration height considers all outflow boundaries.

WARNING MESSAGES #.#
This input line controls the output of various classes of warning messages. The warning messages (if present) reside in processor dependent files located in the directory **Warnings**. The warning message file names are of the form **warnings_#**, where "#" is the processor number.

0.0 = off (default)
1.0 = activate warnings related to wall function evaluations
2.0 = activate warnings related to temperature evaluations
3.0 = activate warnings for supersonic flow present at subsonic inflow/outflow boundaries
4.0 = activate warnings related to variable update realizability violations
5.0 = activate all warnings

QUIET OUTPUT
This input line disables nearly all of the screen output that describes the various steps being performed at the start of a simulation.

8.6 Global Solver Input Data

GLOBAL EULER
This input line specifies that the inviscid Euler equations are to be simulated.

GLOBAL VISCOUS
This input line specifies that the viscous Navier-Stokes equations are to be simulated.

HEXAHEDRAL GREEN-GAUSS GRADIENT
This input line computes the flowfield gradients required for the viscous fluxes for structured grid simulations using an auxiliary cell formed as the average of 2 adjacent cells (this is the default). This input option only pertains to structured grid simulations that utilize the full Navier-Stokes equations.

OCTAHEDRAL GREEN-GAUSS GRADIENT
This input line computes the flowfield gradients required for the viscous fluxes for structured grid simulations using an auxiliary cell with 2 corners defined by the end points of the cell face and the other 2 corners defined by the cell centers that straddle the face. This method offers a compact stencil with minimal interpolation required, but is somewhat more computationally expensive than the **HEXAHEDRAL GREEN-GAUSS GRADIENT** approach. This method is preferred for structured grids with topological singularities, since the full Navier-Stokes flux terms can be formed such that the fluxes telescope at the singular faces (this feature also requires the **ENABLE TOPOLOGY ANALYSIS** option to be active).

ENABLE TOPOLOGY ANALYSIS
This input line forces a grid topology analysis to be performed for structured grid simulations when the grid is known to have topological singularities. If enabled, then a special interpolation is invoked at nodes/cells adjacent to each topological singularity to ensure proper telescoping of the full Navier-Stokes flux contributions (provided that the **OCTAHEDRAL GREEN-GAUSS GRADIENT** approach is active). Enabling this option also corrects the nodal averages performed when generating the volumetric plot data files.

NOTE: The topology analysis can be computationally expensive even though it is only performed once at the start of the simulation. Hence, this feature should not be enabled if the structured grid is known to be free of topological singularities.

FORCE TOPOLOGY ANALYSIS
The topology analysis is only performed once when the simulation is initiated because the computational cost to perform the analysis can be significant for large problems. This input line forces a grid topology analysis to be performed when restarting a simulation that did not previously invoke the topology analysis.

NOTE: This input line should be removed (or disabled by preceding the line with a $) once the topology analysis has been performed, otherwise the grid topology analysis will be recomputed on every restart of the simulation.

PERIODIC BODY FORCE #.#
This input line applies a body force to the momentum and energy equations to allow for the simulation of fully-developed channel or pipe flows with periodic conditions applied at the

inflow/outflow planes (a common practice for large eddy or hybrid Reynolds-averaged/large eddy simulations). The streamwise direction of the simulation is indicated by the value provided, where:

1.0 = applies the body force to the x-momentum equation
2.0 = applies the body force to the y-momentum equation
3.0 = applies the body force to the z-momentum equation

If the value specified is positive, then the magnitude of the body force is set to precisely balance the instantaneous total shear force acting on the channel (or pipe) surfaces. This option forces the mass flow rate to remain constant as the simulation proceeds. Alternatively, a negative value can be provided, which allows for the specification of a fixed value for the total shear force. In this instance, the time-averaged value for the shear stress (in Pascals) must be specified on the next line as shown below:

AVERAGE SHEAR STRESS #.#

UPWIND ADVECTION VARIABLES #.#
This input defines the variables used to reconstruct the left and right state at the cell interfaces for the evaluation of the inviscid fluxes. The following options are available:

0.0 = density, velocity, pressure (default)
1.0 = density, velocity, temperature
2.0 = pressure, velocity, temperature

Depending on the equation set chosen (turbulent transport, thermal equilibrium, thermal nonequilibrium, etc.), additional variables may be required for the reconstruction. The additional equation set dependent variables required for the state reconstruction are as follows:

Turbulent Transport	$\rightarrow$	turbulent transport variables (e.g., k and ω)
Thermal Equilibrium	$\rightarrow$	species mass fractions
Thermal Nonequilibrium	$\rightarrow$	mass fractions and vibrational/electronic temperature

LDFSS SHOCK CONSTANT #.#
This input controls the level of dissipation in the Low Dissipation Flux Split Scheme (LDFSS[2]) of Edwards[17] near shock waves to counteract the carbuncle phenomenon that often appears aft of strong bow shocks. Valid values range between 2.0 and 4.0 with higher values adding additional numerical dissipation. The default value is 2.0.

APPROX ROE INVISCID JACOBIAN #.#
This input line specifies that the inviscid flux Jacobian will be evaluated based on an approximate 1^{st}-order Roe scheme. The numerical value is optional and, if present, controls the level of entropy fix that is applied to the Jacobian. Valid values range between 0.1 and 0.5 (0.1 is used if no value is specified). Larger values tend to make the implicit system more diagonally dominant, but may slow the convergence rate. This input is only relevant when a highly implicit scheme is selected (e.g., **ILU** or **SGS**), and is the default method

used to evaluate the inviscid flux Jacobian.

NOTE: This particular inviscid Jacobian is hand-derived, and treats the Roe matrix ($\|\tilde{\mathbf{A}}\|$) as a constant when forming the Jacobian. The only motivation for choosing this approximate hand-derived Jacobian (in lieu of the **DDT ROE INVISCID JACOBIAN** option) is that the hand-derived version is more computationally efficient. However, this option has yet to be fully implemented for the thermal nonequilibrium equation set, so one of the alternative options that utilize the operator overloading feature of FORTRAN should be invoked for simulations of thermal nonequilibrium flows.

DDT ROE INVISCID JACOBIAN #.#
This input line specifies that the inviscid flux Jacobian will be evaluated based on an exact 1^{st}-order Roe scheme using the operator overloading feature of FORTRAN. The numerical value is optional and, if present, controls the level of entropy fix that is applied to the Jacobian. Valid values range between 0.1 and 0.5 (0.1 is used if no value is specified). Larger values tend to make the implicit system more diagonally dominant at the expense of a reduced convergence rate. This input is only relevant to highly implicit schemes (e.g., **ILU** or **SGS**).

DDT INVISCID JACOBIAN #.#
This input line specifies that the inviscid flux Jacobian will be evaluated from the 1^{st}-order version of the selected inviscid flux scheme using the operator overloading feature of FORTRAN. This input is only relevant to highly implicit schemes (e.g., **ILU** or **SGS**). Specification of a numerical value is optional and, if present, controls the level of diagonal dominance for the implicit system as described below:

- If the Roe inviscid flux scheme is chosen, then the value entered controls the level of entropy fix that is applied. Valid values range between 0.1 and 0.5 (0.1 is used if no value is specified).

- If the LDFSS inviscid flux scheme is chosen, then the value entered controls the level of dissipation that is added. If no value is present (or if a value of 0.0 is specified), then the LDFSS[2] scheme is used to form the Jacobian. Any other value between 0.0 and 1.0 forms the Jacobian using the more dissipative LDFSS[0] variant.

- A numerical value (if entered) has no effect if the HLLC inviscid flux scheme is chosen.

ENABLE TIME STEP REDUCTION LOGIC #.#
This input line enables logic that reduces the time step in regions where the maximum allowed variable update factor (**DQ COEF** setting in the region control specification section of the input file) is being enforced. This logic temporarily reduces the time step to satisfy the specified update constraint, and holds it there for 50 iteration cycles. After 50 iteration cycles, the time step is allowed to return to its normal value with the hopes that the flow feature that caused the large update was an initialization transient. If the time step is reduced at any cell for 10 of these 50 iteration cycles, the time step reduction factor is permanently imposed for that cell.

NOTE: This option is often one of the most effective ways to increase solver robustness when stability problems are limiting the CFL number to unacceptably low values when an implicit time integration scheme is used.

RESET TIME STEP REDUCTION LOGIC #.#
Add this input line to reset the time step reduction logic back to its initial state when restarting a simulation.

UNS ADAPTIVE CFL METHOD #.#
This input line specifies the strategy used to locally reduce the time step for unstructured grid simulations when the adaptive CFL option is active in the region control specification section of the input file. The specific method is determined by the numerical value specified:

0.0 = The CFL number is reduced in regions of high pressure gradients and high mass fraction gradients (this strategy is the default setting and matches what is done for structured grid simulations)

1.0 = In addition to reducing the CFL number in regions of high pressure and/or mass fraction gradients, the viscous time step constraint is enforced for pyramidal cells and all cells that contain a boundary face with a boundary condition based on wall functions

2.0 = The CFL number is much more aggressively reduced in regions of high pressure and/or mass fraction gradients, and the viscous time step constraint is enforced for pyramidal cells and all cells that contain a boundary face with a wall function boundary condition

NOTE: The time step reduction that results when a value of 2.0 is entered can significantly reduce the iterative convergence rate and should only be invoked as a last resort.

CELL SKEW CFL LIMITER #.#
This input line enables logic that reduces the CFL values in highly skewed cells for simulations that solve the full Navier-Stokes equations when using any of the highly implicit schemes (i.e., **ILU** or **SGS**). The value specified determines how the local CFL number varies with respect to the cell skew angle. The valid range for the coefficient is between 1.0 and 10.0. The coefficient entered here is the minimum local CFL number that will be applied for highly skewed cells. This input is disabled by default.

UNS VOLUME CENTER METHOD #.#
This input line specifies the method used to compute the cell center coordinates of each unstructured grid cell. The specific method is determined by the numerical value specified:

0.0 = cell center coordinates are defined based on an arithmetic average of the node Cartesian coordinates that define the cell (default)

1.0 = cell center coordinates are defined using a cell face area weighted (linear) average of the cell face center Cartesian coordinates, where the cell face center coordinates are

computed using an edge length weighted (linear) average of the edge center Cartesian coordinates

2.0 = cell center coordinates are defined using a cell face area weighted (quadratic) average of the cell face center Cartesian coordinates, where the cell face center coordinates are computed using an edge length weighted (quadratic) average of the edge center Cartesian coordinates

NOTE: The weighted approaches reduces sensitivity to cell skewness, potentially improving robustness and iterative convergence.[18]

UNS CELL AVG GRADIENT METHOD #.#
This input line specifies the method for obtaining the cell-average gradients on unstructured grids. The specific method is determined by the numerical value specified:

0.0 = stencil formed via Green-Gauss
1.0 = stencil formed via linear least-squares using 1 level of face-neighbors
2.0 = stencil formed via linear least-squares using 2 levels of face-neighbors
3.0 = stencil formed via linear least-squares based on node-neighbors (default)

NOTE: The Green-Gauss and level-1 linear least-squares should only be invoked for pure hexahedral grids, since these stencils are prone to instabilities for general mixed element unstructured grids. For pure tetrahedral (or mixed element) unstructured grids, the node-neighbor least-squares option should typically be selected. This node-neighbor linear least-squares method is generally considered to be the more robust (and accurate) method, but the resulting stencil size is considerably larger (and more costly) than the face-neighbor stencil options.

UNS NODE CENTERED GRADIENTS #.#
This input line specifies that the linear least squares gradients are to be evaluated at the cell nodes, and subsequently averaged to obtain the cell-centered and face-centered gradients as required.[19] The node-centered gradients are directly averaged to obtain the cell face-centered gradients needed to construct the viscous fluxes. The node-centered gradients are averaged to the cell centers to obtain the cell-centered gradients required by gradient-based source terms. The method used to obtain the gradients required for the higher-order reconstruction of the cell solution to the cell face for the inviscid fluxes is determined by the numerical value specified:

0.0 = face averaging is not performed, i.e., cell-averaged gradients are used by averaging the node-centered gradients to the cell centers

1.0 = face averaging of the node-centered gradients are performed only for faces that are not shared with a hexahedral cell, and cell-averaged gradients (obtained by averaging the node-centered gradients to the cell centers) are used for quadrilateral faces shared with a hexahedral cell (FOC-ANG method described by White et al.[19])

2.0 = the use of face averaging or cell averaging of the node-centered gradients is dependent on the particular combination of cell-face topology (triangle or quadrilateral) and cell topology (tetrahedron, pyramid, prism, or hexahedron) of the cells that share each face (FAC-ANG method described by White et al.[19])

3.0 = face averaging of the node-centered gradients is used for all cell topologies and the node-centered least squares gradient stencil is adjusted based on the cell topology. (F-ANG+ method described by White et al.[20])

NOTE: The node-centered gradient method should be the least expensive way of computing the least-squares gradients. On pure hexahedral grids, the node-centered gradient stencil has a size of 8, whereas the cell-averaged gradient level 2 face-neighbor stencil has a size of 24, and node-neighbor stencil has a size of 26. On a pure tetrahedral grid, the level 2 face-neighbor stencil is the smallest with a size of 16, but gaps are present that can effect stability and accuracy. On nonhexahedral grids, the node-centered gradient stencil will always be smaller than the cell-averaged gradient node-neighbor stencil, since the number of nodes is always less than the number of cells. For example, on pure tetrahedral grids the ratio of nodes to cells is 6.

UNS LEAST-SQUARES WEIGHTS #.#
This input line controls the least-squares weighting that is applied to the gradients that use a least-squares stencil. The gradients are used for 3 purposes:

(a) higher-order variable reconstruction at the cell faces for the inviscid fluxes
(b) cell face gradient formation for the viscous fluxes
(c) source term gradients (e.g., turbulence production terms)

and the use of different least-squares weights for each of these purposes can be beneficial. The value specified for this input defines the least-squares weights that are applied for the gradient construction as indicated below:

-1.0 = a weighting power of -1.0 is uniformly applied

1.0 = a weighting power of 0.0 is uniformly applied

2.0 = a weighting power of 0.0 is used for the inviscid fluxes
a weighting power of -1.0 is used for the viscous fluxes
a weighting power of -1.0 is used for the source terms

3.0 = a weighting power of 0.0 is used for the inviscid fluxes
a weighting power of -1.0 is used for the viscous fluxes
a weighting power of -0.5 is used for the source terms

The default weightings are those used when a value of 2.0 is specified.

UNS LSQ COEF METHOD #.#
This input specifies the method used to obtain the coefficients for the weighted linear least

squares stencil used to compute the cell-averaged and/or node-centered gradients for unstructured grid simulations. The specific method is determined by the numerical value specified:

0.0 = a direct-inverse method is used to compute the inverse of the (3×3) least squares stencil matrix formed to compute the (3×3) least squares stencil coefficient matrix (default)

1.0 = a pseudoinverse method is used to compute the inverse of the $(N \times 3)$ least squares stencil matrix, where "N" is the number of cells present in the stencil, which is then used to compute the (3×3) least squares stencil coefficient matrix

NOTE: This latter method significantly increases the one-time cost of computing the least squares weighting coefficients, but is a more robust alternative.

UNS VISCOUS GRADIENT METHOD #.#
This input line controls the method used to construct the cell face gradients from the cell-average gradients. The available options are as follows:

0.0 = edge-normal correction method
1.0 = face-tangent correction method (default)
2.0 = alpha-damped method

The edge-normal correction method is a common approach for the construction of cell face gradients from the cell-average gradients, but is prone to instabilities as the cell skewness increases. The damping characteristics of the face tangent correction approach offer a substantial improvement in this regard. The alpha-damped method, which is based on an upwind discretization, also exhibits favorable damping characteristics.

UNS ALPHA DAMPING FACTOR
This input line specifies the coefficient used to scale the damping offered by the alpha-damped method. The default is 1.0, but larger values can be specified to increase the damping characteristics of the scheme.

UNS SOURCE GRADIENT METHOD
This input line controls the method used to compute the gradients required for source terms. The available options are:
0.0 = uses the cell-averaged gradients (default)
1.0 = uses a face weighted average of the face-tangent viscous gradients
2.0 = uses a face weighted average of the alpha-damped viscous gradients

UNS SPECTRAL RADIUS VISCOUS JACOBIAN
This input line specifies that the viscous flux Jacobian will be approximated by the spectral radius of the viscous fluxes for unstructured grid simulations. This option can be more robust than the **UNS TLNS VISCOUS JACOBIAN** or **UNS DDT VISCOUS JACOBIAN** options, but the convergence rate will be considerably slower. This option is only recommended when stability problems arise with the use of more precise viscous flux Jacobians.

UNS TLNS VISCOUS JACOBIAN #.#
This input line specifies that the viscous flux Jacobian will be evaluated based on the thin layer Navier-Stokes viscous terms. The numerical value is optional and, if present, controls the level of diagonal dominance that is applied to the Jacobian. Valid values range between 1.0 and 5.0 (1.0 is used if no value is specified, implying no diagonal dominance scaling). Values greater than unity will make the implicit system more diagonally dominant at the expense of a reduced convergence rate. This input is only relevant to the implicit unstructured schemes (e.g., **SGS**). By default, VULCAN-CFD uses this approach to forming the viscous flux Jacobian with a diagonal dominance scaling factor of 1.0.

UNS DDT VISCOUS JACOBIAN #.#
This input line specifies that the viscous flux Jacobian will be evaluated based on the thin layer Navier-Stokes viscous terms using the operator overloading feature of FORTRAN. The numerical value is optional, and if present, controls the level of diagonal dominance that is applied to the Jacobian. Valid values range between 1.0 and 5.0 (1.0 is used if no value is specified, implying no diagonal dominance scaling). Values greater than unity will make the implicit system more diagonally dominant at the expense of a reduced convergence rate. This input is only relevant to the implicit unstructured schemes (e.g., **SGS**).

NOTE: This option retains additional terms that the hand-coded **UNS TLNS VISCOUS JACOBIAN** neglects, but is also considerably more computationally expensive.

UNS SMOOTH LIMITER COEFFICIENT #.#
This input line defines the free coefficient used by the Venkatakrishnan and Nishikawa smooth limiters for variable extrapolation to cell faces when constructing the inviscid upwind fluxes. The default value is 1.0, which maximizes the amount of limiting, but values as high as 100.0 can be specified to reduce the limiting threshold.

NOTE: This Venkatakrishnan smooth limiter coefficient is specified in the region control specification section of the input file for structured grid simulations. This input control had to be moved to the main input section for unstructured grid simulations to accommodate the optional pressure limiter available for unstructured simulations.

UMUSCL LIMITER CONVERGENCE FREEZING CRITERIA #.#
When the residual error hangs after it has dropped a few orders of magnitude, often the limiter is responsible (i.e., limiter buzz). This input option freezes the stencil-based advection scheme limiters for unstructured simulations after the L2-norm of the residual error has dropped "N" orders of magnitude (where "N" is the value specified on this line). The value entered cannot be less than 1.0.

UMUSCL LIMITER ITER NUMBER FREEZING CRITERIA #.#
This input option forces the stencil-based advection scheme limiters to be frozen after "N" steady-state iteration cycles (where "N" is the value specified on this line).

NOTE: The user should ensure that other convergence measures, (i.e., integrated forces,

moments, and mass flow error) have leveled off before enabling any of the limiter freezing options. Otherwise, solution stability may be compromised if flow discontinuities have shifted slightly outside of where the frozen limiter is active.

UNSTRUCTURED MAX TMP MEMORY #.# [units]
This input option determines the amount of memory that VULCAN-CFD can allocate for temporary storage. The memory required for unstructured grid simulations is typically far greater than that for an equivalent structured grid simulation, so this input line provides some level of memory control. The default value used is 1 GB. Larger values may result in slightly reduced simulation times, while performance might degrade slightly if smaller values are specified. The units of the memory, while optional, should also be specified as follows:

KB → kilobytes
MB → megabytes
GB → gigabytes

If no units are provided, VULCAN-CFD will assume the following:

gigabytes (if the value specified is <= 256.0)
megabytes (if the value specified is > 256.0 and <= 1024.0
kilobytes (if the value specified is > 1024.0 and <= 1024.0^2)

VULCAN-CFD will not accept a value outside of this range unless the units of the memory are also provided.

NSTAGE #.#
This line specifies the number of Runge-Kutta stages to be used for a generic user-defined N-stage Runge-Kutta scheme. The line that follows contains the stage coefficients provided as a comma (or blank space) separated real-valued list, e.g.,

NSTAGE 3.0 (no. of Runge-Kutta stages)
0.333333333333 0.5 1.0

This line is only relevant if **R-K** is specified as the time integration scheme in the region control specification section of the input file.

TVD 3-STAGE RK
This line denotes the use of the 3-stage, 3^{rd}-order TVD (Total Variation Diminishing) Runge-Kutta scheme of Osher and Shu.[21] This line is only relevant if **R-K** is specified as the time integration scheme in the region control specification section of the input file.

HEUN 3-STAGE RK
This line denotes the use of the 3-stage, 3^{rd}-order scheme of Heun.[22] This line is only relevant if **R-K** is specified as the time integration scheme in the region control specification section of the input file.

CAR-KEN 5-STAGE RK
This line denotes the use of the 5-stage, 4^{th}-order 2N-storage scheme of Carpenter and Kennedy.[23] This line is only relevant if **R-K** is specified as the time integration scheme in the region control specification section of the input file.

ENABLE HYBRID WALL MATCHING #.#
This input line enables a hybridization of the standard solve-to-wall no-slip boundary condition with wall functions (in particular the Wilcox[24] wall matching functions). This option initially sets all wall function surfaces to the standard solve-to-wall conditions, then on the 2^{nd} pass, the properties are reset to the wall function conditions if the cell y^+ value (based on wall function relationships) is larger than the value specified on this line. If a y^+ value is not specified, then the default y^+ value of 2.5 will be used.

NOTE: The wall matching boundary conditions include logic that will blend the wall function relationships with a no-slip condition that is asymptotically consistent with the solve-to-wall relationships as the y^+ value approaches unity. As a result, it is recommended that the hybrid wall matching option only be enabled when the vast majority of surface points are sufficiently clustered so that the solve-to-wall relationships are valid.

DISABLE OUTFLOW BC RULE
By default, logic is in place to ensure that flow does not enter the domain at boundaries intended to be outflow boundaries (**PRES_OUT**, **MDOT_OUT**, **MIXED_OUT**, **EXTRAP**, and **EXTRAP2**. If the outflow boundary can not be positioned to ensure that flow exits the domain at each outflow face, then the outflow enforcement can be disabled by using this input line.

NOTE: Allowing the flow to enter the computational domain at a specified outflow boundary leads to an ill-posed boundary condition, so the use of this override is discouraged unless absolutely necessary.

RIEMANN STAGNATION BC
The specification of a subsonic inflow condition requires that one flow property be extrapolated from the interior. The Mach number is the variable that is extrapolated by default for the **P0_T0_IN** boundary condition, but this input line will override this behavior and instead extrapolate the outgoing Riemann invariant. This option might be a better choice for farfield stagnation inflow boundary specifications or inviscid simulations, but can be problematic within the boundary layer if a viscous surface intersects the stagnation inflow boundary.

STAGNATION BC RELAX ITERATIONS #.#
The specification of a subsonic inflow condition requires that one flow property be extrapolated from the interior. If the specified initial (reference) conditions are far from the values being imposed at a subsonic inflow boundary, then instabilities may be encountered. This input forces the "fixed" inflow conditions of the **P0_T0_IN** boundary condition (composition, stagnation conditions, transported turbulence variables) to be blended with the corresponding reference conditions for the first "N" iteration cycles (where "N" is the value

specified on this input line).

NOTE: For structured grid simulations, another option is to invoke the nozzle initialization feature that is available for the **P0_T0_IN** boundary condition (provided that the block topology is amiable with it). Initialization via bounding volume geometries (see Chapter 10) offers a similar capability and is available for both structured and unstructured grid simulations.

8.7 Hybrid Advection Input Specification

HYBRID ADVECTION SCHEME #.#
Scale-resolving simulations require the use of numerical schemes with as little dissipation as possible to resolve the small (relative to the grid spacing) turbulent flow structures that develop. VULCAN-CFD utilizes a hybrid advection scheme for structured grid simulations to achieve this goal where a flow-dependent blending function is utilized to determine the switch between the dissipative (e.g., upwind differencing) and nondissipative (e.g., central differencing) inviscid flux operators. The value specified on this line determines the hybrid advection scheme to employ:

0.0 = pure dissipative upwind-biased scheme, i.e., hybrid advection is disabled (this is the default)

1.0 = blend of the dissipative upwind-biased cell face variable reconstruction with a nondissipative symmetric reconstruction[25]

2.0 = dissipative upwind-biased scheme blended with a kinetic energy preserving nondissipative α-split flux formulation[25,26]

3.0 = dissipative upwind-biased scheme blended with the nondissipative α-split flux formulation of Honein and Moin[25,27]

NOTE: The hybrid advection schemes have not yet been implemented for unstructured grid simulations.

HYBRID ADVECTION CD ORDER #.#
This input line controls the order of accuracy of the symmetric reconstruction when the hybrid advection scheme is based on a blend of upwind and symmetric cell face reconstructions (i.e., HYBRID ADVECTION SCHEME 1.0). The value specified on this line determines the order of accuracy, i.e.,

2.0 = 2^{nd}-order symmetric reconstruction
4.0 = 4^{th}-order symmetric reconstruction
6.0 = 6^{th}-order symmetric reconstruction

NOTE: If this input line is not present, then the symmetric reconstruction order of accuracy will be set to be no less than that associated with the dissipative (upwind) reconstruction.

HYBRID ADVECTION SENSOR #.#
This input line specifies the functional form of the sensor used to blend the dissipative and nondissipative inviscid flux operators:

1.0 = functional form of Ducros[25,28]
2.0 = functional form of Larsson[25,29]
3.0 = modified functional form of Larsson[25]
4.0 = same as 3.0, but with no distinction made between compression and expansion waves

NOTE: By default, a smoothing operation is applied spatially to the hybrid advection sensor field. The smoothing operation can be disabled if the sensor option is negated.

HYBRID ADVECTION SENSOR LOWER LIMIT #.#
This input line sets a lower limit on the hybrid advection sensor. The value specified must lie between 0.0 (default) and 1.0. A value of 0.0 places no lower limit on the sensor, and a value of 1.0 completely disables the nondissipative operator. A value between 0.1 and 0.2 is typically sufficient for simulations that cannot be stabilized without retaining some portion of the dissipative operator.

HYBRID ADVECTION SENSOR UPPER LIMIT #.#
This input line sets an upper limit on the hybrid advection sensor if one wants to remove some portion of the dissipative contribution. A value of 0.0 completely disables the dissipative operator, and a value of 1.0 (default) places no artificial upper limit on the sensor.

HYBRID ADVECTION CONTINUITY LOWER LIMIT #.#
This input line sets a lower limit on the hybrid advection sensor for the mass continuity equations. This limit behaves in the same way as the generic lower limit (HYBRID ADVECTION SENSOR LOWER LIMIT), but the limit is only applied to the mass continuity equations. Some level of numerical dissipation is often required to avoid unrealizable extrema in the species mass fractions, and this input addresses this issue without adding dissipation to the other transport equations.

HYBRID ADVECTION MOMENTUM LOWER LIMIT #.#
This input line sets a lower limit on the hybrid advection sensor for the momentum equations. This limit behaves in the same way as the generic lower limit (HYBRID ADVECTION SENSOR LOWER LIMIT), but the limit is only applied to the momentum equations.

HYBRID ADVECTION ENERGY LOWER LIMIT #.#
This input line sets a lower limit on the hybrid advection sensor for the energy equations. This limit behaves in the same way as the generic lower limit (HYBRID ADVECTION SENSOR LOWER LIMIT), but the limit is only applied to the energy equations.

HYBRID ADVECTION TURBULENCE LOWER LIMIT #.#
This input line sets a lower limit on the hybrid advection sensor for the turbulent trans-

port equations . This limit behaves in the same way as the generic lower limit (HYBRID ADVECTION SENSOR LOWER LIMIT), but the limit is only applied to the turbulent transport equations.

HYBRID ADVECTION VORTICITY SCALE COEF #.#
This input line sets the coefficient associated with the vorticity portion of the modified form of the Larsson sensor. Valid values for this coefficient lie between 0.01 and 5.0. Larger values place a stronger weighting on the vorticity component of the sensor as compared to the background pseudovorticity term. The default value is 5.0.

HYBRID ADVECTION BACKGROUND SCALE COEF #.#
This input line sets the coefficient associated with the background pseudovorticity portion of the modified form of the Larsson sensor. Valid values for this coefficient lie between 1.0×10^{-8} and 0.05. Larger values place a stronger weighting on background pseudovorticity portion of the sensor as compared to the vorticity term. The default value is 0.05.

NOTE: Larger values tend to be more appropriate for the pure modified Larsson functional form (HYBRID ADVECTION SENSOR 4.0), while a value of 1.0×10^{-8} tends to be more appropriate for the version of this sensor that makes no distinction between compression and expansion waves (HYBRID ADVECTION SENSOR 3.0).

HYBRID ADVECTION SENSOR FREEZING CRITERIA #.#
Improved subiteration convergence associated with the **SUBIT1** and **SUBIT2** time step options is typically achieved if the hybrid advection sensor is frozen at some point during the subiteration process. This input option forces the sensor to be frozen after the L2-norm of the subiteration residual error has dropped "N" orders of magnitude (where "N" is the value specified for this input parameter). Freezing the sensor after 1 or 2 orders of magnitude drop (1.5 is the default) in the subiteration residual error tends to work well in practice. The value entered for "N" cannot be less than 1.0.

8.8 Thermodynamic Model Specification

GAS/THERMO MODEL #.#
This input line specifies the thermodynamic model for the simulation. The value specified determines the specific thermodynamic formulation:

0.0 = employs a calorically perfect gas model (i.e., the fluid is in thermal equilibrium, obeys the ideal gas law, and the specific heat capacity is constant)

1.0 = employs a thermally perfect gas model (i.e., the fluid is in thermal equilibrium, obeys the ideal gas law, but the specific heat capacity is allowed to vary with temperature)

2.0 = employs a two-temperature thermal nonequilibrium model, where the translational and rotational energy modes are not necessarily equilibrated with the vibrational and electronic energy modes

UNIVERSAL GAS CONSTANT #.#
This input line specifies the value for the universal gas constant in MKS units [J/(kmol K)]. The default value is 8314.34.

MIN. STATIC TEMPERATURE #.#
This input line specifies the minimum allowable temperature in Kelvins (default value is 10.0).

MAX. STATIC TEMPERATURE #.#
This input line specifies the maximum allowable temperature in Kelvins for calorically perfect gas simulations (default value is 15000.0). The maximum allowable temperature for noncalorically perfect gas simulations is set based on the upper temperature bound of the thermal property polynomial curve fits, and will **not** be overwritten by this input.

NOTE: This input specification can be useful for troublesome nonreacting simulations to get past transients that are reporting unrealistically high maximum temperature violations. In the absence of heat addition, the static temperature cannot significantly exceed the inflow stagnation temperature (total temperature can only increase due to non-unity Prandtl number effects). Setting the maximum allowable static temperature to be slightly larger than the incoming total temperature can provide additional simulation robustness if unrealistic temperature excursions are encountered.

T-V EXCHANGE MODEL #.#
This input line specifies the functional form of the translational/vibrational energy exchange source term for simulations of thermal nonequilibrium flows:

0.0 = evaluates the energy exchange term using the pure vibrational energy functional form using the characteristic vibrational temperatures provided in the thermal nonequilibrium database

1.0 = evaluates the energy exchange term using the vibrational/electronic energy value obtained from the species thermodynamic polynomial fits (this is the default)

2.0 = evaluates the energy exchange term using the mixture value for the vibrational specific heat multiplied by the difference in translational/rotational and vibrational/electronic temperature and a single mixture averaged relaxation time (this is the approximation utilized in the NASA Langley developed LAURA code)

8.9 Transport Model Specification

SPECIES VISCOSITY MODEL #.#
This input line defines the functional form utilized for computing the species molecular viscosities:

0.0 = Power law (**only** available for calorically perfect gas structured grid simulations)
1.0 = Sutherland law[30] (default)

2.0 = Sutherland law blended with a linear model when T < Sutherland temperature
3.0 = N/A (reserved for future use)
4.0 = McBride 4-coefficient polynomial

NOTE: Given that a calorically perfect gas simulation implies the use of a single chemical constituent (i.e., mixture formulas are irrelevant), this input can also be invoked with the string **VISCOSITY MODEL** if a calorically perfect gas has been selected.

MIXTURE VISCOSITY MODEL #.#
This input specifies the functional form used to obtain the mixture viscosity from the individual species viscosities:

1.0 = Wilke mixture law[31] (default)

SPECIES CONDUCTIVITY MODEL #.#
This input line defines the functional form utilized for computing the species molecular conductivities:

0.0 = Prandtl number (default)
1.0 = Sutherland law[30] (**only** available for noncalorically perfect gas simulations)
2.0 = N/A (reserved for future use)
3.0 = N/A (reserved for future use)
4.0 = McBride 4-coefficient polynomial[32,33] (**only** available for noncalorically perfect gas simulations)

NOTE: Given that a calorically perfect gas simulation implies the use of a single chemical constituent (i.e., mixture formulas are irrelevant), this input can also be invoked with the string **CONDUCTIVITY MODEL** if a calorically perfect gas has been selected.

MIXTURE CONDUCTIVITY MODEL #.#
This input specifies the functional form used to obtain the mixture conductivity from the individual species conductivities:

0.0 = Prandtl number (default)
1.0 = Mason and Saxena mixture law[34]

SPECIES DIFFUSION MODEL #.#
This input line specifies the functional form used to construct the molecular species diffusion terms:

0.0 = Fickian diffusion law (default)

8.10 Chemical Constituent Data

NO. OF CHEMICAL SPECIES #.#
This input line specifies the number of species tracked in the simulation, followed by 2 ad-

ditional lines that define the species names and the reference mass fraction values, e.g.,

```
NO. OF CHEMICAL SPECIES  7.0
N2     O2     H2  H2O  OH  O   H
0.767  0.233  0.0 0.0   0.0 0.0 0.0
```

The reference mass fraction values are those that correspond to the reference flow conditions provided in the reference condition data section of the input file.

NOTE: If finite rate chemistry is accounted for by a user-supplied CARM (Computer Automated Reduced Mechanism) subroutine, then the order of the species provided here must precisely match the species order in the CARM subroutine. The CARM routines provided with VULCAN list the species present in the mechanism (and the order in which they must be specified) in the header supplied with the routine.

SPECIES THERMODYNAMIC DATA FORMAT #.#
This input line specifies the format of the thermodynamic database file that is utilized, which must immediately be followed by the database file name. The value specified defines the database format:

0.0 = specifies the use of a VULCAN-CFD format database file containing both thermodynamic data and molecular transport data (in Sutherland law form). Three VULCAN-CFD format species database files are provided:

gas_mod.Lewis_1 → contains thermodynamic data as a 7-coefficient McBride thermodynamic property fit for a single temperature interval valid for temperatures between 300 and 5,000 K
gas_mod.Lewis_2 → contains thermodynamic data as a 7-coefficient McBride thermodynamic property fit across two temperature intervals valid for temperatures between 200 and 6,000 K
gas_mod.Lewis_3 → contains thermodynamic data as a 9-coefficient McBride thermodynamic property fit across three temperature intervals valid for temperatures between 200 and 20,000 K

1.0 = specifies the use of a NASA Glenn CEA format thermodynamic database file. A CEA format thermodynamic database file (**therm_cea.inp**) is provided.

NOTE: The NASA Glenn CEA database offers the most flexibility, since this format can house either the 7-coefficient and 9-coefficient McBride thermodynamic property fits for a given species. Moreover, the number of temperature intervals, as well as the temperature interval end points, can be independently defined for each species. This flexibility introduces some computational overhead (on the order of 8% or so) relative to simulations that utilize the **gas_mod.Lewis_2** database.

SPECIES TRANSPORT DATA FORMAT #.#
This input line specifies the format of the transport database file that is utilized. The value

specified defines the database format as follows:

0.0 = specifies the use of Sutherland law transport data as provided in the VULCAN-CFD format database file (this option requires that a VULCAN-CFD database file format be chosen for the SPECIES THERMODYNAMIC DATA FORMAT input line)

1.0 = specifies the use of a NASA Glenn CEA format transport database file, which houses the transport data as a 4-coefficient McBride transport property fit (this option requires that the transport database file name, **trans_cea.inp**, be provided on the next line)

SPECIES NONEQUILIBRIUM THERMODYNAMIC DATA
Additional information (beyond that required for the simulation of flows in thermal equilibrium) is needed to account for thermal nonequilibrium, e.g., translational/rotational specific heat capacity, characteristic internal mode temperatures, degeneracies, relaxation time correlations). This input line provides that information through a database whose file name follows on the next line. Two thermal nonequilibrium database files are provided with VULCAN-CFD. Each database contains identical information, except for the data related to the Millikan/White translational/vibrational relaxation times:

therm_noneq_laura.inp → contains the Millikan/White translational/vibrational relaxation time correlation data extracted from the LAURA (version 5.4-61473) software package developed at NASA Langley

therm_noneq_dplr.inp → contains the Millikan/White translational/vibrational relaxation time correlation data extracted from the DPLR (version 4.03.1) software package developed at NASA Ames

8.11 Chemistry Model Specification

CHEMISTRY MODEL #.#
This input line defines the manner in which chemical reactions are handled:

0.0 = chemically frozen (default)
1.0 = finite rate with Arrhenius and/or mixing controlled chemical kinetics
2.0 = N/A (reserved for future use)
3.0 = finite rate via a user-supplied CARM subroutine
4.0 = finite rate via a user-supplied CARM subroutine with ISAT

NOTE: If a Computer Automated Reduced Mechanism (CARM) subroutine is used, then it must conform to the CHEMKIN[35] CKYWP subroutine structure, i.e.,

```
SUBROUTINE CKWYP (P, T, Y, WDOT)
!
! INPUT:
! P - Pressure [dynes/cm**2]
!       Data type - real scalar
```

```
!  T - Temperature in  [K]
!       Data type - real scalar
!  Y - Mass fractions of the species
!       Data type - real array (1:ncs)
!
!  OUTPUT:
!  WDOT - Chemical molar production rates of the species  [mol/(cm**3 sec)]
!               Data type - real array (1:ncs)
```

To use a specific CARM subroutine, the file must be placed in the Source_code directory of your VULCAN-CFD installation and named **CARM.FD**. Once there, VULCAN-CFD must be recompiled to fold this module into the executable. This is most efficiently accommodated by issuing the following command:

```
install_vulcan  allid
```

NO. OF CHEMICAL REACTIONS #.#
VULCAN-CFD allows for the specification of an arbitrary chemical kinetic mechanism either through databases or as a user-supplied CARM subroutine. If finite rate chemistry is not provided as a CARM model, then a chemical kinetic database is required. This input line specifies the number of kinetic steps present in the database with the file name of the database provided on the next line, e.g.,

```
NO. OF CHEMICAL REACTIONS  7.0
$vulcanpath/Data_base/reac_mod.Larc_7x7
```

NOTE: The $vulcanpath shell variable, which points to the VULCAN-CFD installation path, can be used to access the VULCAN-CFD database repository as illustrated above.

ISAT PARAMETERS #.#
This input section controls various aspects of the In-Situ Adaptive Tabulation process when ISAT is invoked. The value specified here defines the number of ISAT parameters to be explicitly defined. The specific ISAT parameters that are accessible are given below:

MAX. TREE NODES #.#
This line specifies the maximum number of entries (nodes) in the ISAT tree structure (default is 50000).

ERROR TOLERANCE #.#
This line specifies the error tolerance used to determine when a new entry must be computed and added to the tree structure (default is 0.05). Recommended values vary between 0.01 and 0.05.

CHEMICAL TIME SCALE #.#
This line specifies an estimate of the chemical time scale used by ISAT to define reference values (default value is 1.0e-05).

F0(TEMPERATURE) #.#
This line specifies a representative temperature associated with chemically active regions (default value is 1000 K).

F0(PRESSURE) #.#
This line specifies a representative pressure associated with chemically active regions (default value is 1.0e5 Pa).

F0(<species name>) #.#
This line specifies a representative maximum mass fraction value for the given species name within chemically active regions (default value is 0.01). The species name must correspond to a species provided in the NO. OF CHEMICAL SPECIES input line.

NOTE: The **ISAT PARAMETERS** input line must appear after the **NO. OF CHEMICAL SPECIES** input line if any F0 mass fraction values are specified.

PARK RATE EXPONENT #.#
This input specifies the exponent that controls the effective temperature (defined as $T_{tr}^{p} \times T_{ve}^{1-p}$) used for the dissociation reaction rates for flows in thermal nonequilibrium. Typical values used for p vary between 0.5 and 0.7 (0.7 is the default).

PREFERENTIAL DISSOCIATION CONSTANT #.#
The chemical source term in the vibrational/electronic energy equation accounts for the vibrational energy that is lost or gained due to dissociation and recombination reactions. If molecules in higher vibrational states are assumed to be preferentially dissociated over those in lower vibrational states, then this can be readily accounted for by specifying a preferential dissociation constant that is greater than unity. This input line controls the level of preferential dissociation in this regard. A value of 1.0 (default) results in a nonpreferential model, while values greater than 1.0 provide for preferential dissociation.

EXPLICIT CHEMISTRY This input line disables the implicit treatment of the chemical source terms. The activation of this option will reduce the per-iteration cost of accounting for chemical kinetics, but the wide range of time scales associated with most finite rate chemistry models typically require an implicit treatment.

IMPLICIT CHEMISTRY #.#
This input line chooses the manner in which the Jacobians are formed for the point implicit treatment of the chemical source terms:

0.0 = disables implicit chemical source treatment (same as **EXPLICIT CHEMISTRY**)
1.0 = uses an analytically derived Jacobian
2.0 = uses a numerical finite difference Jacobian (default for chemically active flow)

NOTE: For large kinetic mechanisms, e.g., mechanisms with 10 or more species, the an-

alytically derived Jacobian will be significantly more computationally efficient than a numerical one based on finite differences.

NOTE: If a Computer Automated Reduced Mechanism (CARM) is utilized, then the analytic Jacobian option can only be chosen if a subroutine is provided within the CARM module that defines the analytical Jacobian. This routine must conform to the following DWDCTRD subroutine structure:

```
SUBROUTINE  DWDCTRD  (T, C, A, LDA)
!
!  INPUT:
!  T - Temperature in  [K]
!       Data type - real scalar
!  C - Molar species concentration [mol/cm^3]
!       Data type - real array (1:ncs)
!  LDA - Number of species in the skeletal mechanism used to define the CARM
!       Data type - integer scalar
!
!  OUTPUT:
!  A - Molar production rate Jacobian wrt molar concentration and temperature  [cgs units]
!             Data type - real array (1:LDA,1:LDA+1)
```

An SKKKII routine that defines the number of species and reactions in the skeletal mechanism that the CARM is based on must also be present, i.e.,

```
SUBROUTINE  SKKKII  (LDA, NCR)
!
!  OUTPUT:
!  LDA - Number of species in the skeletal mechanism
!       Data type - integer scalar
!  NCR - Number of reactions in the skeletal mechanism
!       Data type - integer scalar
```

8.12 Reference Frame Specification

ANGLE REF. FRAME #.#
This input defines the reference frame for angle of attack and yaw:

0.0 = angle of attack in xy plane, yaw angle in xz plane (default)
1.0 = angle of attack in xz plane, yaw angle in xy plane

ALPHA #.#
This input defines the angle of attack in degrees. Valid values range between -90.0 and +90.0 degrees (measured from the *x*-axis).

BETA #.#
This input defines the angle of yaw in degrees. Valid values range between -90.0 and +90.0 degrees (measured from the x-axis).

DIRECTION COSINE METHOD
This input line allows for the specification of the reference velocity direction using direction cosines rather than angle of attack and yaw. At least 2 (possibly 3) additional input lines are required as described below:

DIR COS(U) #.# specifies the direction cosine for the x-velocity component
DIR COS(V) #.# specifies the direction cosine for the y-velocity component
DIR COS(W) #.# specifies the direction cosine for the z-velocity component

NOTE: 2-D and axisymmetric simulations do not require the **DIR COS(W)** line to be present, but if it is, the value provided must be 0.0.

INTEGRATION REF. PRESSURE #.#
This input defines the reference pressure (in Pascals) used to convert integrated forces and moments from an absolute measure to a relative measure. (i.e., $P - P_\infty$ is used as the integrand). The default value is 0.0, which disables this conversion.

MOMENT REF. X #.#
This input provides the reference x-coordinate value (in meters) for the calculation of moments (default is 0.0).

MOMENT REF. Y #.#
This input provides the reference y-coordinate value (in meters) for the calculation of moments (default is 0.0).

MOMENT REF. Z #.#
This input provides the reference z-coordinate value (in meters) for the calculation of moments (default is 0.0). This input is only relevant to 3-D simulations.

INTEGRATION REF. LENGTH #.#
This input defines the reference length (in meters) used to convert integrated forces and moments to force and moment coefficients. The default value is 0.0, which disables the conversion.

INTEGRATION REF. AREA #.#
This input defines the reference area (in meters squared) used to convert integrated forces and moments to force and moment coefficients. The default value is 0.0, which disables the conversion.

INTEGRATION X SCALE FACTOR #.#
This input defines the scaling factor used to scale the x-direction lengths for the output of force and moment coefficients (default is 1.0). Use this option if the origin of the geometry

does not coincide with the location where moments are to be centered.

INTEGRATION Y SCALE FACTOR #.#
This input defines the scaling factor used to scale the y-direction lengths for the output of force and moment coefficients (default is 1.0). Use this option if the origin of the geometry does not coincide with the location where moments are to be centered.

INTEGRATION Z SCALE FACTOR #.#
This input defines the scaling factor used to scale the z-direction lengths for the output of force and moment coefficients (default is 1.0). Use this option if the origin of the geometry does not coincide with the location where moments are to be centered. This input is only relevant to 3-D simulations.

8.13 Reference Condition Specification

NONDIM #.#
VULCAN-CFD requires that a set of reference flow conditions be specified. These conditions are used to nondimensionalize the governing equations, and may also be used to define boundary conditions. The reference conditions can be specified as either static or total conditions, and the value entered for this input defines this setting:

1.0 = static properties used to set the reference conditions
2.0 = stagnation properties used to set the reference conditions

The specific information required to set the reference conditions depends on the gas model chosen (**GAS/THERMO MODEL**) for the simulation.

Reference condition specification for a calorically perfect gas

If a calorically perfect gas model is chosen, then the specific heat ratio (γ) and the Mach number are required:

SPECIFIC HEAT RATIO #.#
MACH NO. #.#

followed by any 3 of the following properties:

- gas constant in MKS units $\left[\frac{\mathrm{J}}{\mathrm{kg\ K}}\right]$
- pressure [Pa]
- temperature [K]
- density [kg/m^3]

If the reference conditions are based on static flow properties, then the input lines that correspond to the above quantities are as follows:

GAS CONSTANT #.#
STATIC PRESSURE #.#
STATIC TEMPERATURE #.#
STATIC DENSITY #.#

The input specification lines for total condition are defined in a similar fashion:

GAS CONSTANT #.#
TOTAL PRESSURE #.#
TOTAL TEMPERATURE #.#
TOTAL DENSITY #.#

This completes the definition of the reference state for inviscid flow simulations, but viscous flow simulations require additional information to set (or define) reference values for the Reynolds number and transport properties. How these data are entered depends on the value entered for the unit Reynolds number, i.e.,

UNIT REYNOLDS NO. #.#

If the value specified is positive, then the value entered is taken to be the Reynolds number per meter. This line must then be followed by an input specification that depends on the particular transport model chosen. If a power law is used for viscosity (i.e, **VISCOSITY MODEL** setting of 0.0), then the power law exponent must be specified:

POWER LAW EXP #.#

Otherwise, if a Sutherland viscosity law is chosen (i.e, **VISCOSITY MODEL** setting of 1.0 or 2.0), then the Sutherland temperature (in Kelvins) must be specified:

SUTHERLANDS LAW S0 #.#

NOTE: The specification of the Reynolds number is a common practice for calorically perfect gas simulations, where simple transport models are often justified (since there is no variation in composition and the temperature variability is typically mild). For the power law and Sutherland law models, the specification of the Reynolds number implies that only one additional transport model constant be specified (the power law exponent or the Sutherland temperature). The McBride transport model fit does not permit such a reduction in the parameter specification, so this transport model cannot be selected if the unit Reynolds number is provided.

If a negative value is provided for the **UNIT REYNOLDS NO.**, then the specific value entered is ignored, and the Reynolds number is instead computed based on a complete specification of the transport property model constants. The specification of the transport property model constants is dependent on the particular VISCOSITY model chosen.

The power law (**VISCOSITY MODEL** setting of 0.0),

$$\mu = \mu^{\circ} \left(\frac{T}{T^{\circ}}\right)^{n}$$

requires specification of the reference viscosity (μ°, [kg/(m s)]), reference temperature (T°, [K]), and the power law exponent (n):

POWER LAW MU0 #.#
POWER LAW T0 #.#
POWER LAW EXP #.#

The Sutherland law (**VISCOSITY MODEL** setting of 1.0 or 2.0),

$$\mu = \mu^{\circ} \left(\frac{T}{T^{\circ}}\right)^{\frac{3}{2}} \frac{T^{\circ} + S}{T + S}$$

requires specification of the reference viscosity (μ°, [kg/(m s)]), reference temperature (T°, [K]), and the Sutherland temperature (S, [K]):

SUTHERLANDS LAW MU0 #.#
SUTHERLANDS LAW T0 #.#
SUTHERLANDS LAW S0 #.#

A variant of the Sutherland law that switches to a linear temperature relationship is often more accurate at low temperatures. Setting the **VISCOSITY MODEL** to 2.0 invokes this model, where the standard Sutherland expression is replaced with the following equation when the temperature is less than the Sutherland temperature (S).

$$\mu = \mu^{\circ} \left(\frac{S}{T^{\circ}}\right)^{\frac{3}{2}} \frac{T^{\circ} + S}{S + S} \left(\frac{T}{S}\right)$$

The McBride Polynomial (**VISCOSITY MODEL** setting of 4.0),

$$\mu = C \exp\left(a_1 \ln T + a_2\, T^{-1} + a_3\, T^{-2} + a_4\right)$$

requires specification of the number of temperature intervals used for the functional form fit, followed by the lower and upper temperature bound (in Kelvins) and the a_1 - a_4 coefficients for each temperature interval. The following example illustrates the specification of a McBride polynomial for viscosity using 2 temperature intervals:

MCBRIDE POLYNOMIAL 2.0
200.0 1000.0
6.2526577E-01 -3.1779652E+01 -1.6407983E+03 1.7454992E+00
1000.0 5000.0
8.7395209E-01 5.6152222E+02 -1.7394809E+05 -3.9335958E-01

The last specification required is the value for the Prandtl number, which is used to define the thermal conductivity:

PRANDTL NO. #.#

NOTE: 0.72 is a typical Prandtl number value for calorically perfect "air" simulations.

Reference condition specification for noncalorically perfect gases

Considerably less information is required to define the reference state for noncalorically perfect gases, since much of the data is supplied elsewhere through the various database files. The required specifications include the Mach number,

MACH NO. #.#

followed by any 2 of the following thermodynamic properties:

- pressure [Pa]

- temperature [K]

- density [kg/m^3]

If the reference conditions are based on static flow properties, then the input lines that correspond to the above quantities are as follows:

STATIC PRESSURE #.#
STATIC TEMPERATURE #.#
STATIC DENSITY #.#

The input specification lines for total condition are defined in a similar fashion:

TOTAL PRESSURE #.#
TOTAL TEMPERATURE #.#
TOTAL DENSITY #.#

For simulations of flows in thermal nonequilibrium, the reference vibrational/electronic temperature and the translational/rotational temperature are taken to be equilibrated at the reference static temperature value. This default treatment can be overridden if a reference value for the vibrational/electronic temperature is explicitly provided, i.e.,

VIBRATIONAL TEMPERATURE #.#

In this scenario, the reference static temperature value should be entered as (or taken to be) the translational/rotational temperature.

NOTE: The value provided for the reference vibrational/electronic temperature must be no smaller than the reference translational/rotational temperature value.

This completes the definition of the reference state for inviscid flow simulations. For viscous flow simulations, a value for the Prandtl is also required,

PRANDTL NO. #.#

as well as a Schmidt number to define the diffusion coefficient for the Fickian diffusion law:

SCHMIDT NO. #.#

NOTE: The Schmidt number is only relevant if more than one chemical constituent is included in the simulation.

NOTE: If a **MIXTURE CONDUCTIVITY MODEL** other than the constant Prandtl number model is chosen, then the specified reference value for the Prandtl number will be overwritten with the value from the chosen transport property models evaluated at the reference conditions.

8.14 RAS Turbulence Model Specification

TURBULENCE MODEL
This input line signifies that a turbulence model specification will follow on the next line. The available Reynolds-Averaged Simulation (RAS) turbulence models are listed below:

LAMINAR denotes laminar flow, i.e., no turbulence model (this is the default)

SPALART invokes the standard Spalart-Allmaras 1-equation model[36]

SPALART-NEG invokes the "negative" Spalart-Allmaras 1-equation model variant[37]

MENTER-BSL-1994 invokes the 1994 version of the baseline Menter k-ω model[38]

MENTER-SST-1994 invokes the 1994 version of the shear stress transport Menter k-ω model[38]

MENTER-BSL-2003 invokes the 2003 version of the baseline Menter k-ω model[39]

MENTER-SST-2003 invokes the 2003 version of the shear stress transport Menter k-ω model[39]

WILCOX-1998 invokes the 1998 version of the Wilcox k-ω model[40]

WILCOX-2006 invokes the 2006 version of the Wilcox k-ω model[41]

RG-EASM-KO-2003 invokes the 2003 Rumsey-Gatski k-ω based algebraic stress model[42]

NOTE: Simply specifying MENTER-BSL or MENTER-SST will invoke the 1994 variants of the Menter models for backward compatibility.

STRAIN-BASED PRODUCTION
This option utilizes the "exact" production term for the k-equation, implying a production term based on the strain rate magnitude for linear eddy viscosity models (this is the default).

VORTICITY-BASED PRODUCTION
This option utilizes an approximate production term for the k-equation based on the vorticity magnitude. This strategy alleviates the production of artificially high turbulence levels behind strong shock waves, and tends to reduce the likelihood of turbulence variable realizability violations.

STRAIN/VORTICITY-BASED PRODUCTION
This option utilizes an approximate production term for the k-equation based on the Frobenius inner product of the strain rate and vorticity tensors. This strategy is often a good compromise between mitigating artificially high turbulence levels behind strong shock waves, while retaining some of the turbulence production due to strain.

NOTE: Enabling the vorticity-based or strain/vorticity-based turbulence production term is recommended for supersonic flows and strongly recommended for hypersonic flows.

QCR-2000
This option invokes the quadratic constitutive relation of Spalart (2000 variant)[43] for the Reynolds stresses instead of the linear Boussinesq relationship. This model is able to capture normal stress anisotropies, providing improved predictions for flows where this phenomenon is important.

QCR-2013
This option invokes the quadratic constitutive relation of Spalart (2013 variant)[44] for the Reynolds stresses instead of the linear Boussinesq relationship. This variant of the model includes a term that mimics the $\frac{2}{3}\rho k$ contribution of the Boussinesq Reynolds stress tensor nulled in the original QCR formulation.

QCR-2013-V
This option uses magnitude of vorticity instead of the strain rate magnitude in the **QCR-2013** approximation of $\frac{2}{3}\rho k$ when a 2-equation model is not used. This modification is intended to alleviate numerical issues that have been reported in the original strain-based formulation (particularly for wake flows).

NOTE: The 2013 variants of QCR only affect its use with the Spalart model. There is no difference between the 2000 and 2013 variants when used with a turbulence model that includes a transport equation for the turbulence kinetic energy.

THIVET REALIZABILITY #.#
This input option invokes the Reynolds stress limiter of Thivet[45] for any 2-equation turbu-

lence model that does not already have a built-in stress limiter. The value specified controls when the limiter becomes activated with valid values ranging between 0.5 and 1.0 (higher values provide more aggressive limiting).

DURBIN REALIZABILITY #.#
This input option invokes the realizability constraint of Durbin[46] for any 2-equation turbulence model that does not already have a built-in stress limiter. This option is identical to the Thivet realizability constraint, except the turbulent time scale is also limited. The value specified controls when the limiter becomes activated with valid values ranging between 0.5 and 1.0 (higher values provide more aggressive limiting).

TURB. INTENSITY #.#
This input defines the reference turbulent velocity fluctuation intensity relative to the mean reference velocity field. The reference turbulence kinetic energy value is defined based on this entry. This input is only relevant to 2-equation closure models.

TURB. VISC. RATIO #.#
This input defines the ratio of the reference turbulent viscosity to the reference molecular viscosity. If a Spalart model is selected, then the reference transported Spalart variable ($\tilde{\nu}$) is defined based on this entry. If a 2-equation model is selected, then the reference ω value is defined based on this entry and the reference value for k.

TURB. PRANDTL NO. #.#
This input defines the turbulent Prandtl number for the Reynolds heat flux vector. A typical value used in practice is 0.9.

TURB. SCHMIDT NO. #.#
This input defines the turbulent Schmidt number for the Reynolds mass flux vector. Typical values used in practice range between 0.5 and 1.0.

NOTE: The turbulent Schmidt number is only relevant if more than one chemical constituent is included in the simulation.

INIT. MIN. DIST.
The evaluation of the nearest cell center distance to each wall, which is required by some turbulence models, can be a computationally expensive process. As a result, VULCAN-CFD only performs this evaluation at the start of a simulation, and stores the result to the files used for restarting purposes. Under most circumstances, the VULCAN-CFD solver will detect whether the wall distance calculation has to be performed when restarting a simulation (e.g., switching from a turbulence model that does not require the wall distance to one that does or activating/deactivating no-slip boundary conditions). This input option forces the wall distance to be evaluated when restarting a simulation to cover any situation encountered that does not trigger the autodetection logic.

EXCLUDE SPALART FT2 TERM
This option disables the laminar suppression factor (f_{t_2}) that appears in the production and

destruction terms of the Spalart one-equation model. This term was intended to be used with the optional boundary layer trip term of the Spalart model, and was formulated to discourage laminar to turbulent transition upstream of the trip point. The Spalart model implementation in VULCAN-CFD does not include this trip term, so the f_{t_2} factor is unnecessary. The choice to include f_{t_2} should have a negligible impact on the solution provided that the inflow turbulence levels are set sufficiently high ($0.2 \leq \nu_t/\nu \leq 1.2$).

NOTE: Prior to VULCAN-CFD version 7.2, the f_{t_2} factor was disabled by default. In accordance with best practices outlined on the Turbulence Model Resource website,[47] the decision was made to enable this feature as an option, so that the standard Spalart model is used by default.

SPALART SHAT LIMITER 1B
The $\hat{S}$ term in the Spalart model must be greater than zero. This input utilizes the approach recommended by Spalart to limit $\hat{S}$ to no smaller than 0.3 times the vorticity magnitude (see Note 1B of the Turbulence Model Resource[47] description of the Spalart model). This is the default.

SPALART SHAT LIMITER 1C
The $\hat{S}$ term in the Spalart model must be greater than zero. This input utilizes the approach proposed by Allmaras et al.[37] that avoids clipping (see Note 1C of the Turbulence Model Resource[47] description of the Spalart model).

NO 2/3 RHOK
This option disables the $\frac{2}{3}\rho k$ term that appears in the normal components of the Reynolds stress tensor. The removal of this term will force the static pressure to remain nominally constant from the no-slip surface to the boundary layer edge (consistent with boundary layer theory), but the trace of the normal stresses will be zero (for linear eddy viscosity models) instead of recovering the turbulence kinetic energy value.

THIN LAYER SOURCE TERMS
This option forces the thin layer specification (as defined in the block specification section of the input file) to be applied to the turbulent source terms. This option only pertains to structured grid simulations.

ENABLE WILCOX LOW RE TERMS
This option enables the Wilcox low Reynolds number corrections for either the **WILCOX-1998** or **WILCOX-2006** models.

DISABLE WILCOX POPE TERM
This option disables the Pope-inspired correction that addresses the radial/round jet anomaly in the **WILCOX-1998** and **WILCOX-2006** models. The specific form of the radial/round jet anomaly correction used in the **WILCOX-1998** model is particularly prone to produce erroneous results for some 3-D scenarios.

MAX. ALLOWABLE MUT/MU #.#
This option allows the eddy viscosity to be clipped when it exceeds this specified factor times the molecular viscosity value. This option is only intended to control the eddy viscosity growth during the initial transient startup of steady-state simulations, and should in most cases be disabled as convergence is approached. If utilized, typical limiting values are 1×10^5 (or higher).

KARMAN CONSTANT #.#
This input provides the user access to the von Karman constant (κ). This value should only be altered if performing a parametric uncertainty analysis for the turbulence closure model.

EDDY VISCOSITY CONSTANT #.#
This input provides the user access to the leading constant in the eddy viscosity relationship (C_μ) for the 2-equation models. This value should only be altered if performing a parametric uncertainty analysis for the turbulence closure model.

TKE DIFFUSION CONSTANT #.#
This input provides the user access to the constant used for turbulent diffusion (σ_k) of the turbulence kinetic energy. This value should only be altered if performing a parametric uncertainty analysis for the turbulence closure model.

TKE DIFFUSION CONSTANT (OMEGA) #.#
This input provides the user access to the constant used for turbulent diffusion (σ_k) of the turbulence kinetic energy for the $k-\omega$ branch of the **MENTER** turbulence models. This value should only be altered if performing a parametric uncertainty analysis for the turbulence closure model.

TKE DIFFUSION CONSTANT (EPSILON) #.#
This input provides the user access to the constant used for turbulent diffusion (σ_k) of the turbulence kinetic energy for the $k-\epsilon$ branch of the **MENTER** turbulence models. This value should only be altered if performing a parametric uncertainty analysis for the turbulence closure model.

OMEGA DIFFUSION CONSTANT #.#
This input provides the user access to the constant used for turbulent diffusion (σ_ω) of the specific turbulence dissipation rate. This value should only be altered if performing a parametric uncertainty analysis for the turbulence closure model.

EPSILON DIFFUSION CONSTANT #.#
This input provides the user access to the constant used for turbulent diffusion (σ_ϵ) of the turbulence dissipation rate for the $k-\epsilon$ branch of the **MENTER** turbulence models. This value should only be altered if performing a parametric uncertainty analysis for the turbulence closure model.

OMEGA DESTRUCTION CONSTANT #.#
This input provides the user access to the constant used for turbulent destruction (C_{ω_d}) of

the specific turbulence dissipation rate. This value should only be altered if performing a parametric uncertainty analysis for the turbulence closure model.

EPSILON DESTRUCTION CONSTANT #.#
This input provides the user access to the constant used for turbulent destruction (C_{ϵ_d}) of the turbulence dissipation rate for the $k-\epsilon$ branch of the **MENTER** turbulence models. This value should only be altered if performing a parametric uncertainty analysis for the turbulence closure model.

MENTER-SST C_LIM #.#
This input provides the user access to the constant used to limit the Reynolds stresses for the **MENTER-SST** model. The default value should only be altered if performing a parametric uncertainty analysis for the turbulence closure model.

WILCOX-2006 C_LIM #.#
This input provides the user access to the constant used to limit the Reynolds stresses for the **WILCOX-2006** model. The default value should only be altered if performing a parametric uncertainty analysis for the turbulence closure model.

SPALART CV1 CONSTANT #.#
This input provides the user access to the c_{v_1} constant for the **SPALART** model. This value should only be altered if performing a parametric uncertainty analysis for the turbulence closure model.

SPALART CB1 CONSTANT #.#
This input provides the user access to the c_{b_1} constant for the **SPALART** model. This value should only be altered if performing a parametric uncertainty analysis for the turbulence closure model.

SPALART CB2 CONSTANT #.#
This input provides the user access to the c_{b_2} constant for the **SPALART** model. This value should only be altered if performing a parametric uncertainty analysis for the turbulence closure model.

SPALART CW2 CONSTANT #.#
This input provides the user access to the c_{w_2} constant for the **SPALART** model. This value should only be altered if performing a parametric uncertainty analysis for the turbulence closure model.

SPALART CW3 CONSTANT #.#
This input provides the user access to the c_{w_3} constant for the **SPALART** model. This value should only be altered if performing a parametric uncertainty analysis for the turbulence closure model.

SPALART DIFFUSION CONSTANT #.#
This input provides the user access to the constant used for turbulent diffusion (σ) of the

transported Spalart ($\tilde{\nu}$) variable. This value should only be altered if performing a parametric uncertainty analysis for the turbulence closure model.

8.15 RAS Turbulence Chemistry Model Specification

TURBULENCE CHEMISTRY MODEL
This input line signifies that a turbulence-chemistry interaction model specification will follow on the next line. The available Reynolds-Averaged Simulation (RAS) turbulence-chemistry models are listed below:

EDDY DISSIPATION invokes the mixing controlled kinetic model of Magnussen and Hjertager.[48] This model is sometimes referred to as the eddy breakup model.

EDDY DISSIPATION WITH FINITE RATE invokes a combination of single-step finite rate kinetics with the eddy breakup model (the branch chosen is the one with the lower chemical time scale).

1ST EDC CONSTANT #.#
This input sets the "A" constant for the mixing controlled kinetic rate. This constant governs the reaction rate when the limiting constituent is a reactant species (i.e., either fuel or oxidizer). Any non-negative value is permitted for this constant, with larger values providing faster reaction rates (a value of zero disables the chemistry). Typical values used in practice vary between 1.0 and 4.0 (4.0 is the default value).

2ND EDC CONSTANT #.#
This input sets the "B" constant for the mixing controlled kinetic rate. This constant governs the reaction rate when the limiting constituent is the product species. This aspect of the mixing controlled kinetic model is meant to restrict reactions to regions where high temperatures (i.e., combustion products) exist. This branch is somewhat problematic for nonpremixed systems, since the lack of a product species entering the domain prevents the chemical reactions from taking place. This can be remedied by seeding a small amount of product species in either the air stream or the fuel stream, or this branch can simply be disabled. Setting this constant to zero disables the product species branch (this is the default). To enable this branch, the constant must be set to some value greater than zero, but no larger than unity. If enabled, a typical value used for this constant is 0.5.

EDC LIMIT #.#
The chemical time scale is taken to be proportional to the turbulence time scale when the mixing-controlled kinetic model is chosen. This implies that the chemical reaction rate can be arbitrarily large as the turbulence time scale approaches zero. This unrealistic scenario can be avoided by using this input parameter to place a lower bound on the chemical time scale. Any non-negative value is allowed, with a value of zero disabling the chemistry. The value entered for this quantity should be a representative upper bound for $1/\tau_c$, where τ_c is the chemical time scale. This feature is disabled by default.

8.16 LES Turbulence Model Specification

The scale-resolving simulation capabilities are not currently available for unstructured grid simulations. Therefore, all of the input options described in this section only pertain to structured grid simulations.

TURBULENCE MODEL
This input line signifies that a turbulence model specification will follow on the next line. The available Large Eddy Simulation (LES) turbulence models are listed below:

IMPLICIT LES denotes no explicit subgrid model, i.e., numerical dissipation is the implicit subgrid closure (this is the default)

SMAGORINSKY invokes the Smagorinsky algebraic subgrid model[49]

VREMAN invokes the Vreman algebraic subgrid model[50]

YOSHIZAWA invokes the Yoshizawa 1-equation subgrid model[51]

SGS VISCOSITY CONSTANT #.#
This input line specifies the leading constant in the subgrid viscosity relationship (C_μ) for the algebraic subgrid closures. The default value is $(1.0/6.0)^2$.

VREMAN COMPRESSIBILITY CONSTANT #.#
The Vreman closure includes a term that accounts for compressibility, and this input line provides the constant associated with this term. The default value[50] is 0.1, and a value of 0.0 will disable this compressibility extension.

TURB. INTENSITY #.#
This input defines the reference subgrid velocity fluctuation intensity relative to the mean reference velocity field. The reference subgrid kinetic energy value is defined based on this entry. This input is only relevant to 1-equation subgrid closures.

TURB. PRANDTL NO. #.#
This input defines the turbulent Prandtl number for the subgrid heat flux vector.

TURB. SCHMIDT NO. #.#
This input defines the turbulent Schmidt number for the subgrid mass flux vector.

NOTE: The turbulent Schmidt number is only relevant if more than one chemical constituent is included in the simulation.

INIT. MIN. DIST.
The evaluation of the nearest cell center distance to each wall, which is required by some turbulence models, can be a computationally expensive process. As a result, VULCAN-CFD only performs this evaluation at the start of a simulation, and stores the result to the

files used for restarting purposes. Under most circumstances, the VULCAN-CFD solver will detect whether the wall distance calculation has to be performed when restarting a simulation (e.g., switching from a turbulence model that does not require the wall distance to one that does or activating/deactivating no-slip boundary conditions). This input option forces the wall distance to be evaluated when restarting a simulation to cover any situation encountered that does not trigger the autodetection logic.

NOTE: This input line should be removed once the wall distance has been evaluated, otherwise the wall distance will be recomputed on every restart of the simulation.

NO 2/3 RHOK
This option disables the $\frac{2}{3}\rho k$ term that appears in the normal components of the subgrid stress tensor. The removal of this term will force the static pressure to remain nominally constant from the no-slip surface to the boundary layer edge, but the trace of the normal stresses will be zero (for linear subgrid viscosity models) instead of recovering the subgrid turbulence kinetic energy value.

GERMANO DYNAMIC SGS
This input line enables the dynamic subgrid formulation of Germano[52] that assumes structural similarity between the subgrid stress tensor and the test-filter stress tensor. This dynamic formulation is available with either the **SMAGORINSKY** or **VREMAN** subgrid closures.

