



Rocstar v5.0.4

User Guide

Contract Number DE-SC0018481

Illinois Rocstar LLC
108 Hessel Blvd
Champaign, IL 61820-6574



Contents

1	Introduction	1
1.1	<i>Rocstar</i>	1
1.2	Coupling Framework	1
1.3	<i>IMPACT</i>	1
1.4	Structural Domain Solver (<i>CalculiX</i>)	3
1.5	Fluid Domain Solvers	3
1.5.1	<i>Rocflu</i>	4
1.5.2	<i>RocFOAM</i>	4
1.6	Combustion Solver (<i>Rocburn</i>)	4
1.7	Services	5
1.7.1	<i>Rocprop</i>	5
1.7.2	<i>Rocprep</i>	5
2	Installation	5
2.1	Ubuntu	6
2.1.1	Third-Party Library Dependencies	6
2.1.2	Building <i>IMPACT</i>	6
2.1.3	Installing <i>Rocstar Multiphysics</i>	7
2.2	CentOS	7
2.2.1	Third-Party Library Dependencies	7
2.2.2	Building <i>IMPACT</i>	8
2.2.3	Installing <i>Rocstar Multiphysics</i>	9
3	Setting Up an FSI Simulation	9
3.1	Fluid and Burning Coupling	10
3.1.1	Native Data Archive	10
3.1.2	Control Files	10
3.1.3	Data and Grid Files	12
3.2	Fluid and Solid Coupling	15
3.2.1	Writing <i>RocFOAM</i> Input Files	16
3.2.2	Writing <i>CalculiX</i> Input Files	17
4	Case Study: Attitude Control Motor Rocket	20
4.1	Problem Setup	20
4.2	Input Files	21
4.2.1	Control Files	21
4.2.2	Data Files	23
4.3	ACM Output Files	27
4.3.1	<i>Rocflu</i> Output	27
4.3.2	<i>Rocburn</i> Output	28
5	Case Study: Wind Turbine	28
5.1	Problem Setup	28
5.1.1	Mesh	28
5.1.2	Materials	29
5.1.3	Boundary Conditions	29
5.1.4	Loads	30



5.2	Input Files	30
5.2.1	Fluid Input Files	30
5.2.2	Solid Input Files	37
5.2.3	<i>Rocstar</i>	38
5.3	Running a <i>Rocstar</i> Case	38
5.3.1	Configuring the <i>Rocstar</i> File Structure	39
5.3.2	Preprocessing Fluid Input Files	39
5.3.3	Preprocessing Solid Input Files	41
5.3.4	Computing Common Refinement Files	42
5.3.5	Executing <i>Rocstar</i>	42
6	Case Study: Hypersonic Plate	44
6.1	Problem Setup	44
6.1.1	Mesh	44
6.1.2	Materials	45
6.1.3	Boundary Conditions	45
6.2	Input Files	45
6.2.1	Fluid Input Files	45
6.2.2	Solid Input Files	48
6.3	Running a <i>Rocstar</i> Case	50
6.3.1	Configuring the <i>Rocstar</i> File Structure	51
6.3.2	Preprocessing Fluid Input Files	52
6.3.3	Preprocessing Solid Input Files	54
6.3.4	Computing Common Refinement Files	55
6.3.5	Executing <i>Rocstar</i>	55
7	Case Study: Simple Mixer	57
7.1	Problem Setup	58
7.1.1	Mesh	58
7.2	Input Files	58
7.2.1	Fluid Input Files	58
7.2.2	Solid Input Files	61
7.2.3	<i>Rocstar</i>	62
7.3	Running a <i>Rocstar</i> Case	62
7.3.1	Configuring the <i>Rocstar</i> File Structure	63
7.3.2	Preprocessing Solid Input Files	64
7.3.3	Computing Common Refinement Files	65
7.3.4	Executing <i>Rocstar</i>	66
8	Case Study: Wave Rider	68
8.1	Problem Setup	68
8.1.1	Mesh	69
8.1.2	Materials	69
8.1.3	Boundary Conditions	70
8.2	Input Files	70
8.2.1	Fluid Input Files	70
8.2.2	<i>Rocstar</i>	70
8.3	Running a <i>Rocstar</i> Case	70



8.3.1	Preprocessing Fluid Input Files	71
8.3.2	Configuring the <i>Rocstar</i> File Structure	71
8.3.3	Executing <i>Rocstar</i>	72
9	Questions and Feedback	72
9.1	Support Team	73
9.2	Company Contact Information	73



1 