HEINZ DYNAMIC SGS
This input line enables the dynamic subgrid formulation of Heinz[53] that assumes structural similarity between the subgrid stress tensor and the Leonard stress tensor. This dynamic formulation is available with either the **SMAGORINSKY** or **VREMAN** subgrid closures. In general, this dynamic variant will result in less spatial variability than the traditional Germano approach.

SMOOTH DYNAMIC SGS #.#
This input controls the smoothing operator that can be applied to the dynamic subgrid closure constant. The value entered defines the number of smoothing sweeps (the default is 0.0, which disables the smoothing).

DO NOT CLIP DYNAMIC SGS
The application of a dynamic subgrid closure can produce negative values for dynamically determined subgrid coefficients, which often results in numerical instabilities. To mitigate this possibility, VULCAN-CFD clips the subgrid constant at zero (after each smoothing step) if any negative values are detected. This default behavior is overwritten when this input line is present.

SPATIALLY AVERAGE DYNAMIC SGS
This input line enables a spatial average of the dynamic subgrid coefficient in each homogeneous direction that is specified. This option should be employed if the turbulence has any homogeneity associated with it, and if the structured grid topology allows for it.

NOTE: The logic that allows the spatial average to continue across block interfaces requires that the grid blocks be full-face matching at all block interfaces aligned with the homogeneous direction(s).

HOMOGENEOUS-I
This input line denotes that the turbulence is taken to be homogeneous in the i-coordinate direction of each structured grid block. If this input line is present, the i-coordinate cell length portion of the subgrid length scale (i.e., MAX[Δ_x , Δ_y , Δ_z]) will be ignored when defining the subgrid length scale. This information is also used when spatially averaging the dynamic subgrid coefficients.

HOMOGENEOUS-J
This input line denotes that the turbulence is taken to be homogeneous in the j-coordinate direction of each structured grid block. If this input line is present, the j-coordinate cell length portion of the subgrid length scale (i.e., MAX[Δ_x , Δ_y , Δ_z]) will be ignored when defining the subgrid length scale. This information is also used when spatially averaging the dynamic subgrid coefficients.

HOMOGENEOUS-K
This input line denotes that the turbulence is taken to be homogeneous in the k-coordinate direction of each structured grid block. If this input line is present, the k-coordinate cell length portion of the subgrid length scale (i.e., MAX[Δ_x , Δ_y , Δ_z]) will be ignored when defining the subgrid length scale. This information is also used when spatially averaging the dynamic subgrid coefficients.

ISOTROPIC FILTER WIDTH
The subgrid filter width is defined as MAX[Δ_x , Δ_y , Δ_z] by default for pure large eddy simulations. This input line overrides this setting and instead defines the filter width as the cubed root of the cell volume.

IDDES FILTER WIDTH
The subgrid filter width is defined as MAX[Δ_x , Δ_y , Δ_z] by default for pure large eddy simulations. This input line overrides this setting and instead defines the filter width described by Shur et al.[54]

TIME SCALE SGS
This input uses the product of the time step and the maximum of the resolved velocity magnitude or the $\sqrt{2k}$ as a limiting factor when determining the subgrid length scale, i.e., the subgrid filter width is determined by MAX[Δ_x , Δ_y , Δ_z , Δ_t MAX($\|\mathbf{v}\|$, $\sqrt{2k}$)]. The intent of this modification is to increase the subgrid filter width when the time step is larger than the time required to convect the fluid more than a cell width.

8.17 Hybrid RAS/LES Specification

The scale-resolving simulation capabilities are not currently available for unstructured grid simulations. Therefore, all of the input options described in this section only pertain to structured grid simulations.

HYBRID-LES #.#
This input line defines the hybridization strategy for hybrid RAS/LES. The values specified for this input parameter determines the particular approach:

0.0 = disables the hybrid RAS/LES scheme (this is the default)
1.0 = enables the original DES model[55]
2.0 = enables the IDDES model[56]
-2.0 = enables the modified IDDES model of Gritskevich and Menter[57]
3.0 = enables the hybrid model of Edwards and Baurle[58,59]

HYBRID-LES SGS MODEL
The DES (Detached Eddy Simulation) formulations attempt to unify the RAS and LES branches with a single closure applicable to both regimes. The hybrid RAS/LES approach of Edwards and Baurle follows a different philosophy where any 2-equation RAS model can be blended with an independently chosen subgrid closure. This input line defines the subgrid closure for the LES branch when this hybrid strategy is selected:

IMPLICIT LES denotes no explicit subgrid model, i.e., numerical dissipation is the implicit subgrid closure (this is the default)

SMAGORINSKY invokes the Smagorinsky algebraic subgrid model[49]

VREMAN invokes the Vreman algebraic subgrid model[50]

YOSHIZAWA invokes the Yoshizawa 1-equation subgrid model[51]

NOTE: The specification of the algebraic subgrid closure constant (**SGS VISCOSITY CONSTANT**) and all of the options related to the filter width can accompany the above subgrid closure specification. The dynamic subgrid closure options, however, are not yet available for use within the hybrid RAS/LES framework.

HYBRID-LES ALPHA #.#
This input line specifies the constant (α) that controls the blending between the RAS and LES branches of the hybrid model of Edwards and Baurle. The blending function, F, is currently based on the ratio of the wall distance (d) to a modeled form of the Taylor

microscale (λ) via the following relationship,

$$F = \frac{1}{2}\left\{1 - \tanh\left[5\left(\frac{\kappa}{\sqrt{C_\mu}}\eta^2 - 1\right) - \tanh^{-1}(0.98)\right]\right\}$$
$$\eta = \frac{d}{\alpha\lambda}, \quad \lambda = \sqrt{\frac{\mu}{C_\mu\rho\omega}}$$

and the value chosen for α determines the y^+ position where the average LES to RAS transition has completed (defined when $F = 0.99$). The ideal location for the blend is the upper end of the logarithmic layer just before the transition where the wake law starts to deviate from the log law. To best utilize this model, a precursor Reynolds-averaged simulation should be performed so that the velocity profile can be scrutinized to determine this y^+ location. Once determined, the corresponding value for α is provided by the relationship below.

$$\alpha = \sqrt{y^+}$$

HYBRID-LES CDES #.#
This input line specifies the DES length scale constant for the LES branch. The default value is dependent upon the specific RAS model used for the DES formulation:

0.65 → default value for the DES model based on the Spalart 1-equation RAS closure
0.61 → default value for the DES model based on the Menter 2-equation RAS closure
0.72 → default value for the DES model based on the Wilcox 2-equation RAS closures

8.18 Inflow Recycling/Rescaling Control Data

The recycling/rescaling procedure implementation in VULCAN-CFD is specific to structured grid simulations only. Therefore, all of the input options described in this section only pertain to structured grid simulations.

The inflow conditions for many scale-resolving turbulent flow simulations require the introduction of resolved turbulent content into the computational domain. The use of a recycling/rescaling approach[60] is one method for providing this information. The idea behind the recycling/rescaling procedure is to take the variables from some streamwise plane within the computational domain (recycling plane), rescale them based on the turbulent flat plate scaling relationships, and then apply the rescaled properties to the inflow plane as the boundary condition. The recycling plane should be located at least 8 to 10 boundary layer heights downstream of the inflow plane to allow sufficient eddy turnover times to occur before properties are recycled back into the domain. To prevent undesired fluctuations in the freestream (e.g., acoustics) from being reintroduced at the inflow plane, an intermittency function is applied to the rescaled inflow properties. The current VULCAN-CFD implementation does not account for interacting walls (i.e., corners). The features of the VULCAN-CFD implementation of the recycling/rescaling process are given below:

- The recycling/rescaling process can be applied to any inflow boundary specified as a profile condition

- Only the fluctuations about the mean are recycled/rescaled for the primary flow variables to discourage any drift away from the specified mean inflow specification. The transported turbulence variables, however, are directly recycled/rescaled.
- Multiple independent recycling/rescaling specifications are allowed to handle inflows with different incoming boundary layer properties.
- Minimal user input is required to assemble the complete recycling and reintroduction (inflow) planes for structured grids that have been partitioned for efficient load balancing.

The recycling/rescaling process is implemented as a master/slave process, which is illustrated pictorially in Fig. 10. The master process houses the cells located at the inflow and recycling planes for each recycling specification using two entities, namely, the Inflow Block Assembly (IBA), and the Recycling Block Assembly (RBA). At the initialization stage, the grid coordinates for the recycling and inflow planes are sent to the master node to compute and store the grid related information required for subsequent interpolation operations of flow properties between the RBA and IBA. At each time step taken in the simulation, the flow properties in the grid blocks at the recycling location are mapped to the RBA on the master node. The required interpolations (and rescaling) are then conducted between the IBA and RBA entities. After the interpolation is complete, the results stored on the IBA are mapped back to the inflow plane blocks. It should be noted that the only extra information required to construct the IBA and RBA entities are the block number and orientation of the blocks that make up the (J=1,K=1) corner for each of these entities. The remaining information that is required is extracted from the block-to-block connectivity specification of the input file.

RECYCLE INFLOW #.#
This input line defines the number of independent recycling/rescaling specifications to be defined in the lines that follow. The first line for each recycling/rescaling specification is a string/value pair that provides a unique name for the recycling region and a master process ID to associate it with, e.g.,

<recycle_region_name> #.#

The name provided for each recycling/rescaling specification is arbitrary (but limited to 16 characters). While not required, each recycling/rescaling specification should have a different master process ID assigned to it (to the extent possible) for maximum computational efficiency. The master process ID number can be any value from 0 to one less than the number of processors used for the simulation.

NOTE: Once the recycling/rescaling specification name and master process ID have been specified, the remaining inputs required to complete the recycling/rescaling specification can be placed in any order.

BOUNDARY LAYER CELLS #.#
This input line defines the maximum number of wall-normal cells considered for the cur-

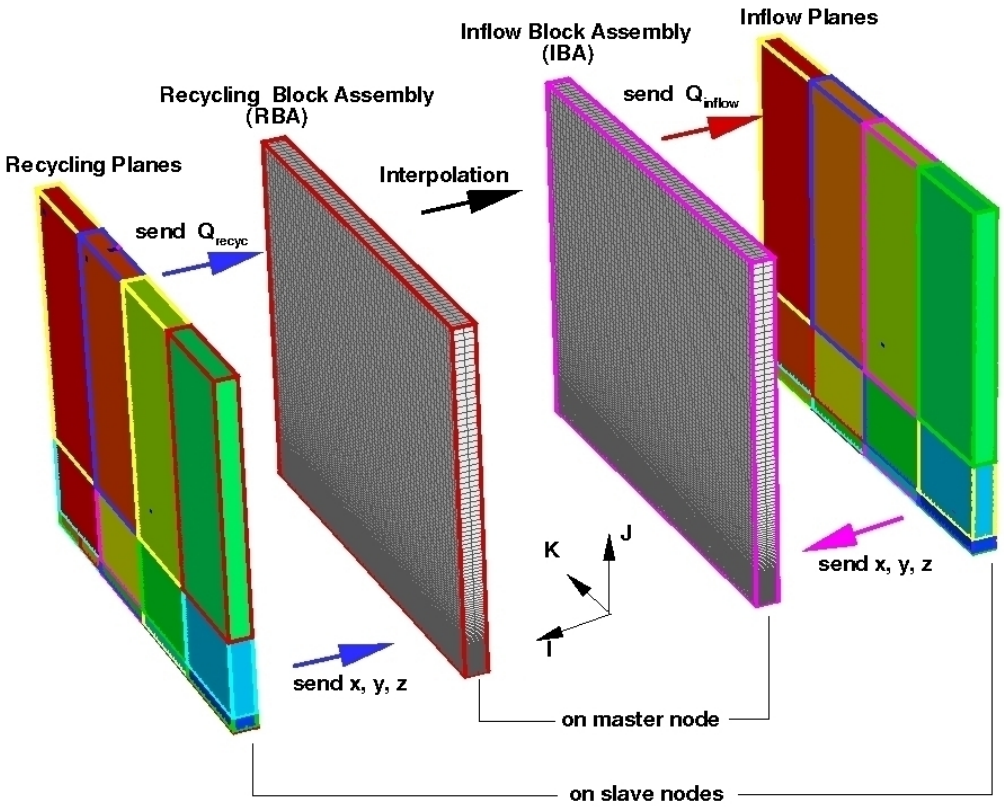


Figure 10: Image depicting the recycling/rescaling process (flow direction is right to left).

rent recycling/rescaling specification. This value should be chosen to either ensure that the recycling/rescaling regions do not overlap, or to provide a limit after which the block topology no longer allows the mapping to a single plane. For example, a group of blocks near the surface may form a "wrap" consisting of pure H-topologies that eventually transition into a more complex topology outside of the boundary layer. In this scenario, the bounding wall-normal cell index cannot exceed the end of the "wrap" topology. A minimum of 20 cells is required for this bound.

INFLOW BLOCK CORNER
This input line provides the block number and axis orientation for the block that contains the (J=1,K=1) corner of the reintroduction plane formed by the IBA (Inflow Block Assembly). This information is supplied as shown below:

INFLOW BLOCK CORNER
BLOCK # I-AXIS J-AXIS K-AXIS

where

BLOCK # $\rightarrow$ specifies the block number that contains the (J=1,K=1) corner of the rein-

troduction plane formed by the IBA

I-AXIS → specifies the structured grid coordinate that corresponds to the streamwise direction of the IBA (allowable input values are: +I, -I, +J, -J, +K, -K, where a positive value implies that the coordinate is aligned with the flow direction)

J-AXIS → specifies the structured grid coordinate that corresponds to the wall-normal direction of the IBA (allowable input values are: +I, -I, +J, -J, +K, -K, where a positive value implies that the surface resides at the MIN boundary of the wall-normal coordinate axis of the block)

K-AXIS → specifies the structured grid coordinate that corresponds to the spanwise direction of the IBA (allowable input values are: +I, -I, +J, -J, +K, -K, where a positive value implies that the spanwise coordinate axis of the block is aligned with the spanwise coordinate of the IBA)

NOTE: The partitioning tools supplied with VULCAN-CFD do not currently map the IBA information to the partitioned state, so the partitioned VULCAN-CFD input file must be manually altered after the structured grid partitioning step has been performed to map the IBA block number to the corresponding partitioned block number. The mapping information required to perform this step can be determined by examining the **MERGE_MAP.DAT** file that is generated by the structured grid partitioning tool (see Chapter 25).

RECYCLE BLOCK CORNER
This input line provides the block number, streamwise station index, and axis orientation for the block that contains the (J=1,K=1) corner of the recycle plane formed by the RBA (Recycle Block Assembly). This information is supplied as shown below:

RECYCLE BLOCK CORNER
BLOCK # I-AXIS I-INDEX J-AXIS K-AXIS

where

BLOCK # → specifies the block number that contains the (J=1,K=1) corner of the recycle plane formed by the RBA

I-AXIS → specifies the structured grid coordinate that corresponds to the streamwise direction of the RBA (allowable input values are: +I, -I, +J, -J, +K, -K, where a positive value implies that the coordinate is aligned with the flow direction)

I-INDEX → specifies the streamwise (I) index value that corresponds to the recycle plane used to form the RBA

J-AXIS → specifies the structured grid coordinate that corresponds to the wall-normal direction of the RBA (allowable input values are: +I, -I, +J, -J, +K, -K, where a positive value implies that the surface resides at the MIN boundary of the wall-normal coordinate

axis of the block)

K-AXIS → specifies the structured grid coordinate that corresponds to the spanwise direction of the RBA (allowable input values are: +I, -I, +J, -J, +K, -K, where a positive value implies that the spanwise coordinate axis of the block is aligned with the spanwise coordinate of the RBA)

NOTE: The streamwise index value chosen for the recycling plane must be at least 2 (or possibly 3) to avoid recycling ghost cell values. A value of 2 or greater is required if the one-sided stencil used for variable extrapolation to the face for inviscid flux construction requires 2 cells (i.e., standard 2^{nd}-order accurate MUSCL schemes). A value of at least 3 is required when using a higher-order scheme (e.g., PPM, WENO, or MUSCL w/ENO limiting).

NOTE: The partitioning tools supplied with VULCAN-CFD do not currently map the RBA information to the partitioned state, so the partitioned VULCAN-CFD input file must be manually altered after the structured grid partitioning step has been performed to map the RBA block number and streamwise (I) index value to the corresponding partitioned state. The mapping information required to perform this step can be determined by examining the **MERGE_MAP.DAT** file that is generated by the structured grid partitioning tool (see Chapter 25).

INFLOW BL THICKNESS #.#
This input line specifies the time-averaged inflow boundary layer thickness. The boundary layer thickness entered must be consistent with the value specified for the **GRID SCALING FACTOR**. For example, if the grid coordinates are supplied in units of inches, then the value entered here must also be provided in inches. This input controls the intermittency function that is employed to damp the oscillations (i.e., acoustics) in the freestream that form due to the recycling process.

BL THICKNESS RATIO #.#
This input line specifies the ratio of the time-averaged boundary layer thickness at the **recycle** plane relative to the time-averaged boundary layer thickness at the **inflow** plane. This value must be greater than or equal to 1.0.

SKIN FRICTION RATIO #.#
This input line specifies the ratio of the time-averaged skin friction at the **inflow** plane relative to the time-averaged skin friction at the **recycle** plane. This value must be greater than or equal to 1.0.

FRICTION VELOCITY RATIO #.#
This input line specifies the ratio of the time-averaged friction velocity at the **inflow** plane relative to the time-averaged friction velocity at the **recycle** plane. This value must be greater than or equal to 1.0.

NOTE: The friction velocity ratio and the skin friction ratio provide the same information

to the recycling process, so the specification of both the **SKIN FRICTION RATIO** and the **FRICTION VELOCITY RATIO** will be flagged as an error.

FREESTREAM VELOCITY #.#
This input line specifies the time-averaged boundary layer edge velocity (m/s). This input parameter is used by the random walk process that shifts the recycled properties in the homogeneous spanwise direction. The intent is to discourage the formation of longitudinal vortices (rollers) that can form due to the recycling procedure. The random walk process is disabled if the value entered on this line is zero, or if this input line is not present.

MAX. INTEGRATION TIME #.#
This input line specifies an upper bound for the total integration time (s) associated with the time-accurate simulation. This input parameter is used by the random walk process that shifts the recycled properties in the homogeneous spanwise direction. The intent is to discourage the formation of longitudinal vortices (rollers) that can form due to the recycling procedure. The random walk process is disabled if the value entered on this line is zero, or if this input line is not present. If used, the value entered here must be larger than the total elapsed flow evolution time utilized in the time-accurate simulation.

NOTE: The current recycling/rescaling implementation does not allow for intersecting walls (i.e., corners). Instead, each recycling/rescaling specification for a given no-slip surface is assumed to be independent (i.e., spatially removed from) all other surfaces that utilize recycling/rescaling.

NOTE: The formation of resolved fluctuations during the recycling/rescaling process can be greatly accelerated if realistic eddy structures are added to the mean velocity field. The **fluct_ext** utility (see Chapter 25) can be used for this purpose. This utility creates a "box" of velocity fluctuations that is written to the file fluct.dat. Placing this file into a subdirectory named **Recycle_files** in the working directory of the VULCAN-CFD simulation will flag the solver to superimpose these fluctuations on the velocity field present in the restart files. This addition of fluctuations only occurs during the initialization of the recycling/rescaling process, and will be skipped if VULCAN-CFD detects that the recycling/rescaling process has previously been initialized.

8.19 Structured Grid Adaptation Control Data

VULCAN-CFD has the capability to perform several classes of grid adaptation for structured grid simulations provided that the block topology satisfies a minimal set of requirements. The classes of grid adaptation supported include:

- Vehicle outer bow shock grid adaptation
- Vehicle surface boundary layer grid adaptation

Both classes only move grid lines (i.e., new grid lines are not created or removed) subject to the requirements described below:

(1) The flow feature being adapted to (bow shock and/or boundary layer) must be completely confined within a single layer of grid blocks that contain the farfield boundary for bow shock adaptation, and/or the surface boundary for boundary layer adaptation. The term "layer" in this context refers to the direction normal to the relevant boundary (i.e, the farfield boundary for bow shock adaptation, or the surface boundary for boundary layer adaptation).

(2) All of the blocks within each layer must contain the same number of cells in the direction normal to the relevant boundary.

(3) Bow shock adaptation, if enabled, is performed along all boundaries with a **REF_IN** condition, so this boundary condition can not be used for any boundary that is not a farfield boundary.

(4) Boundary layer adaptation, if enabled, is performed along **all** surfaces that use a no-slip, integrate-to-wall boundary condition (i.e., **IWALL**, **AWALL**, **TCWALL**, or **REWALL**).

Several images that illustrate allowable block topologies for grid adaptation are given in Figs. 11 and 12. Figure 11 shows the grid topology for a simple blunt body configuration. This grid was generated with a single layer of cells between the freestream boundary and the surface (i.e., block interfaces are only present normal to these boundaries), which satisfies the requirements for both shock and surface grid adaptation. The image shown on the left shows the grid at the start of the simulation. This particular case was previously adapted to a similar freestream condition (so the outer boundary already conforms well to the bow shock shape), but the aggressive shock clustering used will not match the shock location at the new freestream condition. The image on the right shows the final adapted grid at the new inflow conditions. Less aggressive grid clustering has been applied for the new shock position, which now resides much closer to the freestream boundary. A more complicated scenario is presented in Fig. 12. The image on the left shows the block topology generated for a capsule module on reentry. The outer block (red mesh) was sized such that the anticipated steady-state shock position resides within this grid block, which satisfies the requirement for bow shock adaptation. The boundary layer is completely contained within the blue mesh that surrounds the capsule, so this topology could also support boundary layer mesh adaptation if desired. The final shock adapted mesh is shown on the right, illustrating how the freestream boundary has moved to conform with the shock shape. The various input options that control the adaptation process are described below:

USE STR GRID ADAPTION #.#
This input line determines if grid adaptation is to be invoked for structured grid simulations based on the value specified:

0.0 = disables grid adaptation (this is the default)
1.0 = enables grid adaptation

NOTE: The **USE STR GRID ADAPTION** line indicates the start of the structured grid adaptation section, but the subsequent input parameters can be specified in any order.

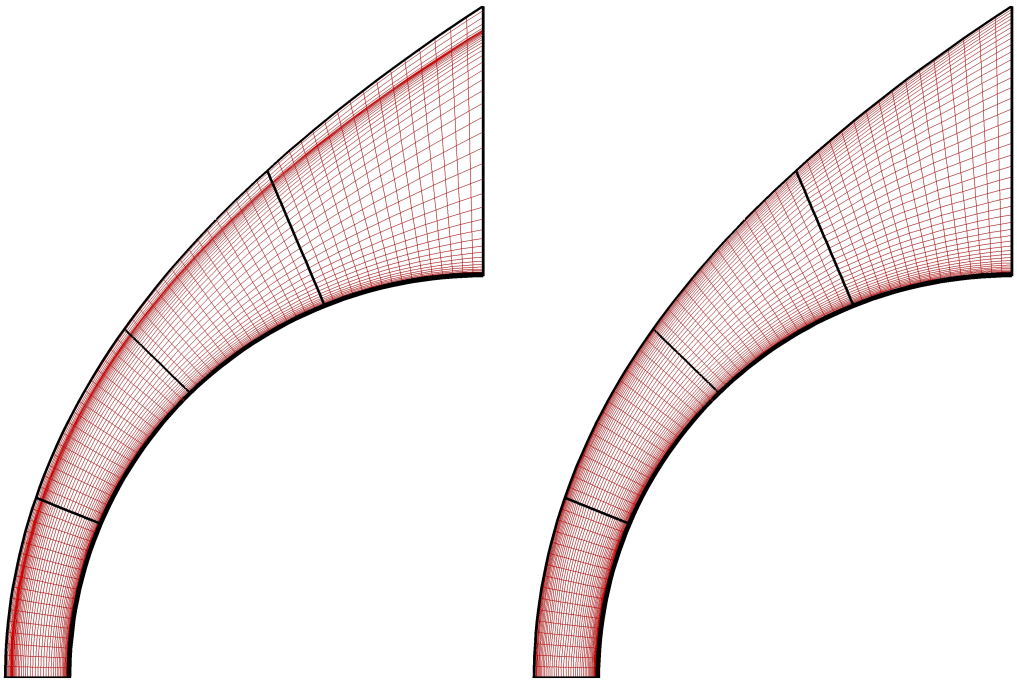


Figure 11: Grid topology for the blunt body: initial grid (left), final adapted grid (right).

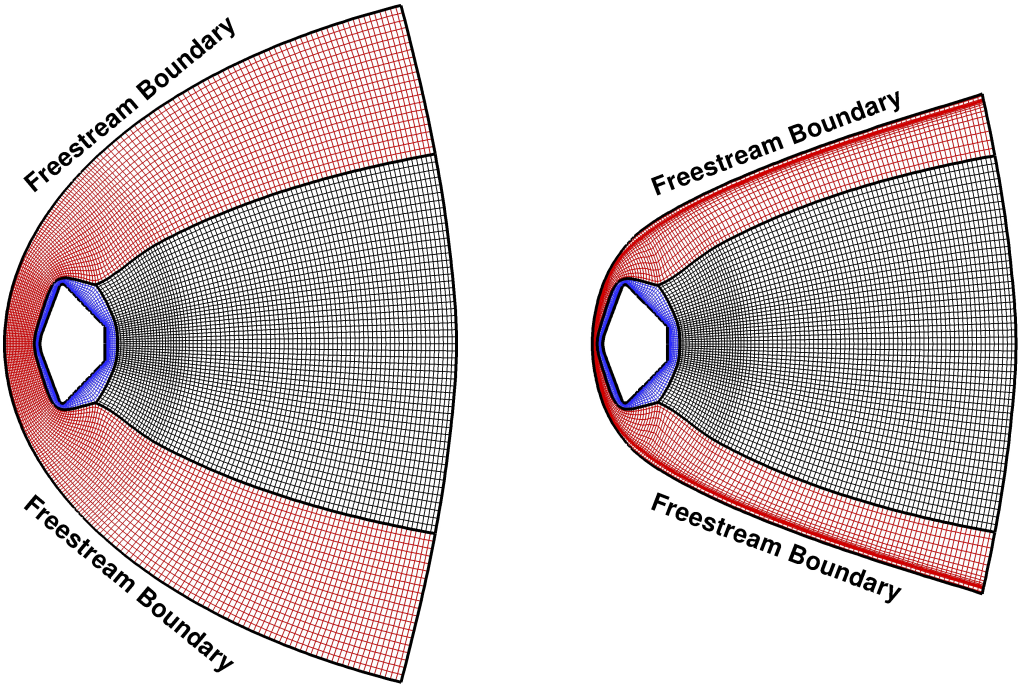


Figure 12: Grid topology for the capsule: initial grid (left), final adapted grid (right).

ADAPTATION ON ITERATION #.#
The grid adaptation should not take place until the flowfield structure being adapted to (bow shock and/or boundary layer) has formed to a sufficient level. This input line specifies the iteration cycle number to start the grid adaptation.

ADAPTATION NUMBER #.#
This input line specifies the number of grid adaptations to perform. A minimum of 3 grid adaptations is required, so the value specified here must be no smaller than 3.0.

ADAPTATION OFF ITERATION #.#
This input line specifies the iteration cycle number to terminate the adaptation process. The value entered here should be large enough to allow at least 3 grid adaptations to take place.

NOTE: Either the **ADAPTATION NUMBER** or **ADAPTATION OFF ITERATION** option should be selected to control when to halt the grid adaptation process. Specifying both of these inputs will be flagged as an error.

ADAPTATION FREQUENCY #.#
This input line specifies the grid adaptation iteration cycle frequency (i.e., the number of iterations between adaptation steps). The number of iterations provided should be large enough to allow the L2-norm of the residual error to drop several orders of magnitude (typically at least 4) between successive grid adaptations.

ADAPTATION L2 STOP #.#
This input line specifies the orders of magnitude drop in the L2-norm of the residual error that should be satisfied between adaptation cycles. If this criteria is satisfied before the next specified grid adaptation cycle is to be performed, then the adaptation is performed and the remainder of the adaptation schedule is shifted accordingly. Typically, at least 4 orders of magnitude drop in the L2-norm of the residual error should be requested, and at least 2 orders is required.

SHOCK ADAPTATION #.#
This input line controls whether bow shock grid adaptation is to be enabled. This adaptation option not only allows the grid to be adapted to the outer bow shock formed around the body, but it can also align the farfield boundary to the bow shock.

0.0 = disables bow shock grid adaptation
1.0 = bow shock grid adaptation/alignment based on a shock jump threshold

SHOCK DETECTION COEF #.#
This input line specifies the pressure jump threshold used to detect the bow shock. The value entered should be close to (but larger than) unity.

SPACING AT SHOCK #.#
This input parameter defines the level of grid clustering performed at the shock surface. A value of 1.0 will attempt to keep the grid spacing nominally constant between the freestream

boundary and the shock surface. Values less than 1.0 will cluster the grid toward the shock surface, and values greater than 1.0 will cluster the grid away from the shock surface (the further the value is from unity, the more aggressive the stretching).

FREESTREAM BOUNDARY #.#
This input line controls the freestream boundary movement during the shock adaptation process based on the value entered for this parameter:

0.0 = keeps the freestream boundary fixed

1.0 = aligns the freestream boundary to the bow shock based on a geometric projection of the shock surface

$1.0 \leftrightarrow 1.25$ = aligns the freestream boundary to the bow shock based on a shock stand-off scaling (larger values result in a larger stand-off distance)

NOTE: The geometric projection of the shock surface option typically places more stringent requirements on the grid adaptation process. As a result, if this option is selected, then at least 5 adaptations of the grid are required instead of 3.

SHOCK SMOOTHING COEF #.#
This input parameter defines the coefficient used to smooth the detected shock surface. The value entered must lie between 0.0 and 0.5 (larger values provide more smoothing, and 0.0 disables the smoothing).

SHOCK TO FS BUFFER #.#
This input line specifies the number of freestream grid lines that lie between the freestream boundary and the shock surface. The default value if 5.

WALL ADAPTATION #.#
This input line specifies the particular strategy used for boundary layer grid adaptation.

0.0 = disables surface grid adaptation
1.0 = adapts the grid to the surface by scaling the original wall spacing
2.0 = adapts the grid to the surface using a specified wall spacing
3.0 = N/A (reserved for future use)
4.0 = adapts the grid to the surface based on a user-specified cell Reynolds number
5.0 = adapts the grid to the surface based on a user-specified cell center y^+ value

WALL PARAMETER #.#
This input line defines how the wall spacing is determined. The form of the value entered here is dependent on the **WALL ADAPTATION** option chosen:

If the chosen **WALL ADAPTATION** option was 1.0, then the value entered here is the desired scaling factor for scaling the wall spacing relative to the original wall spacing.

If the chosen **WALL ADAPTATION** option was 2.0, then the value entered here is the desired constant wall spacing value. The units for the grid spacing must be consistent with the units provided in the grid file. For example, if the grid coordinates are supplied in units of inches, then the spacing entered here must also be in inches.

If the chosen **WALL ADAPTATION** option was 4.0, then the value entered here is the desired surface cell Reynolds number to maintain.

If the chosen **WALL ADAPTATION** option was 5.0, then the value entered here is the desired surface cell center y^+ value to maintain.

WALL SMOOTHING COEF #.#
This input parameter defines the coefficient used to smooth the wall spacing. The value entered must lie between 0.0 and 0.5 (larger values provide more smoothing, and 0.0 disables the smoothing).

8.20 Block, Subblock, and Block Boundary Specifications

CURV BLOCKS #.#
This input line specifies the number of structured curvilinear blocks in the computational grid.

NOTE: Given the legacy of VULCAN-CFD as a structured grid code, this input can also be invoked by using the string **BLOCKS** for backward compatibility.

NOTE: The addition of "CURV" to denote generic structured curvilinear blocks has been added to eventually allow Cartesian structured grid blocks to be included as a separate structured grid type.

UNST BLOCKS #.#
This input line specifies the number of unstructured blocks (partitions) in the computational grid.

BCGROUPS #.#
This input line provides the number of boundary condition settings used to set the specific boundary conditions for the simulation. A full description of the input format for the boundary condition grouping lines is provided in Chapter 9. This chapter also contains the complete list (and description) of the boundary conditions available in VULCAN-CFD.

BCOBJECTS #.#
This input line provides the number of boundary condition objects (defined as a collection of boundary condition groups). A full description of the input format for the boundary condition object lines is provided in Chapter 9.

INIT. SUB-DOMAINS #.#
This input line provides the number of user-specified flowfield initialization volumes used

to tailor the initial state of the simulation. A full description of the input data and format required by each available volume shape is provided in Chapter 10.

LAMINAR SUB-DOMAINS #.#
This input line provides the number of user-specified turbulence suppression volumes used to delay transition from laminar to turbulent flow. A full description of the input data and format required by each available volume shape is provided in Chapter 11.

IGNITION SUB-DOMAINS #.#
This input line provides the number of user-specified ignition source volumes intended to promote ignition for reacting flow simulations. A full description of the input data and format required by each available volume shape is provided in Chapter 12.

TIME SUB-DOMAINS #.#
This input line provides the number of user-specified time history output specifications for unstructured grid simulations. A full description of the input data and format required by each time history specification class is provided in Chapter 13. The output of time history files for unstructured grid simulations requires that T-infinity be enabled.

NOTE: The output of time history files for structured grid simulations is controlled by the **TIME HISTORY I/O** input line described below.

FLOWBCS #.#
This input line provides the number of structured grid boundary condition specification lines that are present in the structured grid boundary condition specification section of the input file (see Chapter 14 for further details).

CUTBCS #.#
This input line provides the number of structured grid block-to-block C(0) connectivity specification lines that are present in the structured grid interface connectivity specification section of the input file (see Chapter 15 for further details).

PATCHBCS #.#
This input line provides the number of structured block-to-block non-C(0) (or patched) connectivity specification lines that are present in the structured grid patched interface connectivity specification section of the input file (see Chapter 16 for further details).

PATCH FILE
This input specifies the name of the file created by the patch preprocessing step that contains the patch interpolation coefficients (including the full path if desired). The file name must be provided on the next line.

NOTE: If this line is not present, and if patched block interfaces are present, then the patch interpolation coefficient file will be written to **patch.dat**.

LAMINAR SUB-BLOCKS #.#
This input provides the number of structured grid turbulence suppression line specifications that are present in the structured grid turbulence suppression specification section of the input file (see Chapter 17 for further details).

IGNITION SUB-BLOCKS #.#
This input provides the number of structured grid ignition line specifications that are present in the structured grid ignition specification section of the input file (see Chapter 18 for further details).

TIME HISTORY I/O #.#
This input provides the number of structured grid time history output line specifications that are present in the structured grid time history output specification section of the input file (see Chapter 19 for further details).

BLOCK CONFIG #.#
This input line provides the number of block (or partition) configuration lines that are present. A block configuration header line and the block configuration line specifications must immediately follow this line, e.g.,

BLOCK CONFIG 3.0 (no. of block configuration lines)

BLK	VISC (N, T, or F)	TURB	REAC	REGION
0	F	Y		1
1	F	N		1
2	F	N		1

The block configuration column entries are described below:

BLK : Specifies the block number
0 → denotes all blocks
>0 → denotes a specific block number

NOTE: If most of the blocks are to be configured in a similar fashion with only a few outliers, then a value of 0 can be used to set a default configuration followed by additional configuration lines that override the default setting (the example above depicts this usage).

VISC : Specifies how the viscous terms are treated
N → disables the viscous terms
T → invokes a thin layer approximation to the viscous terms (structured grid blocks only)
F → invokes all viscous terms

TURB : Enables/Disables the turbulence model
N → disables the turbulence model
Y → enables the turbulence model

NOTE: This entry must be left blank if performing an Euler or laminar flow simulation

REAC : Enables/Disables the chemistry model
N → disables the chemistry model
Y → enables the chemistry model

NOTE: This entry must be left blank if performing a nonreacting flow simulation

REGION : Specifies the region number that the block belongs to

8.21 Region Control Specification

VULCAN-CFD permits the decomposition of the computational domain (at the block level) into disparate "regions" that can be solved sequentially. This decomposition is possible for many supersonic applications, where the upstream flowfield is not affected by downstream flow conditions. In VULCAN-CFD, a region is defined as a collection of grid blocks (or partitions) that are to be solved together using a common set of solver attributes. The number of regions to be configured, and the individual blocks that make up each region, are provided in the block specification section discussed above. The ordering of the regions is determined by the sequential arrangement of each region configuration section. Each region specification section involves up to 7 (elliptic regions) or 8 (parabolic regions) rows of information, where each row consists of a header line followed by the region specification line data. Example elliptic and parabolic region configuration specifications are given below:

```
$———————————— Example Inactive Space Marching Region ————————————$
SOLVER/STATUS
        M/I
VAR-REC   LIMITER   LIM COEF   FLUX SCHEME   ENT FIX (U)   ENT FIX (U+a)
     3          2        0.0          LDFSS           0.0             0.0
SM-ORD  BEG:END  MG BEG:END  VIG COEF  MEAN-LIM  TURB-LIM  SUB-STEP
     2     001  064     001  005      0.95       1.0       1.0         N
FMG-LVLS   NITSF   1ST-ORD SWITCH   REL RES   ABS RES   DQ COEF
       1      500             0        -6.0      -8.0      0.25
TURB CONV   DT RATIO   NON-EQ   POINT IMP   COMP MODEL   CG WALL BC
      2ND         1.0     20.0           Y            N          WMF
SCHEME  T-STEP  STATS  MIN-CFL  VAR-CFL  # CFL  VIS-DT  IMP-BC  REG-RES
   DAF   LOCAL     10      1.0        Y      2       Y       N        N
   1     5
   0.1   1.0
   5.0   5.0

$———————————— Example Active Elliptic Region ————————————$
SOLVER/STATUS
        E/A
VAR-REC   LIMITER   LIM COEF   FLUX SCHEME   ENT FIX (U)   ENT FIX (U+a)
     3          2        0.0          LDFSS           0.0             0.0
FMG-LVLS   NITSC   NITSF   1ST-ORD SWITCH   REL RES   ABS RES   DQ COEF
```

2	1500	5000		-1	-6.0	-8.0	0.25
TURB CONV		DT RATIO	NON-EQ	POINT IMP	COMP MODEL		CG WALL BC
2ND		1.0	20.0	Y	N		WMF
SCHEME	T-STEP	STATS	MIN-CFL	VAR-CFL	# CFL	VIS-DT	IMP-BC	REG-RES
DAF	LOCAL	10	0.5	Y	5	Y	N	N
1	500	1500	1501	2000				
0.5	3.0	3.0	0.5	3.0				

A complete description of each region configuration line will now be presented along with some guidance on recommended settings for certain classes of simulations.

The first row, labeled above as,

SOLVER/STATUS

defines the region solver type (elliptic or parabolic) and status (active or inactive). This specification consists of two parts separated by a slash. The first part designates the solver type, and the second part designates whether the region is active or inactive, e.g.,

E/A → elliptic/active
E/I → elliptic/inactive
M/A → parabolic/active
M/I → parabolic/inactive

NOTE: Elliptic regions can consist of either structured or unstructured blocks (partitions). Parabolic (i.e., space-marching) regions, however, must adhere to the following conditions:

- Parabolic regions must be comprised entirely of structured grid blocks with the streamwise (i.e., space-marching) direction aligned with the i-coordinate.
- Blocks that reside in a parabolic region must not interface in the space-marching direction with other blocks from the same region. In other words, any i-direction block-to-block interfaces must also be region-to-region interfaces.
- All connected blocks within a parabolic region must start from a common space-marched plane. However, they do not have to all end at the same space-marched plane.

The next row sets the parameters that control the convective flux scheme.

VAR-REC LIMITER LIM COEF FLUX SCHEME ENT FIX (U) ENT FIX (U+a)

VAR-REC defines the approach used to reconstruct the flow variables at the cell interfaces from the cell center values based on one of the following methods:

- Monotone Upwind-biased Scheme for Conservation Laws (MUSCL)[61]
- Partially Parabolic Method (PPM) of Colella and Woodward[62]
- Weighted Essentially NonOscillatory (WENO) schemes of Carpenter and Fisher[63]

- Linearity Preserving Unstructured MUSCL (LP-UMUSCL)[64]

where

1 = 1^{st}-order upwind scheme
2 = 2^{nd}-order fully upwind scheme (MUSCL/LP-UMUSCL parameter, κ=−1)
3 = 3^{rd}-order upwind-biased scheme (MUSCL/LP-UMUSCL parameter, $\kappa=\frac{1}{3}$)
4 = 2^{nd}-order upwind-biased Fromme scheme (MUSCL/LP-UMUSCL parameter, κ=0)
5 = 3^{rd}-order upwind-biased Burg scheme (if invoked for unstructured grids) **or**
5 = Mixed $3^{rd}/4^{th}$-order PPM scheme (if invoked for structured grids)
6 = 4^{th}-order WENO 242 scheme
7 = 6^{th}-order WENO 363 scheme

NOTE: The MUSCL, PPM, and WENO schemes are intended for structured grid simulations and the LP-UMUSCL scheme only applies to unstructured grid simulations.

LIMITER defines the functional form of the limiter used to reconstruct the variables at the cell interfaces. The following are the limiter options if an LP-UMUSCL reconstruction method is chosen for an unstructured grid simulation:

0 = none
1 = MLP implementation of the MINMOD limiter
2 = MLP implementation of the van Leer limiter
3 = MLP implementation of the van Albada limiter
4 = MLP implementation of the Venkatakrishnan smooth limiter
5 = MLP implementation of the Koren limiter
6 = MLP implementation of the Nishikawa 6^{th}-order polynomial smooth limiter

NOTE: A less dissipative (but less robust) stencil-based version of the above MLP (Multi-dimensional Limiting Process) options is invoked by negating the MLP option.

The following are the limiter options if a MUSCL reconstruction scheme is chosen for a structured grid simulation:

0 = none
1 = Sweby β TVD limiter family (MINMOD $\rightarrow \beta = 1.0$, SUPERBEE $\rightarrow \beta = 2.0$)
2 = van Leer TVD limiter
3 = van Albada TVD limiter
4 = Venkatakrishnan smooth limiter (valid for $\kappa=\frac{1}{3}$ and κ=0 only)
5 = Koren TVD limiter
6 = UNO limiter (ENO version of the MINMOD limiter)
7 = SONIC-A limiter (ENO version of the van Leer limiter)
8 = SONIC-B limiter (ENO version of the SUPERBEE limiter)

Finally, if the PPM reconstruction scheme is chosen for a structured grid simulation, then the limiter options are:

0 = pure 4^{th}-order symmetric reconstruction
1 = standard PPM scheme
2 = PPM scheme with additional limiting for strong shocks

There are currently no user-supplied limiter options for the WENO schemes.

LIM COEF defines a coefficient needed by some of the limiters that are available for structured grid simulations, **or** activates an enhancement to the limiters used for unstructured grid simulations.

For structured grid simulations that invoke the Sweby limiter, the value entered here is the Sweby β coefficient, which can take on any value between 1.0 and $(3 - \kappa)/(1 - \kappa)$. For structured grid simulations that invoke the Venkatakrishnan limiter, the value entered here is the scaling coefficient used by the cell length parameter associated with this limiter. Allowable values for the scaling coefficient range between 1×10^{-3} and 1×10^{8}, with larger values providing less limiting. A typical value taken for this parameter is 2.0.

For unstructured grid simulations, the value entered here can provide additional limiting based on the local pressure gradient:

0.0 > value >= -1.0 invokes a 2-stage pressure limiter that is sensitive to compressive gradients only (less-dissipative method)
0.0 < value <= 1.0 invokes a 2-stage pressure limiter that is activated by both compression and expansion processes (more-dissipative method)

NOTE: A value of 0.0 disables the additional limiting, while values of ±1.0 maximizes the limiting effect.

FLUX SCHEME defines the inviscid flux scheme from the following options:

ROE	:	Roe flux difference scheme[65]
HLLC	:	Harten, Lax, van Leer, with Contact surface flux scheme[66]
LDFSS	:	Edwards Low Dissipation Flux Split Scheme (LDFSS[2])[17]
PRECOND	:	Edwards low Mach preconditioned flux split scheme[67]
NO FLUX	:	No flux evaluation (useful for 0-D chemical kinetics)

NOTE: The Roe scheme and Edwards low Mach preconditioned flux split scheme are only available for structured grid simulations.

ENT FIX (U) controls various strategies to address potential instabilities that result when the convective eigenvalue approaches zero. The particular strategy that is invoked depends on the specific inviscid flux scheme chosen:

If either the LDFSS or the low Mach preconditioned flux split scheme is chosen, then this entry controls the blend between the LDFSS[2] and LDFSS[0] variants:

0.0 = invokes the Edwards LDFSS[2]
1.0 = blends LDFSS[2] with a more dissipative LDFSS[1] variant

If the HLLC scheme is chosen, then this entry controls the blend between the HLLC and the HLL variant:

0.0 = invokes the standard HLLC
1.0 = blends HLLC with the HLL variant

If the Roe scheme is chosen, then a value between 0.0 and -0.5 invokes a fixed coefficient entropy fix aimed at alleviating shock / grid alignment (carbuncle) problems, while a value between 0.0 and 1.0 invokes a variable coefficient variant controlled by the pressure gradient.

ENT FIX (U+a) controls various strategies to address potential instabilities that result when the acoustic eigenvalue approaches zero. The particular strategy that is invoked depends on the specific inviscid flux scheme chosen:

If the LDFSS scheme is chosen, then this entry controls the switch between the LDFSS[2] and LDFSS[0] variants:

0.0 = invokes the Edwards LDFSS[2]
1.0 = blends LDFSS[2] with a more dissipative LDFSS[1] variant

If the HLLC scheme is chosen, then a value between 0.0 and 0.5 invokes a fixed coefficient entropy fix aimed at the prevention of expansion shocks.

If the Roe scheme is chosen, then a value between 0.0 and -0.5 invokes a fixed coefficient entropy fix aimed at the prevention of expansion shocks, while a value between 0.0 and 1.0 invokes a variable coefficient variant.

If the Edwards low Mach preconditioned flux split scheme is chosen, then the value entered has nothing to do with entropy fixes. Instead, this value controls how to set the minimum velocity limit utilized by the low-velocity preconditioning scheme:

- A value between 0.0 and 1.0×10^3 uses a constant coefficient minimum velocity limiter, where the value entered is a scalar multiplier for the square of the reference velocity value. This multiplier should be chosen such that the resulting minimum velocity is several orders of magnitude smaller than any expected streamwise velocity value outside of boundary layers.
- A value between -1.0 and 0.0 invokes the Darmofal variable coefficient minimum velocity limiter. The magnitude of the value entered should be chosen such that when multiplied by the reference velocity, the result is a velocity that is several orders of magnitude smaller than any expected streamwise velocity value outside of boundary layers.

Finally, there is an optional 7th column that is only relevant for multiregion simulations when a hybrid advection scheme has been specified. This column determines whether the hybrid advection scheme is active in the region:

HYB-ADV enables (**Y**) or disables (**N**) the hybrid advection scheme.

The following items should be considered when choosing an appropriate convective flux scheme:

- In general, the LDFSS and HLLC schemes are typically preferred over the ROE flux difference split scheme. These schemes were formulated to address some of the short-comings of the Roe scheme, and are less likely to require additional entropy fixes.
- For steady-state Reynolds-averaged simulations, a good starting point is the selection of the MUSCL scheme (κ=1/3) with the van Leer TVD limiter. If the convergence stalls prematurely due to "limiter buzz", the Venkatakrishnan smooth limiter can sometimes offer some improvement. However, keep in mind that this limiter does not have the TVD property, and may not perform well if strong shocks are present. The Koren TVD limiter tends to provide slightly better compressible flow feature resolution than the van Leer limiter, but it is also more susceptible to "limiter buzz".
- For scale-resolving simulations, a good starting point is the use of a hybrid advection scheme with the Ducros sensor chosen to blend the upwind reconstruction with a 4^{th}-order symmetric reconstruction. For the upwind reconstruction, the MUSCL (κ=1/3) scheme with the UNO limiter is recommended. This particular ENO limiter is less dissipative than the TVD limiters, and tends to resolve shock waves with minimal overshoots/undershoots. However, the standard PPM scheme is also a reasonable option. If improved eddy resolution is desired, then consider replacing the Ducros sensor with the **HYBRID ADVECTION SENSOR** option 4.0 to activate the nondissipative operator more aggressively in viscous dominated regions of the computational domain. If instead robustness issues are encountered with the Ducros blending, then either activate one of the hybrid advection scheme sensor lower limits, or deactivate the hybrid advection scheme altogether and select one of the WENO convective schemes instead. The WENO schemes offer the least amount of dissipation (as compared to MUSCL and PPM), but the computational cost is typically not warranted when used within the hybrid advection scheme framework.

The row labeled above as:

SM-ORD BEG:END MG BEG:END VIG COEF MEAN-LIM TURB-LIM SUB-STEP

is specific to space marching, and should only be present in parabolic regions. This row provides the specific information controlling the space marching advancement, and the header of this line **must** begin with the string SM.

SM-ORD defines the order of accuracy for the space marching advancement:

1 = 1^{st}-order upwind differencing
2 = 2^{nd}-order upwind differencing

BEG:END defines the extent (start and end) of the space marching advancement. The starting value must either be 1 (if the region has not yet been solved) or 1 greater than the last solved (and converged) plane if the region was previously partially solved. Any ending value greater than or equal to the number of cells in the marching (i.e., i-direction) will solve the region in its entirety.

MG BEG:END defines the range of marching planes that use coarse-to-fine sequencing, as well as any 1^{st}-order specifications that have been provided. Any ending value greater than or equal to the number of cells in the marching (i.e., i-direction) will use coarse-to-fine sequencing and enforce the 1^{st}-order specifications for every plane in the region.

VIG COEF defines the Vigneron coefficient, which is essentially a safety factor for the streamwise parabolic operator. Valid values vary between 0.9 and 0.99. A typical value used in practice is 0.95, but flows with substantial adverse pressure gradients in the marching direction may require lower values for this coefficient.

MEAN-LIM controls the level of streamwise limiting performed for all the equations except the turbulent transport equations, which are controlled separately. Valid values range from 1.0 to ∞ (smaller values imply more limiting).

TURB-LIM controls the level of streamwise limiting performed for the turbulent transport equations. Valid values range from 1.0 to ∞ (smaller values imply more limiting).

SUB-STEP enables (or disables) the use of substepping to refine the marching direction grid resolution of the simulation. If enabled, VULCAN-CFD will subdivide the input grid cells in the marching direction according to a substep schedule provided after the CFL schedule near the end of the region specification control section. Substepping can also be used to provide the additional resolution to allow a 1^{st}-order treatment of the space marching direction that recovers the accuracy of a 2^{nd}-order method without substepping. One reason to go this route is if overshoots/undershoots cannot be eliminated with the streamwise limiter options described above. Valid options are:

N $\rightarrow$ no substepping schedule will be provided, so use the streamwise grid spacing
M $\rightarrow$ a manual substepping schedule will be provided with the CFL schedule

The conditions required to invoke the substepping feature are as follows:

(1) The first plane of any region that has a substep schedule should either not be connected to blocks in any other region, or be connected to blocks that reside in the region that immediately preceded it.

(2) If the first plane of any region that has a substep schedule interfaces with blocks that reside in more than one previously solved region, then:

- One of the interface blocks must reside in the region that immediately precedes the current substepped region.
- The last marching plane in the upstream regions that contain an interface block should all have the same number of substeps.

The following items should be considered when choosing options related to space marching:

- Any coarse-to-fine sequencing performed (i.e., **MG BEG:END**) should typically be limited to only the first couple of planes, since the converged solution from the previous plane is already a very good initial solution for the current plane.
- If convergence issues arise when using 2^{nd}-order upwind differencing in the marching direction, or if overshoots/undershoots cannot be overcome with the simple streamwise limiter options, then consider switching to a 1^{st}-order upwind scheme with the substepping enabled. To leading order, a 1^{st}-order upwind treatment with 4 substeps per marching plane should roughly reproduce the accuracy of a 2^{nd}-order upwind treatment without substepping.

The row labeled above as:

FMG-LVLS NITSC NITSF 1ST-ORD SWITCH REL RES ABS RES DQ COEF

defines the number of full multigrid levels, the number of iterations to perform on each grid level (coarse-to-fine sequencing), any 1^{st}-order specifications, and the convergence criteria. The header of this line **must** begin with the string FMG, and the last entry must contain the string DQ.

FMG-LVLS defines the number of grid levels to be used in coarse-to-fine sequencing of the solver (the maximum number of grid levels is limited to 5). If a multigrid algorithm is also to be employed, then this value should be specified as a negative number to indicate that an additional row of data will be specified to control the multigrid process.

NOTE: The number of grid levels must be set to 1 for unstructured grid simulations.

NITSC ... NITSF sets the number of iteration cycles to be executed on each grid level. The number of iteration values provided must match the number of grid levels (i.e., FMG_LVLS) specified. The iteration cycles must be supplied in ascending order where the first is on the coarsest grid and the last on the finest grid.

1ST-ORD SWITCH defines the point, designated either by iteration number or by grid level, at which to change from a first-order scheme to a higher-order scheme. Negative values designate grid levels, positive values designate iteration numbers. For example, if there are 3 grid levels (fine grid [level 3], medium grid [level 2] and coarse grid [level 1]), then specifying:

-1 $\rightarrow$ switches to higher-order upon completion of the coarse grid iterations
-2 $\rightarrow$ switches to higher-order upon completion of the medium grid iterations

REL RES specifies the relative residual error L2-norm criteria for halting execution on the current plane (space marching regions) or the entire region (elliptic regions). The numerical value specified should represent the criteria in terms of orders of magnitude (e.g., a value of 6 implies that the execution will be halted after the L2-norm of the residual error has been reduced by at least 6 orders of magnitude relative to the value at the first iteration). The number can be entered as a positive or negative value (the sign is ignored), with a value of 0.0 disabling the criteria. Disabling the criteria can be beneficial for time-accurate simulations where the desire is to perform the simulation for a desired time frame regardless of the solution state.

ABS RES specifies the absolute residual error L2-norm criteria for halting execution on the current plane (space marching regions) or the entire region (elliptic regions). The numerical value specified should represent the criteria in terms of orders of magnitude (e.g., a value of 6 implies that the execution will be halted after the L2-norm of the residual error has dropped to a value of 10^{-6}). The number can be entered as a positive or negative value (the sign is ignored), with a value of 0.0 disabling the criteria. Disabling the criteria can be beneficial for time-accurate simulations where the desire is to perform the simulation for a desired time frame regardless of the solution state.

NOTE: Execution will halt when either of these convergence criteria are satisfied, or if all of the specified iterations have been performed.

NOTE: If either of these convergence criteria is disabled, then both of the residual norm exit criteria will be disabled (i.e., all of the specified iterations will be performed).

Consider the following when choosing the number of coarse-to-fine grid sequence levels to consider and the 1^{st}-order switch option:

- Coarse-to-fine sequencing is only available for structured grid simulations. The number of grid levels, n, that can be supported is determined by finding the largest n that returns an integer when substituted into the expression $nc/2^n$. Here, nc is the number of cells in any given i, j, or k direction for each block. The smallest value of n, after considering each coordinate direction for every block, determines the number of grid levels that can be supported by the grid. The utility **grid_plot3d** can be used to extract this information from a given PLOT3D format grid file.

- If using the structured grid partitioner, be sure to specify the desired number of grid levels that must be supported in the **grid_split.inp** file prior to partitioning the grid.

- The goal of coarse-to-fine sequencing is simply to flush out initial condition transients as rapidly as possible, so there is no reason to converge the solution too far on the coarse grid levels (unless grid resolution studies are being performed). In practice, it is usually sufficient to switch grid levels as soon as the L2-norm of the residual error

has dropped 2-3 orders of magnitude from its initial value. A similar argument can be made when determining the number of 1^{st}-order iteration cycles to perform.

- If the 1^{st}-order switch is provided based on iteration cycles, and if the number of initial cycles to be performed 1^{st} order is not known a priori, then set the 1^{st}-order switch to a very large value. Once the decision is made to switch to the higher-order scheme (based on monitoring the solution convergence history), the 1^{st}-order switch can then be reset to zero prior to restarting the simulation. VULCAN-CFD will recognize this as a desire to immediately switch to the higher-order scheme.

DQ COEF specifies the maximum allowable change in the transported variables after each iteration cycle as a fraction of the current value of the transported variable. Valid values range between 0.01 and 1.0.

The row labeled above as:

MG CYCLE CRS LVLS DQ SMTH DQ COEF DAMP MEAN DAMP TURB

controls the multigrid cycle scheme, total number of coarse grid levels to use for the multigrid cycle, and various parameters that control the multigrid process. This row should only be present if the **FMG-LVLS** value was entered as a negative number.

MG CYCLE sets the multigrid cycle scheme

I → each grid level is solved independently (i.e., no multigrid cycling)
V → utilizes a V cycle multigrid algorithm
W → utilizes a W cycle multigrid algorithm

NOTE: Multigrid cycling is currently not available for unstructured grid simulations.

CRS LVLS specifies the number of coarse grids to be used for the multigrid cycle process. This value cannot be smaller than the **FMG-LVLS** value (specified on the previous line) minus 1.

DQ SMTH specifies the coefficient used for implicit smoothing of coarse grid corrections when using the V cycle or W cycle options. Valid values vary between 0.0 and 0.5. Larger values provide more smoothing, and 0.0 disables the smoothing operation.

DQ COEF specifies the maximum allowable coarse grid correction (used for multigrid cycling) as a fraction of the current value of the transported variable. Note that this parameter constrains the coarse grid correction for multigrid cycling, while the **DQ COEF** entered on the FMG-LVLS line constrains the dependent variable update between iteration cycles. Valid values range between 0.01 and 1.0.

DAMP MEAN provides the minimum allowable coefficient used to damp the multigrid forcing function for all equations except the turbulent transport equations. Valid values

range between 0.1 and 1.0 (smaller values provide more damping).

DAMP TURB provides the minimum allowable coefficient used to damp the multigrid forcing function for the turbulent transport equations. Valid values range between 0.1 and 1.0 (smaller values provide more damping).

Consider the following when deciding whether to activate one of the multigrid algorithms:

- The multigrid functionality is only available for structured grid simulations.
- V-cycle and W-cycle multigrid tend to provide the most benefit for subsonic applications. The speedup (if any) is typically marginal for supersonic flow applications.

The row labeled above as:

TURB CONV DT RATIO NON-EQ POINT IMP COMP MODEL CG WALL BC

controls various parameters related to the turbulence transport equations.

TURB CONV defines the spatial order of accuracy used for the convective terms of the turbulence equations.

1ST $\rightarrow$ 1^{st}-order treatment
2ND $\rightarrow$ 2^{nd}-order treatment

DT RATIO Sets the ratio of the time step used to update the turbulent transport equations relative to the time step used for the rest of the equation set. Valid values range between 0.1 and 1.0.

NON-EQ Specifies the maximum allowable degree of nonequilibrium behavior (defined as the production to destruction ratio) in the turbulence kinetic energy equation. This parameter is only active for turbulence models that contain a turbulence kinetic energy equation. A typical value used in the literature is 20.0. A negative value for this parameter invokes the smoothed modified Larsson hybrid advection sensor (see the **HYBRID ADVECTION SENSOR** input description) to null all source terms in the turbulence transport equations in the vicinity of shock waves.

POINT IMP Specifies whether the point implicit scheme is enabled for the turbulence source terms:

Y $\rightarrow$ enables the point implicit treatment
N $\rightarrow$ disables the point implicit treatment

NOTE: The point implicit treatment should in most cases be used to enhance simulation stability (particularly for turbulent wall-bounded flow simulations with the grid clustered so that y^+ is near unity).

COMP MODEL Specifies whether the compressibility correction[40,41] aimed at reducing the spreading rate of compressible free shear flows is enabled:

Y → enables the compressibility correction
N → disables the compressibility correction

NOTE: The compressibility correction should be invoked with caution if large boundary layer separations are present. This correction is known to corrupt attached boundary layer flow physics, and the control functions that disable this feature in the boundary layer may fail in separated flow regions.

CG WALL BC Specifies how the solve-to-wall surface boundaries are handled on coarse grid levels for turbulent flow simulations that utilize 2-equation models.

WMF → uses the wall function version of the surface condition specified (recommended)
STW → uses the solve-to-wall surface condition that was specified

NOTE: The wall function option for coarse grids cannot be invoked if any blocks have turbulence disabled or transition regions enabled.

Consider the following when selecting parameters that control turbulent transport:

- The use of a 1^{st}-order treatment is prevalent in the external flow CFD community, where viscous regions are dominated by attached boundary layers. Convection plays a more pronounced role for free shear layers, so a 2^{nd}-order treatment is advised for general purpose simulations.
- The point implicit treatment of the turbulence source terms should always be active unless an explicit scheme (e.g., Runge-Kutta) is used which requires time accuracy.
- Enable the use of wall matching functions on coarse grid levels to avoid instabilities that may arise if the surface cell y^+ values on the coarse grids get much higher than unity.

The last row labeled above as:

SCHEME T-STEP STATS MIN-CFL VAR-CFL # CFL VIS-DT IMP-BC REG-RES

specifies the time advancement and time stepping scheme, output frequency of iteration statistics, implicit boundary condition treatment, and restart file usage. Several additional input lines are required for some of the time advancement and time step options.

SCHEME specifies how the equations are updated in time (or pseudotime). The following options are available for structured grid simulations:

R-K → Runge-Kutta (the specific Runge-Kutta method is supplied in the global solver data section)
DAF → Diagonalized Approximate Factorization[68]

ILU → Incomplete LU(0)[69] (additional suboptions must be supplied in the implicit scheme control section described below)

and these options are available for unstructured grid simulations:

R-K → Runge-Kutta (the specific Runge-Kutta method is supplied in the global solver data section)
SGS → Symmetric Gauss-Seidel[70] (additional suboptions must be supplied in the implicit scheme control section described below)
SOR → Symmetric Gauss-Seidel[70] with relaxation applied to the final implicit sweep (additional suboptions must be supplied in the implicit scheme control section described below)
SSOR → Symmetric Gauss-Seidel[70] with relaxation applied on all of the implicit sweeps (additional suboptions must be supplied in the implicit scheme control section described below)

The implicit scheme control line for **ILU** has the following format:

SWEEP-DIR	JAC-UPDATE	START	NUM-SLVS	RELAX
0	0	1	3	0.9

The implicit scheme control line for the **SGS**, **SOR**, and **SSOR** schemes has a similar structure, but the sweep direction input is not present, e.g.,

SGS	JAC-UPDATE	START	NUM-SLVS	RELAX
	0	1	10	0.9

NOTE: The relaxation parameter is not used by the **SGS** scheme and can be omitted.

SWEEP-DIR specifies the sweep direction for the **ILU** scheme for 3-D elliptic simulations (this entry is ignored for 2-D/axisymmetric problems or when space marching):

0 = sweep for each block is aligned with the direction of the smallest spectral radius
1 = sweep direction is aligned with the structured grid i-coordinate
2 = sweep direction is aligned with the structured grid j-coordinate
3 = sweep direction is aligned with the structured grid k-coordinate
-# = full 3-D scheme (no sweeping direction)

JAC-UPDATE controls how often the Jacobians are updated. This parameter improves the efficiency of the implicit scheme by reducing the overhead associated with evaluating the Jacobians.

0 = updates the Jacobian less frequently as the simulation converges (recommended)
>0 = specified frequency for the Jacobian updates

START specifies the iteration number to start utilizing the update Jacobian schedule. The Jacobian is updated at every iteration cycle prior to this point (this input is ignored if the

automated algorithm is selected for JAC-UPDATE).

NUM-SLVS specifies the number of linear algebra solves to perform for the algebraic system of equations solved by the implicit scheme. This parameter reduces the factorization error associated with multidimensions and "broken" implicit operators resulting from block-to-block (or partition) interfaces. Values between 3 and 5 are recommended for the **ILU** scheme, and a value of 5 or higher is recommended for the symmetric Gauss-Seidel family of schemes.

RELAX sets the underrelaxation parameter for the **ILU**, **SOR**, or **SSOR** scheme update. Valid values range between 0.5 and 1.0.

T-STEP specifies how the method used to obtain the time step:

LOCAL → uses the provided CFL schedule to determine a spatially varying time step
GLOBAL → uses the minimum value of the local time step at each iteration cycle
DELTAT → uses the provided time step schedule
SUBIT1 → dual-time stepping scheme where the integration time step is provided along with a CFL schedule for the local time step used for integration in pseudotime (subiterations)
SUBIT2 → dual-time stepping scheme where the integration time step is provided along with a scaling factor schedule used to scale the integration time step for the subiteration process

NOTE: The dual-time stepping schemes are only available for use with the implicit time advancement schemes.

The **SUBIT1** and **SUBIT2** time step options require an additional control line to complete the dual-time integration specification:

TIME STEP	SUB-ITS	RES-RED	METHOD	C-N RELAX
5.0e-7	5	3.0	2ND	0.5

TIME STEP specifies the integration time step in seconds.

SUB-ITS specifies the maximum number of subiterations to perform.

RES-RED specifies the relative residual error L2-norm criteria for halting the subiteration process. The numerical value is the convergence criteria in terms of orders of magnitude (e.g., a value of 3 implies that the subiterations will be halted after the L2-norm of the subiteration residual error has been reduced by at least 3 orders of magnitude relative to the value at the first subiteration). The value can be supplied as a positive or negative value (the sign is ignored), but the magnitude cannot be less than 1.0. Note that the subiteration process is also halted if the absolute residual error L2-norm criteria **ABS RES** specified on the FMG-LVLS line has been satisfied (provided that a non-zero value has been specified).

METHOD specifies the method used to form the temporal differencing term.

1ST $\rightarrow$ 1^{st}-order backward difference
2ND $\rightarrow$ 2^{nd}-order backward difference
C-N $\rightarrow$ 2^{nd}-order Crank-Nicolson scheme

C-N RELAX specifies the relaxation coefficient for the Crank-Nicolson scheme. The value specified must lie between 0.25 and 0.5.

STATS provides the iteration cycle frequency for the output of various solver statistics gathered during the time (or pseudotime) advancement. The specific information written out is solver-specific, but contains information such as the maximum residual error and location where it occurred, number of realizability violations, minimum and maximum y^+ value, Jacobian update frequency, etc. A value of 10 is recommended to provide this information without swamping the iteration cycle residual error L2-norm output. Setting the value to 1 can be useful when having difficulties getting a simulation started, as it can provide information to aid with the determination of how (or where) the simulation is failing.

MIN-CFL specifies the minimum CFL value to be used when the **VAR-CFL** option is active and the time stepping scheme is set to either **LOCAL** or **SUBIT1**. A setting between 0.5 and 1.0 is recommended for the **R-K** or **DAF** time advancement schemes, while values between 1.0 and 0.1 times the maximum CFL number are recommended for the strong implicit schemes (i.e., **ILU**, **SGS**, or **SOR**). Smaller values should only be used if robustness issues are encountered.

VAR-CFL activates (**Y**) or deactivates (**N**) an adaptive (spatially variable) CFL strategy that reduces the specified CFL number in regions of high pressure and/or mass fraction gradients. The minimum CFL number imposed by this option is the **MIN-CFL** value provided. This feature allows the time step to be reduced when time accuracy is not required in regions where the flow properties are evolving rapidly. It is recommended that this option be activated when implicit schemes are used for steady-state simulations.

 # CFL determines the number of entries to be supplied in the CFL (or time step) schedule. A minimum of 2 entries is required. The CFL schedule is supplied on the lines that follow unless a strong implicit scheme (i.e., **ILU**, **SGS**, or **SOR**) is selected, where instead the schedule will immediately follow the implicit scheme control specification.

VIS-DT enables (**Y**) or disables (**N**) the viscous eigenvalue constraint when **LOCAL**, **GLOBAL**, or **SUBIT1** time stepping is selected. The viscous eigenvalue constraint should always be enabled when an explicit **R-K** time advancement scheme is used for viscous flow simulations. This constraint is often necessary for the weakly implicit **DAF** scheme as well, but can often be disabled when any of the highly implicit schemes (e.g., **ILU** or **SGS**) are used unless robustness issues are encountered.

IMP-BC enables (**Y**) or disables (**N**) the implicit treatment of the boundary conditions. This option has no effect for simulations that use either the **R-K** and **DAF** time advance-

ment schemes. This flag should typically be enabled when any of the implicit schemes are utilized (e.g., **ILU** or **SGS**).

REG-RES enables (**Y**) or disables (**N**) the reading of restart files for this region. Restart files must be present if this flag is enabled.

Consider the following when selecting parameters that control time advancement:

- The use of the ILU scheme for structured grid simulations is typically recommended for viscous flow problems that resolve the entire boundary layer (i.e., when wall functions are not used). The spatial transfer of information is less sensitive to high aspect ratio grids, so the flowfield in these regions will evolve at a rate that is comparable to the rest of the computational domain.

- The ILU scheme for structured grid simulations that utilize wall matching functions may not outperform the DAF scheme. Grids generated for use with wall functions typically have less severe aspect ratios, which the DAF scheme is well suited for. Moreover, the implicit treatment for the wall matching function boundary conditions have not been fully linearized, and as a result, the maximum stable CFL number may not be high enough to offset the computational cost associated with the ILU scheme.

- The CFL schedule should reflect changes that occur to the numerical algorithm as the simulation proceeds. For example, the CFL value should be reduced at iteration cycles that correspond to advancing to a new grid level and/or when switching from 1^{st}-order to higher-order. This provides the solution some time to adjust to the new settings (grid level or higher-order) while using a lower time step.

- The viscous time step constraint should always be enabled unless a highly implicit scheme is selected (i.e., ILU or SGS), but considerable robustness gains (at the expense of convergence rate) can be realized by enabling this feature for these implicit schemes as well.

- The implicit dual-time stepping scheme permits the use of large time steps for time-accurate simulations, so careful thought should be given when setting the time step value to maintain a high level of temporal accuracy. For hybrid RAS/LES, a useful measure to consider is the cell residence time (e.g., $\Delta x/u_\infty$), which should be kept below unity. For wall-resolved LES, the time step should be even more restrictive, since the simulation is intended to be scale-resolving through the log layer and sub-layer portions of the boundary layer where processes other than convection dominate.

- If the convergence rate of the subiterations is slower than expected, then consider dropping the integration time step. This will reduce the number of subiterations required by the dual-time stepping scheme as well as improve the temporal accuracy.

- The Crank-Nicolson scheme is more accurate than the 2^{nd}-order backward difference scheme. However, this scheme is prone to spurious oscillations that may lead to instabilities if the time step is too high, so keep this in mind when choosing the finite difference scheme for the dual-time stepping method.

- The direct specification of the time step (i.e., DELTAT) often proves to be a better choice than local time stepping (based on a CFL values) for steady-state simulations that have large nearly stagnant regions. A common example is the discharge of a nozzle flow into an ambient environment. If local time stepping with an implicit time integration strategy proves to be problematic, try specifying the time step directly with the VAR-CFL option active.

Examples are provided below that illustrate the various permutations of **SCHEME** and **T-STEP** choice for both elliptic and parabolic regions:

Example 1: Parabolic region using **DAF** with local time stepping.

```
SCHEME  T-STEP  STATS  MIN-CFL  VAR-CFL  # CFL  VIS-DT  IMP-BC  REG-RES
  DAF     LOCAL   10      1.0        Y      2       Y       N        N
  1       5   ← plane no.
  0.1     1.0 ← initial CFL value
  5.0     5.0 ← final CFL value
```

For parabolic regions, the CFL schedule is controlled at the space-marching plane level rather than by iteration counter. The reason behind this philosophy is the fact that startup stability issues (if present) when space-marching are limited to the first couple of marching planes where, for example, a boundary layer may be initiated. Once the boundary layer (or other flow structure) has been initiated, the initial condition for the next marching plane is a very good one based on the converged solution of the previous plane. The ramping between the starting to ending CFL values supplied for a given plane is controlled internally by VULCAN-CFD based on the number of iterations specified for each plane. In this example, the CFL schedule states that the CFL number will start at 0.1 on the first plane and ramp up to 5.0 as the plane is solved. On subsequent planes, the starting CFL number will ramp up from 0.1 to 1.0 (linearly) until plane 5 is reached. Every plane thereafter will start with a CFL of 1.0 and end with a CFL of 5.0.

Example 2: Parabolic region that is substepped using **DAF** with local time stepping.

```
SCHEME  T-STEP  STATS  MIN-CFL  VAR-CFL  # CFL  VIS-DT  IMP-BC  REG-RES
  DAF     LOCAL   10      1.0        Y      2       Y       N        N
  1       5   ← plane no.
  4       4   ← no. of substeps
  0.1     1.0 ← initial CFL value
  5.0     5.0 ← final CFL value
```

The only difference between this example and the previous one is the additional line that sets the number of substeps to perform at each marching plane when the **SUB-STEP** parameter is set to "M". The number of substeps does not have to be the same for each marching plane. However, a discontinuous jump in streamwise spacing will occur between adjacent marching planes that have been subdivided differently.

Example 3: Parabolic region using **ILU** with local time stepping.

```
SCHEME  T-STEP  STATS  MIN-CFL  VAR-CFL  # CFL  VIS-DT  IMP-BC  REG-RES
  ILU    LOCAL    10      1.0        Y      2       N       Y        N
SWEEP-DIR  JAC-UPDATE  START  NUM-SLVS  RELAX
     0          0        1       3       0.9
  1     5   ← plane no.
  0.1   1.0 ← initial CFL value
  5.0   5.0 ← final CFL value
```

NOTE: For this example, the viscous time step constraint was deactivated, since the implicit ILU scheme typically does not require it to be active for stability. Notice also that the CFL schedule appears after the implicit scheme control specification section.

Example 4: Elliptic structured grid region using **DAF** with local time stepping.

```
SCHEME  T-STEP  STATS  MIN-CFL  VAR-CFL  # CFL  VIS-DT  IMP-BC  REG-RES
  DAF    LOCAL    10      0.5        Y      8       Y       N        N
  1     500   1500   1501   2000   2500   2501   3000
  0.5   1.5   1.5    0.5    1.5    1.5    0.5    1.5
```

Here, the CFL values have been reduced at each iteration cycle that corresponds to advancing to a new grid level and/or when switching from first-order to higher-order. This provides the solution some time to adjust to the new settings (grid level or higher-order) while using a lower time step value.

Example 5: Time-accurate structured or unstructured grid region using **R-K** with a time step based on CFL values.

```
SCHEME  T-STEP  STATS  MIN-CFL  VAR-CFL  # CFL  VIS-DT  IMP-BC  REG-RES
  R-K    GLOBAL   10      0.5        Y      2       Y       N        N
  1     50
  0.5   1.5
```

This example illustrates the setup for an elliptic region that is being solved in a time-accurate manner. The use of global time stepping allows the CFL number to be specified (which has known stability bounds for a given Runge-Kutta scheme). In general, the time step value will change for each time increment, but it will not vary spatially at a given time level.

Example 6: Time-accurate structured or unstructured grid region using **R-K** with a specified time step.

```
SCHEME  T-STEP  STATS  MIN-CFL  VAR-CFL  # CFL  VIS-DT  IMP-BC  REG-RES
  R-K    DELTAT   10      0.5        Y      2       Y       N        N
  1     50
  5.0e-7  1.0e-6
```

This example is identical to the previous one, except the time step (in seconds) is being explicitly specified. Specifying the time step directly can be problematic when using an explicit time advancement scheme, since the CFL stability constraint could be violated if enough conservatism is not used when defining the time step schedule. However, this method does allow the user to specify the same time step for each time advancement step (if this is desired).

Example 7: Unstructured grid region using **SGS** with local time stepping.

```
SCHEME  T-STEP  STATS  MIN-CFL  VAR-CFL  # CFL  VIS-DT  IMP-BC  REG-RES
  SGS    LOCAL    10      0.5        Y      5      N       Y        N
SGS  JAC-UPDATE  START  NUM-SLVS
              0      1       10
  1     1000     2500    2501    3000
  0.5   5.0      2.5e1   1.0     2.5e1
```

NOTE: As was done for the **ILU** example, the viscous time step constraint has been deactivated for this point implicit scheme.

Example 8: Unstructured grid region using **SOR** with local time stepping.

```
SCHEME  T-STEP  STATS  MIN-CFL  VAR-CFL  # CFL  VIS-DT  IMP-BC  REG-RES
  SOR    LOCAL    10      0.5        Y      5      N       Y        N
SOR  JAC-UPDATE  START  NUM-SLVS  RELAX
              0      1       10     0.5
  1     1000     2500    2501    3000
  0.5   5.0      2.5e1   1.0     2.5e1
```

Example 9: Time-accurate structured grid region using **DAF** with dual-time stepping.

```
SCHEME  T-STEP  STATS  MIN-CFL  VAR-CFL  # CFL  VIS-DT  IMP-BC  REG-RES
  DAF    SUBIT2   10      0.5        Y      2      Y       N        N
  1       100
  1.0e2   1.0e2
TIME STEP  SUB-ITS  RES-RED  METHOD  C-N RELAX
  5.0e-7      5        3.0     2ND       0.5
```

Notice that the dual-time control section appears after the subiteration CFL schedule.

Example 10: Time-accurate structured grid region using **ILU** with dual-time stepping.

```
SCHEME  T-STEP  STATS  MIN-CFL  VAR-CFL  # CFL  VIS-DT  IMP-BC  REG-RES
  ILU    SUBIT1   10      1.0        Y      2      N       Y        N
SWEEP-DIR  JAC-UPDATE  START  NUM-SLVS  RELAX
    -3         100       1        3      0.9
```

```
  1          100
  1.0e2      1.0e2
TIME STEP   SUB-ITS   RES-RED   METHOD   C-N RELAX
   5.0e-7         5         2.5       C-N         0.5
```

Note that the frequency for updating the Jacobian has been set to a value larger than the number of subiterations to perform. This implies that the Jacobian will be frozen after the first subiteration. This tactic reduces the time required to perform the subiteration process, and given that the solution should not change much between successive time steps, freezing the Jacobian for the entire subiteration process is often acceptable.

This completes the description of the region control specification input section. At this point, a line that begins with "!" must be inserted after the last region specification to indicate that all of the solver control specifications for the VULCAN-CFD input file have been set, e.g.,

```
!****************** End of VULCAN-CFD solver control data ******************!
```

All that remains are the specification of boundary conditions, specialized initializations within user-prescribed volumes (optional), and for structured grid simulations, the mapping of boundary conditions to block boundaries, block connectivity mappings, and any spatial subblock specifications for turbulence suppression, ignition, and/or time history output.

9 Boundary Condition Specification

The number of boundary condition specifications is provided by the **BCGROUPS** line located in the block specification section. Each boundary condition specification is grouped based on a unique name (limited to 12 characters), followed by two additional columns that define the specific boundary condition type and any options associated with the application of the boundary condition. Depending on the boundary condition type and/or options provided, auxiliary lines of information may be required as well. The following boundary condition specification example illustrates many of the various permutations associated with defining the boundary conditions for VULCAN-CFD. The first line of the boundary condition specification is an arbitrary header line. The remaining lines define the 7 boundary conditions listed in this example along with the auxiliary information required by each condition. Details describing the available boundary condition types (and the auxiliary data needed to define them) are provided later in this chapter.

```
BCGROUPS:  NAME          TYPE          OPTION
           AIR-IN        FIX_IN        PHYSICAL_PRFINP
           comb_inf.prf
           FUEL-IN       P0_T0_IN      NORMAL
           O2  H2  H2O  N2   Ptot   Uvel   Vvel  Wvel  Ttot   Tint   Vrat
           0.0 1.0 0.0  0.0  1.0e6  1500.0 0.0   0.0   532.4  0.001  0.01
           OUTFLOW       EXTRAP        PHYSICAL_PRFOUT
           comb_exit.prf
           SYMMETRY      SYMM          PHYSICAL
           ISOL-WALL     IWALL         BLEED_IBL   0.006
           Twall   Rlx
           350.0   1.0
           Mass Flow   Tau_wall   Area Fraction   Model   # Cells   Depth   Rlx
           -0.557656   61.715     0.21425         2       1         2.66    1.0
           COMB-WALL     TCWALLM       PHYSICAL_PRFINP_IBL   0.006
           comb_wall.prf
           PORT-WALL     AWALLM        PHYSICAL
```

9.1 Outflow Boundary Conditions

EXTRAP or **REG_OUT**

Supersonic outflow boundary with 1^{st}-order extrapolation of all variables. This boundary condition is applied at the cell center of the ghost (or halo) cells. If the outflow boundary also corresponds to a block-to-block interface connection to a downstream region, then **REG_OUT** should be specified instead of **EXTRAP**.

EXTRAP2 or **REG_OUT2**

Supersonic outflow boundary with 2^{nd}-order extrapolation of the conserved variables. A check is performed to ensure that the extrapolated variables are realizable. If a realizability violation is detected, then the variables are locally reset based on 1^{st}-order extrapolation. This boundary condition is applied at the cell center of the ghost (or halo) cells. If the outflow boundary also corresponds to a block-to-block interface connection to a downstream

region, then **REG_OUT2** should be specified instead of **EXTRAP2**.

PRES_OUT

Subsonic outflow boundary with static pressure specified. The temperature, velocity components, mass fractions (if applicable), vibrational/electronic energy (if applicable), and any transported turbulence properties are extrapolated from the interior. This boundary condition is applied at the cell face centers of the outflow boundary. To account for the effect that the turbulence kinetic energy has on the static pressure profile, the specified pressure (P_s) is modified based on the relationship,

$$P_b \;=\; P_s \;-\; \frac{2}{3}\rho_b k_b$$

since $P + 2/3\rho k$ is the effective pressure that is approximately invariant across the boundary layer. The outflow density is determined from the equation of state together with the extrapolated values for the composition and temperature (i.e., $R_b = R_i$ and $T_b = T_i$).

$$\rho_b \;=\; \frac{P_b}{R_b T_b} \;=\; \frac{P_b}{R_i T_i} \;=\; \frac{P_b}{P_i}\,\rho_i$$

The above relationship (and the extrapolation condition for the turbulence kinetic energy) results in the following expression for the pressure at the outflow boundary.

$$P_b \;=\; P_s \;-\; \frac{2}{3}\left(\frac{P_b}{P_i}\right)\rho_i k_i \quad\rightarrow\quad P_b \;=\; P_s\left(\frac{P_i}{P_i \;+\; \frac{2}{3}\rho_i k_i}\right)$$

In the expressions above, the subscripts "b" and "i" denote the boundary and interior cell values, respectively. The specified pressure value for this boundary condition can either be provided as a constant for the entire outflow boundary, or the value can vary spatially. A constant outflow pressure is specified by simply adding auxiliary information to this boundary condition group, i.e.,

BCGROUPS:	**NAME**		**TYPE**	**OPTION**
	OUTFLOW		PRES_OUT	PHYSICAL
	Back Pres	Rlx		
	1.0e5	0.0		

where the pressure is supplied in units of [Pa], and the underrelaxation parameter must lie between zero and unity (a value of zero or unity disables the underrelaxation). If enabled, the relaxation coefficient is utilized in the manner shown below.

$$P_s \;=\; (\mathrm{Rlx})\;P_s \;+\; (1.0 - \mathrm{Rlx})\left(P_b + \;\frac{2}{3}\rho_b k_b\right)^{\mathrm{old}}$$

An arbitrary negative value can be provided in the pressure slot to denote that the reference pressure (as provided by the reference condition data) is to be applied as the outflow pressure condition. A spatially varying outflow pressure condition (if desired) is accommodated through a user-provided "profile" file, i.e.,

BCGROUPS:	NAME	TYPE	OPTION
	OUTFLOW	PRES_OUT	PHYSICAL_PRFINP
	outflow_pres.prf		

Details on the profile file contents (and format) are provided later in this chapter.

MIXED_OUT
Mixed subsonic/supersonic outflow condition that defines outflow conditions based on **EXTRAP** or **PRES_OUT** depending on the value of the contravariant Mach number at the outflow boundary. The specification of this boundary condition is identical to that required by the **PRES_OUT** condition. If the local contravariant Mach number is not subsonic, then all conditions are extrapolated (1^{st} order) from the interior. Otherwise, the specified pressure condition is imposed as previously described for the **PRES_OUT** boundary condition.

MDOT_OUT
Subsonic outflow condition with the mass flow rate specified. This boundary condition (applied at the cell face centers of the outflow boundary) is enforced by setting an iteratively determined constant back pressure that establishes the desired mass flow rate at the outflow boundary. To complete the specification of this boundary condition, auxiliary information is required for the desired mass flow rate [kg/s], the underrelaxation parameter, and an initial guess for the back pressure [Pa] that drives the mass flow rate, i.e.,

BCGROUPS:	NAME		TYPE	OPTION
	OUTFLOW		MDOT_OUT	PHYSICAL
	Mass Flow	Rlx	Init Pres	
	0.375	0.0	1.0e5	

If enabled, the relaxation coefficient is applied to the iteratively determined back pressure in the same manner as described for the **PRES_OUT** boundary condition. The specification of a negative value for the initial back pressure will result in the reference pressure (as provided by the reference condition data section) being used as the initial guess for the back pressure.

9.2 Inflow Boundary Conditions

REF_IN
Supersonic inflow condition with all variables defined to the reference state (provided in the reference condition data section). The boundary conditions are applied at the cell face centers of the inflow boundary. If desired, the specified velocity values can be aligned with the computational boundary. The specification of **NORMAL** in the **OPTION** column will align the velocity vector normal to each inflow boundary cell face. The specification of **TANGENT** in the **OPTION** column will align the velocity vector with the grid lines that intersect the inflow boundary (applicable to structured grid simulations only).

FIX_IN
Supersonic inflow condition with all variables fixed. The specified conditions for this boundary condition can be provided as constant values for the entire inflow boundary, or

the values can be spatially varying. Constant inflow conditions require the specification of mass fractions (multicomponent gases only), density [kg/m^3], velocity [m/s], vibrational/electronic temperature [K] (thermal nonequilibrium only), temperature [K], turbulence intensity (if applicable), and turbulent to molecular viscosity ratio (if applicable) on the next 2 lines of input, e.g.,

BCGROUPS:	**NAME**			**TYPE**		**OPTION**					
	INFLOW			FIX_IN		PHYSICAL					
	O2	H2	H2O	N2	Dens	Uvel	Vvel	Wvel	Temp	Tint	Vrat
	0.233	0.0	0.0	0.767	1.162	700.0	0.0	0.0	300.0	0.001	0.01

If desired, the specified velocity values can be aligned with the computational boundary. The specification of **NORMAL** in the **OPTION** column will align the velocity vector normal to each inflow boundary cell face. The specification of **TANGENT** in the **OPTION** column will align the velocity vector with the grid lines that intersect the inflow boundary (applicable to structured grid simulations only). Spatially varying inflow conditions are accommodated through a user-provided "profile" file, i.e.,

BCGROUPS:	**NAME**	**TYPE**	**OPTION**
	INFLOW	FIX_IN	PHYSICAL_PRFINP
	inflow.prf		

Details on the profile file contents (and format) are provided later in this chapter.

NOTE: For thermal nonequilibrium gas models, a vibrational/electronic temperature value less than or equal to the translational/rotational temperature value will enforce thermal equilibrium at the boundary.

NOTE: Specifying a negative value in the static (or translational/rotational) temperature slot implies that static pressure [Pa] is being supplied instead of temperature.

NOTE: This boundary condition is applied at the cell face centers of the inflow boundary if constant conditions are specified. The boundary condition is applied at the ghost cell centers if a profile file is utilized.

P0_T0_IN

Subsonic inflow condition with the total pressure and total temperature specified. If a gas model other than calorically perfect was chosen, then the incoming composition (mass fractions) must also be provided. Inflow conditions for any transported turbulence quantities must also be specified, and for simulations of thermal nonequilibrium flow, the vibrational/electronic temperature must be provided as well. By default, the quantity extrapolated from the interior is Mach number. However, the outward pointing Riemann invariant

$$R^+ = u_n + \frac{2c}{\gamma - 1}$$

can instead be extrapolated if the **RIEMANN STAGNATION BC** input line is provided in the general input section of the input file. Here, u_n is the velocity normal to the boundary,

and c is the speed of sound. This particular boundary condition is enforced at the ghost cell centers adjacent to the inflow plane. The specified conditions for this boundary condition are provided either as constant values for the entire inflow boundary, or the values can vary spatially. Constant inflow conditions require the specification of mass fractions (multicomponent gases only), total pressure [Pa], velocity [m/s], vibrational/electronic temperature [K] (thermal nonequilibrium only), total temperature [K], turbulence intensity (if applicable), and turbulent to molecular viscosity ratio (if applicable) on the next 2 lines of input, e.g.,

BCGROUPS:	**NAME**			**TYPE**		**OPTION**					
	INFLOW			P0_T0_IN		PHYSICAL					
	O2	H2	H2O	N2	Ptot	Uvel	Vvel	Wvel	Ttot	Tint	Vrat
	0.233	0.0	0.0	0.767	7.5e5	45.85	0.0	0.0	532.4	0.001	0.01

The specified velocity values are only utilized to convert the turbulence intensity and/or viscosity ratio to transported turbulence quantities. Therefore, reasonable estimates should be provided when simulating turbulent flows. By default (or by specifying **NORMAL** in the **OPTION** column), the velocity values that result from this boundary condition are aligned to be normal to the inflow boundary of the computational domain. Alternatively, the specification of **TANGENT** in the **OPTION** column (structured grid simulations only) will align the velocity vector with the grid lines that intersect the inflow boundary, or the string **VECTOR** can be used to align the inflow velocity with the direction cosines that result from the specified velocity values.

NOTE: For structured grid simulations of converging and/or diverging nozzles, it is advised to negate the total pressure value to invoke the special nozzle initialization (provided that the block topology is amiable to it). This will provide an improved initial flowfield condition for the grid blocks adjacent to the inflow boundary. This option will sweep through the streamwise planes of the grid block that utilizes this boundary condition to compute total area of each plane. These area values are then used to compute 1-D values of the Mach number based on isentropic flow relationships with choked flow assumed at the minimum area plane.

NOTE: An arbitrary negative value can be provided in the total temperature slot to denote that the reference stagnation conditions (as provided by the reference condition data section) are to be utilized instead of the stagnation properties provided.

NOTE: For thermal nonequilibrium gas models, the vibrational/electronic temperature slot is just a placeholder since thermal equilibrium is currently always enforced at subsonic inflow boundaries with a specified stagnation condition.

Spatially varying inflow conditions are accommodated through a user-provided "profile" file, i.e.,

BCGROUPS:	**NAME**	**TYPE**	**OPTION**
	INFLOW	P0_T0_IN	PHYSICAL_PRFINP
	inflow.prf		

Details on the profile file contents (and format) are provided later in this chapter.

NOTE: The nozzle initialization feature described previously is not available if a profile file is utilized. However, a separate **P0_T0_IN** condition can be defined with specified inflow conditions for the sole purpose of flowfield initialization, e.g.,

BCGROUPS:	**NAME**			**TYPE**		**OPTION**				
	INFLOW			P0_T0_IN		PHYSICAL_PRFINP				
	inflow.prf									
	INFLOW_INI			P0_T0_IN		INITIAL				
	O2	H2	H2O	N2	Ptot	Uvel	Vvel	Wvel	Ttot	Tint	Vrat
	0.233	0.0	0.0	0.767	-7.5e5	45.85	0.0	0.0	532.4	0.001	0.01

The specification of **INITIAL** in the **OPTION** column specifies that this condition is to only be used for initialization purposes and is **not** imposed as a boundary condition. See the next section for further details on this boundary condition option.

MF_T0_IN

Subsonic inflow condition with the mass flux and total temperature specified. If a gas model other than calorically perfect was chosen, then the incoming composition (mass fractions) must also be provided. Inflow conditions for any transported turbulence quantities must also be specified, and for simulations of thermal nonequilibrium flow, the vibrational/electronic temperature must be provided as well. The static pressure is extrapolated from the interior. This boundary condition is applied at the cell face centers of the inflow boundary. The specified conditions for this boundary condition can be provided either as constant values for the entire inflow boundary, or the values can be spatially varying. Constant inflow conditions require the specification of mass fractions (multicomponent gases only), density [kg/m^3], velocity [m/s], vibrational/electronic temperature [K] (thermal nonequilibrium only), total temperature [K], turbulence intensity (if applicable), and turbulent to molecular viscosity ratio (if applicable) on the next 2 lines of input, e.g.,

BCGROUPS:	**NAME**			**TYPE**		**OPTION**				
	INFLOW			MF_T0_IN		PHYSICAL				
	O2	H2	H2O	N2	Dens	Uvel	Vvel	Wvel	Ttot	Tint	Vrat
	0.233	0.0	0.0	0.767	4.91	45.85	0.0	0.0	532.4	0.001	0.01

The product of the specified density and the velocity magnitude provides the desired mass flux at the inflow boundary. However, the velocity magnitude should be a reasonable estimate for the inflow velocity for turbulent flow simulations, since these values are used to convert the turbulence intensity and/or viscosity ratio to transported turbulence quantities. By default (or by specifying **NORMAL** in the **OPTION** column), the velocity values that result from this boundary condition are aligned to be normal to the inflow boundary of the computational domain. Alternatively, the specification of **TANGENT** in the **OPTION** column (structured grid simulations only) will align the velocity vector with the grid lines that intersect the inflow boundary, or the string **VECTOR** can be used to align the inflow velocity with the direction cosines that result from the specified velocity values. Spatially varying inflow conditions are accommodated through a user-provided "profile" file, i.e.,

BCGROUPS:	NAME	TYPE	OPTION
	INFLOW	MF_T0_IN	PHYSICAL_PRFINP
	inflow.prf		

Details on the profile file contents (and format) are provided later in this chapter.

NOTE: The extrapolation of pressure from the interior to close out the state of the inflow boundary condition may result in conditions that are far from what is intended unless the pressure values used to initialize the flowfield domain (i.e., the reference pressure) is a good estimate of the expected inflow pressure value. If this is not the case, then it is recommended to use **FIX_IN** with best estimates provided for each inflow quantity during the initial stages of the iteration process, and then switch back to **MF_T0_IN** once a reasonable pressure field has been established in the vicinity of the inflow boundary. Another approach that is available for structured grid simulations (provided that the block topology is amiable to it) would be the use of **P0_T0_IN** as an initialization condition with the total pressure specified as a negative value to invoke the nozzle initialization feature, e.g.,

BCGROUPS:	NAME			TYPE		OPTION					
	INFLOW			MF_T0_IN		PHYSICAL					
	O2	H2	H2O	N2	Dens	Uvel	Vvel	Wvel	Ttot	Tint	Vrat
	0.233	0.0	0.0	0.767	4.91	45.85	0.0	0.0	532.4	0.001	0.01
	INFLOW_INI			P0_T0_IN		INITIAL					
	O2	H2	H2O	N2	Ptot	Uvel	Vvel	Wvel	Ttot	Tint	Vrat
	0.233	0.0	0.0	0.767	-7.5e5	45.85	0.0	0.0	532.4	0.001	0.01

The specification of **INITIAL** in the **OPTION** column specifies that this condition is to only be used for initialization purposes and is **not** imposed as a boundary condition. See the next section for further details on this boundary condition option.

NOTE: For thermal nonequilibrium gas models, the vibrational/electronic temperature slot is just a placeholder since thermal equilibrium is currently always enforced at subsonic inflow boundaries with a specified stagnation condition.

9.3 Inflow/Outflow Boundary Conditions

CHAR_REF

Farfield condition (inflow or outflow) based on characteristic conditions. For an inviscid flow, the compressible Navier-Stokes equations can be transformed into "characteristic form", which introduce quantities (characteristics) that are conserved across a plane through which the fluid flows. The characteristic variables are entropy

$$s = \frac{P}{\rho^{\gamma}}$$

and the Riemann invariants

$$R^{\pm} = u_n \pm \frac{2c}{\gamma - 1}$$

where u_n is the velocity normal to the boundary, and c is the speed of sound. The R^- invariant is evaluated based on conditions exterior to the computational domain, and R^+ is

determined locally from conditions within the computational domain. Once the Riemann invariants have been evaluated, the fact that these quantities are conserved across the boundary allows the interface contravariant velocity and the quantity $2c/(\gamma - 1)$ to be determined from the expressions below.

$$\begin{aligned} u_n &= \frac{1}{2}\left(R^+ + R^-\right) \\ \frac{2c}{\gamma - 1} &= \frac{1}{2}\left(R^+ - R^-\right) \end{aligned}$$

The sign of the contravariant velocity determines whether the local condition is an inflow ($u_n > 0$) or an outflow ($u_n < 0$) condition. If the local condition is an inflow condition, then the interface entropy is taken to be the reference value (extracted from the input values provided in the reference condition data section), and the Cartesian velocity components are provided by the following expressions,

$$\begin{aligned} u &= u_{ref} + \frac{\eta_x}{\|\nabla\eta\|}\left(u_n - u_{ref}\right) \\ v &= v_{ref} + \frac{\eta_y}{\|\nabla\eta\|}\left(u_n - v_{ref}\right) \\ w &= w_{ref} + \frac{\eta_z}{\|\nabla\eta\|}\left(u_n - w_{ref}\right) \end{aligned}$$

where $\eta_x/\|\nabla\eta\|$, $\eta_y/\|\nabla\eta\|$, and $\eta_z/\|\nabla\eta\|$ are the components of the unit vector normal to the local cell boundary face. The mass fractions (if applicable), and any transported turbulence properties are taken to be the reference values. If the local condition is an outflow condition, then the interface entropy is extrapolated from the interior, and the Cartesian velocity components are obtained from the same expressions shown above, but with the reference velocity components replaced with the local interior Cartesian velocity components. The mass fractions (if applicable), and any transported turbulence properties are extrapolated from the interior.

NOTE: The characteristic form of the equations used to arrive at the expressions above were derived for a single component fluid that obeys an ideal gas equation of state. As a result, the composition of the gas in the cells adjacent to any boundary that utilizes this boundary condition must match that of the reference state (provided by the reference condition data section). Moreover, the gas must be thermally equilibrated in the cells adjacent to these boundaries.

CHAR_FIX

Farfield condition (inflow or outflow) based on characteristic conditions. This boundary condition is identical to **CHAR_REF**, except that the exterior conditions are specified in lieu of using the reference conditions. The specified conditions for this boundary condition can be provided as constant values for the entire inflow boundary, or the values can be spatially varying. Constant inflow conditions require the specification of mass fractions (multicomponent gases only), density [kg/m^3], velocity [m/s], vibrational/electronic temperature [K] (thermal nonequilibrium only), temperature [K], turbulence intensity (if applicable), and turbulent to molecular viscosity ratio (if applicable) on the next 2 lines of input, e.g.,

BCGROUPS:	NAME			TYPE		OPTION					
	INFLOW			CHAR_FIX		PHYSICAL					
	O2	H2	H2O	N2	Dens	Uvel	Vvel	Wvel	Temp	Tint	Vrat
	0.233	0.0	0.0	0.767	1.162	700.0	0.0	0.0	300.0	0.001	0.01

Spatially varying exterior conditions are accommodated through a user-provided "profile" file, i.e.,

BCGROUPS:	NAME	TYPE	OPTION
	INFLOW	CHAR_FIX	PHYSICAL_PRFINP
	inflow.prf		

Details on the profile file contents (and format) are provided later in this chapter.

9.4 Viscous Wall Boundary Conditions

AWALL

No-slip, adiabatic wall condition applied at the cell face centers along the surface. All velocity components are set to zero, and the temperature (including the vibrational/electronic temperature if applicable) is extrapolated to the surface to satisfy the adiabatic condition. The species mass fractions (if present) are extrapolated as well, so that any heat transfer due to species mass diffusion is also zero. The pressure at the surface is set to the interior cell value to comply with the boundary layer assumption that the near-wall pressure variation normal to the surface is negligible. The surface conditions for the transported turbulence variables (when applicable) are set as follows:

- The transported Spalart variable, $\tilde{\nu}$, is set to zero.
- The turbulence kinetic energy, k, is set to zero.
- The specific dissipation rate, ω, is set to $60\nu_w/(\beta d_w^2)$.

The surface condition for the specific dissipation rate is derived from asymptotic considerations (dissipation balancing diffusion in the near-wall region) with a factor of 10 enhancement as suggested by Menter.[71] Here, ν_w is the kinematic viscosity at the surface, d_w is the distance from the surface to the center of the cell adjacent to the surface, and β is the constant associated with the dissipation term of the transport equation for ω.

AWALLM

No-slip, adiabatic wall condition based on the compressible wall matching procedure of Wilcox.[24] This wall boundary condition is only available for the k-ω turbulence models, and should be invoked in lieu of the standard solve-to-wall adiabatic condition (**AWALL**) when the wall cell center y^+ value is expected to be much greater than unity. The VULCAN-CFD Theory Manual[1] provides the details for this particular wall function formulation.

NOTE: If the local cell y^+ value is unity (or smaller), then the surface conditions are set in a manner that is asymptotically consistent with the solve-to-wall conditions. If the local cell y^+ value is greater than unity, but still within the sublayer portion of the boundary layer (i.e., $y^+ \lesssim 10.8$), then the surface conditions are set based on a linear blend (in terms of y^+)

of the wall function conditions and the asymptotically consistent solve-to-wall conditions.

IWALL

No-slip, isothermal wall condition applied at the cell face centers along the surface. All velocity components are set to zero, and the surface temperature is user-specified to provide the isothermal condition. The surface temperature can either be provided as a constant for the entire surface boundary, or the value can vary spatially. A constant surface temperature is specified by simply adding auxiliary information to this boundary condition group, i.e.,

BCGROUPS:	**NAME**		**TYPE**	**OPTION**
	WALL		IWALL	PHYSICAL
	Wall Temp	Rlx		
	276.0	0.0		

where the temperature is supplied in units of [K], and the underrelaxation parameter must lie between zero and unity (a value of zero or unity disables the underrelaxation). If enabled, the relaxation coefficient is used to set the wall temperature

$$T_w \;=\; (\mathrm{Rlx})\, T_s \;+\; (1.0 - \mathrm{Rlx})\, T_w^{\mathrm{old}}$$

where T_w is the wall temperature that is applied and T_s is the specified wall temperature. A spatially varying surface temperature condition (if desired) is accommodated through a user-provided "profile" file, i.e.,

BCGROUPS:	**NAME**	**TYPE**	**OPTION**
	WALL	IWALL	PHYSICAL_PRFINP
	wall_temp.prf		

Details on the profile file contents (and format) are provided later in this chapter. Species diffusion processes (if relevant) to the surface are assumed to be negligible, i.e., species mass fractions are extrapolated. The pressure is also extrapolated to the surface from the interior; a condition consistent with the standard boundary layer assumption that near-wall pressure variability normal to the surface is negligible. The surface conditions for the transported turbulence variables (when applicable) are set as follows:

- The transported Spalart variable, $\tilde{\nu}$, is set to zero.
- The turbulence kinetic energy, k, is set to zero.
- The specific dissipation rate, ω, is set to $60\nu_w/(\beta d_w^2)$.

The surface condition for the specific dissipation rate is derived from asymptotic considerations (dissipation balancing diffusion in the near-wall region) with a factor of 10 enhancement as suggested by Menter.[71] Here, ν_w is the kinematic viscosity at the surface, d_w is the distance from the surface to the center of the cell adjacent to the surface, and β is the constant associated with the dissipation term of the transport equation for ω.

NOTE: If the provided surface temperature is a negative value, then this indicates that the surface condition is to be isothermal if the provided wall temperature (in absolute value)

is smaller than the internal cell temperature. Otherwise, the surface is treated locally as an adiabatic surface. This option is intended for surfaces that are either uncooled, or have a high degree of temperature variability (spatially) that has not been adequately measured. For example, if a surface within a high enthalpy facility test is monitored by only a few surface temperature "health" gauges used to govern how long the test can commence, then this temperature can be taken as an estimate for the maximum surface temperature for the CFD simulation.

IWALLM

No-slip, isothermal wall condition based on the compressible wall matching procedure of Wilcox.[24] This wall boundary condition is only available for the k-ω turbulence models, and should be invoked in lieu of the standard solve-to-wall isothermal condition (**IWALL**) when the wall cell center y^+ value is expected to be much greater than unity. The VULCAN-CFD Theory Manual[1] provides the details for this particular wall function formulation. The surface temperature can either be provided as a constant for the entire surface boundary, or the value can vary spatially. A constant surface temperature is specified by simply adding auxiliary information to this boundary condition group, i.e.,

BCGROUPS:	**NAME**		**TYPE**	**OPTION**
	WALL		IWALLM	PHYSICAL
	Wall Temp	Rlx		
	276.0	0.0		

where the temperature is supplied in units of [K], and the underrelaxation parameter must lie between zero and unity (a value of zero or unity disables the underrelaxation). If enabled, the relaxation coefficient is used to set the wall temperature

$$T_w \;=\; (\mathrm{Rlx})\; T_s \;+\; (1.0 - \mathrm{Rlx})\; T_w^{\mathrm{old}}$$

where T_w is the wall temperature that is applied and T_s is the specified wall temperature. A spatially varying surface temperature (if desired) is accommodated through a user-provided "profile file", i.e.,

BCGROUPS:	**NAME**	**TYPE**	**OPTION**
	WALL	IWALLM	PHYSICAL_PRFINP
	wall_temp.prf		

Details on the profile file contents (and format) are provided later in this chapter.

NOTE: If the local cell y^+ value is unity (or smaller), then the surface conditions are set in a manner that is asymptotically consistent with the solve-to-wall conditions. If the local cell y^+ value is greater than unity, but still within the sublayer portion of the boundary layer (i.e., $y^+ \lesssim 10.8$), then the surface conditions are set based on a linear blend (in terms of y^+) of the wall function conditions and the asymptotically consistent solve-to-wall conditions.

NOTE: If the provided surface temperature is a negative value, then this indicates that the surface condition is to be isothermal if the provided wall temperature (in absolute value) is smaller than the internal cell temperature. Otherwise, the surface is treated locally as an

adiabatic surface. This option is intended for surfaces that are either uncooled, or have a high degree of temperature variability (spatially) that has not been adequately measured. For example, if a surface within a high enthalpy facility test is monitored by only a few surface temperature "health" gauges used to govern how long the test can commence, then this temperature can be taken as an estimate for the maximum surface temperature for the CFD simulation.

TCWALL

No-slip, thermally coupled fluid/surface wall condition applied at the cell face centers along the surface. This boundary condition is identical to **IWALL** except instead of specifying the surface temperature, the surface temperature is determined by equating the fluid to surface heat flux with the heat flux expression that results from one-dimensional heat conduction through the surface.

$$\lambda_f \frac{\partial T}{\partial y} = \lambda_s \frac{\Delta T}{\Delta y}$$

In this expression, λ_f is the thermal conductivity of the fluid adjacent to the surface, λ_s is the thermal conductivity of the surface material, and Δy is the thickness of the material (y is taken to be the coordinate normal to the surface boundary). After discretizing the above equation,

$$\lambda_f \frac{T_i - T_w}{y_i - y_w} = \frac{\lambda_s}{\Delta y} (T_w - T_b)$$

an expression for the wall temperature can be obtained

$$T_w = \frac{\frac{\lambda_f T_i}{y_i - y_w} + \frac{\lambda_s T_b}{\Delta y}}{\frac{\lambda_f}{y_i - y_w} + \frac{\lambda_s}{\Delta y}}$$

where T_w is the wall temperature, T_i is the interior cell center temperature, and T_b is the backside surface temperature. To complete the specification of this boundary condition, the temperature on the backside of the surface, T_b, must be provided along with the ratio of the material conductivity to the thickness of the material (as measured from the fluid surface interface to where the backside temperature is provided). Note that the expression above also applies to the scenario where multiple surface slabs exist between the fluid interface and the location where the backside temperature is known. In this case, the ratio of the material thickness to conductivity appearing in the expression above takes on an effective value provided by:

$$\frac{\Delta y}{\lambda_s} = \sum_{n=1}^{N} \frac{\Delta y_n}{\lambda_{s_n}}$$

where λ_{s_n} and Δy_n are the material conductivity and thickness of slab "n". The backside temperature and effective thickness to conductivity ratio for the material can either be provided as a constant for the entire surface boundary, or the values can vary spatially. The specification of constant values is accomplished by providing auxiliary information to this boundary condition group, i.e.,

BCGROUPS:	**NAME**	**TYPE**	**OPTION**
	WALL	TCWALL	PHYSICAL

Thick/Cond	Back Temp	Rlx
2.1e-5	276.0	0.0

The thickness to conductivity ratio is provided in MKS units [(m^2 K / W)], the backside temperature is provided in Kelvins, and the underrelaxation parameter must lie between zero and unity (a value of zero or unity disables the underrelaxation). If enabled, the relaxation coefficient is used to set the backside temperature

$$T_b \;=\; (\text{Rlx})\, T_s \;+\; (1.0 - \text{Rlx})\, T_b^{\text{old}}$$

where T_b is the backside temperature that is applied and T_s is the specified backside temperature. Spatially varying values (if desired) for the backside temperature and/or material thickness to conductivity ratio are accommodated through a user-provided "profile" file, i.e.,

BCGROUPS:	**NAME**	**TYPE**	**OPTION**
	WALL	TCWALL	PHYSICAL_PRFINP
	wall_temp.prf		

Details on the profile file contents (and format) are provided later in this chapter.

NOTE: If the provided backside temperature is a negative value, then this indicates that the surface conditions are to be determined by balancing the heat flux to the surface with the heat flux through the surface slab (as described above) if the lagged wall temperature is smaller than the internal cell temperature. Otherwise, the surface is treated locally as an adiabatic surface. This option is intended for surfaces that are either uncooled, or have a high degree of temperature variability (spatially) that has not been adequately measured. For example, if a surface within a high enthalpy facility test is monitored by only a few embedded temperature "health" gauges used to govern how long the test can commence, then this temperature can be taken as an estimate for the maximum backside temperature for the CFD simulation.

TCWALLM

No-slip, thermally coupled fluid/surface wall condition based on the compressible wall matching procedure of Wilcox.[24] This wall boundary condition is only available for the k-ω turbulence models, and should be invoked in lieu of the standard solve-to-wall thermally coupled fluid/surface condition (**TCWALL**) when the wall cell center y^+ value is expected to be much greater than unity. The VULCAN-CFD Theory Manual[1] provides the details for this particular wall function formulation. The backside temperature and effective thickness to conductivity ratio for the material can either be provided as a constant for the entire surface boundary, or the values can vary spatially. Specification of constant values is accomplished by providing auxiliary information to this boundary condition group, i.e.,

BCGROUPS:	**NAME**	**TYPE**	**OPTION**
	WALL	TCWALLM	PHYSICAL
	Thick/Cond	Back Temp	Rlx
	2.1e-5	276.0	0.0

The thickness to conductivity ratio is provided in MKS units [(m^2 K / W)], the backside temperature is provided in Kelvins, and the underrelaxation parameter must lie between zero and unity (a value of zero or unity disables the underrelaxation). If enabled, the relaxation coefficient is used to set the backside temperature

$$T_b \;=\; (\mathrm{Rlx})\, T_s \;+\; (1.0 - \mathrm{Rlx})\, T_b^{\mathrm{old}}$$

where T_b is the backside temperature that is applied and T_s is the specified backside temperature. Spatially varying values (if desired) for the backside temperature and/or material thickness to conductivity ratio are accommodated through a user-provided "profile" file, i.e.,

BCGROUPS:	**NAME**	**TYPE**	**OPTION**
	WALL	TCWALLM	PHYSICAL_PRFINP
	wall_temp.prf		

Details on the profile file contents (and format) are provided later in this chapter.

NOTE: If the local cell y^+ value is unity (or smaller), then the surface conditions are set in a manner that is asymptotically consistent with the solve-to-wall conditions. If the local cell y^+ value is greater than unity, but still within the sublayer portion of the boundary layer (i.e., $y^+ \lesssim 10.8$), then the surface conditions are set based on a linear blend (in terms of y^+) of the wall function conditions and the asymptotically consistent solve-to-wall conditions.

NOTE: If the provided backside temperature is a negative value, then this indicates that the surface conditions are to be determined by balancing the heat flux to the surface with the heat flux through the surface slab (as described above) if the lagged wall temperature is smaller than the internal cell temperature. Otherwise, the surface is treated locally as an adiabatic surface. This option is intended for surfaces that are either uncooled, or have a high degree of temperature variability (spatially) that has not been adequately measured. For example, if a surface within a high enthalpy facility test is monitored by only a few embedded temperature "health" gauges used to govern how long the test can commence, then this temperature can be taken as an estimate for the maximum backside temperature for the CFD simulation.

REWALL

No-slip, radiative equilibrium wall condition applied at the cell face centers along the surface. This boundary condition determines the wall temperature by balancing the heat conducted to the vehicle surface with the heat emitted from the vehicle body due to radiation, i.e.,

$$\lambda \frac{\partial T}{\partial y} \;=\; \epsilon\, \sigma \left(T_w^4 - T_{amb}^4 \right)$$

where λ is the thermal conductivity of the fluid adjacent to the surface, ϵ is the surface emissivity, σ is the Stefan-Boltzmann constant, T_w is the wall temperature, and T_{amb} is the ambient temperature that the heat is radiated into. Upon discretization of the heat conduction term, the wall temperature can be obtained through the following transcendental

equation:

$$\lambda \frac{T_i - T_w}{y_i - y_w} = \epsilon \sigma \left(T_w^4 - T_{amb}^4 \right)$$

where T_w and T_i are the wall and interior cell center temperatures. To define the properties for this boundary condition, the surface emissivity (ϵ) must be provided along with the assumed ambient temperature (T_{amb}) as auxiliary information to this boundary condition group, i.e.,

BCGROUPS:	**NAME**	**TYPE**	**OPTION**
	WALL	REWALL	PHYSICAL
	Wall Emis.	Amb. Temp	Rlx
	0.5	248.0	0.05

The ambient temperature is provided in Kelvins, and the underrelaxation parameter must be greater than zero, but less than unity. The transcendental equation for the wall temperature is solved by lagging the wall temperature from the previous iteration, and as a result, a fairly low value for the underrelaxation parameter is required for numerical stability (a value between 0.001 and 0.05 is recommended).

NOTE: The heat load output for these surfaces is the conductive heat load that balances the heat radiated from the body due to radiation.

REWALLM

No-slip, radiative equilibrium wall condition based on the compressible wall matching procedure of Wilcox.[24] This wall boundary condition is only available for the k-ω turbulence models, and should be invoked in lieu of the standard solve-to-wall thermally coupled fluid/surface condition (**TCWALL**) when the wall cell center y^+ value is expected to be much greater than unity. The VULCAN-CFD Theory Manual[1] provides the details for this particular wall function formulation. This boundary condition determines the wall temperature by balancing the heat conducted to the vehicle surface with the heat emitted from the vehicle body due to radiation, i.e.,

$$\lambda \frac{\partial T}{\partial y} = \epsilon \sigma \left(T_w^4 - T_{amb}^4 \right)$$

where λ is the thermal conductivity of the fluid adjacent to the surface, ϵ is the surface emissivity, σ is the Stefan-Boltzmann constant, T_w is the wall temperature, and T_{amb} is the ambient temperature that the heat is radiated into. Upon discretization of the heat conduction term, the wall temperature can be obtained through the following transcendental equation:

$$\lambda \frac{T_i - T_w}{y_i - y_w} = \epsilon \sigma \left(T_w^4 - T_{amb}^4 \right)$$

where T_w and T_i are the wall and interior cell center temperatures. To define the properties for this boundary condition, the surface emissivity (ϵ) must be provided along with the assumed ambient temperature (T_{amb}) as auxiliary information to this boundary condition group, i.e.,

BCGROUPS:	NAME	TYPE	OPTION
	WALL	REWALLM	PHYSICAL

Wall Emis.	Amb. Temp	Rlx
0.5	248.0	0.05

The ambient temperature is provided in Kelvins, and the underrelaxation parameter must be greater than zero, but less than unity. The transcendental equation for the wall temperature is solved by lagging the wall temperature from the previous iteration, and as a result, a fairly low value for the underrelaxation parameter is required for numerical stability (a value between 0.001 and 0.05 is recommended).

NOTE: The heat load output for these surfaces is the conductive heat load that balances the heat radiated from the body due to radiation.

LWBLOW

Surface condition with mass effusion into the domain. This particular boundary condition was taken from the LAURA CFD solver, and it is typically employed to provide a surface mass transfer condition for heat-shield ablation. The conditions specified are the surface mass flux ($\rho_w\ u_{n,w}$) and surface temperature (T_w). If applicable, the species mass fractions, transported turbulence variables, and the vibrational/electronic temperature must also be specified. To complete the boundary condition specification, the wall-normal momentum is extrapolated from the interior (i.e., wall-normal momentum is taken to be conserved).

$$P_w \ + \ \rho_w\, u_{n,w}^2 \ = \ P_i \ + \ \rho_i\, u_{n,i}^2$$

The specified surface conditions can either be provided as constant values for the entire surface boundary, or the values can vary spatially. Specification of constant inflow conditions require values for the mass fractions (multicomponent gases only), density [kg/m^3], velocity [m/s], vibrational/electronic temperature [K] (thermal nonequilibrium only), temperature [K], turbulence intensity (if applicable), and turbulent to molecular viscosity ratio (if applicable) on the next 2 lines of input, e.g.,

BCGROUPS:	NAME	TYPE	OPTION
	WALL	LWBLOW	PHYSICAL

O2	N2	Dens	Uvel	Vvel	Wvel	Temp	Tint	Vrat
0.233	0.767	1.25	0.0	85.0	0.0	300.0	0.001	0.01

The product of the specified density and the velocity magnitude provides the desired mass flux at the inflow boundary. However, the velocity magnitude should be a reasonable estimate for the inflow velocity for turbulent flow simulations, since these values are used to convert the turbulence intensity and/or viscosity ratio to transported turbulence quantities. Spatially varying values (if desired) for the inflow conditions are accommodated through a user-provided "profile" file, i.e.,

BCGROUPS:	NAME	TYPE	OPTION
	WALL	LWBLOW	PHYSICAL_PRFINP
	wall_blow.prf		

Details on the profile file contents (and format) are provided later in this chapter.

NOTE: For thermal nonequilibrium gas models, a vibrational/electronic temperature value less than or equal to the translational/rotational temperature value will enforce thermal equilibrium at the boundary.

9.5 Slip/Symmetry Boundary Conditions

EWALL
Slip wall condition (applied at the ghost cell centers) with the surface wall pressure determined from the inviscid wall-normal momentum equation when the contravariant velocity is zero. This particular slip wall condition is only available for structured grid simulations. The velocity vector is reflected across the slip surface, and the entropy is extrapolated from the interior. The mass fractions (if applicable), vibrational/electronic energy (thermal nonequilibrium simulations only), and any transported turbulence properties are also extrapolated from the interior. This boundary condition is better suited for subsonic flowfields, since the formation of shocks along the slip surface would violate the enforcement of constant entropy.

NOTE: The convective flux is forced to zero at all boundaries that utilize this slip wall condition.

SWALL
Standard slip wall condition (applied at the ghost cell centers). All properties are reflected about the slip surface, resulting in a symmetric specification of all ghost cells relative to the interior cells of the domain.

NOTE: The convective flux is forced to zero at all boundaries that utilize this slip wall condition.

SYMM
Symmetry condition (applied at the ghost cell centers). All properties are reflected about the symmetry plane, resulting in a symmetric specification of all ghost cells relative to the interior cells of the domain. This boundary conditions reflects all grid properties (metrics, volumes, etc.) as well, and as a result, this boundary condition can only be applied at planar boundary surfaces. If the boundary is not planar, the standard slip wall boundary condition must be used instead.

NOTE: This boundary condition is intended to reproduce the behavior of a simulation where the computational domain has been reflected across the plane of symmetry. This entails reflecting the grid coordinates across the symmetry plane when defining ghost cell metrics (which requires the symmetry boundary to be a planar surface). The convective flux is also **not** forced to zero at boundaries that utilize this boundary condition. This avoids the "entropy layer" that forms due to the nulling of the convective fluxes (which removes the natural numerical dissipation associated with upwind treatment of these terms), but some level of mass conservation is sacrificed across the boundary. The amount of mass loss (or

gain) will be proportional to the truncation error of the numerical scheme in the vicinity of the symmetry boundary. If strict mass conservation is desired, then use the standard slip wall boundary condition instead.

NOTE: If using **SWALL** as a substitute for **SYMM**, then add the **EXCLUDE SLIP WALLS** line in the general input section of the input file to prevent these boundaries from being treated as actual solid surfaces in the force and moment accounting.

SYMM_X
Special symmetry condition (applied at the ghost cell centers) for symmetry boundaries that are formed by planes of constant x values. All properties are reflected about the symmetry plane, resulting in a symmetric specification of all ghost cells relative to the interior cells of the domain. This specific boundary condition allows for the possibility of collapsed cell faces at the symmetry boundary (which the generic **SYMM** boundary condition does not allow), since the reflection vector is known.

SYMM_Y
Special symmetry condition (applied at the ghost cell centers) for symmetry boundaries that are formed by planes of constant y values. All properties are reflected about the symmetry plane, resulting in a symmetric specification of all ghost cells relative to the interior cells of the domain. This specific boundary condition allows for the possibility of collapsed cell faces at the symmetry boundary (which the generic **SYMM** boundary condition does not allow), since the reflection vector is known.

SYMM_Z
Special symmetry condition (applied at the ghost cell centers) for symmetry boundaries that are formed by planes of constant z values. All properties are reflected about the symmetry plane, resulting in a symmetric specification of all ghost cells relative to the interior cells of the domain. This specific boundary condition allows for the possibility of collapsed cell faces at the symmetry boundary (which the generic **SYMM** boundary condition does not allow), since the reflection vector is known.

SYMM_AXI
Special symmetry condition applied at the cell face centers of the boundary axis for structured grid simulations that utilize polar grid topologies. This boundary condition requires that the axis of symmetry correspond to the x-axis (i.e., the y and z coordinates must be zero along the axis of symmetry). All scalar properties and the streamwise (x) velocity component are extrapolated to the axis, while the remaining velocity components are set to zero. This boundary condition must be used at the boundary that corresponds to the axis of symmetry for axisymmetric simulations. If this boundary condition is applied to a 3-D simulation, where multiple cells are collapsed in the azimuthal direction to form the axis of symmetry, then all extrapolated properties at each streamwise station are azimuthally averaged to provide a unique set of flowfield properties along the axis of symmetry. The coordinate direction that corresponds to the collapsed direction must be specified in the **OPTION** column of this boundary condition specification by one of the following strings:

IDIR specifies that the grid is collapsed in the i-direction
JDIR specifies that the grid is collapsed in the j-direction
KDIR specifies that the grid is collapsed in the k-direction

NOTE: The process used to average properties in the azimuthal direction assumes that the boundary condition has not been split into multiple segments in the azimuthal direction. This implies that the entire azimuthal extent of a given collapsed cell block boundary must span a single structured grid block.

9.6 Collapsed Cell Boundary Conditions

SINGULAR
Collapsed hexahedral cell face condition for structured grid simulations. This condition, applied at the collapsed cell face centers, averages all properties extrapolated from the interior in each "collapsed" direction. The coordinate direction that corresponds to the collapsed direction(s) must be specified in the **OPTION** column of this boundary condition specification by one of the following strings:

IDIR specifies that the grid is collapsed in the i-direction
JDIR specifies that the grid is collapsed in the j-direction
KDIR specifies that the grid is collapsed in the k-direction
ALL specifies that the grid is collapsed in every direction

NOTE: Special allowances are in place to force a 1^{st}-order convective treatment on the cell faces opposite of the collapsed boundary for all properties extrapolated from the collapsed boundary direction. As a result, the specific values defined by this boundary condition only influence the visualization of the final postprocessed results and will not impact the flowfield solution.

NOTE: The process used to average properties in the "collapsed" directions assumes that the boundary condition has not been split into multiple segments in the "collapsed" direction(s). This implies that the entire extent of the "collapsed" direction(s) for a given collapsed cell face must span a single structured grid block.

NOSLPSNG
Collapsed hexahedral cell face condition for structured grid simulations where the collapsed cells reside on a no-slip surface. This boundary conditions is identical to the **SINGULAR** condition, except the velocity components are set to zero. Any transported turbulence model variables that are zero at no-slip surfaces are also set to zero.

9.7 Periodic Boundary Conditions

C-PERIODIC
Cartesian periodic condition for unstructured grid simulations, where the flow condition is prescribed as spatially periodic along the Cartesian coordinate (x, y, or z) that is normal to the boundary. There must be a matching C-PERIODIC condition (with matching extruded grid coordinates) prescribed for the opposite boundary with a different boundary condition

name.

NOTE: In general, up to three C-PERIODIC boundary condition pairs are possible, but currently VULCAN-CFD only allows one pair to be present.

P-PERIODIC
Polar periodic condition for unstructured grid simulations, where the flow condition is prescribed as spatially periodic along the azimuthal coordinate. The x coordinate must be the streamwise direction, so that the radial and azimuthal coordinate values (r, θ) map to the (y, z) coordinate values. There must be a matching P-PERIODIC condition (with matching rotated grid coordinates) prescribed for the opposite boundary with a different boundary condition name.

NOTE: Only one P-PERIODIC boundary condition pair can be specified.

NOTE: C-PERIODIC and P-PERIODIC boundaries are specified as cut connectivity conditions for structured grid simulations (see Chapter 15).

9.8 Boundary Condition Options

The **OPTION** column of the boundary condition specification provides a means to supply additional "options" to those boundary conditions that support additional (or optional) information. Several of these have already been mentioned in the previous section, but a complete list of allowable strings that can be supplied to the **OPTION** column is provided below:

PHYSICAL	boundary condition applied with no special options
INITIAL	boundary condition used for initialization purposes only
BLEED	effusion of mass (into or out of the domain) at the boundary
<>**_NORMAL**	inflow velocity aligned normal to the boundary face
<>**_TANGENT**	inflow velocity aligned with grid lines that intersect the boundary (structured grids only)
<>**_VECTOR**	inflow velocity aligned with the provided velocity vector
<>**_IDIR**	grid is collapsed in the i-direction (structured grids only)
<>**_JDIR**	grid is collapsed in the j-direction (structured grids only)
<>**_KDIR**	grid is collapsed in the k-direction (structured grids only)
<>**_ALL**	grid is collapsed in every direction (structured grids only)
<>**_PRFINP**	boundary condition data supplied via an external file
<>**_PRFOUT**	internal solution adjacent to (or possibly at) the boundary will be written to a "profile" file that can be used for subsequent simulations
<>**_IBL**	no-slip wall boundary condition blended with the flowfield initialization across the provided boundary layer thickness
<>**_BLEND**	boundary condition blended with the flowfield initialization across the provided distance
<>**_PRFINP_IBL**	no-slip wall thermal condition data supplied via an external

	file and blended with the solution initialization across the provided boundary layer thickness
<>_**PRFOUT_IBL**	no-slip wall condition blended with the solution initialization across the provided boundary layer thickness with boundary data written to a "profile" file for use with subsequent simulations
<>_**PRFOUT_BLEND**	boundary condition blended with the solution initialization across the provided distance with boundary data written to a "profile" file for use with subsequent simulations
<>_**NORMAL_BLEND**	boundary condition blended with the solution initialization across the provided distance and inflow velocity aligned normal to the boundary face
<>_**TANGENT_BLEND**	boundary condition blended with the solution initialization across the provided distance and inflow velocity aligned with grid lines that intersect the boundary (structured grids only)
<>_**VECTOR_BLEND**	boundary condition blended with the solution initialization across the provided distance and inflow velocity aligned with the provided velocity vector

Note that the generic use of <> is intended to represent any one of the strings **PHYSICAL**, **INITIAL**, or **BLEED**. Further details (and limitations) associated with each option are provided in the text that follows.

PHYSICAL
This is the standard option that simply denotes that the boundary condition is a true physical boundary condition to be applied with no special options.

INITIAL
This option denotes that the boundary condition is to be applied only during the initialization process as a means to provide the user some control over how the computational domain is initialized. This option is currently only relevant to structured grid simulations, and can only be used with the following boundary conditions:

FIX_IN
P0_T0_IN

If **FIX_IN** is used for initialization purposes, then a nonzero propagation order index value must be specified in the structured grid boundary condition mapping lines that access this initialization. Chapter 14 contains further details on the propagation feature that is available for flowfield initialization in structured grid simulations. If **P0_T0_IN** is used, then the nozzle initialization feature specific to this boundary condition must be used (see the **P0_T0_IN** boundary condition description for further details).

NOTE: **FIX_IN** is the only boundary condition that can be specified as an initialization condition when a profile file is utilized.

BLEED
This option specifies that a surface mass effusion model (into or out of the domain) will be applied as a source term in the cells adjacent to the surface boundary. This option requires additional input data to define the effusion details and must immediately follow any auxiliary data associated with the provided surface boundary condition. The specific information to be provided is dependent upon the particular effusion model chosen:

Specified Mass Flux Model: This model specifies the effusion condition as a fixed mass flux over the entire effusion region. If the specified mass flow rate is positive, the fluid is to be effused into the domain (with a given composition and total temperature). If the mass flow rate is negative, the fluid is to be effused out of the domain. The input parameters required for this effusion specification are listed below:

- The mass fractions of the fluid being effused into the domain (multicomponent gas simulations only). These values must be specified regardless of whether mass is effused into or out of the domain (the values are ignored if mass is bled out of the computational domain).
- total mass flow rate [kg/s] flowing into (or out of) the flow domain. A positive value implies mass being fed into the domain, while a negative value implies mass being drained out of the domain.
- total temperature of the fluid being effused into the domain. This input is ignored if mass is effused out of the domain,
- fraction of the surface area that is composed of holes relative to the total effusion area. This fraction should also include any corrections desired to account for the effective aerodynamic porosity.
- effusion model type (i.e., set to "1")
- the number of cells normal to the boundary to be considered when applying the effusion source terms. The value entered here is only relevant for structured grid simulations, and should always be set to 1 unless stability problems arise.

Below is an example illustrating the use of this bleed model:

BCGROUPS:	**NAME**		**TYPE**	**OPTION**		
	WALL		IWALL	BLEED		
	Twall	Rlx				
	350.0	1.0				
	Mass Flow	Tot Temp	Area Fraction	Model	# Cells	
	0.27525	300.0	0.275	1	1	

Doerffer-Bohning Model: This model specifies the effusion condition based on the premise that a perforated plate separates the computational domain from an effusion plenum, where the effusion through the plate for this particular model is governed by an empirical correlation.[72] This model enforces the total effusion rate into (or out of) the domain, but the mass

flux at each surface cell interface will adjust based on local flow conditions. If the specified mass flow rate is positive, fluid is fed into the computational domain. If the mass flow rate is negative, fluid is drained out of the computational domain. The input parameters required for this effusion specification are listed below:

- The mass fractions of the fluid being effused into the domain (multicomponent gas simulations only). These values must be specified regardless of whether mass is effused into or out of the domain (the values are ignored if mass is bled out of the computational domain).
- total mass flow rate [kg/s] flowing into (or out of) the flow domain. A positive value implies mass being fed into the domain, while a negative value implies mass being drained out of the domain.
- plenum temperature of the plenum fluid (if the mass flow rate is a positive number, i.e., effusing mass into the domain), or a representative surface shear stress [Pa] for the flow approaching the effusion region if the total mass flow rate is zero or negative.
- fraction of the surface area that is composed of holes relative to the total effusion area. This fraction should also include any corrections desired to account for the effective aerodynamic porosity.
- effusion model type (i.e., set to "2")
- the number of cells normal to the boundary to be considered when applying the effusion source terms. The value entered here is only relevant for structured grid simulations, and should always be set to 1 unless stability problems arise.
- depth of the effusion plenum (the depth should be the value required to recover the plenum volume when multiplied by the total bleed area) Note that the value provided here must match the units provided in the grid file. For instance, if the grid coordinates are supplied in inches, then the plenum depth must also be provided in inches.
- underrelaxation factor (between 0 and 1) used to relax the bleed plenum pressure for numerical stability. A value of zero or unity disables the underrelaxation.

Below is an example illustrating the use of this bleed model:

BCGROUPS:	**NAME**	**TYPE**	**OPTION**				
	WALL	AWALL	BLEED				
	Mass Flow	Tau_wall	Area Fraction	Model	# Cells	Depth	Rlx
	-0.557656	61.715	0.21425	2	1	2.66	1.0

NOTE: The surface shear stress tends to reduce the amount of mass flow that can be bled from the domain. This parameter is known to influence the bleed process, but the correlation of this effect can be problematic to quantify. The Doerffer-Bohning model was originally based on a $1/7$ power law assumption for the approach flow boundary layer profile. In general, the surface shear stress is not known a priori, but this information can be extracted from the vulcan.loads.tec file that VULCAN-CFD outputs. If the approach flow shear stress is not known a priori, it is recommended that the surface shear stress initially be set to zero

for simulations that bleeds mass out of the domain. As the flowfield converges, the shear stress of the approach flow just upstream of the bleed region can be interrogated and used for the bleed specification for a subsequent restart of the simulation.

Slater Model: Similar to the Doerffer-Bohning model, this model specifies the effusion condition based on the premise that a perforated plate separates the computational domain from an effusion plenum, where the effusion through the plate for this particular model is governed by an empirical correlation.[73] It should be mentioned that this model has only been calibrated as a bleed model, but it can be used to effuse mass into the computational domain as well. This model enforces the specified effusion rate, but the mass flux at each surface cell interface will adjust based on local flow conditions. If the specified mass flow rate is positive, fluid is fed into the computational domain. If the mass flow rate is negative, fluid is drained out of the computational domain. The input parameters required for this effusion specification are listed below:

- The mass fractions of the fluid being effused into the domain (multicomponent gas simulations only). These values must be specified regardless of whether mass is effused into or out of the domain (the values are ignored if mass is bled out of the computational domain).
- total mass flow rate [kg/s] flowing into (or out of) the flow domain. A positive value implies mass being fed into the domain, while a negative value implies mass being drained out of the domain.
- plenum temperature of the fluid being effused into the domain. This input is ignored if mass is effused out of the domain,
- fraction of the surface area that is composed of holes relative to the total effusion area. This fraction should also include any corrections desired to account for the effective aerodynamic porosity.
- effusion model type (i.e., set to "3")
- the number of cells normal to the boundary to be considered when applying the effusion source terms. The value entered here is only relevant for structured grid simulations, and should always be set to 1 unless stability problems arise.
- depth of the effusion plenum (the depth should be the value required to recover the plenum volume when multiplied by the total bleed area) Note that the value provided here must match the units provided in the grid file. For instance, if the grid coordinates are supplied in inches, then the plenum depth must also be provided in inches.
- underrelaxation factor (between 0 and 1) used to relax the bleed plenum pressure for numerical stability. A value of zero or unity disables the underrelaxation.

Below is an example illustrating the use of this bleed model:

BCGROUPS:	**NAME**	**TYPE**	**OPTION**				
	WALL	AWALL	BLEED				
	Mass Flow	Plen Temp	Area Fraction	Model	# Cells	Depth	Rlx
	-0.557656	298.15	0.21425	3	1	2.66	1.0

The iterative history of the effusion plenum pressure, temperature, and the mass flow rates into and out of the domain are output to the file **bleed_#.tec** (where "#" is the region number that contains the bleed surface). This file should be monitored to ensure that the bleed plenum properties are converging properly. If oscillations in the plenum properties are noted, this is often a sign that a more conservative (i.e., smaller) underrelaxation parameter may be needed. Care must also be taken when performing simulations that effuse mass out of the computational domain, since it is possible to form a vacuum in the bleed plenum if the requested bleed flow rate cannot be accommodated given the present state of the flow conditions in the vicinity of the bleed region. If this occurs, the bleed flow rate may need to be reduced (in absolute value), and/or the area ratio (porosity) may need to be increased. If this problem is deemed to simply be a transient startup issue, then the plenum properties can be reinitialized on subsequent simulation restarts by negating the number of cells normal to the boundary input entry. This strategy will reinitialize the bleed plenum properties to the area weighted average composition, pressure, and temperature currently present over the surface through which the bleed occurs.

NOTE: The bleed models described above can be used for surfaces that utilize wall functions. However, the wall function relationships implemented in VULCAN-CFD have no corrections to account for the presence of mass effusion. As a result, the predictive accuracy on the impact of mass effusion will be compromised to some extent if activated on surfaces where wall functions have been employed.

NORMAL
This option aligns the velocity components that result from certain boundary conditions to be normal to the inflow boundary of the computational domain. The following inflow boundary conditions support this particular option:

REF_IN
FIX_IN
P0_T0_IN
MF_T0_IN

TANGENT
This option aligns the velocity components that result from certain boundary conditions to be tangent to the grid lines that intersect the inflow boundary of the computational domain. This option is applicable to structured grid simulations only, and if specified for an unstructured grid boundary, will resort to using **NORMAL**. The following inflow boundary conditions support this particular option:

REF_IN
FIX_IN
P0_T0_IN
MF_T0_IN

VECTOR
This option aligns the velocity components that result from certain boundary conditions to

be aligned with the direction cosines that result from the specified velocity values. The following inflow boundary conditions support this particular option:

P0_T0_IN
MF_T0_IN

IDIR, JDIR, KDIR, ALL
These options force a volume weighted average of the interior flowfield data adjacent to a collapsed grid boundary to define the state of the flow at the collapsed boundary. These options are only applicable to structured grid simulations. The specification of either **IDIR**, **JDIR**, or **KDIR** must be chosen to match the collapsed direction of the grid cells at the boundary. If the grid is collapsed in both coordinate directions that vary at the boundary (i.e., a single point), then the string **ALL** must be chosen. The following boundary conditions support these particular options:

SINGULAR
NOSLPSNG
SYMM_AXI (3-D simulations only)

IBL
This option imposes an initial boundary layer for the no-slip surface boundary condition specified. The boundary layer thickness to be imposed must also be provided with units that are consistent with the value specified for the **GRID SCALING FACTOR**. For instance, if the grid coordinates are supplied in units of inches, then the value entered here must also be provided in inches. The example below illustrates how this particular option is specified:

BCGROUPS:	**NAME**	**TYPE**	**OPTION**	
	WALL	AWALL	PHYSICAL_IBL	0.006

This particular example will linearly vary the velocity from zero at the surface to the initialized value in the computational domain over a span of 0.006 grid units. This option can be invoked with any no-slip wall boundary condition.

NOTE: If the global boundary layer initialization option has also been invoked (see the simulation initialization and restart control section of Chapter 8), then both initializations will be superimposed starting with the global initialization.

BLEND
This option initializes a portion of the domain by blending (or extruding) the specified boundary state into the interior over some distance. For unstructured grid boundaries, the distance over which this action will be performed is a true distance that must be provided with units that are consistent with the value specified for the **GRID SCALING FACTOR**. For instance, if the grid coordinates are supplied in units of inches, then the value entered here must also be provided in inches. For structured grid boundaries, the blend is performed in computational space over some fraction of the cells normal to the boundary face (i.e., i-direction, j-direction, or k-direction). The fraction of cells is set to 0.25 by default, but

this value can be changed using the input line **INIT. BLENDING FACTOR**. The example below illustrates how this particular option is specified:

BCGROUPS:	**NAME**	**TYPE**	**OPTION**	
	INFLOW	FIX_IN	PHYSICAL_BLEND	0.0254

This particular example will extrude the specified inflow condition into the computational domain over a span of 0.0254 grid units.

NOTE: The use of **BLEND** for unstructured grid simulations requires that T-infinity be enabled.

NOTE: Any blending distance prescribed with the **BLEND** option will be ignored for boundaries attached to structured grid blocks (i.e., the global value specified by the **INIT. BLENDING FACTOR** will be used instead).

The list of boundary conditions that can be used with the **BLEND** option (and the actions performed) are as follows:

P0_T0_IN	Extrudes the specified conditions
MF_T0_IN	Extrudes the specified conditions
FIX_IN	Extrudes the specified conditions
CHAR_FIX	Extrudes the specified conditions
PRES_OUT	Blends using isentropic relationships or extrudes normal shock conditions
MIXED_OUT	Blends using isentropic relationships or extrudes normal shock conditions
MDOT_OUT	Blends using isentropic relationships or extrudes normal shock conditions

NOTE: The action of blending or extruding normal shock conditions when using the **BLEND** option with **PRES_OUT**, **MIXED_OUT**, or **MDOT_OUT** depends on the reference state Mach number and the ratio of the specified pressure to the reference state pressure. If the reference Mach number is supersonic and the pressure ratio is greater than unity, then a normal shock condition is extruded from the outflow plane based on the pressure ratio. This is a useful way to initialize the flowfield when the back pressure is expected to force the flow to be subsonic (e.g., a scramjet isolator flowfield). If the reference Mach number is subsonic, or if the pressure ratio is less than or equal to unity, then isentropic relations are used to compute the Mach number given the specified pressure and the reference state stagnation pressure. This Mach number is then linearly blended into the interior with the reference Mach number, and all other flow properties are set from isentropic relationships based on the blended Mach number value.

PRFINP
This option specifies that the external conditions required by the boundary condition will be provided via an externally supplied "profile" file rather than specified as constant values. The file name for this input profile data is specified on the next line.

PRFOUT
This option exports a profile file for either direct use as a boundary condition, or for the

purpose of providing the grid coordinates for subsequent interpolation of externally supplied spatially-varying data. This option can be used in conjunction with any boundary condition. If the boundary condition is applied to the ghost cells of the boundary (e.g., **EXTRAP**), then the adjacent interior flowfield data will be exported to the file (in this scenario, the adjacent interior cells would be ghost cells for an inflow boundary of a downstream grid block). If the boundary condition is applied at the boundary face, then the boundary face values will be exported to the file. A common scenario where this might be desired is for no-slip surfaces where a temperature distribution is to be specified. The file name for the exported profile data is specified on the next line.

NOTE: The profile file data are output at the boundary condition application level. If the boundary condition group is spread over multiple block (or partition) faces, then multiple calls to the boundary condition routine will be required to export the profile data in its entirety. To accommodate this scenario, the profile data for each block (or partition) face are output to separate files with a unique integer appended to the file name. Prior to using these profile files in a simulation, the utility **vulcan_prof_mrg** must be executed to merge the profile data into a single file. This utility also sorts the profile data by x, y, z value, which greatly reduces the time required to match profile coordinates with grid boundary coordinates.

9.9 Boundary Condition Profile File Format

Any boundary condition that requires externally provided properties can be specified using either constant conditions (defined directly in the input file), or as spatially varying conditions through an externally supplied file. The file format used to specify spatially varying boundary properties is a native Tecplot[7] format ASCII data file, as illustrated via the following pseudocode:

```
write(iprf,*) 'TITLE = VULCAN-CFD primitive variable profile file'
write(iprf,*) 'VARIABLES = "x" "y" "z" "v1" ... "vnv"'
do l = 1, ngl
   write(iprf,*) 'ZONE T="Profile", I = ', nbf
   do n = 1, nbf
     write x, y, z, (v(m),m=1,nv)
   end do
end do
```

The number of ghost layers (ngl) is always 1 for unstructured grid boundaries, and can be 1, 2, or 3 for structured grid boundaries. The maximum number of structured grid ghost cell layers is governed by the size of the one-sided stencil used for variable extrapolation to the boundary face when constructing the inviscid fluxes. The one-sided stencil size is 2 for second-order schemes and 3 for the higher-order schemes (e.g., PPM or WENO). The order of the layers begins with the ghost cell closest to the boundary. If less than the maximum number of layers is present in the file, then the remaining layers are filled by simply extrapolating the last layer provided. The coordinate and flow variable values must be specified for each boundary face assigned to the boundary condition, where the

number of boundary faces has been denoted by nbf in the code segment above. However, the specific order of the boundary cell faces is of no consequence, since VULCAN-CFD will simply match the coordinate values in the profile file to the face-centered coordinates of the grid boundary. The number of flow variables that must be specified (nv) is dependent on the specific type of simulation being performed. Calorically perfect gas simulations require that values be specified for the following variables:

Density, X-velocity, Y-velocity, Z-velocity, Temperature, Turbulence Variables

Thermally equilibrated multicomponent gas simulations require specifications for the following quantities:

Y_1 ... Y_{ncs}, Density, X-velocity, Y-velocity, Z-velocity, Temperature, Turbulence Variables

Finally, the variables required for multicomponent gases in thermal nonequilibrium simulations are as follows:

Y_1 ... Y_{ncs}, Density, X-velocity, Y-velocity, Z-velocity, Vibrational/Electronic Temperature, Translational/Rotational Temperature, Turbulence Variables

Note that the order of the flow variables must also conform to the descriptions above. The required turbulent transport variables depends on the particular turbulence model utilized. No turbulence variables are specified for laminar flow simulations (or large eddy simulations with algebraic subgrid closure models). The Spalart model requires the specification of the Spalart viscosity ($\tilde{\nu}$), two-equation models require specification of k followed by ω, and one-equation subgrid models require the specification of k_{sgs}. All coordinates and flow variables must be provided in *mks* units. Furthermore, the profile file data for boundary conditions that impose flow properties other than those defined above (e.g., specified inflow stagnation conditions or specified outflow pressure conditions) must be provided in a manner that recovers the desired condition(s). For instance, if profile file data are generated with the intent of specifying the outflow pressure, then the values provided for ρ, T, and Y_m must produce the desired static pressure as governed by the perfect gas relation.

$$P = \rho R(Y_m) T$$

NOTE: When creating a VULCAN-CFD profile data file from an external source for two-equation turbulence model usage, the convention used to define ω must be reconciled. VULCAN-CFD always defines ω as (ϵ/k) rather than the $(\epsilon/k/0.09)$ convention used by Wilcox[40,41] and Menter.[38] If the ω values supplied to VULCAN-CFD follow the Wilcox/Menter convention, then these values must be multiplied by 0.09 when creating the profile data or use with VULCAN-CFD.

In order to match the profile file coordinate values to the coordinates of the grid in an efficient manner, the x, y, and z values must be sorted. This sorting is accomplished by executing the utility code **vulcan_prof_mrg**. This utility also merges the data from multiple profile files to form a master profile file for all boundary cells that require the profile

file data. The need to merge profile file data commonly arises when profile files have been output from a partitioned VULCAN-CFD simulation. To accommodate the conversion of deprecated structured grid conserved variable profile file data files to the present format, all profile files that accompany a **FIX_IN** boundary condition must include one of the following title lines at the beginning of the profile data file:

TITLE = VULCAN-CFD primitive variable profile file
TITLE = VULCAN-CFD conserved variable profile file

so that the flow solver has knowledge of what type of data is present. The appropriate title line is included by default for all profile files written by the flow solver or the **vulcan_prof_mrg** utility. As evidenced by the discussion here, the **vulcan_prof_mrg** utility should (and usually must) be executed to process the profile file data prior to their use in a VULCAN-CFD simulation.

If the profile data are being supplied outside of VULCAN-CFD, or if variable interpolation is required to supply the profile data to a different grid point distribution, it is best to output "dummy" profile files using VULCAN-CFD for the boundaries that are intended to utilize the profile file data. This is accomplished by temporarily replacing the PRFINP boundary condition option with PRFOUT, and executing VULCAN-CFD with the number of iterations set to zero. The purpose of this step is to extract the precise x, y, and z values of the grid boundary faces that require the profile information, which can subsequently be used to either analytically define the profile file data, or used to interpolate data from an external source (e.g., measurements or data generated from another CFD code and/or grid). A discussion on how to use the data interpolation features of Tecplot to interpolate the profile file data from one set of face grid coordinates to another (including the extrusion of 2-D profile file data to a 3-D domain) is provided in the profile file utility codes section of Chapter 25.

As a final note, the profile file framework was designed to allow a common format for both structured or unstructured grid simulations. Hence, a profile file generated from a structured grid simulation can be used to supply the boundary condition for an unstructured grid boundary (and vice-versa). However, a complication arises when a profile file from a structured grid axisymmetric simulation is to be used for an unstructured grid simulation (or conversely, when a profile file from an azimuthally symmetric unstructured grid simulation is to be used for a structured grid simulation) because of differences in how axisymmetric simulations are handled. The structured path constructs a 3-D azimuthally symmetric grid by rotating a supplied 2-D grid for axisymmetric simulations. The unstructured path does not currently support this feature, and instead requires that an azimuthally symmetric sector mesh be provided. As a result, the radial coordinates from a 2-D grid used for an axisymmetric structured grid simulation will not match the face-centered radial coordinates of an equivalent 3-D azimuthal sector mesh generated by rotating the 2-D grid about the streamwise axis as illustrated in Fig. 13. Instead, the radial coordinates are related through the following expression:

$$y_{3D} = y_{2D} \cos(\theta/2)$$

where y_{3D} are the face-centered y-coordinates of the 3-D azimuthal grid, y_{2D} are the radial coordinates of the axisymmetric grid, and θ is the sector angle of rotation used to construct

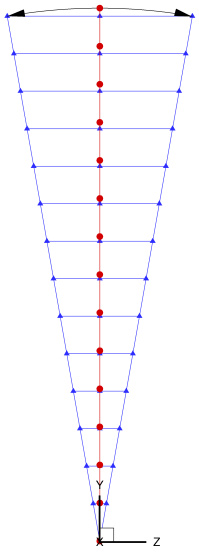


Figure 13: Image depicting the relationship between axisymmetric and 3-D sector grid coordinates (red circles represent the axisymmetric radial distribution, blue triangles show the resulting y-distribution of the axisymmetric grid after being rotated $\pm\ \theta/2$ degrees).

the 3-D azimuthal grid. This transformation must be applied to modify the y-coordinates for the scenarios described above. The **vulcan_prof_mrg** utility will prompt for the sector angle required to perform this transformation (if the user specifies that the profile data is intended for an axisymmetric simulation) when every z-coordinate in the profile file is zero.

9.10 Boundary Condition Objects

By default, VULCAN-CFD will output integrated forces and moments for each boundary condition that is specified. This provides a mechanism for tracking integrated loads for various components of the flight vehicle. For instance, a wing/slat/flap combination could conveniently be grouped in the following manner:

BCGROUPS:	**NAME**		**TYPE**	**OPTION**
	FREESTREAM		REF_IN	PHYSICAL
	OUTFLOW		EXTRAP	PHYSICAL
	WING		IWALL	PHYSICAL
	Twall	Rlx		
	350.0	1.0		
	SLAT		AWALLM	PHYSICAL
	FLAP		AWALLM	PHYSICAL

in order to track the integrated loads on each wing component separately. Further control of the force accounting is provided through the use of boundary condition objects. Boundary condition objects represent groupings of boundary conditions. This feature allows the integrated loads of two or more boundary conditions to be summed together. In the example

above, one could define a boundary condition object that contains the WING, SLAT, and FLAP groupings, i.e,

BCOBJECTS:	**NAME**	**# OF MEMBERS**
	WING+SLAT+FLAP	3
	WING SLAT FLAP	

resulting in the output of integrated loads for the entire wing/slat/flap combination (in addition to the loads on the wing, slat, and flap separately). Each boundary condition object consists of two lines of information. The first line contains the boundary condition object name (an arbitrary choice), and the number of boundary conditions that define the object. The second line lists the boundary conditions that make up the object. The boundary condition object input section must immediately follow the boundary condition specification section. The first line of the boundary condition object specification is an arbitrary header line. The remaining lines define each boundary condition object. The total number of boundary condition objects is provided by the **BCOBJECTS** line located in the block specification section.

10 Specified Volume Initializations

The default flowfield initialization process sets the entire computational domain to the reference state provided in the reference condition data section. While convenient, this simplistic initialization process can lead to simulation startup difficulties in regions where the eventual fluid state is vastly different than the reference state. Examples of flow regions that could benefit from an alternative initialization include:

- Base flow regions
- Recessed cavity flow regions
- Fuel injection ports

VULCAN-CFD allows the user to define custom initializations within a specified volume that includes any combination of rectangular boxes, cylinders, cones, or conical frustums. The total number of initialization specifications is provided by the **INIT. SUB-DOMAINS** line located in the block specification section of the input file. This section (if present) must appear immediately after all boundary conditions (and, if present, boundary condition objects) have been defined. The initialization within each volume can be set to a constant condition, or two states can be linearly blended in a specified direction within the volume. The beginning of this section of the VULCAN-CFD input file requires an arbitrary header line, e.g.,

SUB-DOMAIN INITIALIZATIONS

followed by "N" bounding box specifications where "N" is the number of initializations provided by the **INIT. SUB-DOMAINS** line. Each bounding box specification requires coordinate data to define the extent of the bounding box and 1 (or possibly 2) lines of flow property data to define the initialization. The specification of 2 lines of flow property data implies that the conditions will be linearly blended based on a supplied directional vector (η_x, η_y, η_z). The absence of this vector initializes the volume to constant flowfield values.

There are 2 variants of the rectangular box specification. The first is a Cartesian box where each side of the box is aligned with the x, y, and z directions. The other variant allows the box to be angled in any arbitrary fashion. The Cartesian box is just a special case of the more general box specification, but it requires less user input for situations that can accommodate a Cartesian representation. The full input specification for the Cartesian box initialization has the following form:

<arbitrary name>_BOX η_x η_y η_z
<arbitrary flow property specification header>
mass fractions (multicomponent gases only), pressure [Pa], velocity [m/s], temperature [K]
mass fractions (multicomponent gases only), pressure [Pa], velocity [m/s], temperature [K]
<arbitrary coordinate specification header>
$x_{\min}$ $y_{\min}$ $z_{\min}$
$x_{\max}$ $y_{\max}$ $z_{\max}$

Two examples that illustrate the Cartesian box initialization specification are provided below:

SUB-DOMAIN INITIALIZATIONS

CAVITY_BOX				
Pressure	U-velocity	V-velocity	W-velocity	Temperature
50662.58	0.0	0.0	0.0	1297.775
X	Y	Z		
0.0	-2.0	0.0		
20.0	0.0	0.5		
STEP_BOX	1.0	0.0	0.0	
Pressure	U-velocity	V-velocity	W-velocity	Temperature
50662.58	0.0	0.0	0.0	1297.775
50662.58	4590.118	0.0	0.0	1297.775
X	Y	Z		
0.0	3.5	0.0		
20.0	5.0	0.5		

The first example initializes a rectangular section of the domain to a zero velocity condition. The second example initializes a rectangular section of the domain so that the velocity is zero at $x_{\min}$ and linearly increased to the freestream value of 4590.118 at $x_{\max}$. This latter example is ideal for initializing the flow behind a rearward facing step.

The input specification for the generic box is the same as the Cartesian variant except for the geometry definition details. Instead of supplying 2 lines of data for the minimum and maximum extent of each coordinate direction, 4 lines of data are required to define the extent of the box. The 4 lines of data correspond to the x, y, and z coordinate values of points "D" (first line), "C" (second line), "H" (third line), and "A" (fourth line) shown in Fig. 14.

<arbitrary name>_BOX η_x η_y η_z
<arbitrary flow property specification header>
mass fractions (multicomponent gases only), pressure [Pa], velocity [m/s], temperature [K]
mass fractions (multicomponent gases only), pressure [Pa], velocity [m/s], temperature [K]
<arbitrary coordinate specification header>

x-value of "D"	y-value of "D"	z-value of "D"
x-value of "C"	y-value of "C"	z-value of "C"
x-value of "H"	y-value of "H"	z-value of "H"
x-value of "A"	y-value of "A"	z-value of "A"

Note that the coordinates specified could have originated from any of the 8 box corners provided that a right-handed coordinate system is chosen. Given these coordinates, three vectors are formed (red, green, and blue vectors shown in Fig. 14), which when added together form the box diagonal vector running from point "D" to point "F". The following examples illustrate how to replicate the Cartesian box initializations shown previously us-

ing the generic box input specification.

SUB-DOMAIN INITIALIZATIONS

CAVITY_BOX

Pressure	U-velocity	V-velocity	W-velocity	Temperature
50662.58	0.0	0.0	0.0	1297.775
X	Y	Z		
0.0	-2.0	0.0		
20.0	-2.0	0.0		
0.0	0.0	0.0		
0.0	-2.0	0.5		

STEP_BOX	1.0	0.0	0.0	
Pressure	U-velocity	V-velocity	W-velocity	Temperature
50662.58	0.0	0.0	0.0	1297.775
50662.58	4590.118	0.0	0.0	1297.775
X	Y	Z		
0.0	3.5	0.0		
20.0	3.5	0.0		
0.0	5.0	0.0		
0.0	3.5	0.5		

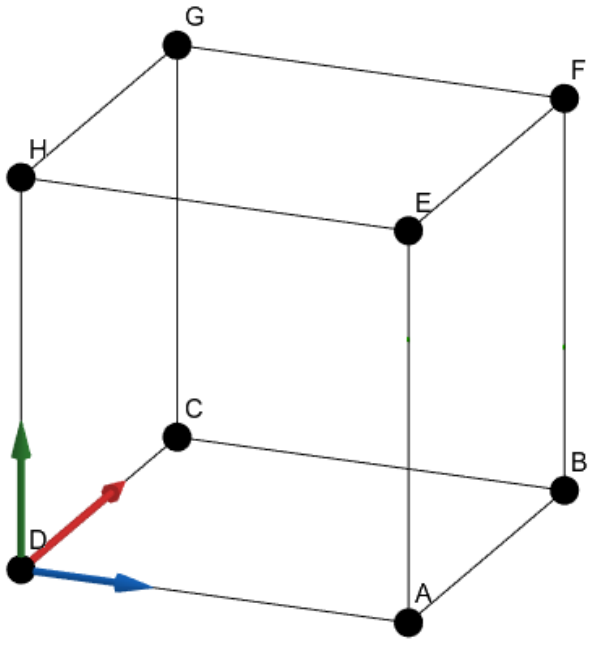


Figure 14: Image depicting the corner coordinates used to define a generic rectangular box subdomain.

The input specification for cylinders, cones, and conical frustums is identical to the box

specifications except for the geometry definition details. Here, 2 lines of data are required to define the starting and ending coordinate values of the cylinder or the cone/frustum axis. This is followed by the specification lines for the cylinder radius,

<arbitrary name>_CYLINDER η_x η_y η_z
<arbitrary flow property specification header>
mass fractions (multicomponent gases only), pressure [Pa], velocity [m/s], temperature [K]
mass fractions (multicomponent gases only), pressure [Pa], velocity [m/s], temperature [K]
<arbitrary axis coordinate specification header>
xbeg ybeg zbeg
xend yend zend
<arbitrary radius specification header>
cylinder radius

or starting and ending radii for a cone (or conical frustum), i.e.,

<arbitrary name>_CONE η_x η_y η_z
<arbitrary flow property specification header>
mass fractions (multicomponent gases only), pressure [Pa], velocity [m/s], temperature [K]
mass fractions (multicomponent gases only), pressure [Pa], velocity [m/s], temperature [K]
<arbitrary axis coordinate specification header>
xbeg ybeg zbeg
xend yend zend
<arbitrary radii specification header>
starting radius ending radius

Examples that illustrate the cylinder and cone/frustum initialization shape specifications are provided below:

SUB-DOMAIN INITIALIZATIONS

INJECTOR_CYLINDER

Air	Fuel	Pressure	U-velocity	V-velocity	W-velocity	Temperature
0.0	1.0	472293.6	389.36	0.0	0.0	235.87
X	Y	Z				
-0.02	0.0	0.0				
-0.01	0.0	0.0				
RADIUS						
0.001						

INJECTOR_CONE 1.0 0.0

Air	Fuel	Pressure	U-velocity	V-velocity	W-velocity	Temperature
0.0	1.0	472293.6	389.36	0.0	0.0	235.87
0.0	1.0	40579.41	706.11	0.0	0.0	106.94
X	Y	Z				
-0.01	0.0	0.0				

0.0 0.0 0.0
RADIUS1 RADIUS2
0.001 0.002

The cylinder specification initializes the constant area section of a fuel injector port to a constant fueling condition. The conical frustum specification initializes the conical extension of the fuel port with starting and ending flow conditions that match the conical expansion process.

NOTE: As indicated in the examples above, the absence of a directional vector on the first line of the specification implies that a single flow condition will be specified. If a vector is provided, then two separate flow conditions must be provided where the first specification corresponds to conditions at the beginning of the volume and the second specification corresponding to the conditions a the end of the volume. Given this convention, each component of the unit vector must be non-negative.

NOTE: The directional vector for box specifications requires 2 components for axisymmetric or two-dimensional simulations and 3 components for three-dimensional simulations. The directional vector for cylinder and conical initialization always has 2 components. The first is the axial direction and the second is the radial direction. The directional vector does not have to be a unit vector, since the vector is normalized after being read in.

NOTE: The specification for the box or axis coordinates requires 3 components (x,y,z) for three-dimensional simulations and 2 components (x,y) for two-dimensional or axisymmetric simulations.

NOTE: All values related to the geometry (coordinates and radii) must be provided with units that are consistent with the value specified for the **GRID SCALING FACTOR**. For instance, if the grid coordinates are supplied in units of inches, then the geometrical values provided here must also have units of inches.

11 Specified Volume Turbulence Suppression

VULCAN-CFD allows the user to define turbulence suppression regions within a specified volume that includes any combination of rectangular boxes, cylinders, cones, or conical frustums. The total number of turbulence suppression specifications is provided by the **LAMINAR SUB-DOMAINS** line located in the block specification section of the input file. This section (if present) must appear immediately after all boundary conditions (and, if present, boundary condition objects and/or subdomain volume initializations) have been defined. The beginning of this section of the VULCAN-CFD input file requires an arbitrary header line, e.g.,

SUB-DOMAIN TURBULENCE SUPPRESSION

followed by "N" bounding box specifications where "N" is the number of turbulence suppressions provided by the **LAMINAR SUB-DOMAINS** line. Each bounding box specification requires coordinate data to define the extent of the bounding box to define the region over which to suppress the turbulence. A directional vector (η_x, η_y, η_z) can also be specified to denote the flow direction for turbulence suppression enforcement that requires this information. This vector is only used by the VULCAN-CFD uncertainty quantification tool (described in Chapter 25) to distinguish between the start and end of the turbulence suppression region.

There are 2 variants of the rectangular box specification. The first is a Cartesian box where each side of the box is aligned with the x, y, and z directions. The other variant allows the box to be angled in any arbitrary fashion. The Cartesian box is just a special case of the more general box specification, but it requires less user input for situations that can accommodate a Cartesian representation. The full input specification for the Cartesian box turbulence suppression has the following form:

<arbitrary name>_BOX η_x η_y η_z
<arbitrary coordinate specification header>
$x_{\min}$ $y_{\min}$ $z_{\min}$
$x_{\max}$ $y_{\max}$ $z_{\max}$

An example that illustrates the Cartesian box turbulence suppression specification is provided below:

SUB-DOMAIN TURBULENCE SUPPRESSION

FBODY_BOX	1.0	0.0	0.0
X	Y	Z	
0.0	-2.0	0.0	
20.0	0.0	0.5	

This example suppresses the turbulence model over a rectangular section of the domain. The unit vector specification on the first line above specifies that the flow direction is aligned with the x-direction ($x_{\min}$ to $x_{\max}$). A negative value would have aligned the flow in the reverse direction ($x_{\max}$ to $x_{\min}$). Note that the specification of a unit vector is currently only

utilized when treating the turbulence suppression extent as an uncertainty with the automated VULCAN-CFD/DAKOTA uncertainty quantification toolkit. This vector determines whether to modify the "min" or "max" extents of the Cartesian BOX specification as part of the automated uncertainty process. This implies that the unit vector must be aligned with the x, y, or z axes of the Cartesian box.

The input specification for the generic box is the same as the Cartesian variant except for the geometry definition details. Instead of supplying 2 lines of data for the minimum and maximum extent of each coordinate direction, 4 lines of data are required to define the extent of the box. The 4 lines of data correspond to the x, y, and z coordinate values of points "D" (first line), "C" (second line), "H" (third line), and "A" (fourth line) shown in Fig. 14.

<arbitrary name>_BOX η_x η_y η_z
<arbitrary coordinate specification header>
x-value of "D" y-value of "D" z-value of "D"
x-value of "C" y-value of "C" z-value of "C"
x-value of "H" y-value of "H" z-value of "H"
x-value of "A" y-value of "A" z-value of "A"

Note that the coordinates specified could have originated from any of the 8 box corners provided that a right-handed coordinate system is chosen. Given these coordinates, three vectors are formed (red, green, and blue vectors shown in Fig. 14), which when added together form the box diagonal vector running from point "D" to point "F". The following example illustrates how to replicate the Cartesian box turbulence suppression shown previously using the generic box input specification.

SUB-DOMAIN TURBULENCE SUPPRESSION

FBODY_BOX

X	Y	Z
0.0	-2.0	0.0
20.0	-2.0	0.0
0.0	0.0	0.0
0.0	-2.0	0.5

The unit directional vector is currently not utilized for the generic box subdomain geometry specification. As a result, this particular geometry cannot be specified as an uncertain turbulence suppression extent for use with the VULCAN-CFD/DAKOTA uncertainty quantification toolkit.

The input specification for cylinders, cones, and conical frustums is identical to the box specifications except for the geometry definition details. Here, 2 lines of data are required to define the starting and ending coordinate values of the cylinder or the cone/frustum axis. This is followed by the specification lines for the cylinder radius,

<arbitrary name>_CYLINDER η_x η_y η_z
<arbitrary axis coordinate specification header>
xbeg ybeg zbeg

xend yend zend
<arbitrary radius specification header>
cylinder radius

or starting and ending radii for a cone (or conical frustum), i.e.,

<arbitrary name>_CONE η_x η_y η_z
<arbitrary axis coordinate specification header>
xbeg ybeg zbeg
xend yend zend
<arbitrary radii specification header>
starting radius ending radius

Examples that illustrate the cylinder and cone/frustum turbulence suppression shape specifications are provided below:

SUB-DOMAIN TURBULENCE SUPPRESSION

SURFACE_CYLINDER 1.0 0.0
X Y Z
-0.02 0.0 0.0
-0.01 0.0 0.0
RADIUS
0.001

SURFACE_CONE 1.0 0.0
X Y Z
-0.01 0.0 0.0
0.0 0.0 0.0
RADIUS1 RADIUS2
0.001 0.002

The cylinder specification suppresses the turbulence model over a cylindrical section of the computational domain. The conical specification suppresses the turbulence model over a conical section of the computational domain.

NOTE: As indicated in the examples above, the presence of a directional vector on the first line of the specification implies that a turbulence suppression direction is being provided.

NOTE: The directional vector for box specifications requires 2 components for axisymmetric or two-dimensional simulations and 3 components for three-dimensional simulations. The directional vector for cylinder and conical turbulence suppression always has 2 components. The first is the axial direction and the second is the radial direction. The directional vector must be aligned with either the axis or the radius of the cylinder/cone geometry.

NOTE: The specification for the box or axis coordinates requires 3 components (x,y,z) for three-dimensional simulations and 2 components (x,y) for two-dimensional or axisymmet-

ric simulations.

NOTE: All values related to the geometry (coordinates and radii) must be provided with units that are consistent with the value specified for the **GRID SCALING FACTOR**. For instance, if the grid coordinates are supplied in units of inches, then the geometrical values provided here must also have units of inches.

NOTE: An alternative approach for specifying turbulence suppression subdomains for structured grid simulations is also available (see Structured Grid Laminar Subblock Specification) that uses structured grid i,j,k bounds instead of physical x,y,z coordinate bounds.

12 Specified Volume Ignition Sources

VULCAN-CFD allows the user to define ignition sources within a specified volume that includes any combination of rectangular boxes, cylinders, cones, or conical frustums. The total number of ignition source specifications is provided by the **IGNITION SUB-DOMAINS** line located in the block specification section of the input file. This section (if present) must appear immediately after all boundary conditions (and, if present, boundary condition objects, subdomain volume initializations, and/or subdomain turbulence suppressions) have been defined. The ignition source can be a power value (denoted by a negative number) or an ignition temperature value (denoted by a positive number). The beginning of this section of the VULCAN-CFD input file requires an arbitrary header line, e.g.,

SUB-DOMAIN IGNITION SOURCE

followed by "N" bounding box specifications where "N" is the number of ignition sources provided by the **IGNITION SUB-DOMAINS** line. Each bounding box specification requires coordinate data to define the extent of the bounding box to define the region over which to supply the ignition source and 1 line of data to define the ignition source.

There are 2 variants of the rectangular box specification. The first is a Cartesian box where each side of the box is aligned with the x, y, and z directions. The other variant allows the box to be angled in any arbitrary fashion. The Cartesian box is just a special case of the more general box specification, but it requires less user input for situations that can accommodate a Cartesian representation. The full input specification for the Cartesian box ignition source has the following form:

<arbitrary name>_BOX
<arbitrary power/temperature specification header>
power [Watts] or temperature [K]
<arbitrary coordinate specification header>

$x_{\min}$	$y_{\min}$	$z_{\min}$
$x_{\max}$	$y_{\max}$	$z_{\max}$

An example that illustrates the Cartesian box ignition source specification is provided below:

SUB-DOMAIN IGNITION SOURCE

CAVITY_BOX
Power [Watts]
-2.5e06

X	Y	Z
0.0	-2.0	0.0
20.0	0.0	0.5

This example enforces the ignition source over a rectangular section of the domain.

The input specification for the generic box is the same as the Cartesian variant except for the geometry definition details. Instead of supplying 2 lines of data for the minimum

and maximum extent of each coordinate direction, 4 lines of data are required to define the extent of the box. The 4 lines of data correspond to the x, y, and z coordinate values of points "D" (first line), "C" (second line), "H" (third line), and "A" (fourth line) shown in Fig. 14.

<arbitrary name>_BOX
<arbitrary power/temperature specification header>
power [Watts] or temperature [K]
<arbitrary coordinate specification header>

x-value of "D"	y-value of "D"	z-value of "D"
x-value of "C"	y-value of "C"	z-value of "C"
x-value of "H"	y-value of "H"	z-value of "H"
x-value of "A"	y-value of "A"	z-value of "A"

Note that the coordinates specified could have originated from any of the 8 box corners provided that a right-handed coordinate system is chosen. Given these coordinates, three vectors are formed (red, green, and blue vectors shown in Fig. 14), which when added together form the box diagonal vector running from point "D" to point "F". The following example illustrates how to replicate the Cartesian box ignition source shown previously using the generic box input specification.

SUB-DOMAIN IGNITION SOURCE

CAVITY_BOX
Power [Watts]
-2.5e06

X	Y	Z
0.0	-2.0	0.0
20.0	-2.0	0.0
0.0	0.0	0.0
0.0	-2.0	0.5

The input specification for cylinders, cones, and conical frustums is identical to the box specifications except for the geometry definition details. Here, 2 lines of data are required to define the starting and ending coordinate values of the cylinder or the cone/frustum axis. This is followed by the specification lines for the cylinder radius,

<arbitrary name>_CYLINDER
<arbitrary power/temperature specification header>
power [Watts] or temperature [K]
<arbitrary axis coordinate specification header>

xbeg	ybeg	zbeg
xend	yend	zend

<arbitrary radius specification header>
cylinder radius

or starting and ending radii for a cone (or conical frustum), i.e.,

<arbitrary name>_CONE
<arbitrary power/temperature specification header>
power [Watts] or temperature [K]
<arbitrary axis coordinate specification header>
xbeg ybeg zbeg
xend yend zend
<arbitrary radii specification header>
starting radius ending radius

Examples that illustrate the cylinder and cone/frustum ignition source shape specifications are provided below:

SUB-DOMAIN IGNITION SOURCE

IGNITOR_CYLINDER
Temperature [K]
1500.0
X Y Z
-0.02 0.0 0.0
-0.01 0.0 0.0
RADIUS
0.001

IGNITOR_CONE
Temperature [K]
1500.0
X Y Z
-0.01 0.0 0.0
0.0 0.0 0.0
RADIUS1 RADIUS2
0.001 0.002

The cylinder specification enforces the ignition source over a cylindrical section of the computational domain. The conical specification enforces the ignition source over a conical section of the computational domain.

NOTE: The specification for the box or axis coordinates requires 3 components (x,y,z) for three-dimensional simulations and 2 components (x,y) for two-dimensional or axisymmetric simulations.

NOTE: All values related to the geometry (coordinates and radii) must be provided with units that are consistent with the value specified for the **GRID SCALING FACTOR**. For instance, if the grid coordinates are supplied in units of inches, then the geometrical values provided here must also have units of inches.

NOTE: An alternative approach for specifying ignition sources for structured grid simulations is also available (see Specified Grid Ignition Subblock Specification) that uses structured grid i,j,k bounds instead of physical x,y,z coordinate bounds.

13 Unstructured Grid Time History Specification

The time history of plot variables specified in the **PLOT FUNCTION** list can be output over a specified portion of the computational domain for unstructured grid simulations performed in a time-accurate manner. The number of time history specifications is provided by the **TIME SUB-DOMAINS** line located in the block specification section of the input file. This section (if present) must appear immediately after all boundary conditions (and, if present, boundary condition objects, subdomain volume initializations, subdomain turbulence suppressions, and/or subdomain ignition sources) have been defined. The beginning of this section of the VULCAN-CFD input file requires an arbitrary header line, e.g.,

SUB-DOMAIN TIME HISTORY I/O

followed by "N" time history specifications where "N" is the number of time history specifications provided by the **TIME SUB-DOMAINS** line. The specific information provided for each time history specification depends on the subdomain class chosen from the following list:

- Entire Computational Domain
- Specific Boundary Surface
- Planes
- Point Probes

All classes require a string that specifies the time history file name and the subdomain class. Coordinate values that define the portion of the computational domain to extract time history data are also required for some subdomain classes (e.g., planes and probes). For each time history file name provided, a separate time history file will be written for each output interval specified on the **OUTPUT TIME HISTORY** input line. The time history interval counter is appended to the file name that is provided. Note that a unique file name is not required for each time history output specification (the one exception being the point probes class, which requires unique file names for each probe specified). The data for time history specifications with the same file name are simply written out to separate Tecplot zones.

The input specification to output the time history for the entire computational domain is entered as follows:

<time_history_file_name>_ALL

where <time_history_file_name> is an arbitrary file name that can include path information. The input specification to output a time history for an external boundary of the computational domain is specified using the boundary condition group name, i.e.,

<time_history_file_name>_BC_<boundary_condition_group_name>

where <boundary_condition_group_name> is the desired group name supplied in the first column of the boundary condition specification. For example, if a time history of the data that lies on the following boundary is desired,

BCGROUPS:	**NAME**	**TYPE**	**OPTION**
	SYMMETRY	SYMM	PHYSICAL

then the time history specification line would be as follows:

<time_history_file_name>_BC_SYMMETRY

There are 2 variants of the plane subdomain class specification. The first is a Cartesian plane where a slice is cut at through the domain at constant x, y, or z stations. The other variant allows the plane to be angled in any arbitrary fashion. The Cartesian plane is just a special case of the generic plane specification, but it requires less user input for situations that can accommodate a Cartesian representation. The input for a constant x plane time history specification (y, z planes are similar) has the following form:

<time_history_file_name>_X-PLANE
<arbitrary coordinate specification header>
x-value

The generic plane specification requires 3 sets of x, y, z coordinates to define the plane and has the following form:

<time_history_file_name>_PLANE
<arbitrary coordinate specification header>

x-value$_1$	y-value$_1$	z-value$_1$
x-value$_2$	y-value$_2$	z-value$_2$
x-value$_3$	y-value$_3$	z-value$_3$

Examples that illustrate the Cartesian and generic plane specification are provided below:

SUB-DOMAIN TIME HISTORY I/O

SLICE_Y-PLANE
Y
2.5
SLICE_PLANE

X	Y	Z
1.0	0.0	0.0
0.0	1.0	0.0
0.0	0.0	1.0

The last subdomain class is the probe (or point) specification, which outputs the time history at a single cell. Unlike the other subdomain classes, this class outputs the time history

data to an ASCII Tecplot format file. All other classes output the data to a binary (.plt) Tecplot format file. An ASCII format is used for this class to allow the data to easily be loaded to external applications for further processing (e.g., a Fast Fourier Transform). The input specification for the probe subdomain class has the following form:

<time_history_file_name>_PROBE
<arbitrary coordinate specification header>
x-value *y*-value *z*-value

The *x*, *y*, and *z* values entered are used to find the nearest cell center (or boundary surface face center) for the probe data extraction. Hence, no data interpolation is involved. Also note that the time history file name provided for each probe specification must be unique.

NOTE: T-infinity must be enabled to output time history files for unstructured grid simulations.

14 Structured Grid Mapping of Boundary Conditions

The number of specifications required to map the boundary conditions to the external boundaries of each structured grid block is provided by the **FLOWBCS** line located in the block specification section. The section that houses the assignment of these conditions resides after the boundary conditions have been defined (and, if present, boundary condition objects and any subdomain specifications), but before any block-to-block connectivity specifications. The first line of input for this section is an arbitrary header line. This line is followed by "N" lines ("N" is the value provided on the **FLOWBCS** line) that map the boundary conditions to the external boundaries of each structured grid block. Each line associates a boundary condition name from the boundary condition section with a block number and spatial indices that define the portion of the external block face that the condition will be applied over. The boundary condition mappings can be specified in any order. An example illustrating how the boundary conditions are mapped to the external boundaries of structured grid blocks is provided below, along with a detailed description of each entry.

BC NAME	BLK	FACE	PLACE	DIR1	BEG	END	DIR2	BEG	END	IN-ORD
AIR-IN	1	I	MIN	J	MIN	MAX	K	MIN	MAX	0
OUTFLOW	1	I	MAX	J	MIN	MAX	K	MIN	MAX	0
COMB-WALL	1	J	MIN	I	MIN	65	K	MIN	MAX	0
COMB-WALL	1	J	MIN	I	89	MAX	K	MIN	MAX	0
SYMMETRY	1	J	MAX	I	MIN	MAX	K	MIN	MAX	0
FUEL-IN	2	J	MIN	I	MIN	MAX	K	MIN	MAX	1
PORT-WALL	2	I	MIN	J	MIN	MAX	K	MIN	MAX	0
PORT-WALL	2	I	MAX	J	MIN	MAX	K	MIN	MAX	0

BC NAME specifies one of the boundary condition group names from the boundary condition group list (see Chapter 9). This entry connects the external block boundary to a specific boundary condition.

BLK Designates the block number associated with the external block boundary.

FACE Designates the face index associated with external block boundary.

I → i-face
J → j-face
K → k-face

PLACE Designates the face index location associated with the external block boundary.

MIN → first face index
MAX → final face index

DIR1 Designates the first direction considered when defining the block boundary "window". For example, if the face index associated with the block boundary is **I**, then the index here must be either **J** or **K**

I → i-direction
J → j-direction
K → k-direction

BEG Designates the starting grid vertex value for the window in the direction that corresponds to the **DIR1** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the window for this coordinate direction.

END Designates the ending grid vertex value for the window in the direction that corresponds to the **DIR1** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the window for this coordinate direction.

NOTE: The **END** vertex value must be greater than the **BEG** vertex value.

DIR2 Designates the second direction considered when defining the block boundary "window". For example, if the face index associated with the block boundary is **I**, then the index here must be either **J** or **K**

I → i-direction
J → j-direction
K → k-direction

BEG Designates the starting grid vertex value for the window in the direction that corresponds to the **DIR2** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the window for this coordinate direction.

END Designates the ending grid vertex value for the window in the direction that corresponds to the **DIR2** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the window for this coordinate direction.

NOTE: The **END** vertex value must be greater than the **BEG** vertex value.

IN-ORD Designates the propagation order if the boundary condition is to be propagated as an initial condition into the block. The propagation order determines the order in which the proportions are to be performed. For example, consider 2 blocks connected in the streamwise direction, where block 1 is the upstream block. If the inflow boundary condition of block 1 is intended to be propagated through block 1 and into block 2, then the **IN-ORD** value for the inflow boundary condition of the first block should be 1, and the **IN-ORD** value for the block 2 side of the block-to-block interface should be 2. If a propagation is not desired, then simply place a 0 in the **IN-ORD** column.

NOTE: VULCAN-CFD provides supporting utilities for use with either the Pointwise[8] or GridPro[9] grid generation packages to export the external block boundary condition mapping section for you. These utilities should be used whenever possible to avoid the tedious (and error-prone) process of specifying these mappings by hand.

15 Structured Grid C(0) Block Interface Conditions

The number of C(0) block interface conditions (i.e., point-matched block interface or cut conditions) is provided by the **CUTBCS** line located in the block specification. The C(0) block connectivity conditions are specified in the input file after the specification of any structured grid external block boundary condition mappings. The first line of input for this section is an arbitrary header line. This line is followed by "N" line pairings ("N" is the value provided on the **CUTBCS** line) that define the connectivity for each of the block interfaces. The connectivity pairings contain 2 input lines (1 line for each side of the block-to-block interface) to define the block numbers that share the interface and the spatial indices defining how the blocks are connected. VULCAN-CFD has no restrictions on how the blocks are connected (e.g., i-faces can be connected to i, j, or k-faces), and the 2 entries for each cut pair may be provided in any order. As a result, VULCAN-CFD can handle structured grids with any number of topological singularities. A 2-D example is provided in Fig. 15, which consists of 2 structured blocks that are connected with an H-O block topology. This topology has four 3-point topological singularities located at the corners of block 1. Block 1 has the i-direction traversing from left to right, and the j-direction is bottom to top. Block 2 has the i-direction aligned circumferentially in a counterclockwise fashion, while the j-direction is aligned radially from the exterior boundary.

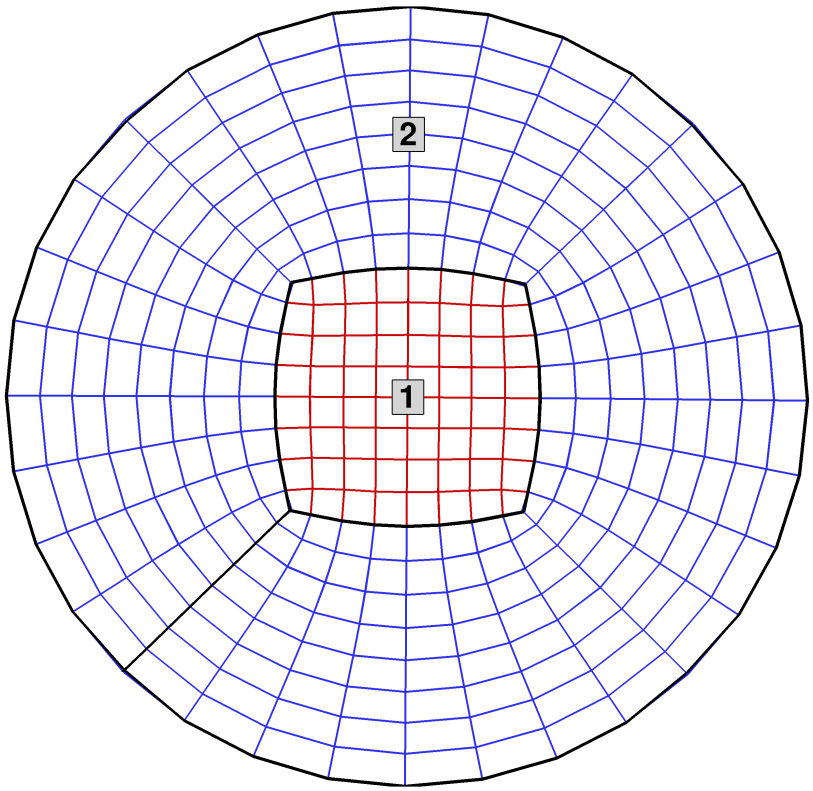


Figure 15: Two block structured grid H-O topology.

The C(0) block interface connectivity specification for this structured grid block topology (with the **CUTBCS** value set to 4) is as follows:

CUT NAME	BLK	FACE	PLACE	DIR1	BEG	END	DIR2	BEG	END	IN-ORD
CUT_1-2(a)	1	J	MIN	I	MIN	MAX	K	MIN	MAX	0
CUT_1-2(a)	2	J	MAX	I	MIN	9	K	MIN	MAX	0
CUT_1-2(b)	1	I	MAX	J	MIN	MAX	K	MIN	MAX	0
CUT_1-2(b)	2	J	MAX	I	9	17	K	MIN	MAX	0
CUT_1-2(c)	1	J	MAX	I	MAX	MIN	K	MIN	MAX	0
CUT_1-2(c)	2	J	MAX	I	17	25	K	MIN	MAX	0
CUT_1-2(d)	1	I	MIN	J	MAX	MIN	K	MIN	MAX	0
CUT_1-2(d)	2	J	MAX	I	25	MAX	K	MIN	MAX	0
CUT_2-2	2	I	MIN	J	MIN	MAX	K	MIN	MAX	0
CUT_2-2	2	I	MAX	J	MIN	MAX	K	MIN	MAX	0

A detailed description of each particular specification is provided below.

CUT NAME specifies the name (limited to 10 characters) for the cut pair. The name provided for each entry (line) of the cut pair must match. Other than this constraint, the name provided for each cut pair is arbitrary, provided that the strings **C-PERIODIC** and **P-PERIODIC** are avoided. The string **C-PERIODIC** is reserved to indicate that the block interface is spatially periodic along a given Cartesian coordinate direction, and **P-PERIODIC** is reserved to indicate that the block interface is spatially periodic in the azimuthal coordinate direction for polar (or cylindrical) computational domains.

NOTE: The P-PERIODIC specification maps the y, z values to r, θ values, where $y = z = 0$ corresponds to the axis of rotation (i.e., the x-axis must correspond to the axis of rotation).

BLK Designates the block numbers on either side of the C(0) block interface. Note that it is permissible to have a block connect to itself as shown in the example above.

FACE Designates the face index associated with each block at the block-to-block interface.

I $\rightarrow$ i-face
J $\rightarrow$ j-face
K $\rightarrow$ k-face

PLACE Designates the face index location associated with each block at the block-to-block interface.

MIN $\rightarrow$ first face index
MAX $\rightarrow$ final face index

DIR1 Designates the first direction considered when defining the block-to-block interface "window". For example, if the face index associated with the block is **I**, then the index here must be either **J** or **K**

I → i-direction
J → j-direction
K → k-direction

BEG Designates the starting grid vertex value for the window in the direction that corresponds to the **DIR1** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the window for this coordinate direction.

END Designates the ending grid vertex value for the window in the direction that corresponds to the **DIR1** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the window for this coordinate direction.

DIR2 Designates the second direction considered when defining the block-to-block interface "window". For example, if the face index associated with the block is **I**, then the index here must be either **J** or **K**

I → i-direction
J → j-direction
K → k-direction

BEG Designates the starting grid vertex value for the window in the direction that corresponds to the **DIR2** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the window for this coordinate direction.

END Designates the ending grid vertex value for the window in the direction that corresponds to the **DIR2** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the window for this coordinate direction.

IN-ORD Designates the propagation order if the conditions present in the neighboring block are to be propagated as an initial condition into the present block. The propagation order determines the order in which the proportions are to be performed. For example, consider 2 blocks connected in the streamwise direction, where block 1 is the upstream block. If the inflow boundary condition of block 1 is intended to be propagated through block 1 and into block 2, then the **IN-ORD** value for the inflow boundary condition of the first block should be 1, and the **IN-ORD** value for the block 2 side of the block-to-block interface should be 2. If a propagation is not desired, then simply place a 0 in the **IN-ORD** column.

NOTE: VULCAN-CFD provides supporting utilities for use with either the Pointwise[8] or GridPro[9] grid generation packages to export the C(0) block-to-block interface connectivity portion of the input file for you. These utilities should be used whenever possible to avoid the tedious (and error-prone) process of specifying this connectivity information by hand.

16 Structured Grid non-C(0) Block Interface Conditions

VULCAN-CFD has allowances for structured grid simulations to permit the exchange of data at block interfaces where the grid nodes do not connect in a one-to-one manner (i.e., non-C(0) interfaces). This capability can greatly simplify the process of generating structured grids in regions of complex geometry, and it also provides a simple means to refine (or coarsen) the grid across block interfaces. The number of non-C(0) block interface conditions (or patch conditions) is provided by the **PATCHBCS** line located in the block specification section. The non-C(0) block connectivity conditions are specified in the input file after the specification of the structured grid external block boundary and C(0) block connectivity sections. The first line of input for this section is an arbitrary header line. This line is followed by "N" line pairings ("N" is the value provided on the **PATCHBCS** line) that provide the input lines (1 line for each side of the block-to-block interface) containing the block numbers that straddle the interface and bounding spatial indices used to determine how the blocks are connected. In contrast to the C(0) block-to-block interface description, each non-C(0) line pair **must** be provided in a specific order. The first line of the patch data pair is the object (receiving) side and the second line is the image (transmitting) side. In other words, data will be transmitted from the image side and received by the object side of the patch pair. In addition, given that the object and image sides of the patch definition do not necessarily have the same bounds, the patch connectivity input specifications are unidirectional. In other words, two separate patch pairs must be provided for the exchange of data in both directions of the patch interface. This is contrasted with the bidirectional specification of C(0) interfaces described previously, where the exchange of data for the reverse direction can easily be deduced from the data provided. Note that VULCAN-CFD will skip the superfluous side of the patch pair when patches are specified at a block interfaces that are also region interfaces. Without true point-to-point connectivity on either side of a non-C(0) block interface, the information required to define these interface conditions cannot be obtained directly from the grid generation software. However, the creation of these interface conditions can be greatly simplified by defining each side of a non-C(0) interface as a boundary condition (with a unique boundary condition group name). This will allow each side of a non-C(0) interface to be exported as a structured grid boundary condition mapping line by the grid generation packages supported by VULCAN-CFD. These lines can then be manually matched up to define the non-C(0) interface conditions.

A 2-D example is provided in Fig. 16, which consists of 2 structured blocks that are connected with a patched interface. Block 1 has the i-direction traversing from left to right, and the j-direction is bottom to top. Block 2 has the i-direction aligned circumferentially in a counterclockwise fashion, while the j-direction is aligned radially from the exterior boundary. In this example, block 1 is simply a coarsened version of a block that provided C(0) continuity. As a result, the grid is C(0) continuous at all 4 corners of block 1, implying that the bounds of the image side of the patch can be chosen to match the bounds of the object side of the patch. This situation is not always encountered in practice, but it should be taken advantage of whenever present to simplify the logic required to determine the donor cells for the patching process. If common bounds are not present between the object and image sides of the interface, then the sum of all image sides must be defined as large as required so that the object faces are fully contained within the domain formed by the image

faces (i.e., overlaps are acceptable, but gaps are not). Otherwise, the patching process will fail to find enough image data to completely define the patch interpolation weighting coefficients for the object side of the patch.

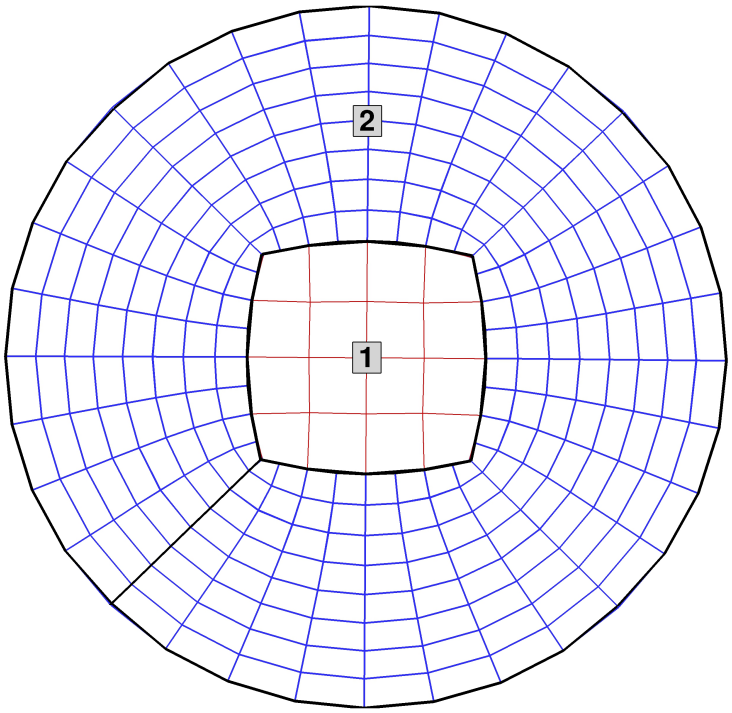


Figure 16: Two block patched grid topology.

The preferred non-C(0) block interface connectivity specification for this structured grid block topology (with the **PATCHBCS** value set to 8) is as follows:

PCH NAME	BLK	FACE	PLACE	DIR1	BEG	END	DIR2	BEG	END	IN-ORD
PCH_1-2(a)	1	J	MIN	I	MIN	MAX	K	MIN	MAX	0
PCH_1-2(a)	2	J	MAX	I	MIN	9	K	MIN	MAX	0
PCH_2-1(a)	2	J	MAX	I	MIN	9	K	MIN	MAX	0
PCH_2-1(a)	1	J	MIN	I	MIN	MAX	K	MIN	MAX	0
PCH_1-2(b)	1	I	MAX	J	MIN	MAX	K	MIN	MAX	0
PCH_1-2(b)	2	J	MAX	I	9	17	K	MIN	MAX	0
PCH_2-1(b)	2	J	MAX	I	9	17	K	MIN	MAX	0
PCH_2-1(b)	1	I	MAX	J	MIN	MAX	K	MIN	MAX	0
PCH_1-2(c)	1	J	MAX	I	MIN	MAX	K	MIN	MAX	0
PCH_1-2(c)	2	J	MAX	I	17	25	K	MIN	MAX	0
PCH_2-1(c)	2	J	MAX	I	17	25	K	MIN	MAX	0
PCH_2-1(c)	1	J	MAX	I	MIN	MAX	K	MIN	MAX	0
PCH_1-2(d)	1	I	MIN	J	MIN	MAX	K	MIN	MAX	0
PCH_1-2(d)	2	J	MAX	I	25	MAX	K	MIN	MAX	0
PCH_2-1(d)	2	J	MAX	I	25	MAX	K	MIN	MAX	0
PCH_2-1(d)	1	I	MIN	J	MIN	MAX	K	MIN	MAX	0

However, the following interface connectivity specification is also valid:

PCH NAME	BLK	FACE	PLACE	DIR1	BEG	END	DIR2	BEG	END	IN-ORD
PCH_1-2(a)	1	J	MIN	I	MIN	MAX	K	MIN	MAX	0
PCH_1-2(a)	2	J	MAX	I	MIN	MAX	K	MIN	MAX	0
PCH_2-1(a)	2	J	MAX	I	MIN	9	K	MIN	MAX	0
PCH_2-1(a)	1	J	MIN	I	MIN	MAX	K	MIN	MAX	0
PCH_1-2(b)	1	I	MAX	J	MIN	MAX	K	MIN	MAX	0
PCH_1-2(b)	2	J	MAX	I	MIN	MAX	K	MIN	MAX	0
PCH_1-2(b)	2	J	MAX	I	9	17	K	MIN	MAX	0
PCH_2-1(b)	1	I	MAX	J	MIN	MAX	K	MIN	MAX	0
PCH_1-2(c)	1	J	MAX	I	MIN	MAX	K	MIN	MAX	0
PCH_1-2(c)	2	J	MAX	I	MIN	MAX	K	MIN	MAX	0
PCH_2-1(c)	2	J	MAX	I	17	25	K	MIN	MAX	0
PCH_2-1(c)	1	J	MAX	I	MIN	MAX	K	MIN	MAX	0
PCH_1-2(d)	1	I	MIN	J	MIN	MAX	K	MIN	MAX	0
PCH_1-2(d)	2	J	MAX	I	MIN	MAX	K	MIN	MAX	0
PCH_2-1(d)	2	J	MAX	I	25	MAX	K	MIN	MAX	0
PCH_2-1(d)	1	I	MIN	J	MIN	MAX	K	MIN	MAX	0

A detailed description of each particular specification is provided below.

PCH NAME specifies the name (limited to 10 characters) for the patch pair. The name provided for each entry (line) of the patch pair must match. Other than this constraint, the name provided for each patch pair is arbitrary.

BLK Designates the block numbers on either side of the non-C(0) block interface.

FACE Designates the face index associated with each block at the block-to-block interface.

I → i-face
J → j-face
K → k-face

PLACE Designates the face index location associated with each block at the block-to-block interface.

MIN → first face index
MAX → final face index

DIR1 Designates the first direction considered when defining the block-to-block interface "window". For example, if the face index associated with the block is **I**, then the index here must be either **J** or **K**

I → i-direction

J → j-direction
K → k-direction

BEG Designates the starting grid vertex value for the window in the direction that corresponds to the **DIR1** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the window for this coordinate direction.

END Designates the ending grid vertex value for the window in the direction that corresponds to the **DIR1** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the window for this coordinate direction.

NOTE: The **END** vertex value must be greater than the **BEG** vertex value.

DIR2 Designates the second direction considered when defining the block-to-block interface "window". For example, if the face index associated with the block is **I**, then the index here must be either **J** or **K**

I → i-direction
J → j-direction
K → k-direction

BEG Designates the starting grid vertex value for the window in the direction that corresponds to the **DIR2** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the window for this coordinate direction.

END Designates the ending grid vertex value for the window in the direction that corresponds to the **DIR2** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the window for this coordinate direction.

NOTE: The **END** vertex value must be greater than the **BEG** vertex value.

IN-ORD Designates the propagation order if the conditions present in the neighboring block are to be propagated as an initial condition into the present block. The propagation order determines the order in which the propagations are to be performed. For example, consider 2 blocks connected in the streamwise direction, where block 1 is the upstream block. If the inflow boundary condition of block 1 is intended to be propagated through block 1 and into block 2, then the **IN-ORD** value for the inflow boundary condition of the first block should be 1, and the **IN-ORD** value for the block 2 side of the patched interface should be 2. If a propagation is not desired, then simply place a 0 in the **IN-ORD** column. Note that propagation conditions should only be specified on the object (receiving) lines of the patch interface conditions, i.e., the first line of each patch pair.

NOTE: Given the non-C(0) nature of patch interfaces, most grid generation packages cannot identify non-C(0) block-to-block interface connectivities. As a result, these conditions must be specified as an external boundary condition, which will be written out to the external block boundary condition mapping section of the input file. It is recommended that

these conditions be assigned a common generic name (e.g., PATCH), so that these lines can easily be identified when removing them from the boundary condition mapping section for placement into the non-C(0) block interface section. From there the lines must be appropriately matched up and mirrored as required to define the non-C(0) connectivity.

The determination of the interpolation weighting coefficients for the non-C(0) patching process involves three distinct tasks. The first task is the identification and gathering of the faces involved with each patch interface. This task is initiated by parsing the non-C(0) block interface connectivity specification of the VULCAN-CFD input file described above. The block faces on either side of the non-C(0) interface are referred to as either the "A" or "B" mesh, respectively. The "B" mesh is considered the donor (or image) mesh, and the "A" mesh is the receiving (or object) mesh. The second task is the identification of all cells in the "B" mesh that overlap a given "A" mesh cell. The final task is the determination of the area fraction for the "A" mesh cells that result from each overlapped "B" mesh cell, which are saved and output to the patch coefficient file. A bad cell list is created (if necessary) that contains information for the cells that have not been properly patched. VULCAN-CFD execution will stop if the **bad_cell.list** file is present. At this point, the user should examine the contents of this file to decide if further refinements in the patch control parameters should be pursued, or if any errors are present in the non-C(0) interface specifications. If the patching errors are not catastrophic (i.e., if some amount of area overlap was found for all of the "A" mesh cells), then no further refinements are required, and executing VULCAN-CFD a second time will simply use the patch file that was created. However, the user has the option at this point to interactively execute the utility that computes the interpolation weighting coefficients via the command:

patcher

to attempt to refine the patching process, and reduce the number of poorly patched interfaces. When executed in interactive mode, the first question that is posed is whether or not to adjust the tolerances on a face-by-face basis for the interfaces that encountered problems. Selecting "y" for this prompt will provide the finest level of control over the patching process, but can be tedious if lots of problems are encountered. Selecting "n" will globally change the default values, and the entire patching process will be repeated based on these changes. If problems were encountered at more than a couple of interfaces, then this option might be a better choice to start with. The next prompt:

```
---------------- Tolerances ----------------
1) Point Outside UV Face      1.00000000E-02
2) Coinciding Point Distance  1.00000000E-04
3) Smallest Area Fragment     1.00000000E-06
q) Quit and Save
```

allows changes to be made to the three parameters that influence the patching process:

(1) Point Outside UV Face → defines the tolerance for determining whether a "B" mesh cell vertex projects onto the given "A" mesh cell. The tolerance entered for

this parameter must lie between 0.0 and 1.0, but typical values used in practice range between 0.001 and 0.05. The default is 0.01.

(2) Coinciding Point Distance → defines the tolerance for determining if a "B" mesh cell vertex coincides with an "A" mesh vertex. This tolerance is scaled relative to a measure of the smallest cell length present on the "B" mesh portion of the patch interface, i.e., tol = MIN(SQRT(smallest "B" mesh area)×tol, tol). The tolerance entered for this parameter must be greater than 0.0, with typical values used in practice ranging between 1.0e-6 and 0.05. The default is 1.0e-4.

(3) Smallest Area Fragment → defines the smallest area fragment percentage of a "B" mesh cell to be considered when projected on the "A" mesh cell. Area fragment percentages below this value are ignored. This tolerance rarely needs to be altered from its default value unless a large number of cells on the "B" mesh are patching to a single cell on the "A" mesh. The tolerance entered for this parameter must be greater than 0.0, with typical values ranging between 1.0e-6 and 0.01. The default is 1.0e-6.

To change any of these parameters, simply select the parameter to change (i.e., enter 1, 2, or 3) followed by the new value for that parameter. When all of the desired changes have been made, enter "q" to save these changes. The final item prompted for is the name of the VULCAN-CFD input file, so that the name of the grid file and the patch interface specifications can be extracted.

17 Structured Grid Laminar Subblock Specification

The turbulence model can be deactivated (or suppressed) over a specified portion of the computational domain for structured grid simulations via user-specified subblocks, where the intent is to provide some control over where (and how) the flow transitions from laminar to turbulent flow. The number of subblock specifications is designated by the **LAMINAR SUB-BLOCKS** line located in the block specification section of the VULCAN-CFD input file. The first line of input for this section is an arbitrary header line. This line is followed by "N" lines ("N" is the value provided on the **LAMINAR SUB-BLOCKS** line) that provide the extent of each subblock in computational space. An example illustrating how the turbulence suppression subblocks are prescribed is provided below:

TRN NAME	**BLK**	**DIR1**	**BEG**	**END**	**DIR2**	**BEG**	**END**	**DIR3**	**BEG**	**END**	**TYPE**
BODY L.E.	1	I	MIN	65	J	MIN	MAX	K	MIN	MAX	LAM

A detailed description of each particular specification is provided below.

TRN NAME specifies the name (limited to 12 characters) for the turbulence suppression subblock.

BLK Designates the block number associated with the turbulence suppression subblock.

DIR1 Designates the first direction considered when defining the turbulence suppression subblock.

I → i-direction
J → j-direction
K → k-direction

BEG Designates the starting grid vertex value for the subblock in the direction that corresponds to the **DIR1** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the subblock for this coordinate direction.

END Designates the ending grid vertex value for the subblock in the direction that corresponds to the **DIR1** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the subblock for this coordinate direction.

DIR2 Designates the second direction considered when defining the turbulence suppression subblock.

I → i-direction
J → j-direction
K → k-direction

BEG Designates the starting grid vertex value for the subblock in the direction that corresponds to the **DIR2** index. The generic strings MIN and MAX can be used to denote the

first and final index locations that define the subblock for this coordinate direction.

END Designates the ending grid vertex value for the subblock in the direction that corresponds to the **DIR2** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the subblock for this coordinate direction.

DIR3 Designates the third direction considered when defining the turbulence suppression subblock.

I → i-direction
J → j-direction
K → k-direction

BEG Designates the starting grid vertex value for the subblock in the direction that corresponds to the **DIR3** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the subblock for this coordinate direction.

END Designates the ending grid vertex value for the subblock in the direction that corresponds to the **DIR3** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the subblock for this coordinate direction.

NOTE: The **END** vertex value must be greater than the **BEG** vertex value for each coordinate direction of the subblock definition.

TYPE Designates how the turbulence suppression is handled. Currently, the only turbulence suppression type available is **LAM**, which fully suppresses the turbulence within the subblock.

NOTE: Only one turbulence suppression type is currently available, but this column is present as a placeholder for options that can provide some control of the lag response when transitioning from laminar to turbulent flow.

NOTE: This approach to specifying turbulence suppression subdomains offers the same functionality as the Specified Volume Turbulence Suppression, but uses structured grid i,j,k bounds instead of physical x,y,z coordinate bounds.

18 Structured Grid Ignition Subblock Specification

Ignition regions can be defined over a specified portion of the computational domain for structured grid simulations via user-specified subblocks to provide a source of energy to initiate the combustion process. The source of energy can be supplied by an elevated temperature value, or as an ignition power source (e.g., spark plug) value. The number of subblock specifications is designated by the **IGNITION SUB-BLOCKS** line located in the block specification section of the VULCAN-CFD input file. The first line of input for this section is an arbitrary header line. This line is followed by "N" lines ("N" is the value provided on the **IGNITION SUB-BLOCKS** line) that provide the extent of each subblock in computational space. An example illustrating how the ignition subblocks are prescribed is provided below:

IGN NAME	**BLK**	**DIR1**	**BEG**	**END**	**DIR2**	**BEG**	**END**	**DIR3**	**BEG**	**END**	**T/P VAL**
SPARK	1	I	41	49	J	MIN	21	K	25	33	1800

A detailed description of each particular specification is provided below.

IGN NAME specifies the name (limited to 12 characters) for the ignition subblock.

BLK Designates the block number associated with the ignition subblock.

DIR1 Designates the first direction considered when defining the ignition subblock.

I → i-direction
J → j-direction
K → k-direction

BEG Designates the starting grid vertex value for the subblock in the direction that corresponds to the **DIR1** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the subblock for this coordinate direction.

END Designates the ending grid vertex value for the subblock in the direction that corresponds to the **DIR1** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the subblock for this coordinate direction.

DIR2 Designates the second direction considered when defining the ignition subblock.

I → i-direction
J → j-direction
K → k-direction

BEG Designates the starting grid vertex value for the subblock in the direction that corresponds to the **DIR2** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the subblock for this coordinate direction.

END Designates the ending grid vertex value for the subblock in the direction that corresponds to the **DIR2** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the subblock for this coordinate direction.

DIR3 Designates the third direction considered when defining the ignition subblock.

I → i-direction
J → j-direction
K → k-direction

BEG Designates the starting grid vertex value for the subblock in the direction that corresponds to the **DIR3** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the subblock for this coordinate direction.

END Designates the ending grid vertex value for the subblock in the direction that corresponds to the **DIR3** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the subblock for this coordinate direction.

NOTE: The **END** vertex value must be greater than the **BEG** vertex value for each coordinate direction of the subblock definition.

T/P VAL Designates the temperature or power source value used to initiate ignition. The specification of a temperature or power value is differentiated based on the sign provided.

> 0 → value specified is the ignition temperature in degrees Kelvin
< 0 → value specified is the ignition power source output in Watts

NOTE: The temperature or power source value must be provided as an integer. This limitation should not pose a problem because any temperature or power value supplied should be orders of magnitude larger than unity to have any impact on the ignition process.

If an ignition temperature is imposed, then the supplied temperature value is utilized to provide energy addition in the subblock through the following process:

(1) The cell internal energy, e, is computed given the current values for total energy (E), velocity (u, v, w) and, if applicable, the turbulence kinetic energy (k) from the expression: $e = E - 0.5\,(u^2 + v^2 + w^2) - k$.

(2) The static temperature is then computed given the value for e, and the current species mass fraction values (if applicable).

(3) If the resulting temperature is less than the supplied ignition temperature, then the temperature in the cell is set to the ignition temperature. Otherwise, the cell properties are not altered and the remaining steps are skipped (implying that the mixture has already ignited in this particular cell).

(4) A new density value is obtained from the perfect gas law given the ignition temperature and the current values for pressure and species mass fraction.

(5) Finally, the conserved flow variables are updated based on the ignition temperature and updated density values with all remaining primitive variables fixed to their current state (i.e., energy is being added as a constant pressure process).

If an ignition power value is imposed, then the specified power is introduced into each of the requested cells as a source term to the total energy equation. The source term is simply the power supplied divided by the total volume enclosed by the cells that occupy the ignition region. If a single ignition region spans multiple grid blocks (or is simply disconnected in physical space), then a common ignition subblock name must be used to indicate that the subblock specification is a subset of a larger ignition region. Hence, the ignition subblock name is the controlling parameter that defines each ignition region. Note that the same power value must be provided for each subblock specification used to define a given ignition region. The specification of the ignition source in this manner is useful when the desire is to model the effectiveness of a given ignition device (e.g., a spark plug or some other power source). If the goal is simply to initiate combustion, then specifying an ignition temperature is typically more convenient.

NOTE: The spatial extents provided for each ignition region are not allowed to overlap/extend into other ignition region subvolumes. This will be flagged as an error due to the ambiguity introduced when applying the power value to the overlapped cells.

NOTE: The introduction of ignition control regions may decrease the robustness of the simulation due to the sudden localized increase in energy that is introduced. If the goal is to simply ignite the flow by any means necessary, then temporarily reducing the activation temperature of the chain initiating steps of the kinetic model may be a better strategy. Another robust option for simulations that utilize a two-equation turbulence model is to temporarily invoke the eddy breakup model described in the turbulence chemistry model specification section of Chapter 8.

NOTE: The specification of ignition regions is intended (as the name implies) to be a means to ignite chemically active flow. However, this functionality can also be used to introduce localized regions of heat addition for non-reacting simulations as well.

NOTE: This approach to specifying turbulence suppression subdomains offers the same functionality as the Specified Volume Ignition Sources, but uses structured grid i,j,k bounds instead of physical x,y,z coordinate bounds.

19 Structured Grid Time History Subblock Specification

The time history of plot variables specified in the **PLOT FUNCTION** list can be output over a specified portion of the computational domain for structured grid simulations performed in a time-accurate manner via user-specified subblocks. The number of subblock specifications is designated by the **TIME HISTORY I/O** line located in the block specification section of the VULCAN-CFD input file. The first line of input for this section is an arbitrary header line. This line is followed by "N" lines ("N" is the value provided on the **TIME HISTORY I/O** line) that provide the extent of each subblock in computational space. An example illustrating how the subblocks are prescribed for time history output is provided below:

TIM NAME	**BLK**	**DIR1**	**BEG**	**END**	**DIR2**	**BEG**	**END**	**DIR3**	**BEG**	**END**
SHK-TUBE	1	I	MIN	MAX	J	MIN	MIN	K	MIN	MIN
SHK-TUBE	2	I	MIN	MAX	J	MIN	MIN	K	MIN	MIN

A detailed description of each particular specification is provided below.

TIM NAME specifies the name (limited to 12 characters) for the time history subblock.

BLK Designates the block number associated with the time history subblock.

DIR1 Designates the first direction considered when defining the time history subblock.

I → i-direction
J → j-direction
K → k-direction

BEG Designates the starting grid vertex value for the subblock in the direction that corresponds to the **DIR1** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the subblock for this coordinate direction.

END Designates the ending grid vertex value for the subblock in the direction that corresponds to the **DIR1** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the subblock for this coordinate direction.

DIR2 Designates the second direction considered when defining the time history subblock.

I → i-direction
J → j-direction
K → k-direction

BEG Designates the starting grid vertex value for the subblock in the direction that corresponds to the **DIR2** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the subblock for this coordinate direction.

END Designates the ending grid vertex value for the subblock in the direction that corresponds to the **DIR2** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the subblock for this coordinate direction.

DIR3 Designates the third direction considered when defining the time history subblock.

I → i-direction
J → j-direction
K → k-direction

BEG Designates the starting grid vertex value for the subblock in the direction that corresponds to the **DIR3** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the subblock for this coordinate direction.

END Designates the ending grid vertex value for the subblock in the direction that corresponds to the **DIR3** index. The generic strings MIN and MAX can be used to denote the first and final index locations that define the subblock for this coordinate direction.

NOTE: The **END** vertex value must be greater than or equal to the **BEG** vertex value for each coordinate direction of the subblock definition.

NOTE: The time history files are only exported when the finest grid level is reached for simulations that invoke coarse-to-fine grid sequencing.

NOTE: The time history files are currently only output to PLOT3D format files (i.e., **PLOT3D OUTPUT** must be enabled), and individual PLOT3D format files are written at each time interval for every subdomain defined above. For ease of postprocessing, these files should be merged using the **time_merge** utility described in Chapter 25. This utility will either merge all of the files into a single ASCII Tecplot file (as separate Tecplot zones), or alternatively merge all of the subdomains into separate PLOT3D function files for each time interval. The first option is convenient for users of Tecplot, while the latter is better suited for users of Fieldview.

20 Output Files

VULCAN-CFD generates several output data files for the purpose of monitoring convergence as well as solution postprocessing. Volumetric flowfield data files are exported in PLOT3D (structured grids only), VTK (unstructured grids only) CGNS, or Tecplot binary formats. The selection of the desired format as well as the specific flow quantities that are available for output are described in the output control data section of Chapter 8. In addition to volumetric data, a surface data file is generated by the postprocessor whenever solid surfaces are present. This file will be written in both Tecplot binary (vulcan.loads.plt) and legacy VTK (vulcan.loads.vtk) formats in the unpartitioned grid state for unstructured grid simulations when T-infinity is enabled. For structured grid simulations (or unstructured grid simulations with T-infinity disabled) the file will be an ASCII Tecplot format file (**vulcan.loads.tec**) written in the partitioned grid state. The surface loads file resides in the **Plot_files** directory and contains the following properties for each surface cell face:

d_w	→	distance from the surface to the surface cell center [m]
y^+	→	"law of the wall" coordinate value at the surface cell center
u_τ	→	friction velocity [m/s]
Area	→	surface face area [m^2]
Pressure Force	→	pressure force vector components [N]
Shear Force	→	viscous shear force vector components [N]
Heat Transfer Rate	→	surface heat load [W]
Wall Temperature	→	surface temperature [K]

Surface integrated values of the pressure forces, shear forces, and heat loads are also output to the screen (or screen output file) for each surface boundary condition and any boundary condition objects derived from surface boundary conditions.

Data related to simulation convergence are output to an ASCII Tecplot format data file named **vulcan.ifam_his_#.tec** (where "#" is the region number) for elliptic regions. This file resides in the working directory of the VULCAN-CFD simulation and is continuously updated as the solver progresses, allowing the convergence of the simulation to be monitored in real-time. The following information is written to this file,

CFL	→	Courant-Friedrichs-Lewy number
$\|\|$ Residual Error $\|\|_2$	→	$\log_{10}(L_2$ residual error norm)
Mass Flow Rate Error	→	mass flow rate error, (mass out - mass in) / (mass in) [%]
Integrated Total Force	→	integrated surface force vector components [N]
Integrated Force Moment	→	integrated surface moment vector components [N]
Heat Transfer Rate	→	integrated surface heat load [W]
Integrated Shear Force	→	integrated shear force vector components [N]

and if requested in the VULCAN-CFD input file (details provided in the output control data section of Chapter 8), some combination of the following are also written to this file:

Total Pressure Loss	→	mass flux weighted total pressure loss [%]
Mixing Efficiency	→	mixing efficiency
Combustion Efficiency	→	combustion efficiency
Fuel Penetration Height	→	fuel penetration height [m]

Integral convergence behavior specific to each inflow, outflow, and surface boundary are also exported to files in the **BC_files** directory for elliptic regions. The integrated surface loads are exported at each solid surface boundary, while integrated total mass flux, momentum component fluxes, and total enthalpy flux are exported at each inflow and outflow boundary. Finally, the convergence history for each region (including parabolic regions) are output by the postprocessor to the file **vulcan.res.tec** in the working directory of the VULCAN-CFD simulation. This file contains both the non-normalized and normalized $\log_{10}(\|$ Residual Error $\|_2)$, where the normalized value is based on the maximum $\|$ Residual Error $\|_2$.

21 Troubleshooting Simulations

This chapter provides some guidance to help with troubleshooting a simulation that is failing or converging poorly. There are a multitude of reasons that can cause a simulation to fail, but the list below provides a process that should help to resolve many of the issues that occur in practice:

- Check to ensure that all boundary conditions have been specified properly and that the flowfield has been initialized as intended. It is always a good idea when attempting a simulation for the first time to set the number of iteration cycles to zero, and execute the VULCAN-CFD solver and postprocessor to generate volumetric plot files, so that the flowfield initialization can be visualized.

- Take advantage of the various options available for providing improved initial conditions. This is particularly important for supersonic and hypersonic flow simulations. For instance, use the boundary layer initialization option for no-slip wall conditions (see Chapter 9) to mitigate the discontinuous jump condition formed at the intersection of the freestream initialization and the no-slip surfaces. This feature can also be used to provide a better initialization of base regions, recessed cavity regions, and injection ports by significantly reducing the velocity values relative to the freestream. Alternatively, specific initializations can be applied within user-specified bounding volumes as described in Chapter 10. The boundary condition propagation feature that is available for structured grid simulations (see Chapter 14) provides even more control over the initialization process. The nozzle initialization feature (available for structured grid simulations via the **P0_T0_IN** boundary condition) is also very beneficial for geometries that contain converging and/or diverging nozzle components.

- If problems still persist even after the initializations and boundary conditions have been set as intended, then halt the simulation and postprocess the solution a few iteration cycles before the problem appears, and examine the plot file data to attempt to locate where the problem is occurring. The iteration statistics that are output by VULCAN-CFD may help in this regard as well. Things to look for include: unrealistically large Mach numbers, very low temperatures, and for turbulent flow simulations, excessive eddy viscosity to molecular viscosity ratios. Pay particular attention to the boundaries of the computational domain. Examples of things to look for near boundaries are the presence of recirculation at an inflow or outflow boundary, subsonic inflow specifications that have become supersonic, shock systems propagated to inflow boundaries, etc. Also check the grid quality in and around the troublesome region to see if the problem may be grid related. Finally, for turbulent flow simulations, ensure that the y^+ value is appropriate (e.g., unity or less for integrate-to-wall boundary conditions). The minimum and maximum y^+ values and their locations are reported in the iteration statistics, but if need be, the y^+ distribution on all no-slip surfaces can be visualized by loading the **vulcan.loads.tec** file into Tecplot.

- If the problem does not appear to be grid related, then consider a more conservative CFL schedule. Also make sure to start the simulation with a 1^{st}-order advection scheme (see the **1ST-ORD SWITCH** option in the region control specification section), since a more dissipative algorithm has a better chance of surviving the initial

transients. Structured grid simulations will typically also benefit from coarse-to-fine grid sequencing during the simulation startup process for the same reasons.

When performing steady-state simulations that are not converging well, consider the following options:

- Ensure that all boundary conditions are well-posed (e.g., check for any mixed subsonic/supersonic flow conditions at inflow and outflow boundaries). Also, for turbulent flow simulations, ensure that the y^+ value is appropriate (e.g., unity or less for integrate-to-wall conditions). Finally, check the quality of the grid in key areas to ensure that the grid is fine enough and appropriately clustered to resolve the features present in the flow.

- If limiters are activated for the advection scheme, try other variants. The smooth non-TVD limiters are typically less susceptible to "limiter buzz", and some TVD limiters are more prone to this phenomenon than others.

- If an implicit time advancement scheme is being used, try reducing the CFL values. Also, for some problems switching from a local time step scheme to a specified time step scheme can improve convergence (just be careful to choose a time step value that is high enough to allow the flowfield to evolve at a reasonable rate, but not so high that instabilities arise). This option is better suited for simulations that do not employ high aspect ratio grids, e.g., grids generated for wall function usage or flowfields that do not contain solid surfaces.

- If none of the above offers any significant improvement in the iterative convergence, then this may indicate that the simulation simply won't support a steady-state solution. This is not an uncommon occurrence for flows with massive flow separation. In this case, switch to a time-accurate solution procedure and continue iterating until the solution has reached a statistically-stationary state. At that point, activate the **UPDATE TIME AVERAGE** input option to force a running time average of the unsteady data. If the unsteadiness is periodic (or nearly so), then the number of time steps required to obtain a good time average will be limited to a value that covers a couple of the periodic cycles. If the unsteadiness is more chaotic, then an appreciable number of time steps will be required to converge out the time-averaged statistics.

22 Thermodynamic and Transport Model Database Formats

The species thermodynamic properties that are required for noncalorically perfect gas simulations are provided as McBride polynomial fits,[32,33] while the species transport properties are provided as either Sutherland law[30] or McBride[32,33] polynomial fits. The curve fit coefficients are supplied to the simulation through the use of database files. VULCAN-CFD currently accommodates two different database formats for the specification of the species thermodynamic and transport properties; the NASA Chemical Equilibrium with Applications (CEA) software format, and a format specific to VULCAN-CFD. The CEA thermodynamic database file format is more general than the VULCAN-CFD format, as it allows for any number of curve fit temperature intervals, and the bounds of each temperature interval can be specified for each species independently. The VULCAN-CFD format requires the same number of temperature intervals to be used for the entire database, and the temperature bounds for each interval must be the same for each species. This simplification provides a slightly more computationally efficient framework (on the order of 7-8% in most cases).

The CEA thermodynamic database file format allows for any number of header lines that begin with "!". The start of the database is indicated by the string "ther", which must be followed by the default (global) temperature interval bounds with an optional trailing character string that specifies the date that the database was released. The remaining lines contain the thermodynamic data specifications for each species, which must adhere to the format provided in Table 1. Note that the CEA database file format allows for species entries that are in a condensed state (either liquid or solid), but VULCAN-CFD will ignore these entries. The CEA format also separates the species into products and reactants (product species are listed first, followed by reactant species). VULCAN-CFD does not distinguish between the two, and instead will simply parse the entire database when searching for matching species names. Further details on the CEA thermodynamic database file format can be found elsewhere.[33] The CEA transport database file format also allows for any number of header lines that begin with "!", and for this file, the start of the database is indicated by the string "tran". The remaining lines contain the transport data specifications for each species, as indicated by the format provided in Table 2. Note that the specification of the temperature intervals must be in ascending order, and only pure species transport property lines will be considered by VULCAN-CFD (i.e., binary interaction specifications are ignored). The CEA thermodynamic and transport database file formats are specified in the VULCAN-CFD input file as follows:

```
SPECIES THERMODYNAMIC DATA FORMAT  1.0
$vulcanpath/Data_base/therm_cea.inp
SPECIES TRANSPORT DATA FORMAT  1.0
$vulcanpath/Data_base/trans_cea.inp
```

The native VULCAN-CFD format gas database houses both the thermodynamic and transport data (in Sutherland law form) in a single file, and is specified in the input file as

```
SPECIES THERMODYNAMIC DATA FORMAT  0.0
$vulcanpath/Data_base/gas_mod.Lewis_#
```

Table 1: CEA Species Thermodynamic Database Format.

Record	Database Entry	Format	Columns
1	Species name	A15	1-15
	Optional comment	A65	16-80
2	No. of temperature intervals	I2	1-2
	Optional identification code	A6	4-9
	Chemical formula	5(A2,F6.2)	11-50
	Phase indicator (0 for gas)	I1	52
	Molecular weight [g/mol]	F13.7	53-65
	Δh_f [J/mol] at 298.15	F15.3	66-80
3	Temperature interval bounds [K]	2(F10.3)	2-21
	No. of coefficients for C_p	I1	23
	Temperature exponents for C_p	7(F5.1)	24-58
	0.0 (reserved for future use)	F5.1	59-63
	$h(298.15) - h(0)$ [J/mol]	F15.3	66-80
4	First 5 coefficients for C_p	5(ES16.9)	1-80
5	Last 2 coefficients for C_p	2(ES16.9)	1-32
	Blank space (reserved for future use)		33-48
	h and s constants of integration	2(ES16.9)	49-80
——	Repeat 3, 4, and 5 for each interval	——	——

Table 2: CEA Species Transport Database Format.

Record	Database Entry	Format	Columns
1	Species name	A16	1-16
	2^{nd} species name or blank space[a]	A16	17-31
	V (for molecular viscosity)	A1	35
	No. of V temperature intervals (0 - 3)	I1	36
	C (for thermal conductivity)	A1	37
	No. of C temperature intervals (0 - 3)	I1	38
	Optional comment	A40	41-80
2	V (if viscosity fits are provided)	A1	2
	Temperature interval bounds [K]	2(F8.1)	3-18
	The 4 coefficients for viscosity	4(E15.8)	21-80
——	Repeat record 2 for each interval	——	——
N[b]	C (if conductivity fits are provided)	A1	2
	Temperature interval bounds [K]	2(F8.1)	3-18
	The 4 coefficients for conductivity	4(E15.8)	21-80
——	Repeat record N for each interval	——	——

[a] 2^{nd} species name of binary pair or blank space for a pure species

[b] N is one greater than the number of molecular viscosity coefficient lines

where "#" is 1, 2, or 3 for the 1-interval, 2-interval, or 3-interval database files. If the efficiency offered by this database format is desired for use with the more accurate CEA transport property fits (instead of the Sutherland law), then the following line should also be present:

SPECIES TRANSPORT DATA FORMAT 1.0
$vulcanpath/Data_base/trans_cea.inp

The VULCAN-CFD database format for the species thermodynamic and transport property coefficients consists of a header section followed by the thermodynamic and transport coefficient specifications for each species. The header section contains the following information:

DB_NCS, DB_VER_NO
DB_THRM_INT, NUM_THRM_COEF, NUM_VISC_COEF, NUM_COND_COEF
TMIN(N), TMAX(N) ← specified for each curve fit interval on separate lines

where:

DB_NCS:	no. of species in the database file
DB_VER_NO:	VULCAN-CFD database format version no.
DB_THRM_INT:	no. of intervals used for the thermodynamic fits
NUM_THRM_COEF:	no. of coefficients in the thermodynamic fits
NUM_VISC_COEF:	no. of coefficients in the molecular viscosity fits
NUM_COND_COEF:	no. of coefficients in the molecular conductivity fits
TMIN(N):	minimum temperature [K] for each fit interval "N"
TMAX(N):	maximum temperature [K] for each fit interval "N"

The remaining lines contain the thermodynamic and transport data specifications for each species. An arbitrary number of comment lines (i.e., lines that begin with a "!") are permitted before each particular species data specification.

DB_SYMB
DB_MW
DB_H0
!
(DB_THRM(N,M), M = 1,NUM_THRM_COEF)
!
(DB_VISC(M), M = 1,NUM_VISC_COEF)
!
(DB_COND(M), M = 1,NUM_VISC_COEF)

where:

DB_SYMB:	species name (chemical symbol)
DB_MW:	molecular weight [g/mol]

DB_H0: $h(298.15) - h(0)$ [J/mol]
DB_THRM(N,M): thermodynamic polynomial fit coefficients $a_1, \ldots$ (a_5 or a_7), b_1, b_2
DB_VISC(M): Sutherland law viscosity coefficients μ° [kg/(m s)], T°_μ [K], S_μ [K]
DB_COND(M): Sutherland law conductivity coefficients λ° [W/(m K)], T°_λ [K], S_λ [K]

Note that the thermodynamic polynomial fit coefficients (and the preceding comment line) must be specified for each thermodynamic curve fit interval "N" in ascending order.

The databases shipped with VULCAN-CFD all have consistent sources for the thermodynamic properties of each species, and to the extent possible, the source of the thermodynamic data for each of the species is provided in the comment sections. If a species is needed that is not already present in the database files, then the following utility codes described in the database utility codes section of Chapter 25 can help with the task of adding them to the database files:

ls_fit → performs a least squares fit between various thermodynamic and transport property functional forms
mix_fit → combines individual species properties into a single mixture species fit (e.g., combine O_2 and N_2 properties to form a single air species fit)

Note that **mix_fit** can also be used to produce a file that contains the species properties as a function of temperature, which can be imported into Tecplot for plotting purposes. This is often useful to ensure that the species properties have been added to the database files properly. As a final note, VULCAN-CFD includes a utility (**perf_ext**) that can be used to extract performance metrics from simulations of air-breathing propulsion systems. This utility uses portions of the NASA Glenn CEA software suite to evaluate species thermodynamic properties and/or perform chemical equilibrium calculations. This utility only accepts the CEA format thermodynamic database file, so if this utility is to be used, then any new species added to the VULCAN-CFD format database files must also be added to the CEA format database file.

23 Thermodynamic Nonequilibrium Model Database Format

The thermal nonequilibrium model implemented in VULCAN-CFD is a two-temperature model that assumes the translational and rotational modes are equilibrated and the vibrational and electronic modes are equilibrated. The rotational modes are also taken to be fully excited which results in constant (temperature independent) specific heats for the translational/rotational energy mode. This feature allows the usage of the thermodynamic polynomial curve fits already employed by VULCAN-CFD to define the temperature-dependent vibrational/electronic specific heats. The sources available for the specific data required to perform simulations of flows in thermal nonequilibrium are sparse as compared with those available for thermal equilibrium. This led to the decision to provide the additional data required for the thermal nonequilibrium processes as a separate database file that contains only a subset of the species available for the simulation of flows in thermal equilibrium. The current implementation of the two-temperature nonequilibrium model in VULCAN-CFD neglects the free electron terms that appear in the vibrational/electronic energy equation. The exclusion of these effects reduces the amount of species specific information required to account for thermal nonequilibrium, and the applications that originally motivated the inclusion of thermal nonequilibrium physics into VULCAN-CFD do not involve free electrons. As a result, the additional species specific data required to perform thermal nonequilibrium simulations via a two-temperature model are limited to:

(1) translational/rotational specific heat value

(2) number of vibrational modes and their characteristic vibrational temperatures

The characteristic vibrational temperatures are required by the Millikan/White correlation used to define the relaxation time for translation/vibration energy exchange. This correlation does not always provide a good fit for the energy exchange process, so the database also supports the explicit specification of Millikan/White coefficient data in instances where improved data are available.

The database file name is specified in the VULCAN-CFD input file on the line that immediately follows the SPECIES NONEQUILIBRIUM THERMODYNAMIC DATA input specification. Two thermal nonequilibrium database files are supplied in the database repository (<your_path_to_vulcan>/Data_base) to provide the additional data required by the two-temperature thermal nonequilibrium model implemented in VULCAN-CFD. The two database files are identical, except for the data section used to override the default Millikan/White translational/vibrational relaxation times:

therm_noneq_laura.inp → contains the Millikan/White translational/vibrational relaxation time correlation data extracted from the LAURA (version 5.4-61473) software package developed at NASA Langley

therm_noneq_dplr.inp → contains the Millikan/White translational/vibrational relaxation time correlation data extracted from the DPLR (version 4.03.1) software package developed at NASA Ames

The VULCAN-CFD nonequilibrium thermodynamics database file requires entries for the following items for each species,

(1) chemical symbol for the species

(2) sum of the translational and fully excited rotational specific heat (C_p) normalized by the universal gas constant

(3) degeneracy and characteristic temperature [K] of each vibrational mode

and the specifications for each species can be separated by an arbitrary number of comment lines that begin with "!". Note that for monatomic species, the vibrational modes are irrelevant, so only 2 lines are specified (the chemical symbol and C_p=2.5). For diatomic molecules only 3 lines will be specified (chemical symbol, C_p=3.5, 1 degeneracy, and the characteristic vibrational mode temperature). Polyatomic species can have additional rotational degrees of freedom, and vibrational modes that depend on the orientation of how the atoms are arranged to form the molecule. Hence, degeneracies may exist as well as multiple vibrational modes. An example that illustrates how these data are provided is given below:

```
!
! Species: Argon
!
Ar
2.5                     Normalized translational/rotational Cp value
!
! Species: Carbon Monoxide
!
CO
3.5                     Normalized translational/rotational Cp value
1,  3122.00E+00         Degeneracy,  Characteristic vibrational temperature [K] of mode 1
!
! Species: Carbon Dioxide
!
CO2
3.5                     Normalized translational/rotational Cp value
1,  1918.70E+00         Degeneracy,  Characteristic vibrational temperature [K] of mode 1
2,   959.66E+00         Degeneracy,  Characteristic vibrational temperature [K] of mode 2
1,  3382.10E+00         Degeneracy,  Characteristic vibrational temperature [K] of mode 3
```

The species information described above is all that is required if the standard Millikan/White correlation [74] is used to define the relaxation time (τ) for the translation/vibration energy exchange process. This correlation is given by the following expression:

$$\tau \;=\; \frac{exp\left[A_{mn}\left(T_{tr}^{-\frac{1}{3}} - B_{mn}\right) - 18.42\right]}{P}$$

where T_{tr} is the translational/rotational temperature [K], and P is the pressure in [atm]. The coefficients A_{mn} and B_{mn} are provided by the following formulas: [74]

$$A_{mn} \;=\; 1.16 \times 10^{-3}\, \mu_{mn}^{\frac{1}{2}}\, \theta_{v,m}^{\frac{4}{3}}$$

$$B_{mn} \;=\; 0.015\,\mu_{mn}^{\frac{1}{4}}$$

where μ_{mn} is the reduced molecular weight of molecule "m" and species "n",

$$\mu_m \;=\; \frac{W_m\,W_n}{W_m + W_n}$$

and $\theta_{v,m}$ is the characteristic vibrational mode temperature of molecule "m". By default, this information is the data used to define the translational/vibrational relaxation time. The values for the A_{mn} and B_{mn} coefficients provided by the Millikan/White expressions can be overwritten based on the data added to the Modified Millikan/White translation/vibration relaxation time section of the database file. The data in this section overwrites the Millikan/White A_{mn} and B_{mn} values based on an optimal fit of the vibrational relaxation time (obtained via measurements) to the Millikan/White functional form. The section of the database that houses this information must immediately follow the species specifications described above, and begin with the following header line:

```
Collision Pair        A_mn      B_mn
```

The remaining lines contain the information that will be used to replace the standard Millikan/White relationships. This information includes the chemical symbols for the "m" molecule and the "n" species, as well as the new values for the A_{mn} and B_{mn} coefficients. If the string **ALL** is provided for the collision partner "n", then this implies that a single value for the A_{mn} (and possibly the B_{mn}) coefficient will be used for molecule "m" for all collision partners. In this instance, the value of B_{mn} can be omitted if the default Millikan/White value is appropriate. Examples that illustrate the specifications that are required for this section of the database file are provided below:

```
Collision Pair        A_mn      B_mn
N2          O         72.40     0.015     # Park, JTHT, Vol. 7, No. 3, 1993
NO          NO        49.50     0.042     # Park, JTHT, Vol. 7, No. 3, 1993
O2          O         47.70     0.059     # Park, JTHT, Vol. 7, No. 3, 1993
CO2         ALL       270.4               # LAURA (version 5.4) database
```

24 Chemical Kinetic Model Database Format

VULCAN-CFD uses its own specific database file format for housing chemical kinetic data. However, a utility is included with VULCAN-CFD that can be used to convert a standard CHEMKIN[35] kinetic database (**chem.inp**) into the VULCAN-CFD format. If the conversion of a CHEMKIN kinetic database to the VULCAN-CFD format is necessary, then the use of this utility **conv_chem** is strongly encouraged to minimize the likelihood of mistakes in the translation process. VULCAN-CFD includes a small library of chemical kinetic database files in the directory:

<your_path_to_vulcan>/Data_base

with a description of the kinetic model (and the reference it came from) listed at the beginning of the file.

The VULCAN-CFD chemical kinetic database file format allows any number of comment lines to be present at the beginning of the file. These lines are intended to house a brief description of the kinetic mechanism along with any reference information that the mechanism was extracted from. The line expected after this informative header section is the following:

REACTION MECHANISM EQUATION LIST

which indicates the start of the database information. This line must immediately be followed by the kinetic mechanism reaction list with a header line that begins with the string REACTION preceding the list. The reaction list is supplied in a user-readable fashion with one kinetic step per line, and the number of kinetic steps provided must match the value specified on the **NO. OF CHEMICAL REACTIONS** line in the VULCAN-CFD input file. An example is provided below:

```
REACTION MECHANISM EQUATION LIST
REACTION     REACTANT SIDE        PRODUCT SIDE
    1        H2 + O2        <=>   2OH
    2        H + O2         <=>   OH + O
    3        OH + H2        <=>   H2O + H
    4        O + H2         <=>   OH + H
    5        OH + OH        <=>   H2O + O
    6        H + OH + M     <=>   H2O + M
    7        2H + M         <=>   H2 + M
```

The reaction list is parsed by VULCAN-CFD to extract the species involved with each kinetic step along with their stoichiometric coefficients. The parsing logic requires that the reaction list adheres to the rules below:

(1) The species names used must exactly match those specified in the chemical constituent data section of the VULCAN-CFD input file.

(2) If a "+" sign is used as part of a species name (e.g., an ion), then it must be the last character of the species name. This limitation results from the fact that the "+" sign is used to delineate species names on either the reactant or product side of the reaction string.

(3) No other string may be used to designate <=> or =>. This string is used to delineate the product and reactant sides of the equation. The <=> designates that the reaction progresses in both the forward and backward direction. The => designates that the reaction progresses in the forward direction only.

(4) Generic catalytic 3^{rd} body species must be designated by an M.

The next line defines the form of the forward (i.e., reactant to product) reaction rate model:

FORWARD REACTION MODEL #.#

The numerical value supplied on this line specifies that the forward reaction rate data are those for either a pure Arrhenius functional form (value of 0.0), or an Arrhenius form with arbitrary reaction orders (value of 1.0). This line is immediately followed by a header line and the Arrhenius pre-exponential factor(s) and activation temperature (in *cgs* units) for each kinetic step, e.g.,

FORWARD REACTION MODEL 0.0

REACTION	A	B	Ta	Kf = A T^B exp(-Ta/T)
1	1.70E13	0.00	24154.6	
2	1.20E17	-0.91	8309.7	
3	2.20E13	0.00	2591.6	
4	5.06E04	2.67	3165.3	
5	6.30E12	0.00	548.5	
6	2.21E22	-2.00	0.0	
7	7.30E17	-1.00	0.0	

The next entry depends on the value specified for the FORWARD REACTION MODEL. If a value of 1.0 was provided, then the number of arbitrary reaction orders must be supplied next, followed by the list of altered reaction orders as shown below:

ARBITRARY REACTION ORDERS 2.0

REACTION	SPECIES	ORDER
22	CH4	0.2
22	O2	1.3

The next section defines the form of the backward (i.e., product to reactant) reaction rate model:

BACKWARD REACTION MODEL #.#

There are 3 available options for defining the backward reaction rate:

0.0 → backward rate data supplied in a pure Arrhenius form
1.0 → backward rate data supplied in an Arrhenius form with arbitrary reaction orders
2.0 → backward rate obtained from the forward rate and the reaction equilibrium constant

If a value of 0.0 or 1.0 is provided, then the specifications described for the forward reaction rates above must also be supplied for the backward rates. If a value of 2.0 is provided, then no additional input is required to define the backward rates (most kinetic models are prescribed in this fashion). The next section defines the catalytic 3^{rd} body efficiency values. By default, VULCAN-CFD assumes a catalytic efficiency of unity. Therefore, only the values that are not unity must be specified. The first line of this section defines the number of catalytic 3^{rd} body efficiency specifications to be provided. This line is immediately followed by a header line and the list of catalytic 3^{rd} body efficiency values, e.g.,

```
CATALYTIC BODY EFFICIENCIES 2.0
REACTION   SPECIES   EFFICIENCY
    0         H2         2.5
    0         H2O       16.0
```

Note that a value of 0 can be specified for the reaction number (as shown above) if the catalytic 3^{rd} body efficiency for the given species applies to all of the 3^{rd} body reactions. The last section of the database defines the data for any pressure-dependent kinetic steps that are present. If the kinetic model does not have any pressure-dependent rate expressions, then no further input is required. Otherwise, the number of pressure-dependent kinetic steps must be specified followed by the information required to account for the pressure dependency. An example that illustrates how to provide this information is given below:

```
PRESSURE DEPENDENT REACTIONS 2.0
REACTION  ARRHENIUS CONSTANTS      FORM  COEFFICIENTS
   16      3.482E16 -0.411  -561.370  TROE  0.50 1.0E-30 1.0E+30 1.0E+30
   17      1.202E17  0.000 22907.917  TROE  0.50 1.0E-30 1.0E+30 1.0E+30
```

The first column is the reaction number associated with the pressure-dependent kinetic rate data. The second, third, and fourth columns contain the additional Arrhenius rate coefficients (A, B, Ta) in cgs units. The remaining columns define the functional form for the pressure dependency. VULCAN-CFD supports the following functional forms:

LIND → Lindemann form[75] (no additional coefficients are required)
SRI → SRI form,[76] followed by the a, b, c, d, and e coefficients
TROE → Troe form,[77,78] followed by the α, T^{***}, T^{*}, and T^{**} coefficients

The pressure dependent kinetic steps are either unimolecular/recombination fall-off reactions, or chemically activated bimolecular reactions. The distinction between these classes is determined by how the catalytic 3^{rd} body is included in the REACTION MECHANISM EQUATION LIST. If the reaction is a unimolecular/recombination fall-off reaction, then the high pressure reaction rate coefficients are provided in the FORWARD REACTION MODEL specification section, and the catalytic 3^{rd} body **must not** be included in the REACTION MECHANISM EQUATION LIST. If the reaction is a chemically activated bimolecular reaction, then the low pressure reaction rates are provided in the FORWARD REACTION MODEL specification section, and the catalytic 3^{rd} body **must** be included in the REACTION MECHANISM EQUATION LIST.

25 VULCAN-CFD Utility Codes

VULCAN-CFD is distributed with several utility codes to aid with various aspects of the CFD simulation process, e.g.,

- Partitioning structured grid simulations for efficient parallel execution
- Translating boundary condition and block-to-block connectivity data files from various sources to the VULCAN-CFD format
- Manipulating thermodynamic, transport, and/or chemical kinetic database files (e.g., conversion of ChemKin format database files, curve fitting thermodynamic and/or transport data)
- Modifying VULCAN-CFD profile files (e.g., adding or deleting flow variables, merging and sorting of profile files)
- Modifying VULCAN-CFD restart files (e.g., adding or deleting flow variables, swapping out turbulence closures, converting thermal equilibrium files to thermal nonequilibrium files)
- Postprocessing VULCAN-CFD results (e.g., extraction of performance metrics, quantification of discretization error)
- Uncertainty Quantification and Optimization with DAKOTA[79,80]
- Unstructured grid adaptation with *refine*[6]

All of the utility codes are installed as part of the usual installation process, i.e.,

install_vulcan new

but they are also installed/reinstalled if the following option is invoked:

install_vulcan utl → compiles all utility codes

These utilities are typically stand-alone FORTRAN 90 codes that are intended to be executed interactively from the command line. Most of these utilities have been written for interactive use where the user is prompted for any required information. The responses to these prompts for most of the utility codes are stored to the file utility_name_tmp.inp so that any subsequent executions can utilize the information contained in this file. To use this recorded input for subsequent execution of the utility, simply remove the "_tmp" from the file name, edit as required, and execute the utility via the command:

utility_name < utility_name.inp

NOTE: A new utility_name_tmp.inp file is initiated upon execution of the utility codes. To prevent an accidental loss of the contents from an existing utility_name_tmp.inp file, a backup file (utility_name_tmp.bak) is created when the utility is executed.

25.1 Structured Grid Partitioning

The VULCAN-CFD package includes a structured grid partitioning utility to enable efficient execution on multicore and/or multinode computational platforms. Automating the partitioning process for structured grid simulations can be difficult, so this utility must currently be performed as a stand-alone preprocessing step. The partitioning utility is installed as part of the usual installation process, i.e.,

install_vulcan new

but it is also installed (or reinstalled) if any of the following options are invoked:

install_vulcan sgld → compiles the structured grid partitioning software
install_vulcan utl → compiles all utility codes
install_vulcan all → compiles everything but utility codes

The installation process will produce the following executable files:

grid_split → partitions the structured grid and VULCAN-CFD input file
restart_split → partitions structured grid VULCAN-CFD restart files
restart_merge → recomposes structured grid VULCAN-CFD restart files back to the unpartitioned state

The partitioning process requires: a VULCAN-CFD input file (developed for the unpartitioned grid), a PLOT3D format structured grid file, and an input file to drive the partitioning process (**grid_split.inp**). A skeletal **grid_split.inp** file can be generated for you by simply entering the preprocess command line execution syntax for parallel VULCAN-CFD execution with the desired number of processors, i.e.,

vulcan 6 host_file -p vulcan_input_file.dat

Note that the skeletal **grid_split.inp** file will only be created if one is not already present in the main working directory (this logic is in place to avoid overwriting any manually created **grid_split.inp** files). Also, the VULCAN-CFD input line that defines the number of processors (if present) must **not** match the number of processors specified on the command line, so that VULCAN-CFD can detect that the input file has not yet been partitioned. The files that are produced by the partitioning process (and used by VULCAN-CFD for structured grid simulations) are as follows:

- A partitioned grid file
- A partitioned VULCAN-CFD input file
- A file mapping the partitioned to parent structured grid blocks (**MERGE_MAP.DAT**)
- A file that houses the mapping of partitioned blocks to processors (**MPI_MAP.TMP**)

The partitioned grid file and VULCAN-CFD input file are the files that must be used for parallel execution. The **MERGE_MAP.DAT** file contains the mapping between partitioned and unpartitioned grid blocks, and is only utilized if the "-r" option is included in the

VULCAN-CFD command line to recompose the volumetric plot files back to the unpartitioned state (for easier postprocessing). The **MPI_MAP.TMP** contains the mapping of child blocks to processors. This file is only needed if there is a desire to manually map the child blocks to processors. Otherwise, then the file can be ignored since the VULCAN-CFD preprocessor will remap the structured blocks to the processors via the same algorithm used by the partitioner.

The input file used to control the partitioning process consists of a mandatory GLOBAL section with any number of optional sections that control the splitting at the Region, Block, or Boundary Condition level. The GLOBAL section **must** be the first section in the file, but the other optional sections can appear in any order. Each section must start with the string "START <section_name>" and end with the string "END". The string "EOF" is used to indicate the end of the input file. Any number of comment (or blank) lines may be present, where comment lines are designated by specifying "#" as the first character in the line. An example **grid_split.inp** file is provided below with each of the optional sections commented out (but included for reference purposes). This is followed by a complete description of the available input parameters that control the partitioning process.

```
START GLOBAL
  SPLIT_OPT          1  # Structured grid splitting algorithm (0 - 2)
  NUM_PROCS         96  # Target number of processors
  LOAD_PERC         95  # Structured grid load balance efficiency (%)
  BLK_MIN_DIM        8  # Minimum structured block dimension
  NUM_LEVELS         2  # Number of grid levels
  OLD_VULCAN_INPUT   vulcan.dat
  NEW_VULCAN_INPUT   vulcan_96_procs.dat
  NEW_GRID           vulcan_96_procs.p3d
END
#
# MULTI-REGION SPLITTING CONTROL
#
#START REGION
#REGION   KEY WORD    VALUE
#   2     SPLIT_OPT     1
#   2     NUM_PROCS     8
#   2     LOAD_PERC    98
#END
#
# STRUCTURED GRID BLOCK SPLITTING CONSTRAINTS
#
#START BLOCK
#BLOCK    KEY WORD    INDEX  BEG    END
#  0      NO_SPLIT      I      1      0
#  0      NO_SPLIT      J      1      0
#  0      NO_SPLIT      K      1      0
#END
#
```

```
# STRUCTURED GRID BOUNDARY CONDITION SPLITTING CONSTRAINTS
#
#START BC
#BC NAME    KEY WORD
#SINGULAR   NOT_ALLOWED
#NOSLPSNG   NOT_ALLOWED
#END
#
# EXTERNAL PARTITIONING OPERATIONS
#
#START EXTERNAL
#SPLIT_RESTART_FILES
#END
#
EOF
```

25.1.1 Global Grid Partitioning Input Parameter Description

The GLOBAL input section is the only section of the **grid_split.inp** file that is required. This section defines the high-level options that control the partitioning process. The following input parameters are available in the GLOBAL input section:

SPLIT_OPT #
This input specifies how structured grids are partitioned at the block level.

0 = partitions the grid at user-specified index locations provided in the BLOCK section.
1 = iteratively partitions the grid until the load balance criteria or grid size threshold is met.

NUM_PROCS #
This input provides the number of processors that the simulation is to be load balanced for.

LOAD_PERC #
This input provides the load balance criteria that is desired when partitioning structured grids using the iterative partitioning algorithm.

BLK_MIN_DIM #
This input specifies the minimum number of cells allowed in any coordinate (i, j, k) direction for each structured grid block during the partitioning process. This value must be large enough to support the stencil size requirements of the numerical scheme on the coarsest grid level. The stencil size requirement is 2 for second-order schemes and 3 for the higher-order schemes (e.g., PPM or WENO). Hence, the value specified here must be either 2^{n-1} or 3^{n-1}, where n is the number of grid levels supported by the partitioned grid.

NUM_LEVELS #
This input specifies the number of grid levels (i.e., fine + number of coarse grids) that the partitioned structured grid must support.

METIS_OPT #
This input specifies the algorithm used to partition structured grid subblocks to the processors.

(0) Partitions structured grid blocks to processors by placing the largest unassigned block on the processor that has the fewest cells currently assigned to it. This is the only option that does not require the METIS (or ParMETIS) package.

(1) Partitions structured grid blocks into k equal size partitions using the METIS multilevel k-way algorithm with the objective of minimizing the number of edges that are cut by the partitioning. This is the only algorithm that is available if ParMETIS is used for the partitioning.

(2) Partitions structured grid blocks into k equal size partitions using the METIS multilevel k-way algorithm with the objective of minimizing the total communication volume.

(3) Partitions structured grid blocks into *k* equal size partitions using a multilevel recursive bisection algorithm with the objective of minimizing the number of edges that are cut by the partitioning.

NOTE: Option "0" matches the algorithm used by the VULCAN-CFD preprocessor to map structured grid blocks to processors, so the **MPI_MAP.TMP** file that is produced by the partitioner can be discarded when this option is selected. However, this file can be retained if there is a desire to manually remap the blocks to processors to reduce communication costs. If this file is modified, then it must be placed in the main working directory of the simulation so that the VULCAN-CFD preprocessor can access it and overwrite the default mapping.

METIS_WGT_FLAG #
This input specifies the weighting algorithm used by METIS (or ParMETIS).

(1) weights assigned based on communication costs only

(2) weights assigned based on computational work load only

(3) weights based on both communication costs and computational work load

OLD_VULCAN_INPUT <file_name>
This input specifies the name of the VULCAN-CFD input file to be partitioned.

NEW_VULCAN_INPUT <file_name>
This input specifies the name of the partitioned VULCAN-CFD input file.

NEW_GRID <file_name>
This input specifies the name of the partitioned PLOT3D (or PLOT2D) grid file for structured grid partitioning.

25.1.2 Multiregion Input Parameter Control

The multiregion control section provides independent control over the partitioning process on a region-by-region basis. This can be particularly useful for simulations that include both elliptic and parabolic regions, where vastly different computational resources are often required. The multiregion control section must begin with the string "START REGION" and end with the string "END". The following partition options, previously defined in the GLOBAL input section, can be overwritten on a region-by-region basis:

SPLIT_OPT #
NUM_PROCS #
LOAD_PERC #

In order to override one or more of these parameters for a particular region, simply prepend the region number to one or more of these input options. For instance, if the GLOBAL value for **NUM_PROCS** was set to 128, and this value needs to be changed to 16 for REGION

"N", then the following region specification section should be provided:

START REGION
#REGION KEY WORD VALUE
N **NUM_PROCS** 16
END

where "N" is the region number that requires the change in the number of processors for load balancing.

25.1.3 Structured Grid Partitioning Control (Block level)

The structured grid block control section provides fine-grain control over the partitioning process on a block-by-block basis. This section must begin with the string "START BLOCK" and end with the string "END". Each entry of this section begins with a block number followed by a keyword. Depending on the input option chosen, some number of additional index parameters may follow. The available input options that control the partitioning process at the block level are provided below, with the understanding that a block number must be prepended to each of these input specifications.

SPLIT <index> VALUE
This input specifies that the grid block is to be split at the specific index location provided. Allowable index strings are i, j, or k, and the index value specified must be greater than 1 and less than the maximum. This option is only allowed when "0" is chosen for SPLIT_OPT.

NO_SPLIT <index> BEG END
This input specifies that the grid block is not to be split between the index range provided. Allowable index strings are i, j, or k. The ending index value can be specified as "0" to indicate the maximum value of that index.

NOTE: The index ranges for this input specification must currently correspond to the minimum and maximum index range, i.e., disabling splitting for the chosen coordinate direction. The specification of index ranges is only being forced to accommodate more general specifications in the future.

NOT_ALLOWED
This input specifies that the block is not to be partitioned.

MIN_DIM #
This input specifies the minimum number of cells allowed in any coordinate (i, j, k) direction of the grid block during the partitioning process. This value must be large enough to support the stencil size requirements of the numerical scheme on the coarsest grid level.

NOTE: If a value of "0" is entered for the block number for any of the input parameters described in this section, then the input will be applied to all of the structured grid blocks.

25.1.4 Structured Grid Partitioning Control (Boundary Condition level)

The structured grid boundary condition control section provides independent control over the partitioning process at the boundary condition level. This section must begin with the string "START BC" and end with the string "END".

NOT_ALLOWED <boundary_condition_type or boundary_condition_group_name>
This input specifies that a partition boundary is not to intersect any boundary that utilizes the specified boundary condition type. Alternatively, the boundary condition group name can be specified instead of the boundary condition type. This constraint is required for any collapsed cell boundary condition that utilizes an averaging operator in the collapsed direction (e.g., SINGULAR or NOSLPSNG).

NOTE: The partitioning process for structured grids current only considers the block index i, j, and k values when partitioning (i.e., the process has no knowledge of the physical x, y, z coordinate values). As a result, the partitioner does not currently allow partition boundaries to intersect non-C(0) block interfaces. If non-C(0) interfaces are only present in one coordinate direction (i.e., i, j, or k) of the block, then the partitioner has one degree of freedom for partitioning the block. If non-C(0) interfaces are present in more than one coordinate direction of a block, then that block will not be partitioned. To minimize the impact of this scenario, the block should be partially partitioned manually to isolate the non-C(0) interfaces from the rest of the grid block. In other words, the size of the blocks that contain the non-C(0) interfaces should be minimized to the extent possible prior to using the VULCAN-CFD partitioner.

25.1.5 Partitioning Example

To illustrate the partitioning process, consider the 2-block structured grid generated for the fuel injector port shown in Fig. 17. The following simple **grid_split.inp** file will attempt to partition this grid for 8 processors, with a load balance efficiency of at least 95%:

```
START GLOBAL
  SPLIT_OPT          1  # Structured grid splitting algorithm (0 - 2)
  NUM_PROCS          8  # Target number of processors
  LOAD_PERC         95  # Structured grid load balance efficiency (%)
  BLK_MIN_DIM        8  # Minimum structured block dimension
  NUM_LEVELS         1  # Number of grid levels
  OLD_VULCAN_INPUT  injector.dat
  NEW_VULCAN_INPUT  injector_8_procs.dat
  NEW_GRID          injector_8_procs.grd
END
EOF
```

If ParMETIS was enabled during the VULCAN-CFD installation process, then the following line is required to execute the partitioner:

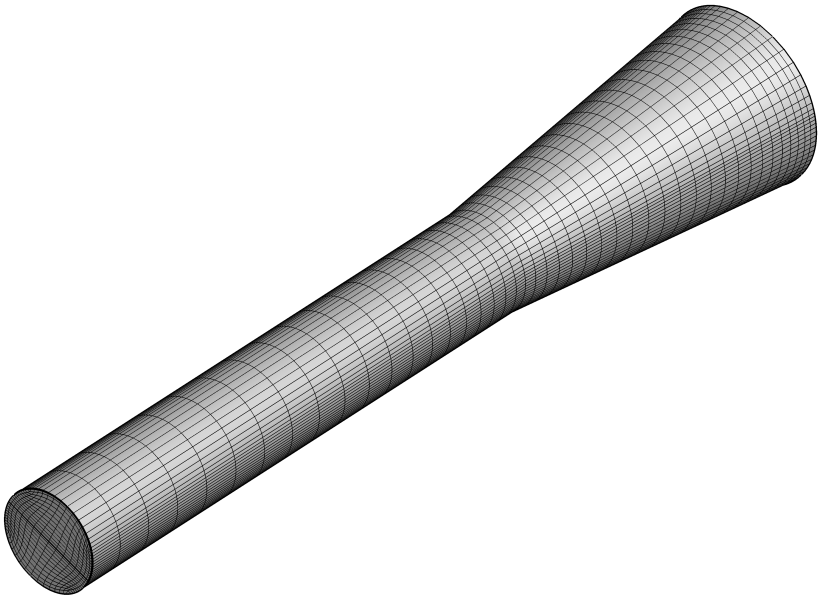


Figure 17: Original (unpartitioned) 2-block grid for the injector port.

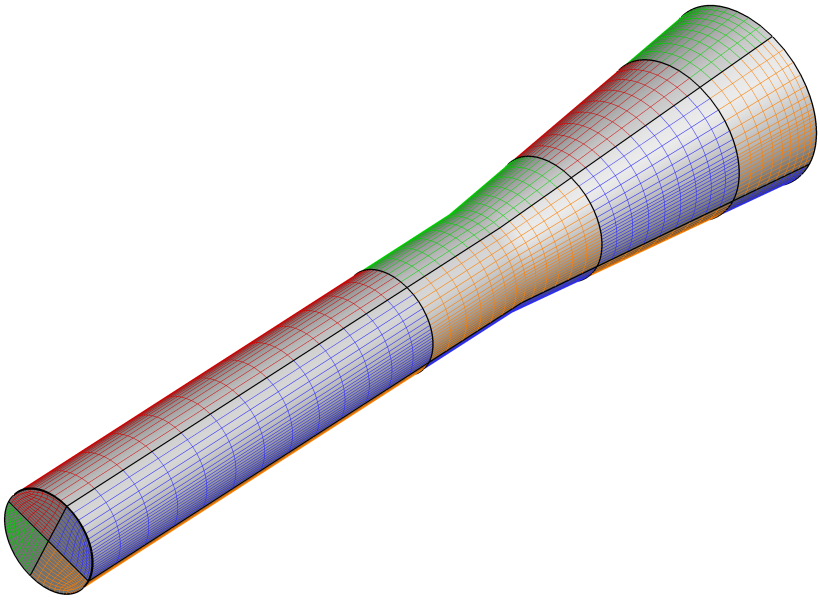


Figure 18: Partitioned 16-block grid for the injector port that achieves at least 95% efficiency.

grid_split <num_cpus> <host_file>

where:

num_cpus denotes the number of processors to be used by ParMETIS
host_file denotes the host file containing the list of MPI hosts

Note that the num_cpus value specified here must be greater than 1, but is not meant to coincide with the number of processors that the problem is being partitioned for. Instead, it should ideally represent the smallest value that will allow the partitioner to execute given the per-node memory constraints on your system. This advice is driven by the fact that the quality of the partitions produced by ParMETIS typically degrades as the number of processors used to perform the partitioning increases. If ParMETIS was not enabled during the installation process, then the partitioner is executed by simply typing "grid_split".

Upon successful execution of the partitioner, the partitioned grid that is produced consists of 16 grid blocks as shown in Fig. 18. The number of grid blocks that results after partitioning can be reduced without sacrificing the load balance efficiency by utilizing the additional control offered at the structured block level. A lower number of structured blocks will reduce the communication overhead associated with the transfer of data across block interfaces, as well as the impact of severing the flow solver implicit operators that are performed at the block level. The following **grid_split.inp** file provides the same load balance efficiency of at least 95%, but with half the number of structured grid blocks (see Fig. 19).

```
START GLOBAL
  SPLIT_OPT          1  # Structured grid splitting algorithm (0 - 2)
  NUM_PROCS          8  # Target number of processors
  LOAD_PERC         95  # Structured grid load balance efficiency (%)
  BLK_MIN_DIM        8  # Minimum structured block dimension
  NUM_LEVELS         1  # Number of grid levels
  OLD_VULCAN_INPUT  injector.dat
  NEW_VULCAN_INPUT  injector_8_procs.dat
  NEW_GRID          injector_8_procs.grd
END
#
START BLOCK
#BLOCK   KEY WORD   INDEX  BEG   END
   0     NO_SPLIT     J     1     0
   0     NO_SPLIT     K     1     0
END
#
EOF
```

Hence, even this simple example illustrates some of the difficulties with automating structured grid partitioning, which is why the partitioning process is not currently transparent to the user for structured grid simulations.

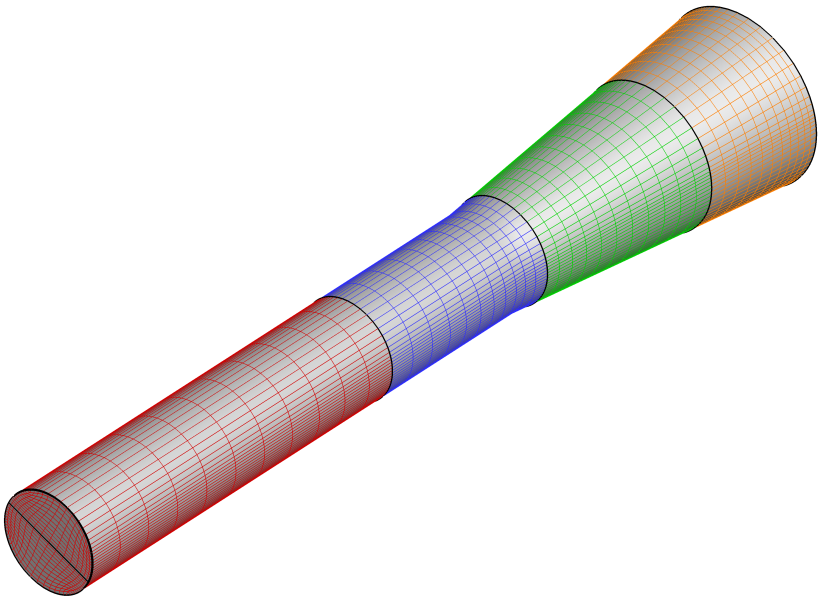


Figure 19: Partitioned 8-block grid for the injector port that achieves at least 95% efficiency.

25.1.6 Partitioning Structured Grid Restart Files

The partitioning utility does not honor the specified initializations (propagation flags) that may have been provided in the IN-ORD column of the structured grid boundary condition, C(0) cut, and/or non-C(0) patch mapping sections of the VULCAN-CFD input file. As a result, all propagation values are zeroed out in the partitioned VULCAN-CFD input file, preventing the desired initialization from being performed. An effective work-around for this issue is to execute VULCAN-CFD using the original (unpartitioned) input file with the number of iterations set to zero. This will create a set of structured grid restart files that contain the desired initialization. These restart files can then be partitioned to match the partitioned grid file using the **restart_split** utility, allowing the partitioned input file to utilize them (with restart flags set appropriately). The input file required by this utility must be named **restart_split.inp**, and contain the following information:

```
# Name of the restart files to be partitioned
RESTART_IN     restart

# Name of the partitioned restart files
RESTART_OUT    restart_split
```

Here, the RESTART_IN specification defines the root name of the restart file to be partitioned, and RESTART_OUT specifies the root name of the partitioned restart files that are created. As is the case for all of the input files associated with the partitioning process, blank lines are ignored and comment lines are indicated by "#". This utility can currently only split structured grid restart data files for regions that are solved in an elliptic fashion

with VULCAN-CFD. The command line syntax for this utility is simply:

restart_split

NOTE: If a **restart_split.inp** file has been created and is present in the working directory used to perform the partitioning, then the structured grid restart files can be partitioned in a seamless fashion as part of the partitioning process by adding the following section to the **grid_split.inp** file:

START EXTERNAL
SPLIT_RESTART_FILES
END

25.1.7 Merging Partitioned Structured Grid Restart & Postprocessing Files

The **restart_merge** utility provides the capability to recompose the partitioned structured grid restart files back to their original unpartitioned state. This capability is useful if one desires at some point to load balance for a different number of processors than what the current partitioning can efficiently support. The input file required by this utility must be named **restart_merge.inp**, and contain the following information:

```
# Name of the partitioned restart files to be merged
RESTART_IN     restart_split

# Name of the merged restart files
RESTART_OUT    restart
```

where RESTART_IN specifies the root name of the partitioned restart files to be recomposed, and RESTART_OUT specifies the root name of the recomposed restart files that are created. As is the case for all of the input files associated with the partitioning process, blank lines are ignored and comment lines are indicated by "#". This utility can currently only recompose structured grid restart data files for regions that have been solved in an elliptic fashion with VULCAN-CFD. To execute this utility from the command line, simply type:

restart_merge

Alternatively, the recomposition of partitioned structured grid restart files can be performed as part of the VULCAN-CFD solver execution process if all of the following conditions are satisfied:

(1) The MERGE_MAP.DAT file (created when executing the partitioner) is present in the working directory of the VULCAN-CFD simulation.

(2) The "-r" (recompose) option has been specified on the VULCAN-CFD command line.

(3) The **restart_merge.inp** is present in the working directory of the VULCAN-CFD simulation.

The PLOT3D format volumetric data files created by the VULCAN-CFD postprocessor are recomposed by the utility **plot3d_merge**. This utility does not utilize an input file, nor is it intended to be executed as a stand-alone application. Instead, the recomposition process for these files is performed during the final VULCAN-CFD postprocessing step, provided that the MERGE_MAP.DAT file (created when executing the partitioner) is present in the working directory of the VULCAN-CFD simulation, and the "-r" (recompose) option has been specified on the VULCAN-CFD command line. Note that the **plot3d.g.fvbnd** file, which contains the information to display data at external boundaries, is not currently included as part of the recomposition process. If this file is desired, the VULCAN-CFD postprocessor must be executed one time using the original unpartitioned grid and VULCAN-CFD input files. This will create a **plot3d.g.fvbnd** which can be saved for future use.

Any Tecplot .szplt format volumetric data files created by the VULCAN-CFD postprocessor for structured grids will be directly recomposed by the VULCAN-CFD postprocessor, provided that the MERGE_MAP.DAT file (created when executing the partitioner) is present in the working directory of the VULCAN-CFD simulation. VULCAN-CFD currently has no mechanism to recompose CGNS format or structured grid Tecplot .plt format data files output by the VULCAN-CFD postprocessor. There is also no current capability to recompose VULCAN-CFD surface data files for structured blocks (**vulcan.loads.tec**) back to the unpartitioned state. If unpartitioned versions of any of these files are desired, then the restart files must be recomposed so that the postprocessor can be executed using the unpartitioned grid and VULCAN-CFD input files.

25.2 Grid Utility Codes

There are several FORTRAN utilities in the Utilities/Grid_codes directory that are intended to operate on grid files or files that contain grid boundary and/or block-to-block connectivity data. The Plot3d directory contains the **grid_plot3d** utility, which performs a variety of user-specified actions on PLOT3D (or PLOT2D) format grid files (e.g., coarsening, scaling, swapping coordinates, changing file formats), while also providing information on the number of grid levels supported by the grid and the number of processors the grid can efficiently utilize. The utility can also be used to partition the grid based on user input for efficient processing on parallel machines. However, in most circumstances it is preferable to use the VULCAN-CFD grid partitioning tool described in 25, which provides more flexibility and allows the partitioning process to operate on the VULCAN-CFD input and output files. The following tasks are currently supported by this utility:

- extract grid data (block dimensions, grid levels, load balancing)
- check for negative volumes
- change the file format from ASCII to FORTRAN UNFORMATTED and vice versa
- swap the order of the spatial coordinates
- extract an i, j, or k plane from a 3-D volume grid
- extrude (or rotate) a planar 2-D grid to obtain a 3-D volume grid
- coarsen any (or all) computational coordinate direction(s)
- scale (and/or shift) the grid by some constant factor
- split a zone at a specified index or at "N" equally spaced intervals

The Pointwise directory contains the VULCAN-CFD plugins required to specify (and output) the boundary condition and grid connectivity information when Pointwise is used to generate computational grids. Interface plugins for both structured (VULCAN-STR) and unstructured (VULCAN-UNS) grid generation are available in the Pointwise/Plugins subdirectory. These plugins have not yet been supplied to Pointwise for redistribution, so precompiled versions (for 64-bit linux systems) are included with VULCAN-CFD. If Pointwise is to be used on a 64-bit linux system, then these precompiled plugin libraries can be simply copied into the plugins directory associated with your installation of Pointwise (this may require root access privileges). If these precompiled plugin libraries are not compatibly with your system, then refer to the postinstallation instructions section of Chapter 1 for details on how to install the plugins on your system. Several glyph scripts are also provided in the Pointwise/Glyph_scripts subdirectory that may prove useful when using Pointwise:

Plot3dMerge_PW_ver1.glf: This script is intended to establish the block connectivity of a PLOT3D format structured grid file that has been imported into Pointwise. Proper connectivity is determined through a splitting/merging process to detect and remove overlapping domains present at block-to-block interfaces, which commonly occurs when importing grid files into Pointwise. The following steps should be performed to effectively use this glyph script once Pointwise is launched:

(1) import the PLOT3D grid coordinate file

(2) set the boundary conditions

(3) execute Plot3dMerge_PW_ver1.glf

Plot3dMerge_PW_ver2.glf: This script is intended to be used when the steps described above fail to merge all of the overlapping domains at block interfaces. This version of the script exhaustively splits and merges domains (that do not have boundary conditions assigned to them) when more than one connector defines a domain edge. This process can result in a large number of domains appearing on block faces, and as a result, should only be used when the **Plot3dMerge_PW_ver1.glf** script is not completely successful.

ViewDomsByCons.glf: This script allows all domains to be viewed that share a selected connector to quickly visualize how the point distribution on a connector affects all neighboring domains.

ConGeometricAutoDimension.glf: This script will autodimension a selected connector (given the edge spacings) with a geometric distribution function given the end spacings, the maximum spacing, and the geometric growth ratio.

The Gridpro directory contains the utility **gridprot**. This executable converts the boundary condition and block-to-block connectivity information from the native Gridpro file format to the boundary condition and cut connectivity mappings needed for insertion into the VULCAN-CFD input file. This utility requires the GridPro connectivity file (.conn_n file). The property map file, **ptymap.vulcan**, which is used to set generic boundary condition classes from within the GridPro GUI, can also be provided to simplify the boundary condition specifications, but the use of this file is optional. The output from this utility is a file that contains the boundary condition specification section of the VULCAN-CFD input file and the structured grid boundary condition and cut connectivity mapping sections. This file is intended to be the starting point for building a complete VULCAN-CFD input file for structured grids generated by GridPro. An additional utility called **gpro2nmf** is also available that converts a native Gridpro connectivity file to a V2K neutral map file for use with V2K (a tool developed by the NASA Langley GeoLab). This file, together with the PLOT3D grid file, allows users to visualize the boundary conditions for VULCAN-CFD using the V2K graphical user interface.

The Heldenmesh directory contains the utility **bc_lump**. This utility imports an AFLR3 format grid file and boundary condition map file (<root_grid_file_name>.mapbc) and combines all separate boundary conditions that share the same boundary condition name into a single entity. The Heldenmesh grid generation software currently exports separate surface boundary condition lines for each boundary surface even if the boundary condition name matches that of some other surface, which unnecessarily increases the number of boundary condition groups used by VULCAN-CFD. This utility examines all of the boundary condition information present in the AFLR3 grid file and assigns a unique boundary condition identification number to boundary conditions that share the same name.

Finally, the Gridgen and V2K directories contain executables that translate boundary condition and block-to-block connectivity information with a functionality that is analogous to the **gridprot** utility. Both Gridgen and V2K are no longer actively developed, but these translation utilities have been retained for users that still have access to these packages. The utility **gridgent** requires a "generic flow solver" boundary condition file to be output by Gridgen along with the glyph script vulcan.glf used to set the boundary conditions from within Gridgen (the use of this file is optional). Similarly, the utility **v2knmapt** requires a neutral map file produced by the GeoLab V2K software (or produced from **gpro2nmf** as described above). The output from each of these utilities is a file that contains the boundary condition specification section of the VULCAN-CFD input file and the structured grid boundary condition and cut connectivity mapping sections.

25.3 Database Utility Codes

There are several FORTRAN codes in the Utilities/Dbase_codes directory that are intended to operate on (or provide data for) thermodynamic, transport, and chemical kinetic database files. The utility **atmos76** calculates flow conditions using the 1976 Standard Atmospheric Tables.[81] This utility prompts for Mach number and either altitude or dynamic pressure and computes the remaining conditions with the following caveats:

- only the lower atmosphere is considered (i.e., geometric altitudes up to 86 km or 282,152 feet)
- a constant molecular weight is assumed (i.e., the variation provided above 80 km is neglected)
- a molecular weight of 28.9651159 g/mol is used rather than the 28.9644 g/mol value specified in the standard to match the air species provided in the McBride thermodynamic database file that assumes a mole percentage of 78.084

The utility **mw_coef** computes the constants used to parameterize the Millikan/White vibrational relaxation time correlation (A_{mw} and b_{mw}). This utility prompts for the molecular weight [g/mol] and characteristic vibrational temperature [K] for the vibrating molecule, and the molecular weight [g/mol] for the colliding species. The output from this utility are the A_{mw} and b_{mw} coefficients required for insertion into the VULCAN-CFD thermal nonequilibrium database file(s).

The utility **ls_fit** performs least-squares fits of thermodynamic and molecular transport data. Transport data can be supplied by any of the following sources: Sutherland law, power law, McBride polynomial, ChemKin polynomial, or tabulated data with two columns of data (T [K] and either μ or λ in MKS units), to produce least-squares fits to one of the following functional forms:

Sutherland law $\rightarrow \quad \mu = \mu^\circ \, (T/T^\circ)^{\frac{3}{2}} \, (T^\circ + S)/(T + S)$

Power law $\rightarrow \quad \mu = \mu^\circ \, (T/T^\circ)^n$

McBride polynomial $\rightarrow \quad \mu = C \, \exp[a_1 \ln T + a_2 T^{-1} + a_3 T^{-2} + a_4]$

ChemKin polynomial $\rightarrow \quad \mu = C \, \exp[a_1 + a_2 \ln T + a_3 (\ln T)^2 + a_4 (\ln T)^3]$

The factor C in the McBride and ChemKin expressions is a conversion factor that is dependent on whether viscosity or thermal conductivity data are being fit. The ChemKin expression expects values of 1.0e-1 (viscosity) and 1.0e-5 (thermal conductivity), while the McBride expression expects values of 1.0e-7 (viscosity) and 1.0e-4 (conductivity). A least-squares fit for thermodynamic properties requires that data be supplied by one of the following sources:

McBride database (7-Coefficient McBride polynomial fit):
$C_p = a_1 + a_2T + a_3T^2 + a_4T^3 + a_5T^4$ and b_1, b_2

McBride database (9-Coefficient McBride polynomial fit):
$C_p = a_1T^{-2} + a_2T^{-1} + a_3 + a_4T + a_5T^2 + a_6T^3 + a_7T^4$ and b_1, b_2

ChemKin database (7-Coefficient McBride polynomial fit):
$C_p = a_1 + a_2T + a_3T^2 + a_4T^3 + a_5T^4$ and b_1, b_2

PEP 7-Coefficient polynomial fit:
$C_p = a_1 + a_2T + a_3T^2 + a_4T^3 + a_5T^4$ and b_1, b_2

Tabulated data: $T[K]$, C_p/R_u , $h/(R_uT)$, s/R_u

The data supplied in any of the forms above are used to produce a least-squares fit for either the McBride 7-Coefficient or the PEP 7-Coefficient polynomial. The output from this utility are the coefficients produced by the least-squares fitting process required for insertion into a VULCAN-CFD thermodynamic (or transport property) database file. Tecplot format data files are also generated that contain the original (source) data and the least-squares fit data to allow visual assessments of the fit data.

The utility **mix_fit** outputs thermodynamic and molecular transport data for a user-specified "mixture" species (e.g., combining N_2, O_2, H_2O, CO, CO_2 to form a single vitiated air species). The mixture composition can be supplied directly as mass fraction (or mole fraction) values, or the output from a CEA equilibrium calculation[32,33] can be parsed to extract the mixture composition. The thermodynamic coefficients for the individual species that make up the "mixture" must be provided by one of the following methods:

- CEA thermodynamic database (therm.inp file)
- ChemKin thermodynamic database (therm.dat file)
- VULCAN-CFD gas model database (e.g., gas_mod.Lewis_2 file)
- direct input of thermodynamic polynomial coefficients (a_1 - a_n, b_1, b_2)

The molecular transport coefficients for the individual species that make up the "mixture" must be provided by one of the following methods:

- CEA transport database (trans.inp file)
- output file generated by the ChemKin **tranfit** utility
- direct input of the transport model coefficients (McBride, ChemKin, Sutherland law, or power law functional forms)

The output from this utility is a tabulated file (Tecplot format) of the thermodynamic properties as a function of temperature, which can be used by the **ls_fit** utility described above to produce a least-squares fit of the polynomial coefficients for the thermodynamic properties

(as well as allow for visualization of the mixture species thermodynamic data). The polynomial coefficients that result from linearly combining (based on composition) the species coefficients that make up the mixture, are also output if the following constraints are satisfied,

- the composition provided is constant (i.e., the composition was not provided from a CEA equilibrium calculation which could vary with temperature)
- number of curve fit temperature intervals is the same for each species used to form the "mixture" species
- the bounding temperatures for each curve fit interval are the same for each species used to form the "mixture" species
- number of polynomial coefficients used to fit the thermodynamic data is the same for each species used to form the "mixture" species

These coefficients can be directly inserted into the desired VULCAN-CFD thermodynamic database file without having to resort to using the **ls_fit** utility. In contrast to the thermodynamic mixture rules, the mixture formulas for viscous transport properties tend to be nonlinear functions of composition. Therefore, the transport model coefficients cannot typically be determined analytically, so the mixture transport properties (μ and λ) are simply written to Tecplot format data files. These files can be used as tabulated data for the **ls_fit** utility described above to produce a least-squares fit of the transport property model coefficients to any of the transport property functional forms supported by the utility.

The utility **conv_chem** converts a ChemKin format chemical kinetics database file (typically named chem.inp) to the VULCAN-CFD format. VULCAN-CFD supports most of the gas phase finite rate chemistry capabilities within ChemKin, but for the few options not supported, a check is performed to ensure that the database being converted does not reference these unsupported features. The following ChemKin features are currently not supported by VULCAN-CFD:

- pressure dependent rates using logarithmic interpolation (Keyword: PLOG)
- pressure dependent rates using Chebyshev polynomials (Keyword: CHEB)
- Landau-Teller reaction rate expressions (Keyword: LT)
- optional rate constant fits (Keywords: FIT1, JAN)
- species temperature dependency option (Keyword: TDEP)
- energy loss event option (Keyword: EXCI)
- momentum transfer collision option (Keyword: MOME)
- radiation wavelength parameter option (Keyword: HV)
- reaction units change option (Keyword: UNIS)

- user-supplied program option (Keyword: USRPROG)

Any occurrence of the above keywords will halt the database file conversion process. The only input required by this utility is the ChemKin database file to be converted. The output is the equivalent chemical kinetics database file in the VULCAN-CFD format.

25.4 Profile File Utility Codes

There are 2 FORTRAN utilities in the Utilities/Profile_codes directory that are intended to operate on VULCAN-CFD profile files that are used to specify boundary conditions that vary spatially. The **vulcan_prof_mrg** utility merges (if required) individual VULCAN-CFD profile files that belong to a common boundary condition group, while also sorting the profile data by x, y, z value. The first action is intended to accommodate the presence of partial profile files that have been output from a partitioned VULCAN-CFD simulation. The second action is performed to reduce the time required to match profile file coordinates with the grid file coordinates. VULCAN-CFD requires that the profile file data be sorted by x, y, z value, so the **vulcan_prof_mrg** utility usually needs to be executed to process the profile file data prior to using it for a VULCAN-CFD simulation. If a deprecated structured grid profile format is detected, then this utility will output the data in the universal format that supports both structured and unstructured grids. Further details (including the variables expected to be present in VULCAN-CFD profile files) are discussed in Chapter 9.

The **vulcan_prof** utility performs a variety of user-specified actions on VULCAN-CFD profile files (e.g., adding or removing flowfield variables, modifying existing flowfield variables, mirroring profile file data). The following tasks are currently supported by this utility:

- add (or remove) flowfield variables
- modify flowfield variables
- mirror a velocity component
- map 2-D/axi profile data to a 3-D profile
- create an analytic profile file
  - fully-developed channel flow
  - temporal mixing layer initial condition
  - isentropic vortex initial condition
  - subsonic manufactured solution
  - supersonic manufactured solution

Several scenarios where this utility might prove useful are listed below:

- Adding species to a profile file generated from an upstream nonreacting inlet simulation for use in a downstream combustor simulation.
- Removing (or adding) species to a simulation when different chemical kinetic models are invoked.
- Creating mirrored profile data intended for simulations at zero angle-of-attack (with symmetry) to use for nonzero angle-of-attack simulations.
- Rotating axisymmetric profile data to create data for a 3-D azimuthal sector mesh (or extruding 2-D profile data over some width for use in a 3-D simulation).

- Adding vibrational/electronic temperature to a profile file created for thermally perfect gas simulations (and vice-versa).

Given that the VULCAN-CFD profile files are output as Tecplot ASCII format files, the above actions (as well as more complicated tasks) could just as easily be performed using Tecplot. A task well suited for Tecplot is the interpolation of profile data to another grid, which is required when the grid that needs the profile data does not match up to the coordinates used to generate the profile data. For this scenario, "dummy" profile files should be created by VULCAN-CFD at the boundaries that need to utilize the profile file data to provide the precise x, y, and z values at the grid boundary faces. This step involves temporarily replacing the PRFINP boundary condition option with PRFOUT, and executing VULCAN-CFD with the number of iterations set to zero. If the boundary that requires the profile data extends over multiple grid partitions (or blocks), then the **vulcan_prof_mrg** utility (described above) can be used to merge the partitioned profile files that have been output into a single file to simplify the interpolation process. Once generated, this "dummy" profile file can be loaded into Tecplot along with the original profile data file for interpolation purposes. Tecplot has three algorithms for variable interpolation: linear interpolation, inverse distance weighting, and kriging. The following are things to consider when choosing an interpolation process:

- Linear interpolation is a good choice from an accuracy perspective, and it also preserves realizability of interpolated quantities. However, any points outside of the original grid range must either be specified to some value, or extrapolated from the interior. This can result in less accuracy near domain boundaries (particularly when the boundaries are not defined by straight lines). This option is also only available for "ordered" (i.e., structured) data, which is typically not the case for VULCAN-CFD profile files that have been sorted by x, y, z value.

- The inverse distance weighted algorithm is very robust, preserves realizability, and allows data to be smoothly extrapolated outside of the original grid range of values. However, it tends to be less accurate since even points far from the interpolation position contribute to some extent.

- The kriging algorithm is more complex than the inverse distance weighted approach, but tends to produce more accurate results. However, it may also result in interpolation extrema which can (in some cases) result in unrealizable interpolated values. If this method is chosen, it is advisable to check the minimum and maximum values of the flow variables that have realizability constraints (e.g., a positivity constraint).

- To output the interpolated profile Tecplot zone(s), select the Output ASCII Output Data Writer (.dat) option, and be sure to select the "Point" format with the precision set to at least 15.

- If interpolating a line of data that defines a 2-D profile to an extruded plane for a 3-D simulation, then perform the following steps once the data files have been loaded into Tecplot:

  – Make a copy of the extruded coordinate (e.g., z).

  - For the extruded 3-D zones, redefine the extruded coordinate to match the value in the 2-D "line" profile (typically $z = 0.0$).
  - Perform the interpolation using the inverse distance weighted algorithm (kriging cannot be used for 2-D interpolation).
  - Overwrite the extruded coordinate with the values copied from the first step and remove the copy of the extruded coordinate.
  - Output the interpolated profile Tecplot zone(s).

  Note that this operation can be handled by **vulcan_prof** if the 3-D coordinates are a simple extrusion (based on the y,z coordinate values) of the 2-D profile coordinates.

- If interpolating a line of data that defines an axisymmetric profile to a 3-D rotated cylindrical grid for a 3-D simulation, then perform the following steps:
  - Make a copy of the rotated coordinates (e.g., y and z).
  - Redefine the first rotated coordinate (r) as the magnitude of the rotated coordinate pair, and redefine the second rotated coordinate (θ) to be zero.
  - Perform the interpolation using the inverse distance weighted algorithm (kriging cannot be used for 2-D interpolation).
  - Overwrite the r and θ values with the coordinate values copied from the first step and remove the copies.
  - Perform the rotational transformation to convert the polar velocity components to Cartesian velocity components, i.e.,
    $v = v_r cos(\theta) + v_\theta sin(\theta)$
    $w = v_\theta cos(\theta) - v_r sin(\theta)$
    where θ is the angle of rotation about the axis determined by each y,z value pair.
  - Output the interpolated profile Tecplot zone(s).

  Note that this operation can be handled by **vulcan_prof** if the 3-D coordinates are a simple rotation (based on the y,z coordinate values) of the 2-D profile coordinates.

25.5 Restart File Utility Codes

The **vulcan_rest** utility, located in the Utilities/Restart_codes directory, performs a variety of user-specified actions on VULCAN-CFD restart files (e.g., changing the number of species, modifying flowfield variables, changing the turbulence model). This utility only operates on restart files that are associated with elliptic regions, but restart file data from parabolic regions should never need to be altered, since any changes would require previously solved streamwise planes to be solved again. The following tasks are currently supported by this utility (some of which are only supported for structured grid simulations):

- add (or remove) conserved variables
- add (or remove) ghost cell layers in structured blocks
- remove structured grid blocks/partitions
- redistribute grid blocks/partitions between regions
- modify conserved variables
- refine (or coarsen) structured restart files
- convert unformatted/binary restart files to ASCII
- convert ASCII restart files to unformatted/binary
- extract a grid level from structured restart files
- convert 2-D structured restart files to 3-D files
- convert structured calorically perfect restart files to thermally perfect files
- duplicate (and possibly mirror) structured restart files
- convert old restart files format to a newer one
- convert between thermal equilibrium and thermal nonequilibrium files
- change the turbulence model
- interpolate structured restart files to a different structured grid
- compute structured grid boundary layer properties

As indicated above, most of the options that involve a change to the grid are only available for structured grid simulations. The reason for this limitation is the fact that changes to an unstructured grid tends to globally alter the indexing of the grid structure making the existing restart files (which are indexed based on the original grid) useless. Typical scenarios where this utility might prove useful are listed below:

- Adding species to a tare nonreacting combustor simulation for use as an initial condition for a reacting simulation.

- Removing (or adding) species to provide a reasonable reacting initial condition when different chemical kinetic models are invoked.

- Mirroring structured restart files obtained from simulations at zero angle-of-attack (with symmetry) to initialize a nonzero angle-of-attack simulation.

- Converting restart files generated from a 1-equation turbulence model to provide a favorable initialization for a simulation that utilizes a 2-equation turbulence model (and vice-versa)

- Using a thermally perfect simulation as an initialization for a simulation in thermal nonequilibrium (and vice-versa).

- Converting unformatted/binary restart header files to ASCII for probing purposes and/or manual modifications.

This utility always prompts for the full name of the input restart header file name (i.e., ROOT_NAME_#_S file), where ROOT_NAME is the restart file name provided in the VULCAN-CFD input file that created the restart files and "#" is the region number associated with the restart file data to be altered. When prompted for the name of the output restart files, only the new ROOT_NAME is expected, since the utility needs to append the proper region number to this root file name. All other prompts are option dependent and should be self explanatory. Also note the following details that pertain to the extraction of boundary layer properties for structured grid simulations:

- This utility has no knowledge of block connectivity, so boundary layer properties can only be extracted for blocks that contain the entire boundary layer height from the no-slip wall to the core flow.

- Checks are performed to see if no-slip surfaces exist on either the MIN or MAX boundaries at each streamwise coordinate station.

- If no-slip surfaces are found, then a core (freestream) vorticity or total enthalpy value is computed depending on user input.

- Starting from each no-slip boundary, the first location with vorticity smaller than the specified fraction of the core vorticity (or the first location with stagnation enthalpy higher than the specified fraction of the core stagnation enthalpy) is used to define the boundary layer edge.

- An integration is then performed across the boundary layer to compute displacement and momentum thicknesses using a simple trapezoidal algorithm.

- If the flow was denoted as axisymmetric, then a circumferential displacement (momentum) area is integrated and later converted to a displacement (momentum) thickness using radial flow relationships.

- Boundary layer properties are output to the screen and to a Tecplot format ASCII data file to allow the extracted boundary layer properties to be visualized as a function of streamwise distance.

In general, the vorticity method only works well when the flow is shock-free and the core flow is relatively uniform. This is usually only the case with flat plate or facility nozzle flow simulations. The total enthalpy method is a more robust option when the core flow is not uniform. However, this option is only applicable for isothermal walls and is a measure of the thermal boundary layer height rather than one determined by the velocity profile.

NOTE: The boundary layer integration is based only on the resolved flow variable profiles supplied in the restart files. Therefore, if wall functions are utilized there will be some error in the integrated properties due to the "missing" profile shape details below the first cell center location.

25.6 Time History File Processing

The time history files for structured grid simulations are currently only output to PLOT3D format files, and individual PLOT3D format files are written at each time interval for every subdomain specified in the structured grid time history output specification section of the VULCAN-CFD input file (see Chapter 19). For ease of postprocessing, these files should be merged using the **time_merge** utility located in the Utilities/Post_Process_codes/Time_files directory. This utility will either merge all of the files into a single ASCII Tecplot file (as separate Tecplot zones), or alternatively merge all of the subdomains into separate PLOT3D function files for each time interval. The first option is convenient for users of Tecplot, while the latter is better suited for users of Fieldview.

NOTE: The block number is appended to the time history file names, so **time_merge** needs to know the largest block number to consider when parsing through the potential time history file names. When **time_merge** prompts for the total number of blocks, the value entered must be at least the largest block (or partition) number that time histories have been exported for. If the largest block number accessed for time history output is not known, then the value entered should be the total number of structured curvilinear blocks present in the simulation (see the block specification section of Chapter 8).

NOTE: The last prompt will ask if the individual PLOT3D format files should be deleted during the merging process. It is recommended to retain the individual PLOT3D format files the first time that this utility is executed to ensure that a mistake was not made during execution of this utility. Once the merge process has been confirmed to be correct, the input file that the utility generates can confidently be used for subsequent merge operations with this particular prompt response reset to enable the deletion of these files.

NOTE: VULCAN-CFD always resets the time history counter to 1 when restarting a simulation that exports time history files. To avoid an unintentional loss of existing time history data, it is important that the user execute the **time_merge** utility (or copy the individual time history files to some other location) before restarting the simulation.

25.7 T-infinity Utilities

Installations with T-infinity enabled will provide access to several unstructured grid capabilities present within **vinf** (Vulcan-INFinity). The **vinf** command line has the following general form:

```
vinf <num_cpus> <subcommand> <options>
```

where:

num_cpus	denotes the number of processors (cores) to be utilized
subcommand	string denoting the specific subcommand to invoke
options	defines the options for the specific subcommand invoked

All options either begin with "-" or "--". The complete list of available subcommands are displayed by simply typing:

```
vinf --help
```

and a detailed description of the options available for each subcommand is displayed by typing:

```
vinf <subcommand> --help
```

The list of subcommands intended for general use and their options are provided below:

cartmesh
The cartmesh subcommand creates a simple Cartesian hexahedral mesh (output in AFLR3 format), which is sometimes useful for generating a quick grid for exploratory or debugging purposes. The subcommand usage line for this feature is as follows:

```
vinf cartmesh --lo <x y z> --hi <x y z>
              -d <nx ny nz>
              -o <mesh_file_name>
```

where:

-o <mesh_file_name>	grid file name (AFLR3 format) [REQUIRED]
-d <nx ny nz>	number of cells in each coordinate direction [REQUIRED]
--lo <x y z>	Cartesian mesh extent lower coordinates
--hi <x y z>	Cartesian mesh extent upper coordinates

If not specified, the lower extent values are all assumed to be zero, and the upper extents are taken to be unity. The grid coordinates are always output as double precision values, and the AFLR3 grid file name must end with either a .lb8.ugrid (little endian output), or .b8.ugrid (big endian output) extension. The grid that is created will have different boundary tags (1-6) applied to each Cartesian coordinate boundary (1=zmin, 2=xmin, 3=ymin, 4=xmax, 5=ymax, 6=zmax).

distance
The distance subcommand computes the closest distance to the set of mesh boundary tags provided, and outputs the result for plotting purposes. The subcommand usage line for this feature is as follows:

```
vinf distance -m <mesh_file_name>
              --tags <bc_tags>
              -o <output_file_name>
```

where:

-m <mesh_file_name>	grid file name (AFLR3 format) [REQUIRED]
-o <plot_file_name>	name of the output plot file [REQUIRED]
--tags <bc_tags>	mesh boundary tags to compute distance from [REQUIRED]
--at <nodes or cells>	specifies whether to compute the distance from nodes or cells

The bc_tags can be a single tag, a space delineated tag list, and/or a range of tags specified with a ":" separator. The distance from nodes is output by default if the --at option is omitted. The extension provided for the output plot file name determines the format of the file (.plt → binary Tecplot file, .vtk → legacy VTK file, .snap → binary snap file).

examine
The examine subcommand prints simple size statistics about a mesh from the AFLR3 grid header as well as other optional information. This subcommand can also examine the contents of (or simply list the fields present in) a specified snap file. The subcommand usage line for this feature is as follows:

```
vinf examine -m <mesh_file_name>
```

where:

-m <mesh_file_name>	grid file name (AFLR3 format) [REQUIRED]
--snap <snap_file_name>	list all fields present in the snap file
--tags	list all mesh boundary tags
--edge-stats <bc_tags>	print cell edge length statistics for a particular mesh boundary
--extent	print the x,y,z extents of the mesh

The mesh file header is the only portion of the grid file loaded when the -m option is all that is specified. If any other option is included then the full mesh must be read in. Therefore, if only the size and cell type make up of the mesh is of interest then it is much faster to only use -m. The bc_tags can be a single tag, a space delineated tag list, and/or a range of tags specified with a ":" separator.

extrude
The extrude subcommand extrudes (or rotates) a 2-D grid file to form a 3-D grid file. This operation creates a single layer of cells in the 3^{rd} dimension from the provided 2-D surface grid. The subcommand usage line for this feature is as follows:

```
vinf extrude -m <mesh_file_name>
             -o <output_grid_name>
```

where:

-m <mesh_file_name>	2-D input grid file name [REQUIRED]
-o <output_grid_name>	3-D output grid file name [REQUIRED]
--revolve	revolve the coordinates to create a 3-D azimuthal sector mesh

If the --revolve option is used to create a 3-D sector mesh (5° included angle), then the axis of rotation is taken to be the x-axis. If the --revolve option is omitted, then the grid is simply extruded in the z-direction. Allowable AFLR3 file format extensions are .lb8.ugrid (little endian), b8.ugrid (big endian), or .ugrid (ASCII). Allowable output grid formats are:

- .lb8.ugrid, b8.ugrid, or .ugrid (AFLR3 format)
- .plt (binary Tecplot format)
- .vtk (legacy VTK format)

fix

The fix subcommand attempts to fix or modify certain aspects of an AFLR3 format unstructured grid file. The subcommand usage line for this feature is as follows:

```
vinf fix -m <mesh_file_name>
         --hanging-edges
         -o <output_grid_name>
```

where:

-m <mesh_file_name>	grid file name (AFLR3 format) [REQUIRED]
-o <output_grid_name>	name of the "fixed" output grid file [REQUIRED]
--missing-faces <bc_tag>	create surface elements on missing faces and assign the provided tag
--hanging-edges	remove certain classes of hanging edges
--double-faces	check for multiply-connected face neighbors
--reorder	reorder nodes with Reverse Cuthill-McKee
--lump-bcs <mapbc_file_name>	reassign all boundary tags with the same name to a single tag
--quilt-tags <merged_tag bc_tags>	merge multiple boundary tags into one merged tag

The bc_tags can be a single tag, a space delineated tag list, and/or a range of tags specified with a ":" separator. The allowable AFLR3 file format extensions are .lb8.ugrid (little endian), b8.ugrid (big endian), or .ugrid (ASCII). Allowable output grid formats are:

- .lb8.ugrid, b8.ugrid, or .ugrid (AFLR3 format)
- .plt (binary Tecplot format)
- .vtk (legacy VTK format)

interpolate
The interpolate subcommand interpolates ".snap" file data from one grid to another grid. The subcommand usage line for this feature is as follows:

```
vinf interpolate --source <source mesh>
                 --snap <source snap file>
                 --target <target mesh>
                 -o <target snap file>
```

where:

--source <source_mesh>	source grid file name (AFLR3 format) [REQUIRED]
--target <target_mesh>	target grid file name (AFLR3 format) [REQUIRED]
--snap <source_snap_file>	snap file to interpolate the results from [REQUIRED]
--smooth	smooth the fields interpolated to the mesh
-o <target_snap_file>	snap file to interpolate the results to

plot
The plot subcommand can be used to visualize grids and simulation results. This subcommand creates smaller visualization files from an AFLR3 grid file, a vulcan_solution.snap file (nodal data), or a VULCAN-CFD restart ".snap" file (cell center data) for efficient visualization. The subcommand outputs either crinkle-cut subvolumes or specified boundary surfaces. The plot subcommand usage line to visualize AFLR3 format grid files is as follows:

```
vinf plot -m <mesh file>
          -o <output file>
```

Field data can be added to the visualization file by also loading a .snap file, e.g.,

```
vinf plot -m <mesh file>
          --snap <snap file>
          -o <output file>
```

The complete option list is provided below:

-m <mesh_file_name>	grid file name [REQUIRED]
-o <output_file>	plot file name for visualization [REQUIRED]
--snap <snap_file_name>	input snap file for field variable visualization
--tags <bc_tags>	extract boundary surface data from list of mesh boundary tags
--surface	extract all boundary surface data (no volume data)
--volume	extract all volume data (no boundary surface data)
--slice <x,y,z = value>	extract crinkle-cut within the provided x, y, or z plane
--sphere <x y z r>	extract crinkle-cut inside sphere centered at x,y,z with radius r
--at <nodes or cells>	specifies whether to average cell data to nodes or vise versa

The extensions provided for the grid and output file names determine the file format. Allowable input grid file extensions are .(l)b8.ugrid (little / big endian AFLR), .ugrid (ASCII), or .meshb. Allowable output file formats are:

- .plt (binary Tecplot format)
- .vtk (legacy VTK format)
- .stl (binary STL file available only for boundary surface output)

The --volume option will only add volume (tet, prism, pyramid, hex) elements to the output file. The --surface option will only add surface (triangle, quad) elements to the output file. The --surface option will also automatically add the surface cell boundary tag as a field to the output file. One common use case of `vinf plot` is to use the --surface option to see the boundary tags when setting up a VULCAN-CFD input file when a boundary condition map (.mapbc) file is not provided. Note also that the filter options for **plot** are Boolean AND operations. For example:

```
vinf plot -m <mesh> --surface --sphere 0 0 0 1
```

will plot surface cells (triangles and quads) that are within 1 grid unit of the origin. All the filtering operations in `vinf plot` are crinkle-cut subvolume filters, so if any part of a cell is within the filtering region, the entire cell is added to the output. For surface extractions see `vinf sampling`.

sampling
The sampling subcommand offers an alternative to the plot subcommand to visualize grids and simulation results. As with the plot subcommand, sampling creates smaller visualization files from an input grid file, a vulcan_solution.snap file, or a VULCAN-CFD restart ".snap" file for efficient visualization. The difference is that the sampling subcommand outputs interpolated volume data (cell or node based) to a triangulated surface mesh rather than crinkle cut subvolumes. The sampling subcommand usage line to visualize AFLR3 format grid files is as follows:

```
vinf sampling -m <mesh file>
              <optional filtering commands>
              -o <output file>
```

Field data can be added to the visualization file by also loading a .snap file, e.g.,

```
vinf sampling -m <mesh file>
              --snap <snap file>
              <optional filtering commands>
              -o <output file>
```

The complete option list is provided below:

-m <mesh_file_name>	grid file name [REQUIRED]
-o <output_file>	plot file name for visualization [REQUIRED]
--snap <snap_file_name>	input snap file for field variable visualization
--plane <x y z n1 n2 n3>	extract a plane centered at x,y,z with normal n1,n2,n3
--sphere <x y z r>	extract a spherical surface centered at x,y,z with radius r
--iso-surface <field_name>	extract an iso-surface for this field variable
--iso-value <value>	extract this iso-surface value

If the field name for the iso-surface extraction includes blank spaces, then the field name must be enclosed within double quotes. If sampling from a vulcan_solution.snap file, then the field name is the variable name exactly as it appears in the tecplot.nam file (alternatively the field names from any .snap file can be extracted using the **snap** subcommand described below). The extensions provided for the grid and output file names determine the file format. Allowable input grid file extensions are .(l)b8.ugrid (little / big endian AFLR), .ugrid (ASCII AFLR), or .meshb. Allowable output file formats are:

- .plt (binary Tecplot format)
- .vtk (legacy VTK format)
- .stl (binary STL file (no field data will be written)

snap
The snap subcommand allows the user to inspect the contents of a ".snap" file. The simple use case is to list the fields within a snap file using:

```
vinf snap -s <snap file>
```

Snap files contain an arbitrary number of flow-field variables corresponding to an unstructured grid. Snap files do not contain the mesh, they accompany a mesh file similar to the distinction between Plot3D grid and solution files. Snap files can support fields located at grid nodes or grid cells. VULCAN-CFD writes two different snap formatted files during normal operation. The postprocessor will export a "vulcan_solution.snap" file that contains all the fields that the user requested in the **PLOT FUNCTION** list of the input file. The solver will also output a snap file containing the VULCAN-CFD restart data as follows:

- uq_n (where n is the nondimensional conserved variable number starting from 0)
  - The species densities are numbered from 0 through number of species minus 1.
  - The next variable is the mixture density.
  - The next 3 variables are the x-, y-, and z-momentum components.
  - For thermal nonequilibrium simulations, the next 2 variables are the vibrational/electronic energy and total energy per unit volume, otherwise the next variable is the total energy per unit volume.
  - The remaining variables are the transported turbulence variables.
- uqg_n (where n is the nondimensional gas variable number starting from 0)

  - Variable 0 is the ratio of specific heats.
  - Variable 1 is the static pressure.
  - Variable 2 is the static temperature.
  - Variable 3 is the static enthalpy.
  - Variable 4 is the specific heat at constant pressure.
  - Variable 5 is the vibrational/electronic temperature (thermal nonequilibrium simulations only).

- utp_n (where n is the nondimensional transport property variable number starting from 0)
  - Variable 0 is the molecular viscosity.
  - Variable 1 is the eddy (or SGS) viscosity.
  - Variable 2 is the molecular conductivity.

The restart snap file is written at cell centers and the solution file is written at nodes. The purpose of the "vulcan_solution.snap" file is to be able to separately postprocess the solution data using the plot or sampling subcommands. The "vulcan_solution.snap" file is also used to output the sensor function used for unstructured grid adaptation.

transform
The transform subcommand performs a variety of grid transformation to an AFLR3 unstructured grid file. The subcommand usage line for this feature is as follows:

```
vinf transform  -m <mesh_file_name>
                --translate <x,y,z>
                -o <transformed_mesh_file_name>
```

where:

-m <mesh_file_name>	input grid file name [REQUIRED]
-o <output_grid_name>	output grid file name [REQUIRED]
--translate <x,y,z>	translate the grid using specified offsets
--scale <value>	scale the grid about (0,0,0)
--scale-x <value>	scale the x-coordinates about (x=0)
--scale-y <value>	scale the y-coordinates about (y=0)
--scale-z <value>	scale the z-coordinates about (z=0)
--rotate-x <value>	rotate coordinates about the x-axis
--rotate-y <value>	rotate coordinates about the y-axis
--rotate-z <value>	rotate coordinates about the z-axis
--reflect-x <value>	reflect coordinates about the x-value
--reflect-y <value>	reflect coordinates about the y-value
--reflect-z <value>	reflect coordinates about the z-value

The extension of the grid file names determine the file format. Acceptable input grid file name extensions are .lb8.ugrid (little endian), b8.ugrid (big endian), or .ugrid (ASCII). Allowable output grid formats are:

- .lb8.ugrid, b8.ugrid, or .ugrid (AFLR3 format)
- .plt (binary Tecplot format)
- .vtk (legacy VTK format)

validate
The validate subcommand performs a variety of mesh validity checks to an AFLR3 unstructured grid file. These are the same checks that are run by VULCAN-CFD during initialization for unstructured simulations. The `vinf validate` command can be run separately to help triage issues related to mesh generation prior to starting the solver. Currently, the following checks are performed:

- All cells have a positive volume.
- All cell faces create a closed surface.
- The mesh contains no hanging edges.
- The mesh contains no hanging nodes.
- All volume cells have face neighbors (no missing surface faces).
- All nodes are contained within at least one cell.

The only argument for this subcommand is the name of the AFLR3 grid file, i.e.,

`vinf validate -m <mesh_file_name>`

where:

`-m <mesh_file_name>` grid file name (AFLR3 format) [REQUIRED]

Allowable AFLR3 file format extensions are .lb8.ugrid (little endian), b8.ugrid (big endian), or .ugrid (ASCII).

25.8 Scale-Resolving Utility Codes

The Utilities/LES_tools directory contains the utility **fluct_ext** that can be used to superimpose resolved boundary layer fluctuations on existing time-averaged mean boundary layer(s) that are present in the restart file data. The use of this utility will accelerate the formation of resolved eddy structures for simulations that use the recycling/rescaling process described in the inflow recycling/rescaling control data section of Chapter 8. This utility prompts for the extent of the computational domain where resolved eddy structures are desired (typically the recycling/rescaling region of the domain) as well as estimates of the boundary layer thickness and friction velocity for the boundary layer(s) in this region. These data are used by the utility to scale an existing box of resolved boundary layer structures to the supplied portion of the computational domain. The output data generated from this process are written to the file fluct.dat. Placing this file into a subdirectory named **Recycle_files** in the working directory of the VULCAN-CFD simulation will flag the solver to superimpose these fluctuations on the velocity field present in the restart files. The addition of fluctuations only occurs during the initialization of the recycling/rescaling process, and will be skipped if VULCAN-CFD detects that the recycling/rescaling process has previously been initialized.

NOTE: This utility is only applicable to scale-resolving simulations that utilize structured grids. Moreover, the utility currently only allows resolved fluctuations to be added to boundary layers attached to no-slip surfaces that correspond to "J" boundaries of the structured grid.

NOTE: If resolved fluctuations are desired for a simulation that does not utilize the recycling/rescaling process, then one way to take advantage of this feature is to create a temporary VULCAN-CFD input file that utilizes the recycling process with the number of iterations set to zero. VULCAN-CFD can then be executed as described above to generate a new set of restart files that have the eddy structures superimposed on the mean velocity profiles. These restart files can then be utilized for the scale-resolving simulation that does not utilize the recycling/rescaling process.

25.9 Grid Convergence Index Extraction

The **gci_ext** utility, located in the Utilities/Post_Process_codes/GCI directory, computes the Grid Convergence Index (GCI)[82] for simulation output quantities based on extracted values from multiple grid resolutions. The GCI is a grid convergence estimator derived from the generalized Richardson extrapolation formula and can be written as follows:

$$\mathrm{GCI} = F_s \frac{|f_1 - f_2|}{r^p - 1} \tag{1}$$

where f is some output parameter of interest evaluated at two different grid resolutions (f_1 and f_2), r is the grid refinement ratio, and p is order of accuracy of the numerical scheme. Note that r is defined based on the refinement used for each coordinate direction, which in general is defined by:

$$r = \left(\frac{\text{total number of fine grid cells}}{\text{total number of coarse grid cells}} \right)^{\frac{1}{n}}$$

where n is the spatial dimension of the problem (e.g, 3 for 3-D). Finally, F_s is a safety factor with recommended values taken to be either 3 (if the observed order of accuracy is assumed to be the theoretical value of the numerical scheme) or 1.25 (if the observed order of accuracy has been rigorously determined). The determination of the observed order of accuracy requires that simulations be performed using a minimum of three grids. If the values of the output quantities are in the asymptotic convergence range (implying that the output quantity is converging to its grid-resolved value monotonically at a constant rate), then the order of accuracy can be extracted using the relationship below.

$$p = \ln\left(\frac{f_3 - f_2}{f_2 - f_1} \right) \; / \; \ln(r) \tag{2}$$

In order to confirm that the observed order of accuracy extracted from this expression is the true value, an additional grid level must be utilized in order to check whether a consistent value of "p" can be obtained. However, this is rarely practical. Instead, it is recommended that the value of p be scrutinized to see if it is a reasonable value, i.e.,

- Is the value between unity and the theoretical value?
- Is a similar value of p obtained for multiple output quantities?

If the answer to both questions is yes, then this provides some evidence that the value of p extracted is a true value and can probably be trusted in the GCI expression. If the answer to either of these questions is no, then the asymptotic convergence requirement has not been met on one or more of the grids used. When this occurs, only the output values from the finest two grid levels should be used to compute the GCI using the theoretical order of accuracy and a safety factor of at least 3.

The **gci_ext** utility accepts either the direct input of an output quantity at multiple grid refinement levels, or files containing spatially varying output quantities (one for each refinement level) can be provided. The first option is the only one that will attempt to compute

the observed order of accuracy (assuming that values from at least 3 refinement levels have been provided). The extraction of p is not attempted for spatially varying output quantities, since it is highly unlikely that a local quantity will be "grid converged" at all spatial locations. Regardless of how the data are supplied to this utility, the refinement ratio (r) must be uniform. Further details about the specific data requirements for this utility are described below:

- The direct input option will extract the observed order of accuracy if more than 2 levels of refinement are considered. If the extracted value is realizable (i.e., if the value is at least unity but no greater than the design order of accuracy), then the extracted value is used to compute the GCI. Otherwise, the user is prompted to input the order of accuracy to be used in the GCI expression.

- If spatially varying output quantities are provided through data files, then the user is always prompted for the order of accuracy to be used in the GCI expression.

- When prompted for the order of accuracy, a value between unity and the design spatial order of accuracy of the numerical scheme should be entered (typically 2).

- If a value of unity is entered for the order of accuracy, then a safety factor of 1.25 will be used. The reasoning here is that a value of unity would only be entered if the user has a strong belief that unity is the true value due to flux limiting (or 1^st-order boundary condition treatments). Any other order of accuracy value that is entered will use a safety factor of 3 in the GCI expression.

- When spatially varying output quantities are provided from data files, then each file must only contain data from a specific grid refinement level. Generally speaking, the data in each of these files must have also been interpolated to a common set of grid points. The one exception is the common scenario when the refinement level is 2 (i.e., every other grid point is removed when coarsening the grid). In this case, the output quantities from each refinement level can reside on the host grid that generated the data.

NOTE: When interpolating the output quantities to a common grid, it is recommended that the common grid be at least as refined as the finest grid used for the simulations so that the interpolation errors are minimized. If using Tecplot, a simple approach to extract the output data along a common set of grid points is to use the "Extract Precise Line" option with the number of extracted points set to a large enough value to provide a point distribution that is no coarser than that of the finest grid used for the simulations.

25.10 Propulsion Flowpath Performance Extraction

The use of CFD for the design and analysis of propulsion systems often requires the extraction of "one-dimensional" figures of merit (e.g., total pressure recovery, mixing and combustion efficiency) from simulation data. The extraction of one-dimensional properties from multidimensional simulation data is realized by applying some sort of data reduction technique to a family of computational surfaces (or lines in two dimensions) as illustrated in Fig. 20. The surfaces of interest will generally correspond to the cross-flow planes of the propulsion system flowpath. A variety of data reduction techniques exist for the extraction of performance measures from multidimensional data sets, but in general, the techniques can be categorized as either a weighted or flux-based approach. The VULCAN-CFD Theory Manual[1] provides details on the following data reduction techniques from both categories:

- Area weighted average
- Mass flux weighted average
- Conserved mass/momentum/energy (CMME) flux average
- Conserved mass/momentum/energy flux average with the Langley distortion method
- Conserved mass/momentum/energy flux average with the AFRL distortion method
- Conserved mass/energy/entropy (CMES) flux average

Each of these data reduction techniques are available in the **perf_ext** utility (located in the Utilities/Post_Process_codes/Perform directory).

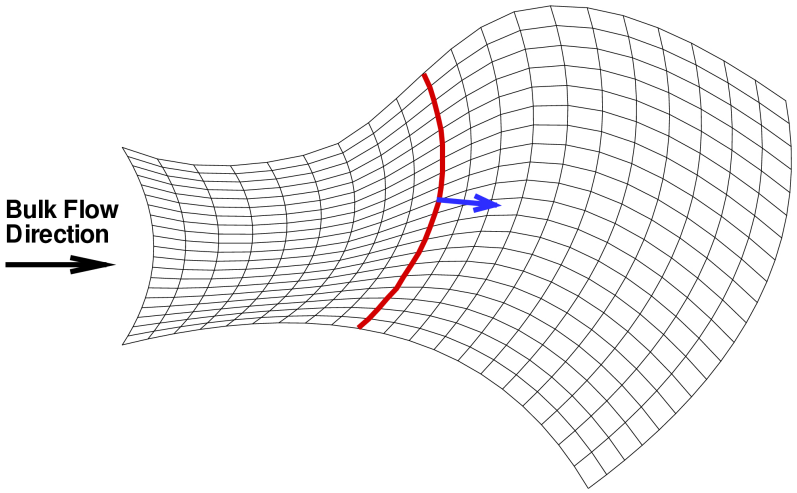


Figure 20: Surface of integration and the unit normal for the extraction of system performance metrics.

The **perf_ext** utility operates on the provided simulation flowfield data to produce an ASCII Tecplot file (**perf_ext.tec**) that contains the performance metrics at each streamwise

station. This utility is quite flexible with regards to how the simulation data can be provided, and as a result, the specific information requested during **perf_ext** execution will be dependent on the responses provided from previous prompts. The prompt sequence listed below captures the high-level dependencies that determine the specific questions that appear during **perf_ext** execution:

(1) Enter the desired approach to one-dimensionalize the data

- Area weighted average
- Mass flux weighted average
- Conserved flux (mass/momentum/energy) average
- Conserved flux (mass/momentum/energy) average with Langley distortion
- Conserved flux (mass/momentum/energy) average with AFRL distortion
- Conserved flux (mass/energy/entropy) average

(2) Enter the desired approach to obtain the flux information

- Flux data computed from nodal PLOT3D or ASCII Tecplot files
- Structured grid simulation flux data provided by VULCAN-CFD

(3) Enter the file name(s) that contain the simulation data

- If data resides in PLOT3D format files then enter the grid file name, function file name, and the variable name file
- If data resides in an ASCII Tecplot format file then enter the file name

  NOTE: If a Tecplot format file is provided, the data can either be structured grid simulation data or cross-sectional planes of data that have been extracted from volumetric simulation data using the Tecplot Slice Extraction Tool

(4) If structured grid simulation data has been provided then

- Enter the list of structured grid blocks to consider
- Enter the structured grid streamwise direction (i, j, or k)
- Enter the information required to merge planar integrated data [optional]

  NOTE: Merging is only required when multiple structured grid blocks are involved in the streamwise plane integration process

(5) Enter the direction cosines that best define the 1-D streamwise direction

(6) Select the performance metric(s) desired and provide any auxiliary required to compute the metric(s)

The following is a list of additional details related to the information requested during the execution of **perf_ext**:

- When working with unstructured grid simulation data, the volumetric simulation data must first be processed by Tecplot to extract a sequence of streamwise planes. This step is accomplished by using the Tecplot Slice Extraction Tool, where a start and end streamwise plane is selected along with "N" intermediate stations. These planes must then be exported to Tecplot zones, so that they can be written out to an ASCII Tecplot data file.

- When working with structured grid simulation data, all of the options described above are available. However, if the structured grid block topology does not accommodate the construction of a single contiguous cross-sectional plane (in computational space) that spans the entire cross section, then the Tecplot Slice Extraction option must be used.

- When prompted for the direction cosines, setting all components of the direction cosine vector to zero will align the streamwise direction with the 1-D velocity vector at each streamwise plane. If there is no desire to maintain a uniform 1-D unit vector for every streamwise plane, then this option can be chosen to minimize the distortion due to flow alignment when the CMES method or one of the CMME methods with distortion is chosen.

- Given that the weighted approaches and the CMME method produce uniform flow properties that are not true one-dimensionalized properties (i.e., the "1-D" velocity direction is not controlled by the user) the value entered for the direction cosines will only affect the output of the 1-D area distribution when these techniques are utilized.

- The prompts associated with Step (4) above only pertain to structured grid simulations when the volumetric plot files output by VULCAN-CFD are utilized (i.e., the user will not be prompted for this information if the flowfield data are supplied via the Tecplot Slice Extraction option).

- When prompted for the specified streamwise direction in Step (4), the direction provided must be the same for all blocks that are involved in the one-dimensionalization process. If this is not the case, then the Tecplot Slice Extraction option must be used.

- If planar integrated data need to be merged in Step (4) due to the integration plane spanning multiple structured grid blocks, then connectivity information is required to allow the integrated values to be merged across all blocks that span the cross-sectional plane. The merge process involves picking one block as a master, and the remaining blocks simply pass their integrated data to the chosen master block. Two lines of information are required to merge each slave block to the master block. The first line of the pair is always the master block number followed by its streamwise index range that needs to be merged with the slave block. The second line of the pair is the slave block number followed by the streamwise index range to be merged with the master block. A streamwise index value of "0" can be specified as a wild card to denote the last index of the block. To illustrate this process, consider the structured block topology shown in Fig. 21. If Block 4 is taken as the master for the merge process, then blocks 1, 2, 3, and 5 must be merged with block 4. This choice will result in a single Tecplot zone of one-dimensionalized data, since block 4 spans the entire

streamwise extent of the computational domain. The following lines are required to properly merge the integrated data for this example:

```
4                       (Number of block merge pairs)
4     1     9           (Merge block 1 to block 4)
1     1     0
4     10    41          (Merge block 2 to block 4)
2     2     0
4     10    41          (Merge block 5 to block 4)
5     2     0
4     42    0           (Merge block 3 to block 4)
3     2     0
```

Note that the merge of block 2 with block 4 began with a streamwise index of 2 rather than 1 to avoid double bookkeeping the shared block interface that was already accounted for when block 1 was merged with block 4. A similar scenario occurs when merging block 3 with block 4. Alternatively, blocks 1, 2, and 3 can be chosen as master blocks for the merge process. With this choice, the following lines would be required to properly merge the integrated data:

```
4                       (Number of block merge pairs)
1     1     0           (Merge block 4 to block 1)
4     1     9
2     1     0           (Merge block 4 to block 2)
4     9     41
2     1     0           (Merge block 5 to block 2)
-5    1     0
3     1     0           (Merge block 4 to block 3)
4     41    0
```

resulting in 3 Tecplot zones of one-dimensionalized data. This example activated an additional feature when merging block 5 to block 2 that allows the streamwise coordinate values in the slave block to be ignored when merging. This does not affect the one-dimensional property values, but it will affect the spatial location assigned to them. Given the skewness of the grid lines in the downstream portion of block 5, invoking this option will provide a more visually appealing line plot of the one-dimensionalized data.

The output from the **perf_ext** utility is an ASCII Tecplot data file (**perf_ext.tec**) that contains the performance metrics at each streamwise station. The location of the streamwise station is defined as the x, y, or z coordinate value averaged over each streamwise plane. The particular choice is determined by the coordinate with the largest variation over the streamwise extent of the domain. The following performance metrics are available:

- stream thrust, F
- total pressure recovery, $P_{\circ}^{rec}$

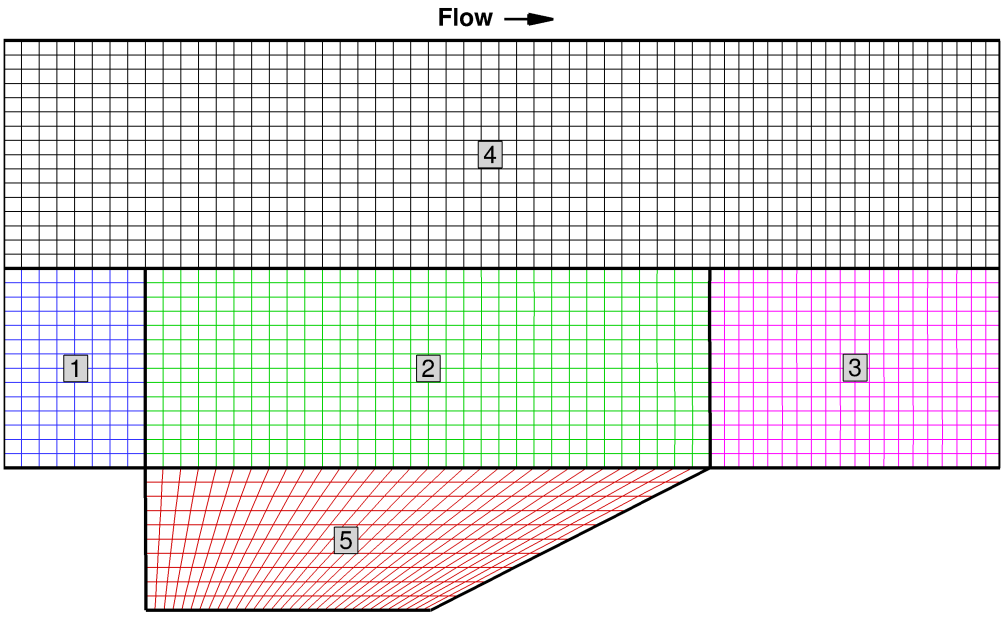


Figure 21: Sample structured grid block topology to illustrate the integration merge process.

- inlet kinetic energy efficiency, η_{KE}
- inlet adiabatic kinetic energy efficiency, η_{KE_a}
- inlet compression efficiency, η_B
- mixing efficiency, η_m
- combustion efficiency based on reactant depletion, η_c^r
- combustion efficiency based on product formation, η_c^p
- combustion efficiency based on heat of formation, η_c^h
- combustion efficiency based on total temperature, $\eta_c^{T_\circ}$
- thrust potential via expansion to a desired pressure, TP_P
- thrust potential via expansion to a desired area, TP_A
- pattern factor based on static (or total) temperature
- weighted average of the streamwise component of vorticity

Other potentially useful properties (that are not specifically performance metrics) can be output as well, such as:

- area (integrated area projected to the 1-D direction)
- mass flow rate ($\dot{m}$)

- equivalence ratio (ER)
- weighted averages of user-specified variables

Finally, any of the above properties can be tabulated at specific user-supplied streamwise stations to the file **perf_ext.txt**.

The 1-D area and the mass flow rate are always output by **perf_ext**. The 1-D area is defined as the integrated area of the streamwise plane projected to the specified 1-D direction,

$$\text{Area} \;=\; \vec{\mathbf{n}} \cdot \int \vec{n}\, dA \tag{3}$$

and the mass flow rate, $\dot{m}$, is provided by the following relationship.

$$\dot{m} \;=\; \int \left[\rho \left(\vec{\mathrm{v}} \cdot \vec{n}\right)\right]\, dA \tag{4}$$

The evaluation of the mass flow rate requires the multidimensional density and velocity values to be available unless the fluxes are output directly by VULCAN-CFD (see the **OUTPUT FLUXES** input specification in the output control data section of Chapter 8). The stream thrust vector at each streamwise plane is given by,

$$\vec{\mathrm{F}} \;=\; \int \left[\rho \left(\vec{\mathrm{v}} \cdot \vec{n}\right) \vec{\mathrm{v}} \;+\; P\, \vec{n}\right]\, dA \tag{5}$$

which also requires the presence of density and velocity variables (as well as pressure), unless the fluxes are output directly by VULCAN-CFD. This vector quantity can be converted to a scalar value by taking the dot product of this vector with the specified 1-D direction as shown below.

$$\mathrm{F} \;=\; \vec{\mathbf{n}} \cdot \vec{\mathrm{F}} \tag{6}$$

The total pressure recovery is a generic measure of flow losses through the propulsion system, and is defined by the relation:

$$P_{\circ}^{rec} \;=\; \frac{\mathbf{P}_{\circ}}{\mathrm{P}_{\circ}^{\mathrm{ref}}} \tag{7}$$

where $\mathbf{P}_{\circ}$ is the one-dimensionalized value of the total pressure and $\mathrm{P}_{\circ}^{\mathrm{ref}}$ is a reference total pressure value (typically taken as the freestream value). $\mathbf{P}_{\circ}$ is either the weighted average of total pressure (if a weighted approach is used), or the 1-D value recovered from the flux equations if a conserved flux approach is used.

The kinetic energy and compression efficiency metrics are typically associated with high speed inlet systems, and represent different attempts to quantify how efficient the compression processes are. The kinetic energy efficiency at station "i" is obtained from the following ratio of kinetic energy measures:

$$\eta_{KE} \;=\; \frac{\mathbf{h}_{\circ}^{i} \;-\; \mathbf{h}_{\infty}^{*}}{\mathbf{h}_{\circ_{\infty}} \;-\; \mathbf{h}_{\infty}} \tag{8}$$

where $\mathbf{h}_{\circ}$ is the total enthalpy, $\mathbf{h}$ is the static enthalpy, and ∞ denotes a freestream value. The quantity $\mathbf{h}_{\infty}^{*}$ is a freestream enthalpy value obtained by isentropically expanding the

1-D state to the freestream pressure. As a result, the numerator in this expression is not the local kinetic energy. Instead, it is a modified kinetic energy value based on the freestream enthalpy recovered by this ideal process. The Mollier diagram provided in the left image of Fig. 22 graphically illustrates this measure of inlet efficiency. Given this definition, the kinetic energy recovery of an isentropic compression process without surface heat transfer would be 100% efficient, so this measure of inlet efficiency accounts for both flow losses and heat losses through the inlet surfaces. The adiabatic kinetic energy efficiency is an alternative measure of kinetic energy efficiency that only accounts for the flow losses, i.e.,

$$\eta_{KE_a} = \frac{\mathbf{h}_{o_\infty} - \mathbf{h}^*_\infty}{\mathbf{h}_{o_\infty} - \mathbf{h}_\infty} \tag{9}$$

so the flow and heat losses can be book-kept separately if desired. The compression efficiency metric is defined by the following relationship:

$$\eta_B = \frac{\mathbf{h}^* - \mathbf{h}_\infty}{\mathbf{h}^i - \mathbf{h}_\infty} \tag{10}$$

where $\boldsymbol{h^i}$ is the static enthalpy at station "i" and $\mathbf{h^*}$ is the enthalpy at station "i" obtained through an isentropic compression process from the freestream state to the local 1-D pressure value. The Mollier diagram provided in the right image of Fig. 22 graphically depicts this measure of inlet efficiency. Given this definition, the compression efficiency would be 100% for an isentropic compression process without any heat loss through the surfaces. The compression efficiency tends to produce a fairly wide range of values as the freestream Mach number and amount of inlet compression is varied. This is not the case for the kinetic energy efficiencies, which often require 3 digits of precision at high freestream Mach numbers in order to discriminate between performance differences.

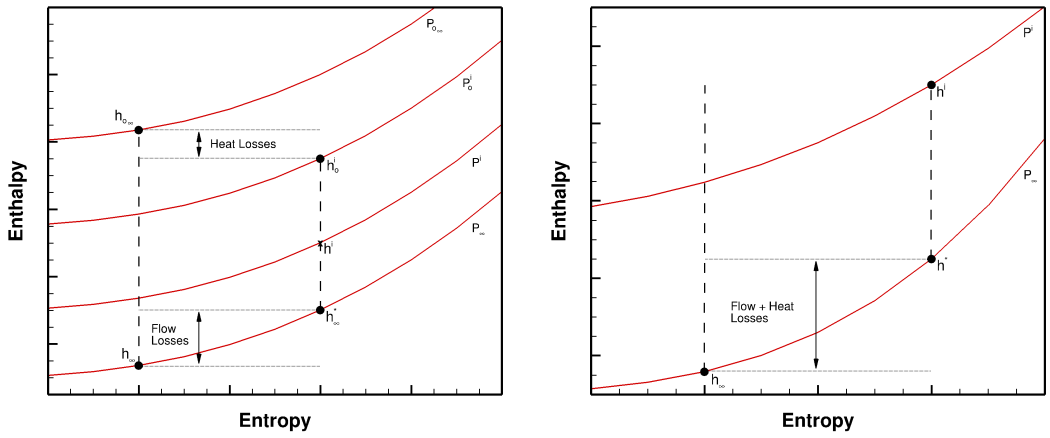


Figure 22: Mollier diagrams illustrating the thermodynamic process used to define kinetic energy (top) and compression (bottom) efficiency.

Performance metrics specific to the combustor component are the mixing efficiency and combustion efficiency. The combustor performance in terms of these metrics is strongly dependent on the fueling condition of the combustor, which is typically categorized by the

equivalence ratio. The equivalence ratio is defined as the fuel/oxidizer ratio being supplied to the combustor relative to the stoichiometric value, i.e.,

$$\mathrm{ER} \;=\; \frac{\dot{m}_f/\dot{m}_o}{\left(Y_f/Y_o\right)_{st}} \tag{11}$$

where an ER value less than unity implies the combustor is operating at a fuel lean condition, while values greater than unity denote fuel rich operating conditions. The mixing efficiency[83] expression used by **perf_ext** depends directly on the fueling condition of the combustor:

$$\eta_m \;=\; \frac{\int \alpha_R \rho U dA}{\int \alpha \rho U dA} \tag{12}$$

where α is the mass fraction of the least available reactant (i.e., fuel for lean operating conditions or oxidizer for fuel rich operation), and α_R is the portion of α that could react if complete reaction took place without further mixing. The expression for α_R can be written as

$$\alpha_R = \begin{cases} \alpha, & \alpha \le \alpha_{st} \\ \alpha_{st}\left(\frac{1-\alpha}{1-\alpha_{st}}\right), & \alpha > \alpha_{st} \end{cases} \tag{13}$$

where α_{st} is the stoichiometric value. Note that when evaluating the mixing efficiency for reacting simulations, the fuel (or oxidizer) mass fractions must be interpreted as the conserved "elemental" values to separate the mixing process from any depletion of fuel (or oxidizer) that results from combustion. The **perf_ext** utility provides four different definitions for the combustion efficiency. The simplest measure is based on the depletion of the least available reactant:

$$\eta_c^r \;=\; \frac{\dot{m}_\alpha}{\dot{m}_\alpha^{in}} \tag{14}$$

where α again denotes either the fuel or oxidizer based on the operating condition of the combustor, and $\dot{m}_\alpha^{in}$ is the mass flow rate of α entering the combustor. This measure is a reasonable choice when the fuel directly breaks down to form radicals and/or combustion products. However, it should not be used for heavy hydrocarbon kinetic systems where the heavy hydrocarbon fuel breaks down into lighter hydrocarbons before reacting with the oxidizer. The combustion efficiency based on product formation is obtained from the following expression:

$$\eta_c^p \;=\; \frac{\dot{m}_p}{\dot{m}_\alpha^{in}} \tag{15}$$

where $\dot{m}_p$ is the mass flow rate of all species that are designated as combustion "products" scaled by the elemental factors for each element present in the lean constituent (α), i.e.,

$$\dot{m}_p \;=\; \sum_{m=1}^{ne} W_m \sum_{n=1}^{np} \frac{N_{mn}}{W_n}\, \dot{m}_n \tag{16}$$

In this expression, ne is the number of elements present in species α, np is the number of species that were designated as products of combustion, N_{mn} is the number of moles of element "m" present in species "n", and W is the species (or element) molecular weight.

The **perf_ext** utility assumes that the fuel is either hydrogen or some hydrocarbon, implying that the potential product species are H_2O, CO_2, and CO. For fuel lean operating conditions, the elemental H and C atom counts for each product species present in the system are used to compute $\dot{m}_p$, while the elemental O atom count of each product species is used for fuel rich operation. If none of the above product species are present, then the utility will prompt the user for the species to be taken as the main product of the chemically reacting system. This measure is almost identical to the fuel depletion efficiency when only a single product species exists as would be the case for a hydrogen fueled system. The downside of this measure arises when multiple product species are present that share the same elements (e.g., CO_2 and CO). This measure has no knowledge of the difference in heat release between the formation of competing products, so it does not detect the more efficient formation of CO_2 relative to CO. The combustion efficiency based on enthalpy of formation provides this feature:

$$\eta_c^h \;=\; \frac{h(\mathbf{Y_m}, \mathrm{T_{ref}}) \;-\; h(\mathbf{Y_m^{in}}, \mathrm{T_{ref}})}{h(\mathbf{Y_m^*}, \mathrm{T_{ref}}) \;-\; h(\mathbf{Y_m^{in}}, \mathrm{T_{ref}})} \tag{17}$$

where the superscript "i" denotes 1-D values at the current streamwise station, the superscript "*" denotes some ideal state for the current streamwise station, $\mathbf{Y_m^{in}}$ denotes the mass fractions of all constituents that enter the combustor, and $\mathrm{T_{ref}}$ is the reference temperature associated with the thermodynamics model (i.e., 298.15 K). The **perf_ext** utility sets the ideal composition based on an equilibrium calculation using the 1-D enthalpy and pressure at the current streamwise station. This combustion efficiency metric is often the most informative measure, since it is based directly on the enthalpy of formation. The combustion efficiency based on total temperature rise is defined as follows:

$$\eta_c^{T_\circ} \;=\; \frac{\mathbf{T}_\circ \;-\; \mathrm{T}_\circ^{\mathrm{ref}}}{\mathbf{T}_\circ^* \;-\; \mathrm{T}_\circ^{\mathrm{ref}}} \tag{18}$$

where the superscript "i" denotes 1-D values at the current streamwise station, the superscript "*" denotes some ideal state for the current streamwise station, and $\mathrm{T}_\circ^{\mathrm{ref}}$ is a user-provided reference value (typically taken to be representative of the total temperature entering the combustor). The **perf_ext** utility defines the ideal total temperature by performing an equilibrium calculation using the 1-D total enthalpy and total pressure at the current streamwise station. This measure of combustion efficiency is unique in that it accounts for heat losses through the combustor surfaces. However, its usefulness degrades when the total temperature rises to levels where excessive dissociation occurs at chemical equilibrium.

There are 2 thrust potentials available in **perf_ext** that provide the potential thrust that can be gained when all (or part) of the engine nozzle is not considered by the CFD simulation. These performance metrics are particularly useful when performing direct-connect simulations that neglect the engine aft nozzle component, or partial free-jet facility simulations where a portion of the aft nozzle is absent. The first thrust potential considers the expansion to a user-specified pressure, and the second one considers the expansion to a user-specified area. Both thrust potentials take the current streamwise station as the "inflow" station for the subsequent ideal isentropic expansion process. The composition is also frozen at the "inflow" state throughout the expansion process. The expression for the thrust potential is as follows:

$$\mathrm{TP} \;=\; \dot{m}\,\mathbf{v_e} \;+\; (\mathbf{P_e} - \mathrm{P_{ref}})\;\mathbf{A_e} \;-\; \dot{m}\;(\vec{\mathbf{v}}_i \cdot \vec{\mathbf{n}}) \;-\; (\mathbf{P_i} - \mathrm{P_{ref}})\;\mathbf{A_i} \tag{19}$$

where $\dot{m}$ is the total mass flow rate (sum of the fuel and air streams) through the propulsion system, $\vec{\mathbf{v}}_i \cdot \vec{\mathbf{n}}$, $\mathbf{P_i}$, and $\mathbf{A_i}$ are the 1-D velocity, pressure, and area values at streamwise station "i", $\mathbf{v_e}$, $\mathbf{P_e}$, $\mathbf{A_e}$ are the velocity, pressure, and area at the end of the expansion process, and P_{ref} is the freestream reference pressure. The introduction of P_{ref} is intended to accommodate a force accounting system that utilizes gauge pressure and should be set to zero otherwise. In order to close this equation, values for $\mathbf{v_e}$, $\mathbf{P_e}$, and $\mathbf{A_e}$ must be determined. The following auxiliary expressions that govern the ideal isentropic expansion process provide the relationships to close the thrust potential expression:

$$\text{mass conservation:} \quad \dot{m} = \boldsymbol{\rho_e}\, \mathbf{v_e}\, \mathbf{A_e} \tag{20a}$$

$$\text{adiabatic flow:} \quad \mathbf{h}_{\circ_i} = h_{\circ_e}(\mathbf{Y^i_m}, \mathbf{T_e}) \tag{20b}$$

$$\text{insentropic flow:} \quad \mathbf{s_i} = s_e(\mathbf{Y^i_m}, \mathbf{T_e}, \mathbf{P_e}) \tag{20c}$$

$$\text{ideal gas law:} \quad \mathbf{P_e} = \boldsymbol{\rho_e}\, R(\mathbf{Y^i_m})\, \mathbf{T_e} \tag{20d}$$

To obtain the thrust potential value based on an isentropic expansion to a user-supplied pressure (TP_P), the $\mathbf{P_e}$ value is specified and the above equations can be rearranged to yield $\mathbf{v_e}$ and $\mathbf{A_e}$. Similarly, to obtain the thrust potential value based on an isentropic expansion to a user-supplied area (TP_A), the $\mathbf{A_e}$ value is specified and the above equations can be rearranged to yield $\mathbf{v_e}$ and $\mathbf{P_e}$.

The pattern factor is a metric that aims to quantify the level of distortion present in the inviscid core of the flowfield. This metric may be applied to any component of the propulsion system, but it is most commonly applied to the exit of the combustor to determine the uniformity of the combustion process. Two temperature-based pattern factors are available in **perf_ext**; one based on static temperature and the other based on total temperature. The pattern factor at a given streamwise station is computed as follows:

$$PF = \frac{T_{\max} - \mathbf{T}}{\mathbf{T}} \tag{21}$$

where $\mathbf{T}$ is the 1-D static (or total) temperature and $T_{\max}$ is the maximum static (or total) temperature extracted from the streamwise station. Small values of the pattern factor indicate less flow distortion.

When using the **perf_ext** utility to extract any performance metrics of interest, consider the following items:

- If the desire is simply to plot the streamwise distribution of the performance metrics, then the best option is usually the mass flux weighted average. This approach tends to preserve the physical features of the flow without sudden dispersions that often appear when a flux-based approach is selected.

- If the desire of the performance extraction is to couple multidimensional CFD data with one-dimensional engineering analysis tools, then choose the flux-based option that most closely matches the fluxes that are conserved in the 1-D engineering analysis tool. This will ensure that the same fluxes (to the extent possible) are conserved between the tools, minimizing any discrepancies that result due to the dimensionalization reduction process.

- The Tecplot Slice Extraction option is the most general approach available to prepare the simulation data for the one-dimensionalization procedure. However, this option also introduces the largest amount of error in the performance extraction process due to the following steps that are required:

  – The VULCAN-CFD postprocessed data files contain flowfield data that have been interpolated from cell centers to cell nodes. This can be viewed as a filtering (or smoothing) operation on the cell center data supplied by the flow solver.

  – These data must then be processed by the Tecplot Slice Extraction Tool to interpolate the nodal data at each streamwise station to an unstructured quadrilateral grid defined at each streamwise plane. This step introduces yet another filtering (or smoothing) operation.

  – At a given streamwise plane, the discrete data provided at the nodes of each quadrilateral cell are the data used by **perf_ext** to construct the fluxes required by the chosen one-dimensionalization process. These fluxes are constructed by averaging the properties at the nodes of each quadrilateral to obtain values at the quadrilateral face centers. The variable reconstruction used by VULCAN-CFD to construct the fluxes does not (in general) match this simplistic approach, resulting in an error that can be viewed as truncation error.

  The errors introduced during each step in the process outlined above will manifest themselves most noticeably in the integrated flux quantities themselves (e.g., the mass flow rate). The errors are typically much less visible for the performance metrics that depend on ratios of integrated fluxes due to error cancellation effects.

- The use of VULCAN-CFD flux output via the **OUTPUT FLUXES** input specification (see the output control data section of Chapter 8) is the preferred approach if the integrated flux output is desired, since it provides a direct reflection of the flux conservation achieved by the CFD solver. For example, the extracted mass flow rate at each streamwise station will provide a precise measure of the mass conservation achieved by the CFD solver to the level of iterative convergence reached in the simulation. This option is only available for structured grid simulations that contain a block topology that accommodates the construction of a single contiguous cross-sectional plane that spans the entire cross section of the internal flowpath.

  NOTE: The process that recomposes the postprocessed volumetric data files back to the unpartitioned state does not operate on the flux files output by VULCAN-CFD. Instead the partitioned restart files must first be recomposed using the **restart_merge** utility as described in the structured grid partitioning section of Chapter 25. The VULCAN-CFD postprocessor can then be executed using this restart data via an input file designed for use with the unpartitioned grid.

- If the flux information is not directly provided by VULCAN-CFD, then the volumetric data files must contain the following variables:

  - For weighted approaches, the first variable must be density followed by the velocity components (this is the minimal set). Any additional variables to be weighted or used to compute performance metrics can be supplied in any order.

- For flux-based approaches, the first variable must be density followed by the velocity components. The remaining required variables (which can be provided in any order) are total enthalpy, pressure, sequence of mass fractions, and Mach number. If the CMES method is chosen, then the entropy must be supplied as well.

- If the flux information is provided directly by VULCAN-CFD, then the only volumetric plot file variable needed is the Mach number (and the entropy if the CMES approach is selected).

- The flux-based options for one-dimensionalization typically require nonlinear equation root solves when extracting the one-dimensional properties from the integrated fluxes. Several root-finding strategies have been implemented to provide a robust root solve process, but there may be instances when a solution was not obtained (particularly when using the CMME approach). If convergence issues are encountered, the following message will appear at the end of **perf_ext** execution:

```
Debug/Warning information was produced!
Check the file perf_ext.deb for details
```

  If this message appears, be sure to check the contents of this file to determine if the specific warnings/errors listed are acceptable.

- The CMME and CMES methods both potentially have multiple roots that satisfy the flux conservation equations.[84] For the CMES method, the root choice depends on whether the one-dimensionalized flow properties correspond to subsonic or supersonic flow. In most instances, the root that provides a Mach number that is closest to the mass flux weighted value is the value that is retained. The only time that this might not be the case is when the CFD simulation flowfield output files are utilized in lieu of the Tecplot Slice Extraction option. In this case, a check is performed to attempt to detect an expansion shock, and if detected, the subsonic root is chosen. The root for the CMME method depends on how the one-dimensionalized Mach number compares to $(1/\gamma)^{\frac{1}{2}}$, where γ is the specific heat ratio. The possibilities are either two subsonic solutions that satisfy the integrated flux expressions, or a subsonic and a supersonic solution. As with the CMES method, the root chosen is typically the one that produces a Mach number closest to the mass flux weighted value. The one exception is when both a subsonic and a supersonic root is detected, and the Tecplot Slice Extraction option is not used. In this case, the subsonic root is chosen if an expansion shock is detected.

- Given the possibility that nonlinear equation root solves will be required during the performance extraction process, the VULCAN-CFD plot variables should be output as 64-bit data files. See the **64 BIT BINARY** option in the output control data section of Chapter 8.

25.11 Anisotropic Tetrahedral Mesh Adaptation via *refine*

Unstructured mesh adaptation with VULCAN-CFD is enabled by using the NASA open source mesh adaptation tool *refine*[6] (https://github.com/nasa/refine). The use of this capability often requires a high level of subject matter expertise with algorithmic input specifications due to the inevitable existence of highly anisotropic tetrahedral cells. This often necessitates the use of smaller time steps (CFL values) than typically employed when using well-crafted mixed element grids, as well as customized algorithmic choices to achieve adequate iterative convergence. Hence, the use of this capability is not recommended for novice users.

While *refine* is included in the VULCAN-CFD distribution, there are three additional prerequisite libraries that are required to perform unstructured grid mesh adaptation: Engineering Sketch Pad,[3] OpenCascade,[4] and TetGen.[85] These packages must be installed and available prior to installing VULCAN-CFD to utilize the unstructured mesh adaptation workflow.

25.11.1 Engineering Sketch Pad and OpenCascade

The Engineering Sketch Pad (ESP)[86] (https://acdl.mit.edu/ESP) is a geometry creation and manipulation system designed for aerospace applications. It contains the open source Constructive Solid Modeler (OpenCSM)[87] package, the Engineering Geometry Aerospace Design System (EGADS)[88] package, and a lighter version of this package (EGADSlite).[89] While ESP can be compiled from source files, users are *strongly* encouraged to install the prebuilt binary distributions available from MIT at https://acdl.mit.edu/ESP/PreBuilts). At the time of this writing, ESP120 is the latest version, and is the version that has been tested with VULCAN-CFD. ESP contains a "hardened" version of OpenCascade to support the creation and import of geometry files. The *refine* package that ships with VULCAN-CFD must be configured to link against both of these libraries. In order to use the unstructured grid adaptation capabilities of VULCAN-CFD, the paths to your installations of ESP and OpenCascade must be specified in the **vulcan_config** file:

- Download the ESP prebuilt binary tarball for your system, for Linux systems see https://acdl.mit.edu/ESP/PreBuilts/ESP120Lin.tgz.
- Extract this tarball to the desired location on your system.
- Modify your **vulcan_config** file to set the `OPENCASCADE_PATH` and `ESP_PATH` variables to point to the OpenCASCADE-7.4.1 and EngSketchPad folders (see Chapter 2).

The environment variable settings for the ESP and OpenCascade paths should be defined as follows (details dependent on the specific versions installed):

```
setenv OPENCASCADE_PATH <path_to_opencascade>/ESP120/OpenCASCADE-7.4.1
setenv ESP_PATH <path_to_esp>/ESP120/EngSketchPad
```

25.11.2 Tetgen

TetGen is also required for use with *refine* during the initial case setup step, so it must also be installed in order to obtain the initial .meshb file. TetGen is not needed once this file is obtained, so this software does not have to be present on the compute cluster if the initial case setup step is performed elsewhere (for example a workstation). This installation is also not needed if you are using a stand-alone version of *refine* to perform the initial bootstrap step. The TegGen package source code is available from http://tetgen.org. At the time of this writing, the latest version is 1.6.0, and this is the version tested with VULCAN-CFD. Unlike ESP and OpenCascade which are linked to the *refine* executable by the VULCAN-CFD build system, TetGen is executed within a bash shell.

- Download the TetGen source code from: http://tetgen.org.
- Follow TetGen's installation instructions and install TetGen to the desired location.
- Add the TetGen install location to the `PATH` variable in your `bash.rc` (the `PATH` variable must be set in your **bash** environment, which is a departure from the tcsh environment required to use VULCAN-CFD).

25.11.3 Unstructured Adaptation Process

The *refine* unstructured grid adaptation process implemented in VULCAN-CFD enables automation by essentially removing the human-in-the-loop intervention that would otherwise be required for grid generation (initially and during any step in the grid adaptation process). While interactive grid generation is eliminated in this automated process, this process places much stricter requirements on the geometry models, which must be water-tight and free of any unintended deviations. Approximations required to close gaps or correct flaws in the CAD geometry when manually generating grids often have no noticeable impact on the resulting solution. However, any irregularities in the geometry will inevitably be identified by the solution error estimation procedure in the adaptation process, which will amplify the impact of any ad hoc corrections made to the geometry. As a result, great care is required to provide the cleanest possible representation of the geometry in order to use this automated adaptation process.

The orchestration of VULCAN-CFD and *refine* is handled by the **vul-adapt** script found in the Scripts directory, but the use of this script presumes you have performed the initial case setup using the ESP and *refine*+TegGen steps shown in the unstructured grid adaptation workflow in Fig. 23. The ESP step creates the ".egads" file that houses the geometry description of your domain, and the *refine*+TetGen step creates the ".meshb" file that contains the initial mesh and the grid-to-geometry association. After generating the initial ".egads" geometry file, the *refine* bootstrap operation creates the initial .meshb file. This operation imports the geometry to form the initial surface and volume grids and exports a grid file that also contains geometry and grid-to-geometry association. Appendix B describes the steps required to generate these files, but novice users of ESP are also highly encouraged to work through the ESP training materials (https://acdl.mit.edu/ESP/Training) to gain a better understanding of how to create and manipulate geometry using this package.

NOTE: If using the prepackaged *refine* software provided with VULCAN-CFD located in $vulcanpath/Utilities/T_infinity/vulcan-infinity/install/bin), then be sure to replace any call to **ref** with the corresponding **vinf_ref** executable instead. These are the same executables described in the *refine* documentation with "`vinf_`" prepended to clarify the use of *refine* installed with VULCAN-CFD. For example:

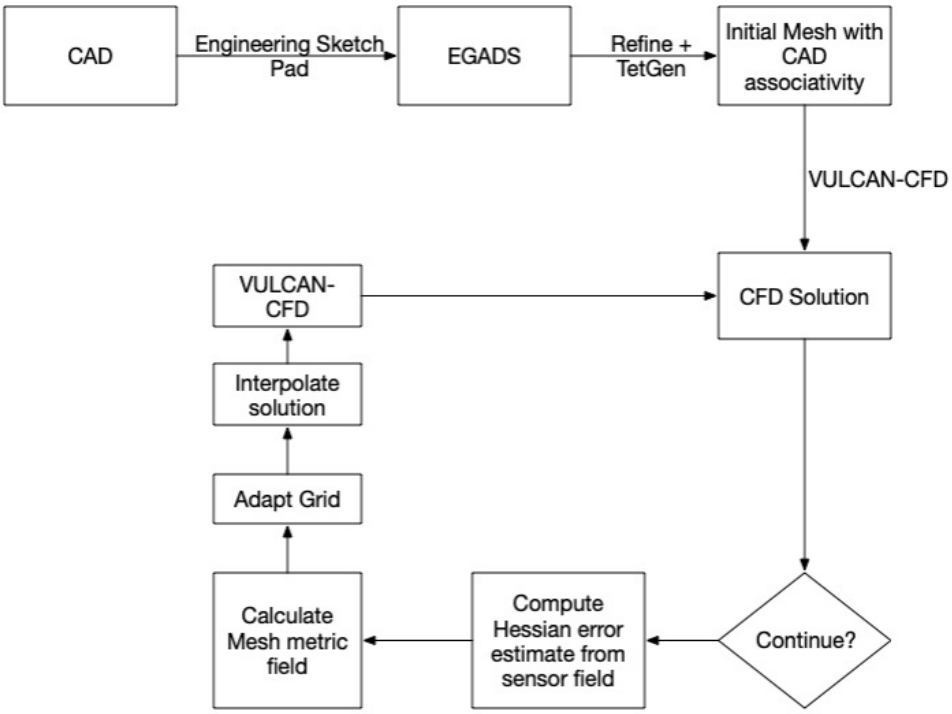


Figure 23: Unstructured grid adaptation workflow.

```
ref –> vinf_ref
mpiexec refmpi –> mpiexec vinf_refmpi
mpiexec refmpifull –> mpiexec vinf_refmpifull
```

Once the initial steps of the adaptation process have been completed, the remaining workflow is automated via the bash script **vul-adapt** and a user-supplied bash script that defines the input control parameters for the adaptation process. The user-supplied script provides access to all relevant parameters that control the adaptation process, and a description of these parameters is provided below.

project_name
This is the project name associated with the ".egads" and ".meshb" files created during the initial steps of the adaptation process. The project name supplied here must correspond to the root name of both of these files, i.e.,

<project_name>.egads
<project_name>.meshb

This parameter must be specified (there is no default).

initial_complexity
This is the initial grid complexity. The grid complexity is proportional to the number of nodes in the adapted grid (approximately equal to half the number of nodes). The value specified should be just large enough to resolve geometrical features, but nothing beyond that. A non-zero value for this parameter must be specified.

input_filename
This is the name of the VULCAN-CFD input file to use when performing the CFD simulation for each adaptation step. The default file name is the supplied project name with a .vulcan_input extension.

restart_in
This is the root name of the restart files used to provide a warm start for CFD simulation in subsequent adaptation steps. The default is Restart_files/restart, but this default value is only valid if it matches the **RESTART IN** file name provided in the VULCAN-CFD input file.

restart_out
This is the root name of the restart files that are output by the VULCAN-CFD simulation for each adaptation step. The default is Restart_files/restart, but this default value is only valid if it matches the **RESTART OUT** file name provided in the VULCAN-CFD input file.

complexity_stride
This is the stride between changes made to the grid complexity. The default is 3, which implies that the grid complexity will remain roughly constant for 3 consecutive adaptation cycles before increasing the complexity. During these cycles, the number of grid nodes is basically fixed, implying that the grid nodes will only be redistributed.

complexity_multiplier
This is the grid complexity multiplier used to increase the grid complexity. The default is 1.5, which specifies that the complexity will be increased by a factor of 1.5 on adaptation cycles that request an increase in complexity.

sensor_field
This is the plot variable name used to form the Hessian matrix that controls the grid node placement when adapting the grid. Any scalar plot variable provided in the **PLOT FUNCTION** list of the VULCAN-CFD input file can be used for this purpose. The string provided must match a variable name written to the **Plot_files/tecplot.nam** file. Popular choices are Mach Number, Temperature [K], or Entropy [J/kg/K].

NOTE: The creation of the vulcan_solution.snap file, which contains the plot field variables

in the ".snap" file format, is only triggered when a T-infinity postprocessing file format is provided in the VULCAN-CFD input file. This implies that one of the following file formats must be specified:

TECPLOT OUTPUT 1.0
VTK OUTPUT 1.0

limit_aspect_ratio
This input parameter specifies the maximum allowed cell aspect ratio. If this input line is not present, then no limit is applied to the cell aspect ratio.

lump_bc_filename
This input line specifies the name of the boundary condition map (.mapbc) file used to lump boundary tags with a common name. Depending on how the geometry is defined within the ESP ".egads" file, there may be multiple mesh boundaries with the same name, implying that the creation of the boundary condition section of the VULCAN-CFD input file will be more tedious than it needs to be. Assigning the .mapbc file to **lump_bc_filename** will minimize the number of VULCAN-CFD boundary condition lines that must be defined. If a .mapbc file is not specified for this entry, then the VULCAN-CFD boundary condition list must match every mesh boundary tag assigned when defining the geometry within ESP.

skip_initial_mesh_gen
This input line specifies whether to skip the initial mesh generation step. This option allows the grid from some pre-existing adaptation case for the same geometry to be used as a starting point. Another usage scenario might be a desire to employ an external tool to impose boundary layer clustering to the grid. The default is false.

twod
This input line specifies that the geometry described in the ESP ".egads" file is a 2-D geometry that requires an extrusion to create a 3-D grid with a single layer of cells in the z-direction.

To assist with the creation of the adaptation input control script, a sample script (**vul-adapt-example-user-input.sh**) is provided in the Scripts directory. To utilize the adaptation workflow, simply copy this template to the adaptation working directory, and modify it as required for your specific problem. The VULCAN-CFD input file created to perform the CFD simulations must also reside in this directory together with the ".egads" geometry description file and the initial ".meshb" file created by the initialization step described previously. With these files in place, the following command (issued from the adaptation working directory) carries out the adaptation workflow:

vul-adapt <your_vul-adapt-input>.sh

Each iteration of the adaptation process will be performed in subfolders named iteration-n, where "n" is the adaptation cycle. The current adaptation cycle is also written to the file **current_grid_level** within the adaptation working directory. The adaptation cycle num-

ber in this file governs where any restart of the adaptation process will start from. Some additional comments that describe how to effectively work within this scripted adaptation workflow are provided below:

- Think of the early adaptation steps as simply a means to provide better initializations for later adaptation steps, i.e., there is usually no reason to converge the early adaptation steps very far.

- Adaptation of turbulent simulations should rely on wall functions when two-equation models are utilized, since there is not yet an adaptation control mechanism to ensure that the first cell center y^+ value is near unity (the Spalart model is considerably more forgiving in this regard).

- It may be advantageous to disable turbulence early in the adaptation process, since obtaining turbulent solutions on the initial grids can be challenging until sufficient adaptation steps are performed to reasonably resolve boundary layers.

- The adaptation workflow can be restarted to either further converge (or initiate) the VULCAN-CFD simulation at the desired adaptation cycle, or simply create the grid for the next adaptation step using the current adaptation step simulation results. However, there is currently no mechanism to restart any substep of the process that creates the grid for the next adaptation step.

- To skip the CFD simulation step and simply use pre-existing simulation files to adapt the grid for the next adaptation cycle, create an empty file named **SKIP_VULCAN** in the current adaptation iteration directory prior to reissuing the **vul-adapt** command. To skip only the VULCAN-CFD solve step (but still postprocess the simulation results to define the adaptation Hessian), create an empty file named **SKIP_SOLVE** prior to reissuing the **vul-adapt** command.

- If the solution diverges on a given adaptation step, then halt the adaptation process to allow the simulation parameters to be adjusted (e.g., lower the CFL number) and then perform the simulation manually. Alternatively, the number of flow solver iterations can be reduced to stop the simulation prior to the point of divergence. The idea here is to simply move on to the next grid assuming that the failure was just an anomaly with the specific grid used. For either strategy, use the **SKIP_VULCAN** option to restart the scripted adaptation workflow to only adapt the grid for the next adaptation cycle.

NOTE: When restarting the adaptation process, be sure that the adaptation cycle number in the **current_grid_level** file matches the adaptation cycle folder number that you wish to restart from. If downstream iteration-n folders are present, then be sure to delete those folders as well.

25.12 Uncertainty Quantification via DAKOTA

Computational fluid dynamics has grown to be an indispensable tool for the design and development of aerospace devices. Unfortunately, the quantification of uncertainties is rarely addressed with anything other than sensitivity studies, so the degree of confidence associated with the numerical results remains exclusively with the subject matter expert that generated them. This practice must at some point be replaced with a formal uncertainty quantification (UQ) process for CFD to play an expanded role in the system design, development, and flight certification process. The probability box (P-Box) approach to uncertainty quantification offers a simple, relatively low cost, nonintrusive methodology that encompasses the basic ingredients required to handle both aleatoric (random) and epistemic (lack of knowledge) uncertainty sources. The nonintrusive nature of the approach allows the computational fluid dynamicist to quantify the uncertainties with the flow solver treated as a "black box", allowing proprietary CFD tools to be readily utilized. Moreover, a large fraction of the process can be automated, allowing the uncertainty assessment to be readily adapted into the engineering design and development workflow. Towards this end, a suite of tools has been developed to automate the P-Box approach to UQ via the DAKOTA package[79,80] for VULCAN-CFD users.

25.12.1 Probability Box Approach to Uncertainty Quantification

The P-Box UQ framework is documented in detail by Roy and Oberkampf,[90] so only the salient features are described here. The first crucial step in this UQ process is the determination and characterization of each source of uncertainty. The determination of potentially relevant uncertainty sources should not be taken lightly, since the impact of any important uncertainty sources that are missed here will not be accounted for, compromising the uncertainty quantification effort. Hence, all members of the design and development team should be involved at some level during this critical initial stage of the process. Once the potential sources of uncertainty have been identified, the next step is the classification of each uncertainty as being either aleatoric or epistemic. Aleatoric uncertainty is any uncertainty source that can be considered as inherently random. These sources of uncertainty are typically not readily reducible, and are treated mathematically using probability theory. Examples of aleatoric uncertainty sources might include variability in facility flow conditions (which manifest themselves as boundary condition uncertainties) and/or geometry variability due to manufacturing tolerances. Epistemic uncertainty sources are typically regarded as some sort of bias error. These sources are represented simply as intervals with no belief that any single value within the interval is more likely to occur than any other value within the interval. Examples of epistemic uncertainties might include turbulence closure assumptions, onset of laminar to turbulent transition, and discretization error. These uncertainties, unlike aleatoric sources, can often be reduced as additional knowledge is gained. For example, turbulence closure uncertainties may be reduced by utilizing higher fidelity models, and discretization errors can be reduced by increasing the resolution of the computational grids employed in the simulations.

Once the uncertainties have been identified and classified, the next step is to perform a sensitivity analysis to estimate their influence on the system response (or output quantity of interest). The purpose of this step is to rank the relative importance of each uncertainty

source toward the prediction of the system response. This information is used to reduce the dimension of the uncertainty space by removing uncertainty sources that are deemed to have a negligible impact on the system response variability. Where this line is drawn is a function of the computational resources available to the CFD practitioner and the time that the "customer" is willing to accept for the additional information provided by an uncertainty analysis. Moreover, some thought should be given to cast the relevant system responses as integral quantities (such as thrust or combustion efficiency) or some other nonlocal flow property. This will provide a smoother output metric that is likely to reduce the number of expensive CFD simulations required by the UQ analysis.

Given the set of relevant uncertainty sources being considered, the next step is to propagate them through the model to determine their impact on the system response. The manner in which this is done is dependent on the characterization of the uncertainty sources. As discussed previously, aleatoric parameters are random in nature and sampled based on an assigned probability distribution. If multiple aleatoric uncertainty sources are considered, then each aleatoric parameter is independently sampled based on its Probability Density Function (PDF). These values are then propagated through the computational model to obtain a single value for the quantity of interest. This process is repeated "N_a" times to obtain a random set of system response values from which a Cumulative Distribution Function (CDF) is formed. The CDF is a nondecreasing function that provides the probability that a random variable (in this case, the system response) will take on a value less than or equal to some chosen value. This function is formed by first sorting the system responses from its smallest to largest value forming the abscissa of Fig. 24. The CDF is constructed by assigning a probability of $1/N_a$ to each response and adding them up as one traverses the abscissa from left to right. The end result is an "S" curve, as shown in the top image of Fig. 24. A steeper curve implies that the variability about the mean value (i.e., the variance of the response) is relatively small and vice versa.

Epistemic parameters have no probabilistic structure associated with them, so any value within the assumed interval is assigned an equal possibility of occurring. Therefore, propagating randomly sampled epistemic values (in the absence of any aleatoric uncertainties) simply results in a bounded interval for the system response, and all system responses between the minimum and the maximum value obtained after propagation of "N_e" samples of the epistemic uncertainties are considered equally probable. The typical scenario in real-world problems is the situation where both aleatoric and epistemic uncertainty sources are present. In this case, one independently samples from each of the epistemic parameters prior to performing the uncertainty analysis for the aleatoric parameters as described in the previous paragraph to produce a CDF that represents a conditional probability. In other words, this CDF provides the probability of the system response "conditioned" on the specific epistemic samples that were chosen. Repeating the process for different sets of epistemic parameters would yield (in general) different CDFs. Therefore, when both aleatoric and epistemic uncertainty sources are present, a nested loop structure is formed with the epistemic parameters sampled in the outer loop, and the aleatoric parameters sampled in the inner loop. This process generates a family of conditional CDFs with the bounding curves representing a probability box, as shown in the bottom image of Fig. 24. The end result of this process is an interval-valued probability representation for the system response.

The computational expense of performing the process outlined above is excessive if the uncertainties are propagated directly through the CFD software using a classical Monte

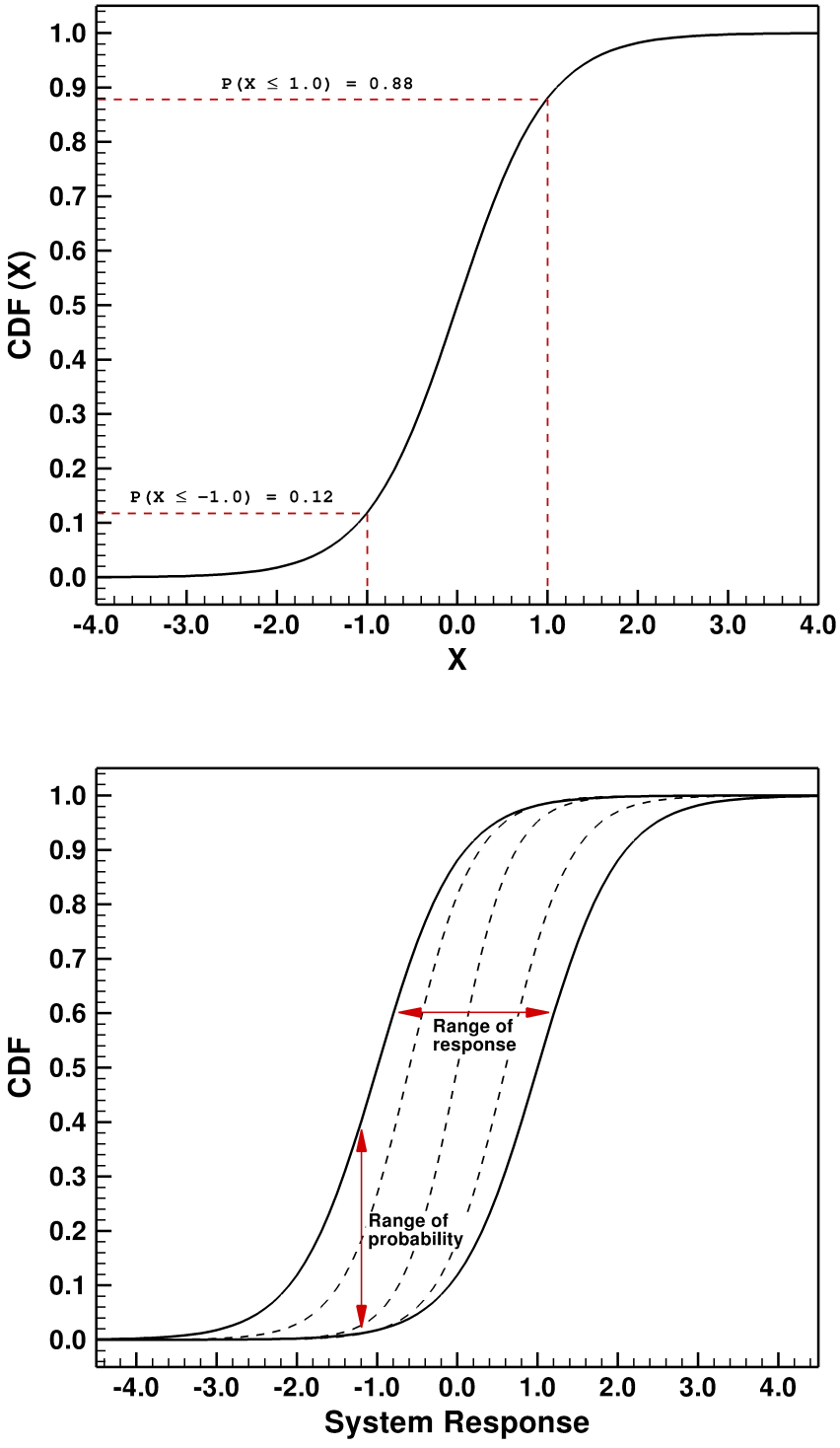


Figure 24: Representative CDF (top) and probability box (bottom) images.

Carlo sampling approach. An effective way to reduce this cost is to construct a metamodel for the system response. Metamodeling is a technique by which a system response is approximated using a limited set of data over a defined design space (in this case, the aleatoric and epistemic parameters). This technique is also known as reduced order modeling, surrogate modeling, or emulation. The use of metamodels is advantageous in situations where many calls to an expensive computational model are required (e.g., uncertainty propagation). Directly accessing the high-fidelity CFD solver in an on-demand manner during the uncertainty propagation process is almost always impractical due to the time required to obtain steady-state solutions from the discretized Navier-Stokes equations. Instead, a finite number of CFD simulations are performed at predefined locations within the design (or in this case uncertainty) space. The metamodel is then formed by fitting the system responses extracted from the CFD solutions as some function of the design (uncertainty) space variables.

In summary, the P-Box uncertainty quantification framework encapsulates the uncertainty information into a "probability box" formed by an ensemble of cumulative distribution functions. Each cumulative distribution function captures the aleatoric uncertainty for a given sample of the epistemic uncertainty space. This framework allows for the inclusion of all sources of uncertainty (random uncertainties, model-form uncertainties, and numerical uncertainties), and when utilizing a metamodel for the uncertainty propagation, can provide an affordable means to quantify uncertainties. The remainder of this section describes the toolset developed to automate the P-Box UQ process using the DAKOTA toolkit and VULCAN-CFD.

25.12.2 Toolkit for Uncertainty Quantification

A suite of tools (**uq_setup**, **uq_final**, **ext_vul_resp**) has been developed to automate the UQ process when using VULCAN-CFD to perform the high-fidelity computational analysis. These tools utilize the DAKOTA toolkit for the various steps required by the P-Box approach to UQ, while hiding most of the UQ process details from the user. In order to use these tools, the DAKOTA toolkit (developed at the Sandia National Laboratory) must be installed on your system. Prebuilt binaries of this package for various systems are publicly available from the DAKOTA website,[91] as are downloads of the source files for custom installations. Note that the prebuilt binaries have been installed using OpenMPI, so OpenMPI must be available to use this installation option. If not, or if some other MPI package was used for the VULCAN-CFD installation, then DAKOTA must be custom installed from the source files. Once installed, be sure to add the path to the DAKOTA binary files to the default path defined in your **.tcshrc** (or **.cshrc**) file. To test the installation, use the following command, which will display the version and release date if the installation was successful:

```
dakota.sh --version
```

NOTE: The tools written to automate the UQ process rely on a loose coupling between DAKOTA and VULCAN-CFD, implying that all of the information passed between the two packages are obtained by parsing the ASCII output files produced by each package. The parsing routines written for this purpose were created using DAKOTA version 6.12 and tested against later versions. Given that the format and content of the DAKOTA ASCII

output files can change with version number, versions older than 6.12 may not be compatible.

The automated UQ workflow enabled using the toolset described here involves the following human-in-the-loop actions:

(1) Set up and simulate the baseline CFD case.

(2) Create the input file (uq_input.dat) that governs the UQ execution process.

(3) Choose a DAKOTA template (pure aleatoric, pure epistemic, or mixed) and define the uncertain variables and desired outputs (responses) of interest.

(4) Execute **uq_setup** to set up the UQ simulation directory structure and VULCAN-CFD input files for each CFD simulation required.

(5) Perform the CFD simulations.

(6) Execute **uq_final** to process the CFD outputs (responses) of interest and generate the P-Box data.

However, the effort required by the CFD practitioner to perform each of these actions is minimal, and nearly all of the details of the UQ framework have been hidden from the user. The remaining text in this section describes each of the above actions in detail, along with guidance to aid with the choices that must be made when setting up the UQ workflow.

Step (1) provides the VULCAN-CFD input file with all of the input settings required to perform the CFD simulations for the UQ process. The preparation of this input file is required by the UQ process, but performing the baseline simulation is optional. Performing the simulation is highly recommended, however, since this provides the opportunity to ensure that algorithmic parameters are set in a robust and efficient manner, while also providing a converged solution state to use as an initial condition for all of the additional simulations required for the UQ process (mitigating the cost of flushing out simulation transients). Hence, the motivation behind this step is to reduce the computational cost incurred by the additional high-fidelity CFD simulations that must be performed during the stochastic UQ workflow. A second baseline CFD case setup may also be required depending on the specific input parameters chosen as uncertain variables. There are currently three scenarios that require an additional baseline setup:

- Treating the numerical discretization error as an epistemic uncertainty
- Treating the turbulence model as an epistemic uncertainty when the number of transported turbulence equations varies
- Treating the chemical kinetics model as an epistemic uncertainty when the number of species and/or the species list varies

The requirement for setting up a second baseline case for handling numerical discretization uncertainty stems from the need to apply the GCI expression to transform the coarse-to-fine grid uncertainty estimates to a true numerical discretization uncertainty estimate. In order

to apply this transformation, the random samples for all the other uncertain variables must be held fixed for a given fine and coarse grid simulation pair. This stipulation necessitates independent control of the fine and coarse grid CFD simulations. The requirement for a second baseline case setup for the remaining two scenarios arises due to the potential for input dependencies. For example, the Spalart turbulence model requires inflow boundary condition specifications for the Spalart viscosity variable, while two-equation k-ω models require specifications to define the turbulence kinetic energy and turbulence frequency variables. Hence, in addition to changing the turbulence model specification itself, other input specifications may require modifications. To seamlessly handle the presence of a second baseline case (when required), the following baseline simulation directory naming conventions must be adhered to:

- The directory for the coarse grid simulation must be named <CASE_NAME>_crs.
- The directory for the one-equation turbulence closure model (Spalart model for RAS or Yoshizawa model for LES) must be named <CASE_NAME>_trb.
- The directory that houses the chemical kinetic model that does not match the species list of the primary baseline case must be named <CASE_NAME>_chm.

Here, <CASE_NAME> refers to the directory name provided in the uq_input.dat file as described below.

NOTE: Appropriate logic is in place to check for the existence of a second baseline case if the selected uncertain variables warrant one, and only one secondary baseline setup is currently allowed for a given UQ assessment (so any combination of the above scenarios is not allowed).

Step (2) involves the creation of a small input file (uq_input.dat) to provide some high-level control over the automated UQ process. The input lines for this file consist of a keyword followed by the specific input string (or value) for the keyword. At least 2 blank spaces must be present between the keyword and the corresponding string/value provided for the keyword. The inputs can be specified in any order, and an optional trailing descriptor can be present. Any number of comment lines (which begin with #) or blank lines can be present as well. An example input file is provided below, and is followed by a brief description of each available input parameter.

```
#
# Input file for coupling VULCAN-CFD with DAKOTA for UQ
#
CASE_NAME        RC22          (VULCAN-CFD simulation setup folder)
VULCAN_INPUT     rc22.inp      (VULCAN-CFD input file file name)
SIMULATION_PATH  CFD_cases     (folder housing the CFD simulations)
DAKOTA_TEMPLATE  dakota_UQ     (DAKOTA template file root name)
DAKOTA_RESTART   No            (DAKOTA restart flag)
POST_CODE        ext_vul_resp  (utility to extract the CFD outputs)
GCI_REFINE       2             (grid refinement factor used for GCI)
```

CASE_NAME
This input specifies the name of the directory that contains the baseline VULCAN-CFD simulation setup created in step (1).

NOTE: Any secondary baseline directories that are required for the UQ process must also use this directory name with the appropriate extension as described above.

VULCAN_INPUT
This input specifies the name of the VULCAN-CFD input file used in the baseline VULCAN-CFD simulation setup created in step (1).

NOTE: The same name must be used for the secondary baseline simulation (if applicable).

SIMULATION_PATH
This input specifies the name chosen for the directory that houses each VULCAN-CFD simulation folder required by the UQ process. This directory will be created when **uq_setup** is executed (if it does not exist). Each simulation folder will be named <CASE_NAME>_#, where "#" is the sample number assigned by DAKOTA.

NOTE: All coarse grid VULCAN-CFD simulations (if numerical discretization uncertainty is desired) will be housed in the directory <SIMULATION_PATH>_crs.

DAKOTA_TEMPLATE
This input specifies the root name of the DAKOTA template file used to define the UQ problem (i.e., portion of the file name that precedes .template). The following skeletal templates are provided in Utilities/Dakota_tools/templates to serve as a starting point:

dakota_UQ_aleat.template	→	aleatoric UQ analysis using CFD results directly
dakota_UQ_aleat_rsur.template	→	aleatoric UQ analysis using a surrogate model
dakota_UQ_epist.template	→	epistemic UQ analysis using CFD results directly
dakota_UQ_epist_rsur.template	→	epistemic UQ analysis using a surrogate model
dakota_UQ_mixed_rsur.template	→	mixed UQ analysis using a surrogate model

These skeletal templates cover all but 1 of the possible scenarios for UQ based on the probability box framework. The missing template is the mixed aleatoric/epistemic UQ analysis without the use of a surrogate model. This template has been omitted due to the extreme cost of using CFD for the nested loop uncertainty propagation in the mixed UQ scenario.

DAKOTA_RESTART
This input specifies whether a restart of the DAKOTA UQ process is to be used. Valid values are Yes or No (or Y/N). This specification is only required when restarting the DAKOTA UQ process to include additional CFD simulations for better resolution of the uncertainty space.

NOTE: When restarting the DAKOTA UQ process, the number of additional samples (i.e., CFD simulations) will double on each restart. For instance, if 10 CFD simulations were

initially performed, then 10 additional simulations will be required on the first restart, 20 additional simulations on the second restart, etc. This doubling of samples is a requirement when using Latin Hypercube Sampling (LHS) in Dakota.

NOTE: A restart of the UQ process can only be performed after a complete UQ cycle has been performed (i.e., after **uq_setup** and **uq_final** have both been executed).

NOTE: The total number of CFD cases is tallied by **uq_final** and output to a file named uq_samples.dat. When a restart is performed, this file and any previous DAKOTA restart files created during execution of **uq_setup** (pre_dakota.rst) and **uq_final** (post_dakota.rst), are copied to files with a ".old" extension. If something goes wrong during the restart, then these files can be used to revert the restart files back to the state before the restart was attempted.

POST_CODE
This input specifies the name of the utility used to extract the desired outputs (responses) from the CFD simulations. The utility **ext_vul_resp** is the only code provided for this purpose, which can extract the following output from various VULCAN-CFD output sources:

- Any integral property written to the VULCAN-CFD output stream by the postprocessor. Integral properties can be extracted at the global (i.e., all blocks), boundary condition object, or boundary condition group level.
- Any property in the integrated force and moment history file (**vulcan.ifam_his_#.tec**, where "#" is the region number), or any property in the boundary condition specific force and moment history files. The specific properties extracted are the values present in the last line of these files.
- Any property output by the performance extraction utility (see Chapter 25). The specific properties extracted from the **perf_ext.tec** file are the values present in the last line of the file. Properties extracted from the optional **perf_ext.txt** tabular file are those from the first axial station present.
- Isolator incident shock train position (requires that Tecplot be available). The Tecplot macro files required for this extraction (**process_loads.mcr** and **isol_shock_xyz.mcr**) are located in the Utilities/Dakota_tools/macros folder. The latter macro file requires editing to specify the control parameters for the shock train location extraction, and both macro files must reside in each simulation folder. Hence, these macro files must be present in the baseline VULCAN-CFD simulation directory (as well as any secondary baseline simulation directories) created during the first step of the automated UQ process.

If other outputs are desired, then modifications to this utility will have to be made to accommodate that.

NOTE: The **isol_shock_xyz.mcr** requires the specification of an ASCII Tecplot format file containing the x, y, and z coordinates used to define the streamwise lines that the surface pressure or shear stress is interpolated to. More than one line can be provided as separate

Tecplot zones to track the shock train along multiple locations (e.g., the shock position along a line that follows the corner of 2 walls can be tracked independently from a line that corresponds to where a symmetry plane intersects a wall). The x, y, and z coordinates supplied in this file must be ordered (i.e., structured) with the streamwise coordinate aligned with the streamwise flow direction. For example, if x is the dominant streamwise coordinate, then the x coordinate values must start at the most upstream isolator x-station and end at the most downstream isolator x-station. The x, y, and z coordinates must also have units consistent with the value specified for the **GRID SCALING FACTOR**.

SKIP_POST_CODE
This input is used to skip the extraction of responses from VULCAN-CFD during execution of **uq_final**. The extraction of the CFD metrics can be time consuming, so if the responses have already been extracted, subsequent executions of **uq_final** can skip this step. This option is particularly useful when experimenting with different surrogate model functional forms.

GCI_REFINE
This input specifies the grid refinement factor used to coarsen/refine the grids, e.g., if the coarse grid was obtained by removing every other grid node from the fine grid then the refinement factor would be two (see the Grid Convergence Index Extraction in Chapter 25 for further details on the Grid Convergence Index).

NOTE: This input line is optional and should only be present if numerical discretization uncertainty is to be quantified. If specified, then a second baseline directory (named <CASE_NAME>_crs) must be present that contains the coarse grid simulation setup.

GCI_SAFE
This input specifies the safety factor used for the GCI transformation (the default value is 3.0). This input has no effect if the **GCI_REFINE** input line is not present.

GCI_ORDER
This input specifies the numerical scheme order of accuracy used for the GCI transformation (the default value is 2.0). This input has no effect if the **GCI_REFINE** input line is not present.

LIST_AVAILABLE_UNCERTAIN_VARIABLES
This input will list all potential uncertain variables that are present in the baseline VULCAN-CFD input file provided by the **VULCAN_INPUT** line. This option is intended to help with the creation of the DAKOTA template file by providing the entire list of available uncertain variables (and the proper syntax) for the DAKOTA input file. If this input line is present, the automated uncertainty process will end after listing these variables, so this line **must** be removed (or commented out) prior to proceeding with the actual uncertainty quantification process.

Step (3) involves choosing a DAKOTA template file from the available templates in Utilities/Dakota_tools/templates for the particular P-Box UQ analysis desired. These tem-

plate files are essentially DAKOTA input files for uncertainty quantification using the P-Box framework, but are missing a few input specification details required to output the probabilistic data. Complete DAKOTA input files for the specific tasks posed to DAKOTA are created on-demand from these templates when **uq_setup** and **uq_final** are executed. The DAKOTA input file created from any template processing performed by **uq_setup** will have the prefix "pre_" appended to it, while any DAKOTA input file created by **uq_final** will have the prefix "post_". These input files are retained only for informational purposes only, and are not intended to be modified by the user. All interfacing with Dakota is handled by modifying the DAKOTA template file, and the only modifications required for a specific UQ problem are:

- Selection of the uncertain VULCAN-CFD input variables and the parameters that characterize their variability
- Selection of the CFD outputs (responses) of interest
- Selecting the number of random samples used for the CFD analyses and the functional form of the surrogate (if surrogate models are used)

Sample lines for these inputs are provided as comments (comment lines are denoted by a leading "#") in each of the templates to easily distinguish the portions of the template that must be modified. Simply remove the "#" and modify the lines as required for a given UQ problem. Note that other lines in the template can be modified as well to alter any of the DAKOTA settings, but this is not necessary. The template for a mixed epistemic/aleatoric uncertainty analysis that uses a surrogate model for the uncertainty propagation is shown below, which is likely to be the most common template chosen in practice. This is followed by a discussion of the various choices available for defining uncertain variables, the available responses that can be extracted using **ext_vul_resp**, as well as guidance towards choosing an appropriate surrogate model functional form and the number of random samples (which translates to CFD simulations) required to construct it.

```
# Dakota input template for mixed uncertainty propagation w/surrogate

environment
  tabular_data
    tabular_data_file = 'dakota_table.dat'
  top_method_pointer = 'UQ_MIXED'

method
  id_method = 'UQ_MIXED'
  sampling
    sample_type lhs backfill
    seed = 17
    samples = 10
    model_pointer = 'EP_MODEL'
  output verbose
```

```
model
  id_model = 'EP_MODEL'
  variables_pointer = 'EP_VAR'
  nested
    sub_method_pointer = 'ALEATORIC'

variables
  id_variables = 'EP_VAR'
# continuous_interval_uncertain = 1
#   lower_bounds = 0.5
#   upper_bounds = 0.7
#   descriptors  = 'TURB._SCHMIDT_NO.'
# discrete_uncertain_set
# string = 1
#   elements_per_variable = 2
#   elements = 'MENTER-BSL'  'MENTER-SST'
#   descriptors = 'TURBULENCE_MODEL'

method
  id_method = 'ALEATORIC'
  sampling
    sample_type lhs
    seed = 17
    samples = 100
    distribution cumulative
    model_pointer = 'SURROGATE'
  output verbose

model
  id_model = 'SURROGATE'
  variables_pointer = 'AL_VAR'
  surrogate global
    dace_method_pointer = 'DACE'
    reuse_points all
#   polynomial quadratic
#   gaussian_process surfpack
#   neural_network
    metrics = 'root_mean_squared' 'mean_abs' 'max_abs' 'rsquared'
    press

variables
  id_variables = 'AL_VAR'
# normal_uncertain  = 1
#   means           = 8.6185e+05
#   std_deviations  = 4.3093e+03
```

```
#   descriptors     = 'TOTAL_PRESSURE'
#  uniform_uncertain = 2
#    lower_bounds    = 0.000  -1.000
#    upper_bounds    = 0.001  -0.990
#    descriptors     = 'OH'    'N2'

method
  id_method = 'DACE'
  sampling
    sample_type lhs
    seed = 17
#   samples = 50
    model_pointer = 'DACE_MODEL'
  output verbose

model
  id_model = 'DACE_MODEL'
  single

interface
  analysis_drivers = 'dakota_script'
    fork
      file_tag

responses
# response_functions = 2
# descriptors        = 'COMBUSTION_EFFICIENCY'  'Qdot_[W]'
  no_gradients
  no_hessians
```

Almost any model parameter present in the VULCAN-CFD input file can be treated as an uncertain variable. Common examples are:

- Turbulence model constants
- Turbulence Schmidt and Prandtl numbers
- Specified boundary condition values
- Sutherland or power law transport coefficients (McBride polynomial transport coefficients cannot currently be treated as uncertain)

Uncertainties in individual reaction rate parameters present in the chemistry model database file may also be treated as uncertain variables, but there is currently no mechanism to handle individual thermodynamic or transport property database parameters. Finally, several VULCAN-CFD input strings can be considered as an uncertain categorical variable:

- Turbulence model string

- Chemistry model database file name
- Grid file name

These categorical uncertainties represent a choice of 2 values, and must be classified as epistemic variables. Note that the grid file name categorical uncertainty is only meant to be used to account for the numerical uncertainty when 2 unrelated grid files are employed. If there is a known refinement factor between the grid files, then the **GCI_REFINE** string should be added to the uq_input.dat file as described previously to provide a bounding discretization error estimate.

Epistemic variables (labeled as EP_VAR in this sample template) have no probabilistic structure associated with them, so these variables only require specifications to define the type of variable and the range of permissible values. Epistemic variables may be provided as a sequence of discrete specifications (e.g., a categorical variable) or as a continuous range of values between specified lower and upper bounds. DAKOTA defines real-valued continuous uncertain variables using the keyword "continuous_interval_uncertain", and these variables are specified in the following manner:

```
continuous_interval_uncertain = N
  lower_bounds =  VALUE_1,  VALUE_2, ...,  VALUE_N
  upper_bounds =  VALUE_1,  VALUE_2, ...,  VALUE_N
  descriptors  = 'NAME_1', 'NAME_2', ..., 'NAME_N'
```

Here, "N" is the number of continuous uncertain variables, "VALUE_1" to "VALUE_N" are the lower (or upper) bounds for each variable, and "NAME_1" to "NAME_N" are the VULCAN-CFD input strings assigned as continuous uncertain variables. Note that DAKOTA does not permit blank spaces in the descriptor names, so any blank spaces present in the VULCAN-CFD input string must be replaced with a single underscore as shown in the sample template above. Discrete uncertain variable specifications can be real-valued, integer-valued, or string-valued. DAKOTA defines discrete uncertain variables using the keyword "discrete_uncertain_set", and these variables are specified as follows:

```
discrete_uncertain_set
string = N
  elements_per_variable = M_1, M_2, ..., M_N
  elements = STRING_1,  STRING_2,  ...,  STRING_M_1
                                .
                                .
                                .
             STRING_1,  STRING_2,  ...,  STRING_M_N
  descriptors  = 'NAME_1', 'NAME_2', ..., 'NAME_N'
```

Here, "N" is the number of discrete uncertain string variables, "M_1" to "M_N" specify the number of allowable values for each of the "N" string variables, and each row of "STRING_1" to "STRING_M" are the specific allowable strings for each of the "N" string variables. The "descriptors" line specifies the VULCAN-CFD input strings that correspond

to each of the "N" string variables (with any blank spaces replaced with an underscore). Any real-valued variables (or integer-valued variables) are specified in an analogous fashion, but with "real = N" (or "integer = N") used instead of "string = N". Finally, any combination of real-valued, integer-valued, and string-valued specifications is permitted:

```
discrete_uncertain_set
string = N
  elements_per_variable = M_1, M_2, ..., M_N
  elements = STRING_1,  STRING_2, ...,  STRING_M_1
                                .
                                .
                                .
             STRING_1,  STRING_2, ...,  STRING_M_N
  descriptors  = 'NAME_1', 'NAME_2', ..., 'NAME_N'
real = N
  elements_per_variable = M_1, M_2, ..., M_N
  elements = REAL_1,  REAL_2, ...,  REAL_M_1
                              .
                              .
                              .
             REAL_1,  REAL_2, ...,  REAL_M_N
  descriptors  = 'NAME_1', 'NAME_2', ..., 'NAME_N'
```

NOTE: When defining the list of values for discrete_uncertain_set variables, each numerical value must be supplied in increasing order and all string values must be alphabetical. If not, DAKOTA will display the following error message:

Error: Set values for each discrete_uncertain_set_string variable must increase

NOTE: If the turbulence model is chosen as a categorical discrete_uncertain_set variable then the following rules apply:

- Any combination of RAS models is allowed, but if the Spalart model is one of them then it must have its own secondary baseline directory.
- Any combination of SGS models is allowed, but if the Yoshizawa model is one of them then it must have its own secondary baseline directory.
- LAMINAR can be used as a turbulence model string for SGS model uncertainty (i.e., ILES), but it is not an allowable option for RAS model uncertainty.

Aleatoric variables (labeled as AL_VAR in the sample template above) have a probabilistic structure associated with them, so these variables require specifications to define their probability distribution function as well as the range of allowable values. DAKOTA offers a wide variety of distribution functions for aleatoric uncertain variables:

- uniform: probability distribution characterized by a lower and an upper bound with equal probability assigned to all values

- normal: probability distribution characterized by a mean and standard deviation with probability governed by a Gaussian distribution (bell curve)
- loguniform: probability distribution characterized by a lower and an upper bound (the natural logarithm of these variables has a uniform distribution)
- lognormal: probability distribution characterized by a mean and standard deviation (the natural logarithm of these variables has a Gaussian distribution)
- gamma: flexible probability distribution for non-negative random variables characterized by α and β parameters (the exponential and chi-squared distributions are special cases from this class)
- beta: flexible probability distribution for random variables that vary between 0.0 and 1.0 characterized by α and β parameters

However, the most common distributions used in practice are the uniform and normal distributions. The specification of variables with uniform and normal distributions was illustrated in the sample template above (loguniform and lognormal specifications are similar). An example of the specifications required for variables with gamma and beta distributions is shown below:

```
gamma_uncertain = N
  alphas        =  ALPHA_1,  ALPHA_2, ...,  ALPHA_N
  betas         =  BETA_1,   BETA_2,  ...,  BETA_N
  descriptors   = 'NAME_1', 'NAME_2', ..., 'NAME_N'
beta_uncertain = N
  alphas       =  ALPHA_1,  ALPHA_2, ...,  ALPHA_N
  betas        =  BETA_1,   BETA_2,  ...,  BETA_N
  lower_bounds =  VALUE_1,  VALUE_2, ..., VALUE_N
  upper_bounds =  VALUE_1,  VALUE_2, ..., VALUE_N
  descriptors  = 'NAME_1', 'NAME_2', ..., 'NAME_N'
```

The treatment of mass fractions as random variables requires special attention due to the requirement that all mass fractions must sum to unity. This implies that any adjustment made to a mass fraction value will require adjustments to one or more other mass fractions. Currently, there are 2 approaches in place to handle this dependency. The first option is to simply specify the list of species to be treated as uncertain variables such that non-negative values are always sampled. If specified in this manner, random values will be chosen for each species in the list, and then all mass fractions (including those not chosen as uncertain variables) will be adjusted by dividing each mass fraction value by the sum of all the mass fraction values. The second option requires the inclusion of a species to lump all mass fraction corrections to (typically an inert species). If this option is used, then this species must be included as an uncertain variable with its lower and upper bounds set to negative values (for mass fractions specified as epistemic continuous_interval_uncertain variables or as aleatoric uniform random variables), or the mean value set to a large (in magnitude) negative number and the standard deviation set to a small positive number (for

mass fractions specified as aleatoric normal random variables). The intent here is to ensure that every random value sampled for this mass fraction results in a negative value, since this unrealizable value is used to denote that this is the species that will have its mass fraction reset by the requirement that all mass fractions sum to unity. If specified in this manner, random values will be chosen for each uncertain mass fraction variable, but the species with a negative mass fraction will have its sampled value replaced with the value required to ensure that all mass fractions sum to unity. This latter approach was used in the sample template above.

Uncertain boundary condition values in the VULCAN-CFD input file are specified by setting the DAKOTA "descriptor" name to the boundary condition header string label that accompanies the uncertain variable, and appending it to the boundary condition name with a colon. For example, the descriptor name 'WALL:Wall_Temp' would be used to specify the wall temperature as an uncertain variable for the following boundary condition:

```
WALL            IWALL       PHYSICAL
Wall_Temp   Rlx
  276.0     0.0
```

If a boundary condition value is to be treated as an uncertain variable, then the boundary condition header line must contain the same number of blank-delineated strings as there are boundary condition values themselves, and each label in the header line must be unique. This implies that each label present on the header line cannot contain blank spaces.

The transition onset locations provided in the VULCAN-CFD input file can be treated as uncertain variables by setting the DAKOTA "descriptor" name to the turbulence suppression subdomain name, and appending it to the string "TRANSITION:". For example, the descriptor name "TRANSITION:FBODY" would be used to specify the transition onset location as an uncertain variable for the following turbulence suppression line:

```
FBODY_BOX   1.0         0.0         0.0
X           Y           Z
0.0         -2.0        0.0
20.0        0.0         0.5
```

Note that a unit directional vector is required to treat the turbulence suppression extent as an uncertain variable. The unit vector line above specifies that the flow direction is aligned with the x-direction ($x_{\min}$ to $x_{\max}$), indicating that the $x_{\max}$ value is the uncertain parameter. A negative value would have aligned the flow in the opposite direction ($x_{\max}$ to $x_{\min}$), indicating that the $x_{\min}$ value is the uncertain parameter. Hence, this vector determines whether to modify the "min" or "max" extents of the Cartesian BOX specification, and implies that the unit vector must be aligned with the x, y, or z axes of the Cartesian box. The cylinder/conical frustum turbulence suppression class extents can also be treated as uncertain, e.g.,

```
SURFACE_CONE    1.0     0.0
X       Y       Z
-0.01   0.0     0.0
0.0     0.0     0.0
RADIUS1 RADIUS2
0.001   0.002
```

In this example, the axial extent of the cylinder/conical frustum has been selected as the turbulence suppression direction to vary. The example below would instead vary the radial extent of the turbulence suppression region:

```
SURFACE_CONE        0.0         1.0
X           Y       Z
-0.01       0.0     0.0
0.0         0.0     0.0
RADIUS1     RADIUS2
0.001       0.002
```

Following the convention used for the Cartesian box, the sign of the nonzero unit vector component determines whether the "min" (-1.0) or "max" (+1.0) extents will be modified. Note that the spatial extent of the generic box turbulence suppression class is not currently permitted to be treated as uncertain.

The ignition source provided in the VULCAN-CFD input file is treated as an uncertain variable by setting the DAKOTA "descriptor" name to the ignition source subdomain name, and appending it to the string "IGNITION:". For example, the descriptor name "IGNITION:CAVITY" would be used to specify the ignition power value as an uncertain variable for the following ignition source line:

```
CAVITY_BOX
Power [Watts]
-2.5e06
X           Y           Z
0.0         -2.0        0.0
20.0        0.0         0.5
```

Note that the spatial extents of the ignition source cannot currently be treated as an uncertain parameter.

Finally, the treatment of individual database reaction rate parameters as uncertain variables requires a special naming convention since there are no uniquely known strings that can be matched. There are 3 reaction rate parameters (A, b, and T_a) for a given Arrhenius form reaction rate [$AT^b \exp(-T_a/T)$]. The descriptor that must be supplied for any of these reaction rate parameters for kinetic step "n" is

```
RF_A_n
RF_B_n
RF_T_n
```

If the reaction model database provides independent control of Arrhenius factors for the backward rate, then the descriptors supplied must be as follows:

```
RB_A_n
RB_B_n
RB_T_n
```

and any arbitrary reaction rate orders to be treated as uncertain parameters must be specified with the descriptor:

RG_<species_string>_n

where <species_string> is the name of the species that has the arbitrary reaction order, and "n" is the kinetic step number.

As mentioned previously, the utility **ext_vul_resp** is the code provided for the purpose of extracting outputs of interest from the CFD simulations, and is currently the only **POST_CODE** option that is available. This utility is embedded as a process within **uq_final**, so it is not intended to be executed manually. The **ext_vul_resp** utility extracts various integral properties that are output directly from VULCAN-CFD (i.e., force and moment history files, postprocessor screen output) or from files generated by the performance extraction utility. The list of output properties that this utility can extract is provided below:

- The following screen output properties written by the VULCAN-CFD postprocessor:
  - Boundary condition integral properties (group or object level):

PRESSURE_FORCE_X,Y,Z	→	pressure force vector component
VISCOUS_FORCE_X,Y,Z	→	viscous force vector component
MOMENTUM_FORCE_X,Y,Z	→	momentum force vector component
FORCE_SUM_X,Y,Z	→	total force vector component
MOMENT_SUM_X,Y,Z	→	total moment vector component
SURFACE_QDOT	→	surface heat transfer rate
MASS_FLOW	→	mass flow rate

  - Global integral properties:

PRESSURE_FORCE_X,Y,Z	→	pressure force vector component
VISCOUS_FORCE_X,Y,Z	→	viscous force vector component
FORCE_SUM_X,Y,Z	→	total force vector component
MOMENT_SUM_X,Y,Z	→	total moment vector component
SURFACE_QDOT	→	surface heat transfer rate
MASS_FLOW	→	mass flow rate
TOTAL_PRESSURE	→	total pressure
TOTAL_PRESSURE_LOSS	→	total pressure loss
EQUIVALENCE_RATIO	→	fuel to air equivalence ratio
MIXING_EFFICIENCY	→	mixing efficiency
COMBUSTION_EFFICIENCY	→	combustion efficiency

- Any property written to the VULCAN-CFD force and moment history file
- Any property written to the force and moment history files specific to each boundary condition
- Any property output by the performance extraction utility.
- Isolator incident shock train position (requires the availability of Tecplot)

If output quantities other than those listed above are desired, then the **ext_vul_resp** utility must be modified accordingly.

Integral properties extracted from the VULCAN-CFD postprocessor output stream can be those at the "BC GROUP", "BC OBJECT", or "GLOBAL" level. Properties desired at the "BC GROUP" level must be prepended by the string "GROUP_<BC_NAME>_", where <BC_NAME> is the boundary condition name provided in the VULCAN-CFD input file. For example, the extraction of the heat load along all surfaces assigned to the following boundary condition:

```
WALL              IWALL          PHYSICAL
Wall_Temp   Rlx
  276.0     0.0
```

would require this "descriptors" entry in the responses section of the template:

descriptors = 'GROUP_WALL_SURFACE_QDOT'

Similarly, properties at the "BC OBJECT" level must be prepended by the string "OBJECT_<OBJECT_NAME>_", where <OBJECT_NAME> is the name assigned to the object specification in the VULCAN-CFD input file. Integral property strings without a GROUP or OBJECT string prepended to it are taken to be a global integral property. For instance, the extraction of the total viscous force in the x-direction would utilize the following "descriptors" entry in the responses section of the template:

descriptors = 'VISCOUS_FORCE_X'

NOTE: The **ext_vul_resp** utility assumes that the VULCAN-CFD output file has the same name as the VULCAN-CFD input file provided in the uq_input.dat file, but with any .inp or .dat extensions replaced with .out. If this naming convention is not followed, then the VULCAN-CFD output file name (followed by a colon) must be prepended to each response string, e.g.:

descriptors = '<output_file_name>:VISCOUS_FORCE_X'

or

descriptors = '<output_file_name>:GROUP_WALL_SURFACE_QDOT'

NOTE: The extraction of any integral parameters that are not part of the standard output (e.g., COMBUSTION_EFFICIENCY) requires the appropriate input line to be present in the VULCAN-CFD input file.

Properties extracted from the **vulcan.ifam_his_#.tec** file ("#" is the region number) are specified exactly as they appear in the VARIABLES header line of the file (but with blank spaces replaced by underscores), and prepended by the string "vulcan.ifam_his_#.tec:". Prepending the file name to the response allows a specific region number to be targeted when multiple regions are present. For example, the extraction of the shear force in the

x-direction from a force and moment history file with the following VARIABLES header line:

```
VARIABLES = "Cycle" "CFL" "log10(L2(Res))" "Mdot Error (%)"
            "F<sub>x</sub> [N]" "F<sub>y</sub> [N]"
            "M<sub>x</sub> [N-m]" "M<sub>y</sub> [N-m]"
            "Qdot [W]" "S<sub>x</sub> [N]" "S<sub>y</sub> [N]"
```

would require a "descriptors" entry in the responses section of the template specified as:

descriptors = 'vulcan.ifam_his_1.tec:S_x_[N]'

Responses extracted from the boundary condition specific force and moment history files (located in the BC_files directory) are handled in precisely the same fashion.

NOTE: The prepended file name string "vulcan.ifam_his_#.tec:" can be omitted if the property is being extracted from the force and moment history file for the first region.

Properties extracted from the **perf_ext.tec** file are specified based on the labels used in the VARIABLES header line of the file (but with blank spaces replaced by underscores), and prepended by the string "perf_ext.tec:". For example, the extraction of temperature from a **perf_ext.tec** file with the following VARIABLES header line:

```
VARIABLES = "X [m]" "Area [m<sup>2</sup>]" "Mass Flow Rate [kg/s]"
            "Mach Number"
            "Temperature [K]"
```

would require a "descriptors" entry in the responses section of the template provided as:

descriptors = 'perf_ext.tec:Temperature_[K]'

Similarly, properties extracted from the **perf_ext.txt** file are specified as they appear in the tabulated output (prior to the "=" sign) with blank spaces replaced by underscores and prepended by the string "perf_ext.txt:". For example, the extraction of temperature from a **perf_ext.txt** file with the following table entries:

```
------------------------------------------------------------
Properties at axial station =  9.50952E+00 [m]
------------------------------------------------------------
Area [m^2]                                  =  3.72028E-01
Mass Flux [kg/s]                            =  1.42482E+02
Mach Number                                 =  3.18278E+00
Static Temperature [K]                      =  4.33231E+02
```

would require this "descriptors" entry in the responses section of the template:

descriptors = 'perf_ext.txt:Static_Temperature_[K]'

NOTE: The execution of the performance extraction utility is triggered by the presence of a perf_ext.inp file in each simulation folder. Hence, a valid perf_ext.inp file must be present in the baseline VULCAN-CFD simulation directory (as well as any secondary baseline simulation directories) created during the first step of the automated UQ process.

NOTE: If the performance extraction utility relies on crossflow plane data extracted from Tecplot, then a Tecplot macro file named **slice.mcr** must also be present in each simulation folder to create this flow data file. Hence, the slice.mcr file must be present in the baseline VULCAN-CFD simulation directory (as well as any secondary baseline simulation directories) created during the first step of the automated UQ process.

The remaining items that require attention in the DAKOTA template file are the choice of surrogate model functional form and the size of the sample space (i.e., number of CFD simulations) to adequately define it. Surrogate models (also known as metamodels or response surface approximations) are functionals derived from higher fidelity computational models (e.g., CFD data) to provide a computationally efficient means to quantitatively capture the trends in the underlying data (CFD outputs of interest). Propagation of input uncertainties via surrogate models is often a highly successful strategy due in large part to the fact that the input parameter uncertainty space is typically a small fraction of the entire input parameter space. In other words, the surrogate only has to capture the trends in the responses over a small portion of the overall parameter space, which is contrasted with the role played by surrogates for design optimization where the surrogate has to function adequately over a large fraction of the parameter space. DAKOTA offers a variety of surrogate model forms, but the most common choices are either a 2^{nd}-order polynomial fit or a Gaussian process model. Several metrics are also available to assess the quality of the surrogate fits.

If the uncertainty space avoids discontinuous (or nearly discontinuous) changes in the response (e.g., an operability limit like inlet start/unstart) then the system response trends are often well captured by a 2^{nd}-order polynomial. The minimum sampling requirement for a 2^{nd}-order polynomial fit is $(n + 2)(n + 1)/2$, where "n" is the number of uncertain input variables. However, additional samples should always be utilized to avoid overfitting. The R^2 surrogate quality measure is commonly used to gauge the quality of polynomial surrogates. This metric measures the amount of variability captured by the model relative to the underlying data used to build it, so values close to unity (~0.95 or greater) indicate a good fit. The other metrics output by DAKOTA are true error measures (e.g., L_2 or L_∞ norms) of the difference between the surrogate approximation and the data used to construct the surrogate. Small values for these measures (relative to the magnitude of the response itself) indicate a good fit. The surrogate models can be further scrutinized using cross-validation techniques to predict how well a surrogate might generalize to unseen data. For this purpose, DAKOTA uses prediction error sum of squares, which is commonly referred to as leave-one-out cross validation.

Gaussian process (kriging) response surfaces might offer a more accurate alternative to

a polynomial fit when the uncertainty space displays a nearly discontinuous variability in the system response because they include a hyper-parametric error model to handle slope discontinuities as well as local minima and maxima. The minimum sampling requirement for the default Gaussian process model in DAKOTA is $(n + 2)(n + 1)/2 + 2n$. However, DAKOTA recommends using a sample size that is at least twice this value (particularly if the response variability is not smooth). Note that the R^2 surrogate quality metric is not relevant for kriging since the training points are exactly reproduced. This also implies that all standard error measures will be zero, so the leave-one-out cross validation error measures must be used to assess the accuracy of a Gaussian process surrogate.

Step (4) in the automated UQ process simply requires the execution of **uq_setup**. This utility parses the uq_input.dat file to extract the provided VULCAN-CFD input file name and baseline setup folder(s), and internally executes DAKOTA to sample the uncertainty space provided in the template file created in the previous step. The sampled values provided by DAKOTA are then used to modify the baseline VULCAN-CFD input file, which are placed (along with the supporting files present in the baseline setup) in unique subdirectories within the <SIMULATION_PATH> folder(s).

NOTE: Any errors encountered while executing DAKOTA will be output to a file named dakota.err, which provides any error messages output by DAKOTA. The DAKOTA output is also retained in the file named pre_<dakota_template_root_name>.out.

Step (5) requires a user-supplied process to perform the CFD simulations that were set up in the previous step. This step is difficult to automate without knowledge of the time required to perform the simulations, which depends on the specifics of the simulations themselves and the computing environment (HPC resource) available to the researcher. This observation drove the decision to loosely couple VULCAN-CFD with DAKOTA when automating the UQ process, and offers the highest level of flexibility to the user to optimize this aspect of the workflow. Moreover, the CFD practitioner is likely to be very knowledgeable with how best to get their simulations through the HPC resources available to them. For simulations that allow one or more CFD cases to complete before exceeding the batch queue limits, looping through the cases within the batch script itself is a reasonable option, e.g.,

```
#!/bin/tcsh
#
#PBS -q normal
#PBS -l select=32:ncpus=16:mpiprocs=16
#PBS -l walltime=24:00:00
#
# User-defined VULCAN-CFD related specifications
#
set vulc_opt = -sg
set inp_file = vulcan.inp
set out_file = ${inp_file:r}.out
set err_file = ${inp_file:r}.err
```

```
#
# Extract the total number of CPUs available for parallel processing
#
set num_proc = `wc -l < $PBS_NODEFILE`
#
# Define job_name to be the name of this batch submittal file
#
set job_name = $0
#
# Define the total number of CFD cases that need to be simulated
#
set max_cases = 20
#
# Define the number of CFD cases for each batch submittal
#
set cases_per_submittal = 4
#
# Enter the <SIMULATION_PATH> defined in the uq_input.dat file
#
cd <path_to_main_UQ_directory>/<SIMULATION_PATH>
#
# Find the next case number to be simulated by looping
# through the case numbers until a file name is found
# that matches the current case number
# NOTE: The file "1" (first case) must initially be present within the
#       <SIMULATION_PATH> directory to initialize the looping process!
#
set case = 0
while ( $case <= $max_cases )
  @ case = $case + 1
  if (-e $case) then
    break
  elseif ($case == $max_cases) then
    echo " "
    echo "The case number in $exec_dir was not between 1 and $max_cases"
    echo " "
    exit  # Something went wrong ...
  endif
end
#
# Loop over the number of simulations performed by this batch submittal
# NOTE: <CASE_NAME> is the case name supplied in the uq_input.dat file
#
icnt = 0
#
while ( $icnt < $cases_per_submittal )
```

```
  #
  @ icnt = $icnt + 1
  #
  cd <CASE_NAME>_"$case"
  vulcan $num_proc .hosts $vulc_opt $inp_file $out_file >& $err_file
  #
  /bin/rm -f ../$case
  if ($case < $max_cases) then
    @ case = $case + 1
    cd .. ; touch $case
  else
    exit  # All of the cases have been simulated, so exit the script
  endif
  #
end
#
# All of the cases have not been simulated, so resubmit the batch job
#
cd $HOME ; qsub $job_name
```

The final step in the automated UQ process simply requires the execution of **uq_final**. This utility parses the uq_input.dat file to determine the VULCAN-CFD simulation path, the code used to extract the requested CFD responses (e.g., **ext_vul_resp**), and any parameters related to the GCI if numerical discretization uncertainty is desired. If Tecplot is available (i.e., if **tec360** is located), then this utility will execute Tecplot in batch mode to produce images of the CFD convergence history (L2 norm of the residual error and the mass flow rate error) to allow the level of convergence to be readily assessed (example images are provided in Fig. 25). These image files are located in the <SIMULATION_PATH> folder provided in the uq_input.dat file. This is followed by the internal execution of the chosen CFD response extraction utility to provide the response values for DAKOTA. Finally, DAKOTA is executed (internally) to process the extracted responses and create the CDF output data files for convenient postprocessing of the probability boxes. A sample image that displays the P-Box from a UQ exercise performed for a scramjet combustor is provided in Fig. 26. These data files (named Pbox_#.dat, where "#" is the response number) are located in the Results folder. If the DAKOTA template was set up to utilize surrogate models, then **uq_final** also outputs the R^2 surrogate quality measure and the L_1, L_2, and L_∞ norms of the surrogate error. These measures are output using the leave-one-out cross validation technique as well to provide additional information on the quality of the surrogate models.

NOTE: The surrogate quality checks are meaningless unless more than the minimum number of simulations required to build the surrogate have been performed.

NOTE: Any errors encountered while executing DAKOTA will be output to a file named dakota.err, which provides any error messages output by DAKOTA. The DAKOTA output is also retained in the file named post_<dakota_template_root_name>.out.

NOTE: If the **GCI_REFINE** input line is present in the uq_input.dat file, then the GCI transform is applied to the coarse and fine grid probability boxes and placed in the Results folder. If the response offset between all corresponding coarse and fine grid CDF curves is monotonic, then the discretization error is applied as a one-sided uncertainty. Otherwise, this is an indication that the grids utilized are far from being asymptotically convergent and the discretization error is applied as a two-sided uncertainty. The original fine grid and coarse grid probability box files are retained in the directory Results/Results_crs if there is a desire to examine them.

NOTE: The Tecplot macro files used to create the images of the CFD residual and mass flow rate errors are retained in the <SIMULATION_PATH> folder to provide a convenient means of assessing the convergence of other properties present in the VULCAN-CFD force and moment history files.

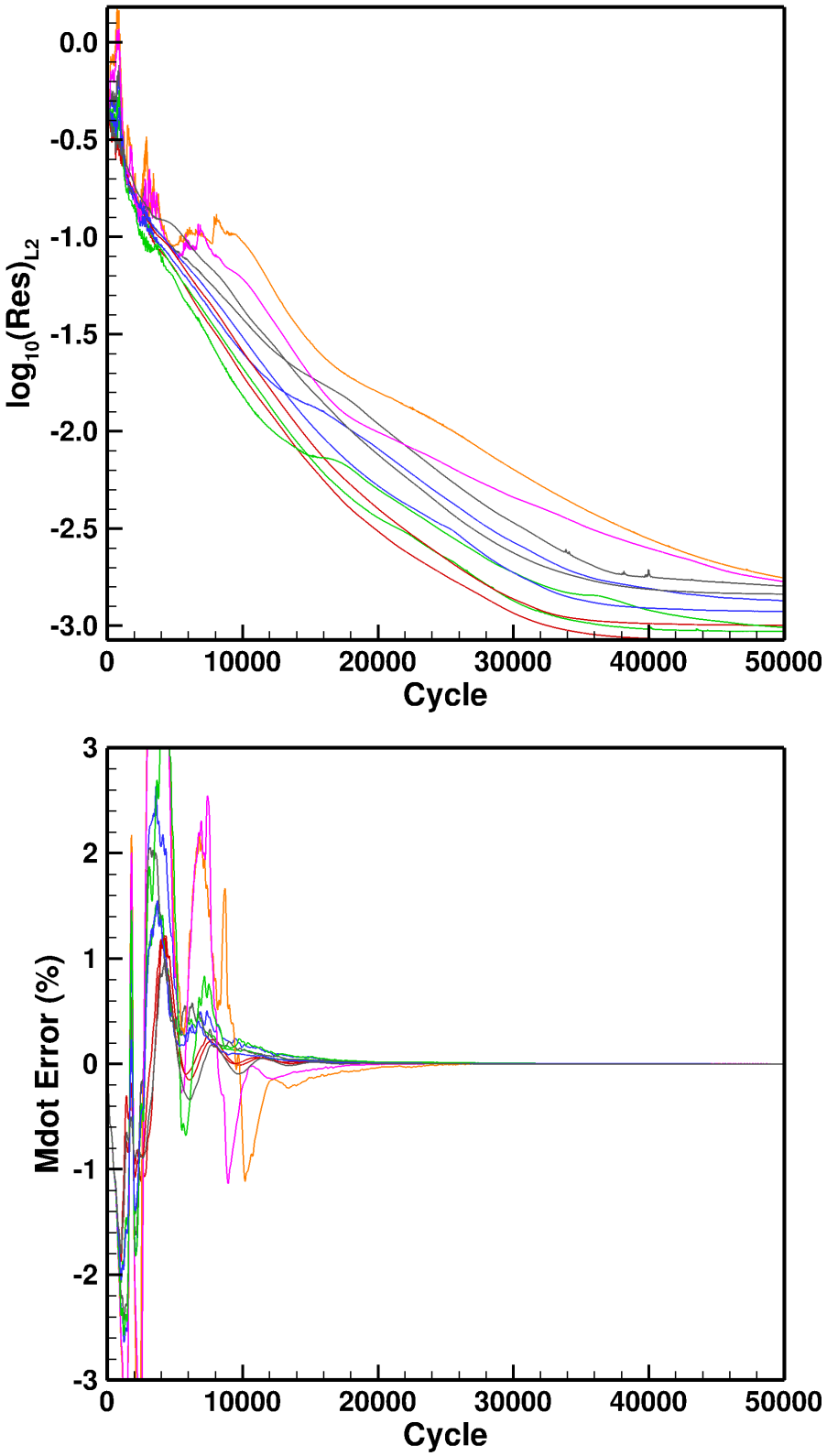


Figure 25: Representative residual error (top) and mass flow (bottom) image output.

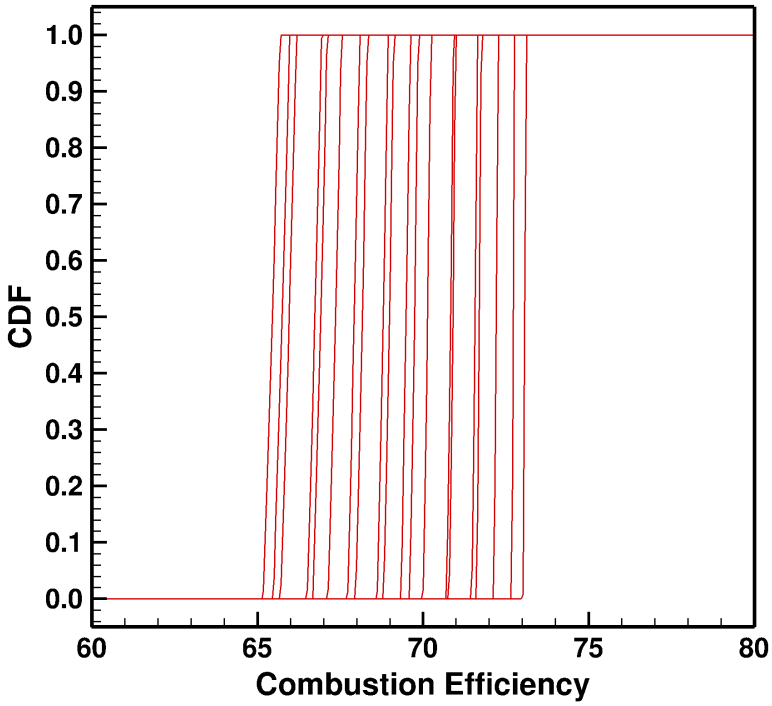


Figure 26: Representative P-Box image output by the automated UQ framework.

25.13 Design Optimization via DAKOTA

In many instances, computational fluid dynamics simulations are utilized as part of the design cycle when developing aerospace devices. This process all too often requires extensive human-in-the-loop interaction to:

(1) Set up and simulate each CFD case in the design cycle based on the design variable values supplied by the optimization driver.

(2) Extract the system responses to evaluate the objective function being used to drive the optimization process.

(3) Transfer the objective function result back to the optimization driver program and repeat.

The first two items above may require the use other software packages in addition to the CFD solver. For instance, if the optimization involves changes to geometry, then a grid generator may be required to generate the grid for iteration of the design cycle. This complication makes full automation of the design process impossible in a general sense. However, it is feasible to automate certain aspects of the optimization process to lessen the workload of the engineer. Towards this end, an automated process has been developed that couples VULCAN-CFD with the DAKOTA package.[79,80] This coupling involves an optimization process workflow that invokes the Efficient Global Optimization (EGO) method,[79] which is a global optimization method that balances exploitation of regions that appear promising in the design space with exploration to examine regions in the design space that have yet to be covered extensively. If the design variables are all VULCAN-CFD inputs, and the responses used to evaluate the objective function are all VULCAN-CFD outputs, then the optimization process is fully automated. Circumstances that require a new grid be developed (or further postprocessing to extract the responses required to form the objective function) can be accommodated provided that user-defined scripts are introduced to handle these extra steps.

25.13.1 Toolkit for Design Optimization & Parameter Calibration

A suite of tools (**opt_driver**, **opt_obj_eval**, and **ext_vul_resp**) has been developed along with several Tecplot macro files to automate the optimization process when using VULCAN-CFD to perform the high-fidelity computational analysis. These tools utilize the DAKOTA toolkit (in particular the EGO method) as the optimizer driver. In order to use these tools, the DAKOTA toolkit (developed at the Sandia National Laboratory) must be installed on your system. Prebuilt binaries of this package for various systems are publicly available from the DAKOTA website,[91] as are downloads of the source files for custom installations. Note that the prebuilt binaries have been installed using OpenMPI, so OpenMPI must be available to use this installation option. If not, or if some other MPI package was used for the VULCAN-CFD installation, then DAKOTA must be custom installed from the source files. Once installed, be sure to add the path to the DAKOTA binary files to the default path defined in your **.tcshrc** (or **.cshrc**) file. To test the installation, use the following command, which will display the version and release date if the installation was successful:

```
dakota.sh --version
```

NOTE: The tools written to automate the optimization process rely on a loose coupling between DAKOTA and VULCAN-CFD, implying that all of the information passed between the two packages are obtained by parsing the ASCII output files produced by each package. The parsing routines written for this purpose were created using DAKOTA version 6.12 and tested against later versions. Given that the format and content of the DAKOTA ASCII output files can change with version number, versions older than 6.12 may not be compatible.

The optimization workflow using the toolset described here involves the following actions:

(1) Set up and simulate the baseline CFD case.

(2) Create the input file (opt_input.dat) that governs the optimization process.

(3) Set up the DAKOTA template by defining the design variables and the outputs (responses) required to evaluate the objective function for the design process.

(4) Execute **opt_driver** to set up the initial simulation directory structure and VULCAN-CFD input files for the initial batch simulations required to initiate the EGO process.

(5) Perform the CFD simulations.

(6) Execute **opt_obj_eval** to process the CFD outputs (responses) of interest and evaluate the optimization objective function.

(7) Repeat steps 4-6 for each EGO iteration (or use the **opt_script** script to perform these actions).

This workflow assumes that the design (or calibration) variables are all VULCAN-CFD input file variables. In this scenario, the only human-in-the-loop tasks are those associated with steps 1-3 that require setting up the files used to set up the optimization process, and steps 4-6 to generate the initial data required to initialize the EGO process. The steps required for each subsequent EGO design iteration is fully automated. The remaining text in this section describes each of the above steps in detail. If one or more design variables are not VULCAN-CFD input file variables (e.g., parametric geometry), then another user-defined process would need to be inserted between steps 4 and 5 to accommodate the variation of these variables, e.g., a scripted geometry and grid creation process.

Step (1) provides the VULCAN-CFD input file with all of the input settings required to perform the CFD simulations for the optimization process. The preparation of this input file is required, but performing the baseline simulation is optional. Performing the simulation is highly recommended, however, since this provides the opportunity to ensure that algorithmic parameters are set in a robust and efficient manner, while also providing a converged solution state to use as an initial condition for all of the additional simulations required for each subsequent optimization step (mitigating the cost of flushing out simulation transients). Hence, the motivation behind this step is to reduce the computational cost incurred by the additional high-fidelity CFD simulations that must be performed during each optimization iteration.

Step (2) involves the creation of a small input file (opt_input.dat) to provide some high-level control over the optimization process. The input lines for this file consist of a keyword followed by the specific input string (or value) for the keyword. At least 2 blank spaces must be present between the keyword and the corresponding string/value provided for the keyword. The inputs can be specified in any order, and an optional trailing descriptor can be present. Any number of comment lines (which begin with #) or blank lines can be present as well. An example input file is provided below, and is followed by a brief description of each available input parameter.

```
#
# Input file for coupling VULCAN-CFD with DAKOTA for Optimization
#
CASE_NAME         RC22             (VULCAN-CFD simulation setup folder)
VULCAN_INPUT      rc22.inp         (VULCAN-CFD input file file name)
SIMULATION_PATH   CFD_cases        (folder housing the CFD simulations)
DAKOTA_TEMPLATE   dakota_opt_EGO   (DAKOTA template file root name)
POST_CODE         ext_vul_resp     (utility to extract the CFD outputs)
CONVERGENCE       0.01             (objective function exit criteria)
RESPONSE_GOALS    resp_goals.dat   (file housing response value goals)
```

CASE_NAME
This input specifies the name of the directory that contains the baseline VULCAN-CFD simulation setup created in step (1).

VULCAN_INPUT
This input specifies the name of the VULCAN-CFD input file used in the baseline VULCAN-CFD simulation setup created in step (1).

SIMULATION_PATH
This input specifies the name chosen for the directory that houses each VULCAN-CFD simulation folder required by the optimization process. This directory will be created when **opt_driver** is executed (if it does not exist). Each simulation folder will be named <CASE_NAME>_#, where "#" is the sample number assigned by DAKOTA.

DAKOTA_TEMPLATE
This input specifies the root name of the DAKOTA template file used to define the optimization/calibration problem (i.e., portion of the file name that precedes .template). A skeletal EGO template (dakota_opt_EGO.template) is provided in Utilities/Dakota_tools/templates to serve as a starting point.

POST_CODE
This input specifies the name of the utility used to extract the desired outputs (responses) from the CFD simulations. The **ext_vul_resp** utility is one code available for this purpose. This utility can extract the following integral outputs from various VULCAN-CFD output sources:

- Any integral property written to the VULCAN-CFD output stream by the postprocessor. Integral properties can be extracted at the global (i.e., all blocks), boundary condition object, or boundary condition group level.

- Any property in the integrated force and moment history file (**vulcan.ifam_his_#.tec**, where "#" is the region number), or any property in the boundary condition specific force and moment history files. The specific properties extracted are the values present in the last line of these files.

- Any property output by the performance extraction utility (see Chapter 25). The specific properties extracted from the **perf_ext.tec** file are the values present in the last line of the file. Properties extracted from the optional **perf_ext.txt** tabular file are those from the first axial station present.

The use of this utility requires the specification of response goal values in the file name provided by the RESPONSE_GOALS entry of the **opt_input.dat** file. The objective function formed when using this utility is the L2 norm of the difference between the extracted response values and the user-supplied response goal values provided in this file. If discrete surface data responses are desired, then **tec360** can be selected as the POST_CODE utility (if Tecplot is available on your system), along with the Tecplot macro file, **perc_err.mcr**, located in the Utilities/Dakota_tools/macros folder. This macro file defines the objective function as the root-mean-square percent difference between surface data extracted from the VULCAN-CFD loads file and "truth" surface data from a user-supplied file (whose file name is provided by the RESPONSE_GOALS entry). This Tecplot utility expects one of the following surface response names to be provided in the DAKOTA template file:

vulcan.loads.plt:PRESSURE
vulcan.loads.plt:TEMPERATURE
vulcan.loads.plt:SHEAR_STRESS
vulcan.loads.plt:HEAT_FLUX

where the prepended string vulcan.loads.plt should be replaced with vulcan.loads.tec for structured grid simulations. This user-supplied data file must be a Tecplot format file with grid coordinates (in units consistent with the value specified for the **GRID SCALING FACTOR**) and "truth" values for one of the following surface properties: temperature [K], pressure [Pa], shear stress [Pa], or heat flux [W/m^2]. The surface property variable names must be specified as T_w, P_w, <greek>t</greek>_w, or q_w to match the names extracted from the VULCAN-CFD loads file. An example Tecplot VARIABLES header line is provided below when optimizing/calibrating to truth data for surface pressure:

```
VARIABLES = "x" "y" "z" "P<sub>w</sub>"
```

NOTE: The **perc_err.mcr** Tecplot macro file expects the VULCAN-CFD loads data to have been processed by the **process_loads.mcr** macro file. Both of these Tecplot macro files are located in the Utilities/Dakota_tools/macros folder, and to use them they must be copied to the main working directory used for the optimization process. Modifications are

not required to either of these macro files unless the "truth" data has entries on both sides of any symmetry plane used in the CFD simulations. If this scenario exists, then edit the **process_loads.mcr** macro file and change one or more of the following lines to a nonzero digit to mirror the VULCAN-CFD loads data across the appropriate symmetry plane.

```
#
# Set options to mirror the CFD data (a nonzero value adds mirrored zones)
#
$!VarSet |MIRROR_X| = 0
$!VarSet |MIRROR_Y| = 0
$!VarSet |MIRROR_Z| = 0
```

NOTE: The **ext_vul_resp** utility is a stand-alone executable and the path to this utility is determined on-the-fly. Hence, no actions are required to copy this utility to the main optimization directory.

If an objective function is desired based on responses other than surface loads properties (or responses extracted using the **ext_vul_resp** utility), then a user-supplied utility will have to be written for this purpose. Any customized response extraction utility must be a macro, script, or executable that outputs the objective function value to an ASCII file named Results/obj_fun.dat. Appropriate modifications will also have to be made to the **opt_obj_eval.F** routine to utilize any user-supplied utility.

RESPONSE_GOALS
This input specifies the name of the file that contains the response goals (or truth data) used to evaluate the objective function. If the **ext_vul_resp** utility was specified as the POST_CODE entry, then this file must include the response goal values for each response (1 per line), i.e.,

```
Response value 1
Response value 2
       .
       .
       .
Response value N
```

If the **tec360** was specified as the POST_CODE entry for use with the **perc_err.mcr** Tecplot macro file, then the response goals file must provide the desired surface property "truth" data as an ASCII Tecplot format file as described above.

CONVERGENCE
The EGO algorithm attempts to find the global optimum by balancing exploitation of regions that appear promising in the design space with exploration to examine regions in the design space that have yet to be covered extensively. The number of high-fidelity CFD simulations that must be performed to satisfy both of these criteria is often excessive, so this input specification provides a means to halt the optimization process once the responses

of interest are sufficiently close to their desired values (i.e., the objective function is sufficiently small). The value entered here is some value of the objective function that satisfies the design requirements. For instance, if the response of interest is combustion efficiency, and a combustion efficiency of at least 95% is required, then a value of 0.05 (or less) could be entered here to halt the design process once this requirement has been met.

LIST_AVAILABLE_DESIGN_VARIABLES
This input will list all potential design variables that are present in the baseline VULCAN-CFD input file provided by the **VULCAN_INPUT** line. This option is intended to help with the creation of the DAKOTA template file by providing the entire list of available design variables (and the proper syntax) for the DAKOTA input file. If this input line is present, the optimization process will end after listing these variables, so this line **must** be removed (or commented out) prior to proceeding with the actual optimization process.

Step (3) involves setting up the DAKOTA template file for the optimization/calibration problem. A skeletal EGO method template (dakota_opt_EGO.template) is provided in Utilities/Dakota_tools/templates to serve as a starting point. This template file is a DAKOTA input file for optimization using the EGO algorithm, and only requires modifications to specify the specific design variables (and their range of variability) for the specific optimization problem and the responses used to construct the objective function that drives the optimization process. Sample lines for these inputs are provided as comments (comment lines are denoted by a leading "#") in the template to easily distinguish the portions of the template that must be modified. Simply remove the "#" and modify the design and response variable lines as required for the given optimization problem. Note that other lines in the template can be modified as well to alter any of the DAKOTA settings, but this is not necessary. Note that **opt_driver** and **opt_obj_eval** use this template file to create the actual input files for DAKOTA execution. However, these input files are retained only for informational purposes only, and are not intended to be modified by the user. All interfacing with Dakota is handled by modifying the DAKOTA template file. The template for the EGO method setup in DAKOTA is shown below. This is followed by a discussion of the various choices available for defining design variables and the available responses that can be extracted using **ext_vul_resp**.

```
# Dakota input template for parameter optimization using EGO

method
  id_method = 'OPT_EGO'
  efficient_global
    seed = 17
    max_iterations = 1
# output verbose

variables
# continuous_design = 1
#   lower_bounds  0.3
#   upper_bounds  1.0
```

```
#   descriptors  'TURB._SCHMIDT_NO.'
# continuous_design = 2
#   lower_bounds  0.3  0.72
#   upper_bounds  1.0  0.90
#   descriptors  'TURB._SCHMIDT_NO.'  'TURB._PRANDTL_NO.'

interface
  analysis_drivers = 'dakota_script'
    fork
      file_tag

responses
# objective_functions = 1
# descriptors         = 'vulcan.loads.plt:PRESSURE'
  no_gradients
  no_hessians
```

Almost any continuous real-valued model parameter present in the VULCAN-CFD input file can be treated as a design variable. Common examples are:

- Turbulence model constants
- Turbulence Schmidt and Prandtl numbers
- Specified boundary condition values

Individual reaction rate parameters present in the chemistry model database file may also be treated as design variables, but there is currently no mechanism to handle individual thermodynamic or transport property database parameters. DAKOTA defines real-valued continuous design variables using the keyword "continuous_design", and these variables are specified in the following manner:

```
continuous_design = N
  lower_bounds =  VALUE_1,  VALUE_2, ...,  VALUE_N
  upper_bounds =  VALUE_1,  VALUE_2, ...,  VALUE_N
  descriptors  = 'NAME_1', 'NAME_2', ..., 'NAME_N'
```

Here, "N" is the number of continuous design variables, "VALUE_1" to "VALUE_N" are the lower (or upper) bounds for each variable, and "NAME_1" to "NAME_N" are the VULCAN-CFD input strings assigned as continuous design variables. Note that DAKOTA does not permit blank spaces in the descriptor names, so any blank spaces present in the VULCAN-CFD input string must be replaced with a single underscore as shown in the sample template above.

The treatment of mass fractions as design variables requires special attention due to the requirement that all mass fractions must sum to unity. This implies that any adjustment made to a mass fraction value will require adjustments to one or more other mass fractions. Currently, there are 2 approaches in place to handle this dependency. The first option is to

simply specify the list of species to be treated as design variables such that non-negative values are always sampled. If specified in this manner, random values will be chosen for each species in the list, and then all mass fractions (including those not chosen as design variables) will be adjusted by dividing each mass fraction value by the sum of all the mass fraction values. The second option requires the inclusion of a species to lump all mass fraction corrections to (typically an inert species). If this option is used, then this species must be included as a design variable with its lower and upper bounds set to negative values. The intent here is to ensure that every random value sampled for this mass fraction results in a negative value, since this unrealizable value is used denote that this is the species that will have its mass fraction reset by the requirement that all mass fractions sum to unity. If specified in this manner, random values will be chosen for each mass fraction variable, but the species with a negative mass fraction will have its sampled value replaced with the value required to ensure that all mass fractions sum to unity.

Boundary condition values in the VULCAN-CFD input file are specified as design variables by setting the DAKOTA "descriptor" name to the boundary condition header string label that accompanies the design variable, and appending it to the boundary condition name with a colon. For example, the descriptor name 'WALL:Wall_Temp' would be used to specify the wall temperature as a design variable for the following boundary condition:

```
WALL            IWALL       PHYSICAL
Wall_Temp   Rlx
  276.0     0.0
```

If a boundary condition value is to be treated as a design variable, then the boundary condition header line must contain the same number of blank-delineated strings as there are boundary condition values themselves, and each label in the header line must be unique. This implies that each label present on the header line cannot contain blank spaces.

The transition onset locations provided in the VULCAN-CFD input file can be treated as design variables by setting the DAKOTA "descriptor" name to the turbulence suppression subdomain name, and appending it to the string "TRANSITION:". For example, the descriptor name "TRANSITION:FBODY" would be used to specify the transition onset location as an design variable for the following turbulence suppression line:

```
FBODY_BOX   1.0         0.0         0.0
X           Y           Z
0.0         -2.0        0.0
20.0        0.0         0.5
```

Note that a unit directional vector is required to treat the turbulence suppression extent as a design variable. The unit vector line above specifies that the flow direction is aligned with the x-direction ($x_{\min}$ to $x_{\max}$), indicating that the $x_{\max}$ value is the design parameter. A negative value would have aligned the flow in the opposite direction ($x_{\max}$ to $x_{\min}$), indicating that the $x_{\min}$ value is the design parameter. Hence, this vector determines whether to modify the "min" or "max" extents of the Cartesian BOX specification, and implies that the unit vector must be aligned with the x, y, or z axes of the Cartesian box. The cylinder/conical frustum turbulence suppression class extents can also be specified as design variables , e.g.,

```
SURFACE_CONE        1.0         0.0
```

```
X           Y           Z
-0.01       0.0         0.0
0.0         0.0         0.0
RADIUS1     RADIUS2
0.001       0.002
```

In this example, the axial extent of the cylinder/conical frustum has been selected as the turbulence suppression direction to vary. The example below would instead vary the radial extent of the turbulence suppression region:

```
SURFACE_CONE        0.0         1.0
X           Y           Z
-0.01       0.0         0.0
0.0         0.0         0.0
RADIUS1     RADIUS2
0.001       0.002
```

Following the convention used for the Cartesian box, the sign of the nonzero unit vector component determines whether the "min" (-1.0) or "max" (+1.0) extents will be modified. Note that the spatial extent of the generic box turbulence suppression class is not currently permitted to be treated as design variables.

The ignition source provided in the VULCAN-CFD input file is treated as a design variable by setting the DAKOTA "descriptor" name to the ignition source subdomain name, and appending it to the string "IGNITION:". For example, the descriptor name "IGNITION:CAVITY" would be used to specify the ignition power value as a design variable for the following ignition source line:

```
CAVITY_BOX
Power [Watts]
-2.5e06
X               Y               Z
0.0             -2.0            0.0
20.0            0.0             0.5
```

Note that the spatial extents of the ignition source cannot currently be treated as a design parameter.

Finally, the optimization (or calibration) of individual database reaction rate parameters requires a special naming convention since there are no uniquely known strings that can be matched. There are 3 reaction rate parameters (A, b, and T_a) for a given Arrhenius form reaction rate $[AT^b \exp(-T_a/T)]$. The descriptor that must be supplied for any of these reaction rate parameters for kinetic step "n" is

```
RF_A_n
RF_B_n
RF_T_n
```

If the reaction model database provides independent control of Arrhenius factors for the backward rate, then the descriptors supplied must be as follows:

RB_A_n
RB_B_n
RB_T_n

and any arbitrary reaction rate orders to be treated as calibration parameters must be specified with the descriptor:

RG_<species_string>_n

where <species_string> is the name of the species that has the arbitrary reaction order, and "n" is the kinetic step number.

As mentioned previously, the utility **ext_vul_resp** and **tec360** are the **POST_CODE** utilities currently available for extracting outputs from CFD simulations to evaluate the objective function of the design problem. The **perc_err.mcr** Tecplot macro file (used when **tec360** is chosen) evaluates the average root-mean-square percent difference between surface data extracted from the **vulcan.loads.tec** (or **vulcan.loads.plt**) file and "truth" surface data from a user-supplied file. The **ext_vul_resp** utility is a general purpose tool that extracts various integral properties that are output directly from VULCAN-CFD (i.e., force and moment history files, postprocessor screen output) or from files generated by the performance extraction utility. The list of output properties that this utility can extract is provided below:

- The following screen output properties written by the VULCAN-CFD postprocessor:
  - Boundary condition integral properties (group or object level):

PRESSURE_FORCE_X,Y,Z	→	pressure force vector component
VISCOUS_FORCE_X,Y,Z	→	viscous force vector component
MOMENTUM_FORCE_X,Y,Z	→	momentum force vector component
FORCE_SUM_X,Y,Z	→	total force vector component
MOMENT_SUM_X,Y,Z	→	total moment vector component
SURFACE_QDOT	→	surface heat transfer rate
MASS_FLOW	→	mass flow rate

  - Global integral properties:

PRESSURE_FORCE_X,Y,Z	→	pressure force vector component
VISCOUS_FORCE_X,Y,Z	→	viscous force vector component
FORCE_SUM_X,Y,Z	→	total force vector component
MOMENT_SUM_X,Y,Z	→	total moment vector component
SURFACE_QDOT	→	surface heat transfer rate
MASS_FLOW	→	mass flow rate
TOTAL_PRESSURE	→	total pressure
TOTAL_PRESSURE_LOSS	→	total pressure loss
EQUIVALENCE_RATIO	→	fuel to air equivalence ratio
MIXING_EFFICIENCY	→	mixing efficiency
COMBUSTION_EFFICIENCY	→	combustion efficiency

- Any property written to the VULCAN-CFD force and moment history file

- Any property written to the force and moment history files specific to each boundary condition
- Any property output by the performance extraction utility.

An L2-norm of any property selected relative to the goal value provided in the user-supplied **RESPONSE_GOALS** file is the objective function that drives the design process when the **ext_vul_resp** tool is selected for the **POST_CODE** entry.

Integral properties extracted from the VULCAN-CFD postprocessor output stream can be those at the "BC GROUP", "BC OBJECT", or "GLOBAL" level. Properties desired at the "BC GROUP" level must be prepended by the string "GROUP_<BC_NAME>_", where <BC_NAME> is the boundary condition name provided in the VULCAN-CFD input file. For example, the extraction of the heat load along all surfaces assigned to the following boundary condition:

```
WALL              IWALL          PHYSICAL
Wall_Temp   Rlx
  276.0     0.0
```

would require this "descriptors" entry in the responses section of the template:

descriptors = 'GROUP_WALL_SURFACE_QDOT'

Similarly, properties at the "BC OBJECT" level must be prepended by the string "OBJECT_<OBJECT_NAME>_", where <OBJECT_NAME> is the name assigned to the object specification in the VULCAN-CFD input file. Integral property strings without a GROUP or OBJECT string prepended to it are taken to be a global integral property. For instance, the extraction of the total viscous force in the x-direction would utilize the following "descriptors" entry in the responses section of the template:

descriptors = 'VISCOUS_FORCE_X'

NOTE: The **ext_vul_resp** utility assumes that the VULCAN-CFD output file has the same name as the VULCAN-CFD input file provided in the opt_input.dat file, but with any .inp or .dat extensions replaced with .out. If this naming convention is not followed, then the VULCAN-CFD output file name (followed by a colon) must be prepended to each response string, e.g.:

descriptors = '<output_file_name>:VISCOUS_FORCE_X'

or

descriptors = '<output_file_name>:GROUP_WALL_SURFACE_QDOT'

NOTE: The extraction of any integral parameters that are not part of the standard output (e.g., COMBUSTION_EFFICIENCY) requires the appropriate input line to be present in the VULCAN-CFD input file.

Properties extracted from the **vulcan.ifam_his_#.tec** file ("#" is the region number) are specified exactly as they appear in the VARIABLES header line of the file (but with blank spaces replaced by underscores), and prepended by the string "vulcan.ifam_his_#.tec:". Prepending the file name to the response allows a specific region number to be targeted when multiple regions are present. For example, the extraction of the shear force in the x-direction from a force and moment history file with the following VARIABLES header line:

```
VARIABLES = "Cycle" "CFL" "log10(L2(Res))" "Mdot Error (%)"
            "F<sub>x</sub> [N]" "F<sub>y</sub> [N]"
            "M<sub>x</sub> [N-m]" "M<sub>y</sub> [N-m]"
            "Qdot [W]" "S<sub>x</sub> [N]" "S<sub>y</sub> [N]"
```

would require a "descriptors" entry in the responses section of the template specified as:

descriptors = 'vulcan.ifam_his_1.tec:S_x_[N]'

Responses extracted from the boundary condition specific force and moment history files (located in the BC_files directory) are handled in precisely the same fashion.

NOTE: The prepended file name string "vulcan.ifam_his_#.tec:" can be omitted if the property is being extracted from the force and moment history file for the first region.

Properties extracted from the **perf_ext.tec** file are specified based on the labels used in the VARIABLES header line of the file (but with blank spaces replaced by underscores), and prepended by the string "perf_ext.tec:". For example, the extraction of temperature from a **perf_ext.tec** file with the following VARIABLES header line:

```
VARIABLES = "X [m]" "Area [m<sup>2</sup>]" "Mass Flow Rate [kg/s]"
            "Mach Number"
            "Temperature [K]"
```

would require a "descriptors" entry in the responses section of the template provided as:

descriptors = 'perf_ext.tec:Temperature_[K]'

Similarly, properties extracted from the **perf_ext.txt** file are specified as they appear in the tabulated output (prior to the "=" sign) with blank spaces replaced by underscores and prepended by the string "perf_ext.txt:". For example, the extraction of temperature from a **perf_ext.txt** file with the following table entries:

```
------------------------------------------------------------
Properties at axial station =  9.50952E+00 [m]
------------------------------------------------------------
Area [m^2]                                    =  3.72028E-01
```

```
Mass Flux [kg/s]                                =  1.42482E+02
Mach Number                                     =  3.18278E+00
Static Temperature [K]                          =  4.33231E+02
```

would require this "descriptors" entry in the responses section of the template:

descriptors = 'perf_ext.txt:Static_Temperature_[K]'

NOTE: The execution of the performance extraction utility is triggered by the presence of a perf_ext.inp file in each simulation folder. Hence, a valid perf_ext.inp file must be present in the baseline VULCAN-CFD simulation directory created during the first step of the optimization process.

NOTE: If the performance extraction utility relies on crossflow plane data extracted from Tecplot, then a Tecplot macro file named **slice.mcr** must also be present in each simulation folder to create this flow data file. Hence, the slice.mcr file must be present in the baseline VULCAN-CFD simulation directory (as well as any secondary baseline simulation directories) created during the first step of the optimization process.

Step (4) in the optimization process simply requires the execution of **opt_driver**. This utility parses the opt_input.dat file to extract the provided VULCAN-CFD input file name and baseline setup folder(s), and internally executes DAKOTA to sample the design space provided in the template file created in the previous step. The sampled values provided by DAKOTA are then used to modify the baseline VULCAN-CFD input file, which are placed (along with the supporting files present in the baseline setup) in unique subdirectories within the <SIMULATION_PATH> folder(s).

NOTE: Any errors encountered while executing DAKOTA will be output to a file named dakota.err, which provides any error messages output by DAKOTA. The DAKOTA output is also retained in the file named pre_<dakota_template_root_name>.out.

Step (5) requires a user-supplied process to perform the CFD simulations that were set up in the previous step. This step is difficult to automate without knowledge of the time required to perform the simulations, which depends on the specifics of the simulations themselves and the computing environment (HPC resource) available to the researcher. This observation drove the decision to loosely couple VULCAN-CFD with DAKOTA when automating the optimization process, and offers the highest level of flexibility to the user to optimize this aspect of the workflow. Moreover, the CFD practitioner is likely to be very knowledgeable with how best to get their simulations through the HPC resources available to them. For simulations that allow one or more CFD cases to complete before exceeding the batch queue limits, looping through the cases within the batch script itself is a reasonable option, e.g.,

```
#!/bin/tcsh
#
#PBS -q normal
```

```
#PBS -l select=32:ncpus=16:mpiprocs=16
#PBS -l walltime=24:00:00
#
# User-defined VULCAN-CFD related specifications
#
set vulc_opt = -sg
set inp_file = vulcan.inp
set out_file = ${inp_file:r}.out
set err_file = ${inp_file:r}.err
#
# Extract the total number of CPUs available for parallel processing
#
set num_proc = `wc -l < $PBS_NODEFILE`
#
# Define job_name to be the name of this batch submittal file
#
set job_name = $0
#
# Define the total number of CFD cases that need to be simulated
#
set max_cases = 20
#
# Define the number of CFD cases for each batch submittal
#
set cases_per_submittal = 4
#
# Enter the <SIMULATION_PATH> defined in the opt_input.dat file
#
cd <path_to_main_optimization_directory>/<SIMULATION_PATH>
#
# Find the next case number to be simulated by looping
# through the case numbers until a file name is found
# that matches the current case number
# NOTE: The file "1" (first case) must initially be present within the
#       <SIMULATION_PATH> directory to initialize the looping process!
#
set case = 0
while ( $case <= $max_cases )
  @ case = $case + 1
  if (-e $case) then
    break
  elseif ($case == $max_cases) then
    echo " "
    echo "The case number in $exec_dir was not between 1 and $max_cases"
    echo " "
    exit  # Something went wrong ...
```

```
  endif
end
#
# Loop over the number of simulations performed by this batch submittal
# NOTE: <CASE_NAME> is the case name supplied in the opt_input.dat file
#
icnt = 0
#
while ( $icnt < $cases_per_submittal )
  #
  @ icnt = $icnt + 1
  #
  cd <CASE_NAME>_"$case"
  vulcan $num_proc .hosts $vulc_opt $inp_file $out_file >& $err_file
  #
  /bin/rm -f ../$case
  if ($case < $max_cases) then
    @ case = $case + 1
    cd .. ; touch $case
  else
    exit  # All of the cases have been simulated, so exit the script
  endif
  #
end
#
# All of the cases have not been simulated, so resubmit the batch job
#
cd $HOME ; qsub $job_name
```

Step (6) in the optimization process simply requires the execution of **opt_obj_eval**. This utility parses the opt_input.dat file to determine the VULCAN-CFD simulation path, the code used to extract the requested CFD responses (e.g., **ext_vul_resp**), and the name of the response goals (or "truth" data) file. If Tecplot is available (i.e., if **tec360** is located), then this utility will execute Tecplot in batch mode to produce images of the CFD convergence history (L2 norm of the residual error and the mass flow rate error) to allow the level of convergence to be readily assessed (example images are provided in Fig. 27). These image files are located in the <SIMULATION_PATH> folder provided in the opt_input.dat file. This is followed by the internal execution of the chosen CFD response extraction utility to provide the response values for DAKOTA. Finally, DAKOTA is executed (internally) to process the extracted objective function to populate the DAKOTA table file used for the next iteration of the EGO optimization process. The DAKOTA table file contains the complete history of the EGO design process, and has the following format:

%eval_id	interface	Variable 1	...	Variable "n"	Response 1	...	Response "n"
1	NO_ID	V1	...	V1	R1	...	R1

2	NO_ID	V2	...	V2	R2	...	R2
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
N	NO_ID	VN	...	VN	RN	...	RN

Note that the response values in the table are actually the objective function values constructed from the CFD responses. Hence, this last column can be extracted to visualize the convergence of the optimization process.

NOTE: Any errors encountered while executing DAKOTA will be output to a file named dakota.err, which provides any error messages output by DAKOTA. The DAKOTA output is also retained in the file named post_<dakota_template_root_name>.out.

NOTE: The Tecplot macro files used to create the images of the CFD residual and mass flow rate errors are retained in the <SIMULATION_PATH> folder to provide a convenient means of assessing the convergence of other properties present in the VULCAN-CFD force and moment history files.

The EGO algorithm relies on the construction of a Gaussian process surrogate for the objective function, which gets updated on each iteration of the optimization/calibration process. This surrogate is used to probe the design space to determine the design variable values to utilize in the CFD simulation for the next design iteration. In order to start the iteration process, a minimum number of CFD simulations ($(n + 2)(n + 1)/2$, where n is the number of design variables) is required to build the initial Gaussian process surrogate. This is the number of CFD simulations that must be performed in step (5) described above. After completing steps (1) through (6) to initialize the design process, the subsequent iterations simply repeat steps (4) - (6) with only 1 CFD simulation required for step 5. If this CFD simulation can be completed before exceeding the batch queue limits, then the **opt_script** driver can be used to perform each design iteration. This script expects 2 arguments:

- number of cores to invoke for the VULCAN-CFD simulation
- number of EGO design iterations to perform

and completely automates the remainder of the optimization process. An example batch script that utilizes **opt_script** is shown below:

```
#!/bin/tcsh
#
#PBS -q normal
#PBS -l select=32:ncpus=16:mpiprocs=16
#PBS -l walltime=24:00:00
#
# Extract the total number of CPUs available for parallel processing
#
set num_proc = `wc -l < $PBS_NODEFILE`
```

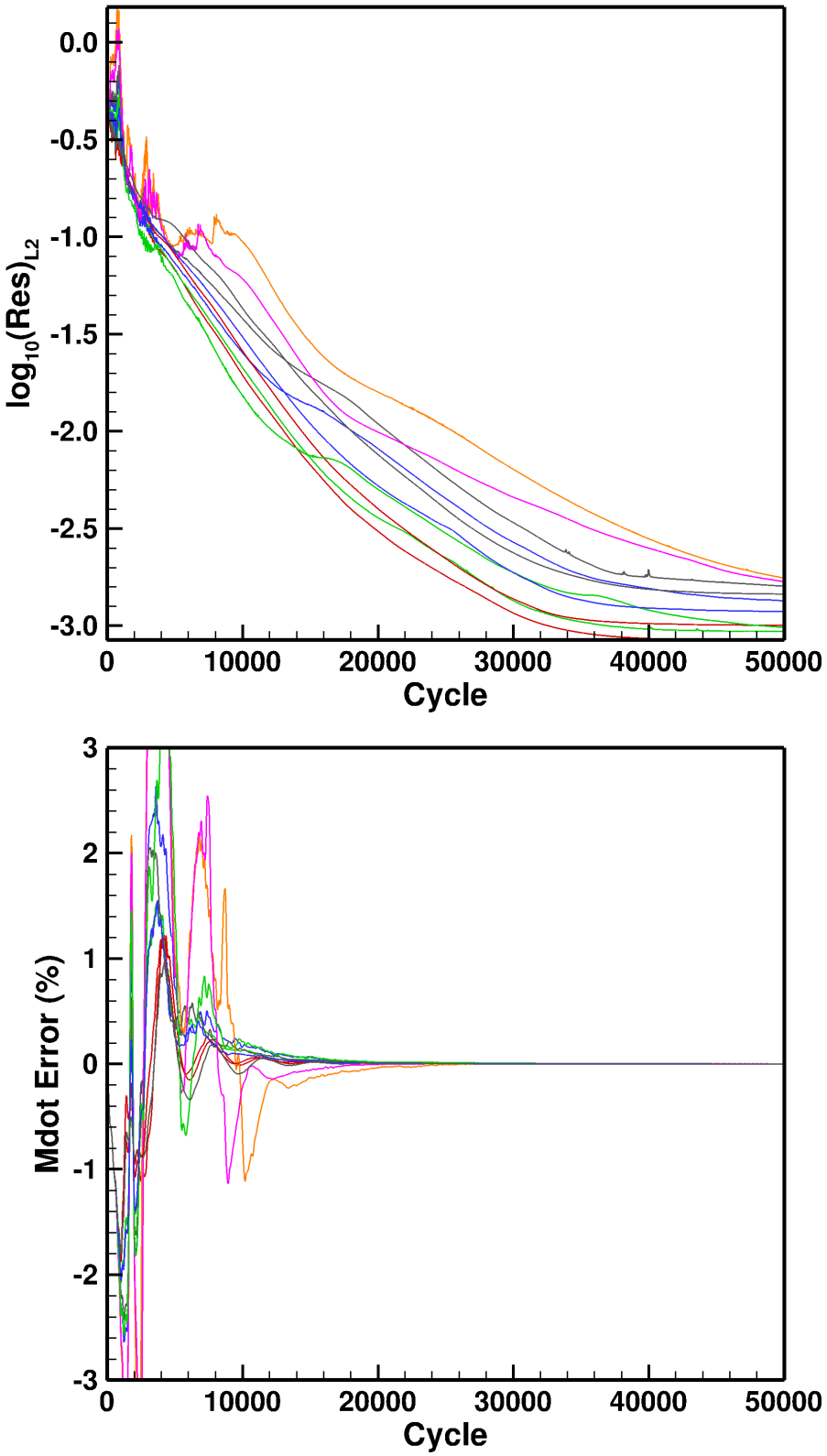


Figure 27: Representative residual error (top) and mass flow (bottom) image output.

```
#
# Define job_name to be the name of this batch submittal file
#
set job_name = $0
#
# Define the number of EGO iterations performed by a batch submittal
#
set design_iterations_per_submittal = 4
#
# Enter the main optimization directory
#
cd <path_to_main_optimization_directory>
#
opt_script $num_proc $design_iterations_per_submittal
#
if ($status != 0) then
  echo " "
  echo "An error flag was returned, batch job not resubmitted"
  echo " "
endif
#
if (-e STOP_OPTIMIZATION) then
  set msg = `/bin/cat STOP_OPTIMIZATION`
  if ("$msg" == "The objective function criteria has been met") then
    echo " "
    echo "The objective function criteria was met, job not resubmitted"
    echo " "
  else if ("$msg" == "Specified number of design cycles completed") then
    echo " "
    echo "The objective function criteria not met, job was resubmitted"
    echo " "
    cd $HOME ; qsub $job_name
  else
    echo " "
    echo "An error flag was returned, batch job not resubmitted"
    echo " "
  endif
else
  echo " "
  echo "Job appeared to end prematurely, batch job not resubmitted"
  echo " "
endif
```

If the wall clock time required to perform the CFD simulation for each design iteration cannot be completed before exceeding the batch queue limits, then steps (4) - (6) need to be performed manually.

As a final note, if any step of the optimization process fails, then care must be taken to sync the process back to the appropriate state. If an error was encountered while performing the **opt_driver** step, then:

(1) Resolve whatever caused the error.

(2) Review the DAKOTA table file and <SIMULATION_PATH>/EGO_history files to determine whether to remove the last line present in each file. The last line in each of these files should correspond to the last successful EGO design cycle.

(3) Remove the last line present in either (or both) files if deemed necessary.

(4) If the case directory was created within <SIMULATION_PATH> for the next EGO iteration, then remove this folder and all of its contents.

(5) Execute **opt_script** to resume the optimization process.

If an error was encountered while performing the VULCAN-CFD simulations, then:

(1) Make adjustments to the VULCAN-CFD input file as required to allow the simulation to complete successfully.

(2) Manually execute **opt_obj_eval**.

(3) Execute **opt_script** to resume the optimization process.

If an error was encountered while performing the **opt_obj_eval** step, then:

(1) Resolve whatever caused the error.

(2) Manually execute **opt_obj_eval**.

(3) Execute **opt_script** to resume the optimization process.

25.14 Externally Supplied Utility Codes

Some utilities have been made available by the VULCAN-CFD user community. These tools are not directly supported by the VULCAN-CFD development team, but have been added to the VULCAN-CFD distribution due to the usefulness of the capabilities they provide:

- The **Utilities/Tecplot_tools** directory houses a suite of tools that can perform a variety of postprocessing steps to enhance the VULCAN-CFD user experience. Most of these tools are only applicable to structured grid simulations, and require that the user have access to the commercially available Tecplot[7] package. The features offered by this toolset are listed below, but refer to the documentation in the **Utilities/Tecplot_tools/doc** directory for a complete description and usage instructions.
  - export a subset of the VULCAN-CFD output stream with reference conditions, total simulation time, and integrated boundary condition information to a smaller file for communication to others
  - extract auxiliary information from the VULCAN-CFD input file for inclusion into Tecplot .plt files as auxiliary data (e.g., VULCAN-CFD version, solver settings), which can be dynamically referred to when creating figures from simulation data
  - automate the execution of **perf_ext** if a perf_ext.inp file is found
  - execute **vpp** (Versatile PostProcessor) to extract boundary surfaces, specific block planes, or slices from the volumetric plot file to a separate smaller file that is easier to plot, store, and share with others
  - execute **vuln** to convert the cell-centered loads file data to a node based file that is consistent with the volumetric plot file
  - execute **integralPdA** to extract pressure loads if a data file with experimental gauge locations is found (this utility integrates surface pressure loads using a resolution that matches that of the experimental data)
  - one-dimensionalize surface data at streamwise stations with **surf1d**
  - one-dimensionalize volumetric data at streamwise stations with **massflow3d** (this utility does not offer the full array of performance metrics and dimension reduction options included with **perf_ext**, but it does offer an automated topology algorithm for merging the integration process across structured grid blocks)
  - execute **blprops** to determine boundary layer properties as well as create a file with boundary layer edge information to aid with visualizing flowfield properties at the boundary layer edge
  - execute the Langley Isolator Model Tecplot macro to assess scramjet isolator performance (NOTE: this capability is ITAR restricted and must be requested separately from the VULCAN-CFD distribution tar file)
  - automate the execution any custom Tecplot macros that are found and the exporting of images to files for any Tecplot layout files that are found

The EGO optimization workflow iteratively performs steps (4) - (6).

References

1. Baurle, R. A., White, J. A, Drozda, T. G., and Norris, A. T.: VULCAN-CFD Theory Manual: Ver. 7.2.0. NASA Technical Report TM-2022-0008776, 2022.

2. METIS. glaros.dtc.umn.edu/gkhome/views/metis, 2022.

3. Engineering Sketch Pad. acdl.mit.edu/ESP, 2022.

4. OpenCascade. www.opencascade.com, 2022.

5. OConnell, M. D., Druyor, C. T., Thompson, K. B., Jacobson, K. E., Anderson, W. K., Nielsen, E. J., Carlson, J.-R., Park, M. A., Jones, W. T., Biedron, R. T., Kleb, B., and Zhang, X.: T-infinity: The Dependency Inversion Principle for Rapid and Sustainable Multidisciplinary Software Development. AIAA Paper 2018-3856, June 2018.

6. Park, M. A., Kleb, B., Jones, W. T., Krakos, J. A., Michal, T., Loseille, A., Haimes, R., and Dannenhoffer, J. F. III: Geometry Modeling for Unstructured Mesh Adaptation. AIAA Paper 2019-2946, June 2019.

7. Tecplot. www.tecplot.com, 2022.

8. POINTWISE. www.pointwise.com, 2022.

9. GridPro. www.gridpro.com, 2022.

10. PLOT3D Structured Grid Formats. www.nas.nasa.gov/Software/FAST/RND-93-010.walatka-clucas/htmldocs/chp_21.formats.html#HDR2022.

11. AFLR3 UGRID 3D Format. www.simcenter.msstate.edu/software/downloads/doc/ug_io/3d_grid_file_type_ugrid.html, 2022.

12. FieldView. www.ilight.com/en/products/product-comparison, 2022.

13. EnSight. www.ensight.com, 2022.

14. ParaView. www.paraview.org, 2022.

15. Heldenmesh. heldenaero.com/heldenmesh/, 2022.

16. Anderson, W. K., Wood, S. L., and Jacobson, K. E.: Node Numbering for Stabilizing Preconditioners Based on Incomplete LU Decomposition. AIAA paper, June 2020.

17. Edwards, J. R.: A Low Diffusion Flux-Splitting Scheme for Navier-Stokes Calculations. *Computers & Fluids*, vol. 26, no. 6, 1997, pp. 635–659.

18. Nishikawa, H.: A Face-Area-Weighted Centroid Formula for Finite-Volume Method that Improves Skewness and Convergence on Triangular Grids. *Journal of Computational Physics*, vol. 401, no. 109001, 2020, pp. 1–26.

19. White, J. A, Nishikawa, H., and Baurle, R. A.: A 3-D Nodal-Averaged Gradient Approach for Unstructured-Grid Cell-Centered Finite-Volume Methods for Application to Turbulent Hypersonic Flow. AIAA Paper 2020-0652, Jan. 2020.

20. White, J. A, Nishikawa, H., and O'Connell, M. D.: F-ANG+: A 3-D Augmented-Stencil Face-Averaged Nodal- Gradient Cell-Centered Finite-Volume Method for Hypersonic Flows. AIAA Paper 2022-1848, Jan. 2022.

21. Osher, S. and Shu, C.-W.: High Order Essentially Nonoscillatory Schemes for Hamilton-Jacobi Equations. *SIAM Journal on Numerical Analysis*, vol. 28, no. 4, 1991, pp. 907–922.

22. Heun, K.: Neue Methoden zur Approximativen Integration der Differentialgleichungen Einer Unabhangigen Veranderlichen. *Z. Math. Phys.*, vol. 45, 1900, pp. 23–38.

23. Carpenter, M. H. and Kennedy, C. A.: Fourth-Order 2N-Storage Runge-Kutta Schemes. NASA Technical Report TM-1994-109112, 1994.

24. Wilcox, D. C.: Wall Matching, a Rational Alternative to Wall Functions. AIAA Paper 89-0611, Jan. 1989.

25. White, J. A, Baurle, R. A., Fisher, T. C., Quinlan, J. R., and Black, W. S.: Low Dissipation Advection Schemes Designed for Large Eddy Simulations of Hypersonic Propulsion Systems. AIAA Paper 2012-4263, June 2012.

26. Pirozzoli, S.: Stabilized Non-Dissipative Approximations of Euler Equations in Generalized Curvilinear Coordinates. *Journal of Computational Physics*, vol. 230, no. 8, 2011, pp. 2997–3014.

27. Honein, A. E. and Moin, P.: Higher Entropy Conservation and Numerical Stability of Compressible Turbulence Simulations. *Journal of Computational Physics*, vol. 201, no. 2, 2004, pp. 531–545.

28. Ducros, F. Ferrand, V., Nicoud, F., Weber, C., Darracq, D., Gacherieu, C., and Poinsot, T.: Large-Eddy Simulation of the Shock/Turbulence Interaction. *Journal of Computational Physics*, vol. 152, no. 2, 1999, pp. 517–549.

29. Larsson, J.: Large Eddy Simulations of the HyShot II Scramjet Combustor Using a Supersonic Flamelet Model. AIAA Paper 2012-4261, Aug. 2012.

30. Sutherland, W.: The Viscosity of Gases and Molecular Force. *Philosophical Magazine*, vol. 36, no. 223, 1893, pp. 507–531.

31. Dorrance, W. H.: *Viscous Hypersonic Flow*. McGraw-Hill, 1962.

32. McBride, B. J. and Gordon, S.: Computer Program for Calculation of Complex Chemical Equilibrium Composition and Applications, I. Analysis. NASA Reference Publication 1311, Oct. 1994.

33. McBride, B. J. and Gordon, S.: Computer Program for Calculation of Complex Chemical Equilibrium Composition and Applications, II. Users Manual and Program Description. NASA Reference Publication 1311, June 1996.

34. Mason, E. A. and Saxena, S. C.: Approximate Formula for the Thermal Conductivity of Gas Mixtures. *Physics of Fluids*, vol. 1, no. 5, 1958, pp. 361–369.

35. Kee, R. J., Miller, J. A., and Jefferson, T. H.: CHEMKIN: A General Purpose, Problem-Independent, Transportable, FORTRAN Chemical Kinetics Code Package. Sandia National Laboratories Report SAND80-8003, 1980.

36. Spalart, P. R. and Allmaras, S. R.: A One-Equation Turbulence Model for Aerodynamic Flows. *Recherche Aerospatiale*, 1994, pp. 5–21.

37. Allmaras, S. R., Johnson, F. T., and Spalart, P. R.: Modifications and Clarifications for the Implementation of the Spalart-Allmaras Turbulence Model. ICCFD7 Paper 1902, July 2012.

38. Menter, F. R.: Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications. *AIAA Journal*, vol. 32, no. 8, 1994, pp. 1598–1605.

39. Menter, F. R., Kuntz, M., and Langtry, R.: Ten Years of Industrial Experience with the SST Turbulence Model. *Proceedings of the Fourth International Symposium on Turbulence, Heat and Mass Transfer*, Begell House, Inc., 2003, pp. 625–632.

40. Wilcox, D. C.: *Turbulence Modeling for CFD*. DCW Industries, Inc., 2nd ed., 1998.

41. Wilcox, D. C.: *Turbulence Modeling for CFD*. DCW Industries, Inc., 3rd ed., 2006.

42. Rumsey, C. L. and Gatski, T. B.: Summary of EASM Turbulence Models in CFL3D with Validation Test Cases. NASA Technical Report TM-2003-212431, 2003.

43. Spalart, P. R.: Strategies for Turbulence Modelling and Simulation. *International Journal of Heat and Mass Transfer*, vol. 21, 2000, pp. 252–263.

44. Mani, M., Babcock, D. A., Winkler, C. M., and Spalart, P. R.: Predictions of a Supersonic Turbulent Flow in a Square Duct. AIAA Paper 2013-0860, Jan. 2013.

45. Thivet, F., Knight, D. D., Zheltovodov, A. A., and Maksimov, A. I.: Insights in Turbulence Modeling for Crossing Shock Wave Boundary Layer Interactions. *AIAA Journal*, vol. 39, no. 6, 2001, pp. 985–995.

46. Durbin, P. A.: On the K-Epsilon Stagnation Point Anomoly. *International Journal of Heat and Fluid Flow*, vol. 17, 1996, pp. 89–90.

47. Turbulence Model Resource. https://turbmodels.larc.nasa.gov, 2022.

48. Magnussen, B. F. and Hjertager, B. H.: On Mathematical Modeling of Turbulent Combustion with Special Emphasis on Soot Formation and Combustion. *Sixteenth Symposium (International) on Combustion*, 1976, pp. 719–729.

49. Smagorinsky, J.: General Circulation Experiments with the Primitive Equations. *Monthly Weather Review*, vol. 91, no. 3, March 1963, pp. 99–164.

50. Vreman, A. W.: An Eddy-Viscosity Subgrid-Scale Model for Turbulent Shear Flow: Algebraic Theory and Applications. *Physics of Fluids*, vol. 16, no. 10, Oct. 2004, pp. 3670–3681.

51. Yoshizawa, A. and Horiuti, K.: A Statistically-Derived Subgrid Scale Kinetic Energy Model for Large-Eddy Simulation of Turbulent Flows. *Journal of the Physical Society of Japan*, vol. 54, 1985, pp. 2834–2839.

52. Germano, M., Piomelli, U., Moin, P., and Cabot, W. H.: A Dynamic Subgrid-Scale Eddy Viscosity Model. *Physics of Fluids*, vol. 3, no. 7, July 1991, pp. 1760–1765.

53. Heinz, S. and Gopalan, H.: Realizable Versus Non-Realizable Dynamic Subgrid-Scale Stress Models. *Physics of Fluids*, vol. 24, no. 11, Nov. 2012, pp. 1760–1765.

54. Shur, M. L., Spalart, P. R., Strelets, M. K., and Travin, A. K.: Detached-Eddy Simulaion of an Airfoil at High Angle of Attack. 4th Int. Symp. Eng. Turb. Modeling and Measurements, May 1999.

55. Spalart, P. R., Jou, W.-H., Strelets, M. K., and Allmaras, S. R.: Comments on the Feasibility of LES for Wings, and on a Hybrid RANS/LES Approach. 1st AFOSR International Conference on DNS/LES (invited), Aug. 1997.

56. Shur, M. L., Spalart, P. R., Strelets, M. K., and Travin, A. K.: A Hybrid RANS-LES Approach with Delayed-DES and Wall-Modelled LES Capabilities. *International Journal of Heat and Fluid Flow*, vol. 29, no. 6, 2008, pp. 1638–1649.

57. Gritskevich, M. S., Garbaruk, A. V., Schütze, J., and Menter, F. R.: Development of DDES and IDDES Formulations for the k-ω Shear Stress Transport Model. *Flow Turbulence Combustion*, vol. 88, no. 3, April 2012, pp. 431–449.

58. Baurle, R. A., Tam, C.-J., Edwards, J. R., and Hassan, H. A.: Hybrid Simulation Approach for Cavity Flows: Blending, Algorithm, and Boundary Treatment Issues. *AIAA Journal*, vol. 41, no. 8, Aug. 2003, pp. 1463–1480.

59. Choi, J.-L., Edwards, J. R., and Baurle, R. A.: Compressible Boundary Layer Predictions at High Reynolds Number Using Hybrid LES/RANS Methods. *AIAA Journal*, vol. 47, no. 9, 2009, pp. 2179–2193.

60. Boles, J. A., Choi, J.-L., Edwards, J. R., and Baurle, R. A.: Multi-Wall Recycling/Rescaling Method for Inflow Turbulence Generation. AIAA Paper 2010-1099, Jan. 2010.

61. van Leer, B.: Towards the Ultimate Conservation Difference Scheme. II. Monotinicity and Conservation Combined in a Second Order Scheme. *Journal of Computational Physics*, vol. 14, 1974, pp. 361–370.

62. Colella, P. and Woodward, P.: The Piecewise Parabolic Method (PPM) for Gasdynamical Simulations. *Journal of Computational Physics*, vol. 54, no. 1, 1984, pp. 174–201.

63. Fisher, T. C., Carpenter, M. H., Yamaleev, N. K., and Frankel, S. H.: Boundary Closures for Fourth-order Energy Stable Weighted Essentially Non-oscillatory Finite Difference Schemes. *Journal of Computational Physics*, vol. 230, no. 1, 2011, pp. 3727–3752.

64. Nishikawa, H.: On the Loss and Recovery of Second-Order Accuracy with U-MUSCL. *Journal of Computational Physics*, vol. 417, Sept. 2020, pp. 1–10.

65. Roe, P.: Approximate Reimann Solvers, Parameter Vectors, and Difference Schemes. *Journal of Computational Physics*, vol. 43, 1981, pp. 357–372.

66. Troe, E. F., Spruce, M., and Speares, W.: Restoration of the Contact Surface in the Harten-Lax-van Leer Reimann Solver. *Shock Waves*, vol. 4, 1994, pp. 25–34.

67. Edwards, J. R. and Liou, M.-S.: Low Diffusion Flux-Splitting Methods for Flows at all Speeds. *AIAA Journal*, vol. 36, no. 9, 1998, pp. 1610–1617.

68. Krist, S. L., Biedron, R. T., and Rumsey, C. L.: CFL3D User's Manual (Version 5.0). NASA Technical Report TM-1998-208444, 1998.

69. Litton, D. K., Edwards, J. R., and White, J. A.: Algorithm Enhancements to the VULCAN Navier-Stokes Solver. AIAA Paper 2003-3979, June 2003.

70. White, J. A., Baurle, R. A., Passe, B. J., Spiegel, S. C., and Nishikawa, H.: Geometrically Flexible and Efficient Flow Analysis of High Speed Vehicles via Domain Decomposition, Part 1, Unstructured-grid Solver for High Speed Flows. 2017 JANNAF CS/APS/EPSS/PSHS Joint Meeting, Dec. 2017.

71. Menter, F. R.: Influence of Free Stream Values on k-ω Turbulence Model Predictions. *AIAA Journal*, vol. 30, June 1992, pp. 1657–1659.

72. Doerffer, P. P. and Bohning, R.: Modelling of Perforated Plate Aerodynamics Performance. *Aerospace Science and Technology*, vol. 4, no. 8, Nov. 2000, pp. 525–534.

73. Slater, J. W.: Improvements in Modeling 90-degree Bleed Holes for Supersonic Inlets. AIAA Paper 2009-0710, Jan. 2009.

74. Millikan, R. C. and White, D. R.: Systematics of Vibrational Relaxation. *Journal of Chemical Physics*, vol. 39, no. 12, 1963, pp. 3209–3213.

75. Lindemann, F. A., Arrhenius, S., Langmuir, I., Dhar, N. R., Perrin, J., and Lewis, W. C. McC.: The Radiation Theory of Chemical Action. *Transactions of the Faraday Society*, vol. 17, 1922, pp. 598–606.

76. Stewart, P., H., Larson, C. W., and Golden, D. M.: Pressure and Temperature Dependence of Reactions Proceeding via a Bound Complex. 2. Application to $2CH_3 \rightarrow C_2H_5 + H$. *Combustion and Flame*, vol. 75, no. 1, 1989, pp. 25–31.

77. Troe, J.: Theory of Thermal Unimolecular Reactions in the Fall-Off Range. I. Strong Collision Rate Constants. *Ber. Bunsenges. Phys. Chem.*, vol. 87, 1983, pp. 161–169.

78. Gilbert, R. G., Luther, K., and Troe, J.: Theory of Thermal Unimolecular Reactions in the Fall-Off Range. II. Weak Collision Rate Constants. *Ber. Bunsenges. Phys. Chem.*, vol. 87, 1983, pp. 169–177.

79. Adams, B. M., Bohnhoff, W. J., Dalbey, K. R., Ebeida, M. S., Eddy, J. P., Eldred, M. S., Hooper, R. W., Hough, P. D., Hu, K. T., Jakeman, J. D., Khalil, M., Maupin, K. A., Monschke, J. A., Ridgeway, E. M., Rushdi, A. A., Seidl, D. T., Stephens, J. A., Swiler, L. P., and Winokur, J. G.: Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.12 User's Manual. Sandia Technical Report SAND2020-5001, 2020.

80. DAKOTA. dakota.sandia.gov, 2022.

81. U.S. Standard Atmosphere, 1976. NASA Technical Report TM-X-74335, 1976.

82. Roache, P. J.: *Verification and Validation in Computational Science and Engineering*. Hermosa Publishers, 1998.

83. Mao, M., Riggins, D. W., and McClinton, C. R.: Numerical Simulation of Transverse Fuel Injection. NASA Contractor Report 1089, 1990.

84. Baurle, R. A. and Gaffney, R. L.: Extraction of One-Dimensional Flow Properties from Multi-Dimensional Data Sets. *Journal of Propulsion and Power*, vol. 24, no. 4, 2008, pp. 704–714.

85. TetGen. http://tetgen.org, 2022.

86. Haimes, R. and Dannenhoffer, III, J. F.: The Engineering Sketch Pad: A Solid-Modeling, Feature-Based, Web-Enabled System for Building Parametric Geometry. AIAA Paper 2013-3073, June 2013.

87. Dannenhoffer, III, J. F.: OpenCSM: An Open-Source Constructive Solid-Modeler for MDAO. AIAA Paper 2013-0701, Jan. 2013.

88. Haimes, R. and Drela M.: On the Construction of Aircraft Conceptual Geometry for High-Fidelity Analysis and Design. AIAA Paper 2012-0683, Jan. 2012.

89. Haimes, R. and Dannenhoffer, III, J. F..: EGADSlite: A Lightweight Geometry Kernel for HPC. AIAA Paper 2018-1401, Jan. 2018.

90. Roy, C. J. and Oberkampf, W. L.: A Comprehensive Framework for Verification, Validation, and Uncertainty Quantification in Scientific Computing. *Computer Methods in Applied Mechanics and Engineering*, vol. 200, no. 25-28, 2011, pp. 2131–2144.

91. DAKOTA Downloads. dakota.sandia.gov/download.html, 2022.

Appendix A

Batch Script Example

A simple PBS (Portable Batch System) script for executing VULCAN-CFD in batch mode is provided below for the parallel execution of either structured grid or unstructured grid simulations. The VULCAN-CFD simulation specific details are supplied early in the script as shell variables, and other than the PBS resource specification, this is the only section of the script that needs to be modified for each simulation. Note that for structured grid cases, this script assumes that the grid has previously been partitioned, implying that the input file name provided corresponds to the partitioned input file.

```
#!/bin/tcsh
#
#PBS -q normal
#PBS -l select=32:ncpus=16:mpiprocs=16
#PBS -l walltime=24:00:00
#
# User-defined VULCAN-CFD related specifications
#
set vulc_opt = -psg
set inp_file = vulcan.inp
set out_file = ${inp_file:r}.out
set err_file = ${inp_file:r}.err
set exec_dir = $HOME/Vulcan/Case
#
# Extract the total number of CPUs available for parallel processing
# and create a host file containing the list of hosts assigned by PBS
#
set num_proc = `wc -l < $PBS_NODEFILE`
echo " "
echo "This job has allocated a total of $num_proc cpus"
echo " "
/bin/cat $PBS_NODEFILE |& tee $exec_dir/.hosts
echo " "
#
# Enter the working directory for the VULCAN-CFD simulation
#
cd $exec_dir
#
# Execute VULCAN-CFD
#
vulcan $num_proc .hosts $vulc_opt $inp_file $out_file >& $err_file
```

Appendix B

Geometry/Grid Initialization for Unstructured Grid Adaptation

Unstructured grid adaptation with VULCAN-CFD is enabled by using the NASA open source mesh adaptation tool *refine*.[6] The adaptation workflow is initiated by defining water-tight geometry for the computational domain, and generating an initial grid that maintains the grid-to-geometry association. The Engineering Sketch Pad (ESP)[86] is the package used to define the geometry for this purpose. ESP is a geometry creation and manipulation system designed for aerospace applications. It contains the open source Constructive Solid Modeler (OpenCSM)[87] package, the Engineering Geometry Aerospace Design System (EGADS)[88] package, and a lighter version of this package (EGADSlite).[89] The **vinf** utility can be used to manage the interaction between *refine* and ESP to create the initial mesh <project_name>.meshb and geometry <project_name>.egads files.

B.1 Geometry Creation

ESP was designed to natively construct geometry in a convenient manner for CFD analysis. The import of STEP and IGES geometry files is possible as well (see Table B1), so that both can be combined to define the CFD domain. ESP also has capabilities to remove or replace problematic regions of the imported geometry, but repairing the geometry in the system used to create it is recommended. A detailed description of these ESP capabilities with examples is provided in the ESP documentation and tutorials. The user must start with an OpenCSM script <project_name>.csm configured to dump a single SolidBody (or FaceBody for 2-D and axisymmetric simulations) to an EGADS file for downstream processes. The *refine* process in VULCAN-CFD expects the same project name used for the <project_name>.csm script to be used for this file, so the following OpenCSM command should be used to perform this operation:

```
dump <project_name>.egads
```

For 2-D or axisymmetric simulations, the FaceBody should be in the *x*-*y* plane, where the *y*-axis is interpreted as the radius. The user is referred to https://acdl:mit:edu/ESP/Training for more extensive documentation and tutorials for working with and building geometry with the OpenCSM CAD system.

The success of Boolean operations when constructing the computational domain can be sensitive to new surface-surface intersections created. A small perturbation of the input solid models or manual adjustments to the tolerances may be required to recover from failed Boolean operations. The water-tightness, smoothness, topology, and other properties of imported models can vary wildly. Inconvenient topology and loose boundary representation tolerances may impede the geometry construction, initial grid generation, and subsequent grid adaptations. The geometry properties are examined at the completion of the grid bootstrap process (described later), and can provide the feedback needed to repair or improve the suitability of a geometry model. An interim SolidBody or FaceBody can be dumped, bootstrapped, and triaged to debug Boolean operations used to construct complex domains.

Table B1: Engineering Sketch Pad Capabilities.

Pro	Con
ESP constrained sketcher	
Fastest way to create simple outlines and grow into 3-D	Requires planning for robust sketch constraints
ESP manual sketcher	
Most robust way to build geometries in ESP	Requires math to develop the models
ESP solids	
Simple solids and Boolean operations to make geometry	Requires planning of multiple construction steps for increasing complexity
Import STEP	
Quickest way if you already have a model	Needs manual sewing of faces or one or more `MANIFOLD_SOLID_BREP`
Import IGES	
If it is all you have it can work; IGES Type 186, Manifold Solid B-Rep Objects import as solids	May require sewing the patches together to get a SolidBody
Discrete grids	
SLUGS (part of ESP) creates a watertight set of surfaces from `.stl`	Manual process

ESP incorporates the concept of attributes that follow entities during construction in OpenCSM and evaluation in EGADS. These attributes persist to allow information to be conveyed between the geometry creation steps and the analysis tools. This allows a boundary condition name (bc_name) attribute to be defined in the OpenCSM script for every face (in 3-D):

```
select face
attribute bc_name inlet_wall
```

or edge (in 2-D):

```
select edge
attribute bc_name inlet_wall
```

The OpenCSM `select` command has a rich set of methods for identifying entities and groups of entities in the model. Once applied, these attributes persist through subsequent geometry construction operations. It is convenient to apply these attributes as early as possible in the OpenCSM script to individual components of the domain (e.g., farfield, symmetry planes, surface sections, etc.) and allow ESP to track these attributes during complex assembly operations. Review the OpenCSM documentation and tutorials for further details.

NOTE: EGADS is used to create an initial tessellation of the geometry for visualization in OpenCSM and to initialize grid generation. A failure of EGADS to tessellate a face will be indicated with a warning of this form

EGADS Warning: Face 304 -> EG_fillTris = -24 (EG_tessThread)!

If a warning is issued, the face indicated should be investigated, because a failure to tessellate often indicates an issue with the face topology or edge curve and surface parameterized curve tolerance.

B.2 Grid to Geometry Association

The *refine* adaptation process requires the grid-to-geometry association, which is built by bootstrapping the initial grid from the geometry created as described above. The grid-to-geometry association requirement prevents the use of an arbitrary initial grid for this process, and a high degree of automation is possible because the initial grid generation can be automated given valid geometry. The geometry bootstrapping and initial grid generation process is invoked by the **s2s** subcommand of **vinf**:

vinf <num_cpus> s2s --csm <project_name>.csm

This command (if successful) produces a coarse initial grid file (<project_name>.meshb) with resolution of curvature and other geometric features, a <project_name>.mapbc file that tags each boundary face group (including any group names set on face attributes in the OpenCSM script file), and the EGADS geometry file (<project_name>.egads). If the

bootstrap was successful these files are all that is needed to begin the adaptation trajectory section of Chapter 25.

B.3 Troubleshooting Initialization Failures

The initialization process will fail if EGADS cannot create an initial tessellation of all the faces in the domain, or if the initial volume grid could not be filled with TetGen. The geometry can be triaged in the event of a failure. The EGADS tessellation process is controlled through a .tParams attribute, and *refine* will attempt to iteratively adjust the .tParams attribute on faces and edges to recover missing faces and report these attempts to standard output. The indices of these faces and edges indicate places where there is an opportunity to improve the geometry. If this automatic process fails, the .tParams attribute can be set manually in OpenCSM and the existing attributes will locally deactivate automatic adjustment. The .tParams attribute is a set of 3 parameters. The first is the maximum length of an edge segment or face triangle side. The second limits the deviation between the centroid of the discrete object and underlying geometry. The third is the maximum interior dihedral angle in degrees. A zero for any of these items deactivates that parameter. The EGADS tessellation is exported to a Tecplot format <project_name>-init-surf.tec surface file and a <project_name>-init-geom.tec discrete geometry file at the beginning of a bootstrap. If long triangle sides in the curvature metric remain at the end of the bootstrap, starting with a finer EGADS tessellation can prevent these "stuck" triangle sides. These triangle sides are indicated by large maximum ratio written to the standard output and the "l" variable in the <project_name>-adapt-prop.tec Tecplot file.

A list of suggestions to speed up the execution of TetGen is provided to standard output by *refine* when TetGen is invoked. If TetGen requires an unusually long time to complete, check these suggestions to limit inserted vertices, loosen element shape measure targets, or limit grid optimization. Accommodating a wide range of scales (driven by geometry topology or curvature) between adjacent faces can increase execution time. A self intersection of the surface will lead to a failure of TetGen. When the grid generation process fails, the surface is examined and surface triangle-triangle intersection locations are reported to standard output with face indices. A <project_name>-intersection.tec Tecplot file is also written with intersection locations. This intersection file is not written if the volume grid generation succeeds.

Failures in grid adaptation occur due to an interaction of geometry properties, problems with the solution error estimation procedure, and grid adaptation protections to maintain a valid grid for the flow solver. These failures are influenced by the grid-adapted solution and cannot be predicted with complete confidence. Feedback reported on the geometry is provided to help understand the geometric contribution to failures. Modifications of the geometry require manual interaction, and it is advised to attempt a pathfinder simulation to see if these issues create downstream problems before investing in the manual interaction for repair.

The bootstrap process reports locations on the geometry with inconvenient edge and face topology, unusually high curvature, small feature size, or large boundary representation tolerance. Sliver faces are marked with `# sliver`, short edges are marked with `# short edge`, and curvature is ignored at locations with `# curve/tol`. These locations are reported to standard output and written to a Tecplot format <project_name>-adapt-triage.tec

file. The <project_name>-adapt-geom.tec file renders the discrete surface, edges, and parameter curves of the geometry with the boundary representation in the `gap` variable. Other geometry variables include entity parameterization `(p0, p1)` and curvature `(k0, k1)`. The discrete representation of the geometry can be used to scan the model for loose boundary representation tolerances (distance between edge and pcurve vertices). The `gap` variable can be filtered to show locations larger than the target viscous spacing (e.g., y^+=1). The tolerance may have a component tangential to the edge curve and a component perpendicular to the edge curve, where the perpendicular component is more likely to create a step in the surface grid. A large perpendicular component of face surface and edge curve mismatch is likely to cause downstream grid adaptation failures. When vertices are identified by a gap larger than the target viscous spacing, examining the discrete edge and pcurve Tecplot zones provides more context for the large tolerance.

The other forms of inconvenient geometry are isolated to efficiency concerns (i.e., additional refinement to resolve geometry features that are not required by the solution error estimate). The information provided by triage of the geometry can be used to prioritize required repairs of the geometry for analysis or potential improvements to the geometry to make grid generation more efficient. For example, a short edge in the geometry is accommodated by resolving the nearby surface and volume grid to incorporate this small edge with limits on grid gradation (smooth size variation). Modifying the geometry to eliminate small edges may improve efficiency by removing the constraint of this edge. In practice, these small geometry features are mostly an aesthetic concern. Loose boundary representation tolerances may be a larger concern for subsequent grid adaptation operations.

B.4 Initial Grid Improvements

The initial volume grid created in the bootstrap process does not conform to the anisotropic curvature and geometry feature size metric used to create the surface grid. This is addressed using the interpolated geometry-based metric to provide a suitable initial inviscid grid. Boundary layer resolution suitable for viscous flow simulations can be built implicitly during grid adaptation, or boundary layer formation can be accelerated by adapting to the u^+ given by the Spalding law-of-the-wall relationship. If adaption to u^+ is desired, then the following **s2s** subcommand should be invoked:

```
vinf s2s -m <project name>.meshb
         --spalding
         --target-cell-count <desired number of cells>
         --yplus <estimate of $y^+=1$ distance>
         --tags <list of viscous wall boundary tags>
```

The target grid size takes priority over the y^+=1 normal wall spacing, so larger normal wall spacing will result when using a smaller target size.

As a final note, a pathfinder examination of the geometry properties can be undertaken by adapting to the Spalding u^+ metric at high target mesh sizes. This examination can be performed without the flow solver, where the target mesh size is approached by increasing

the mesh size with a series of grid adaptations. This process can reveal potential problems with the geometry early, rather than later when the true solution adaptation process is performed.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01-11-2022	Technical Memorandum	

4. TITLE AND SUBTITLE
VULCAN-CFD User Manual: Ver. 7.2.0

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Baurle, R. A., White, J. A., O'Connell, M. D., Drozda, T. G., and Norris, A. T.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
147016.02.07.05.03

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
NASA Langley Research Center
Hampton, Virginia 23681-2199

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSOR/MONITOR'S ACRONYM(S)
NASA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
NASA/TM–20220008781

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Category 02
Availability: NASA STI Program (757) 864-9658

13. SUPPLEMENTARY NOTES
An electronic version can be found at http://ntrs.nasa.gov.

14. ABSTRACT

VULCAN-CFD offers a comprehensive set of capabilities to enable the simulation of continuum flowfields from subsonic to hypersonic conditions. The governing equations that are employed include allowances for both chemical and thermal nonequilibrium processes, coupled with a wide variety of turbulence models for both Reynolds-averaged and large eddy simulations. Simulations can be performed using structured multiblock meshes or fully unstructured meshes. A parabolic (i.e., space-marching) treatment can also be used for any subset of a structured mesh that can accommodate this solution strategy. The flow solver provides geometric flexibility for structured grid simulations by allowing for arbitrary face-to-face C(0) continuous and non-C(0) continuous block interface connectivities. The unstructured grid paradigm allows for mixed element meshes that contain any combination of tetrahedral, prismatic, pyramidal, and hexahedral cell elements. The flow solver is also fully parallelized using MPI (Message Passing Interface) libraries, allowing for efficient simulations on High Performance Computing (HPC) systems. This document provides the installation and usage instructions for the software.

15. SUBJECT TERMS
Computational Fluid Dynamics, Aerodynamics, Turbulence, 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			STI Information Desk (help@sti.nasa.gov)
U	U	U	UU	328	19b. TELEPHONE NUMBER *(Include area code)* (757) 864-9658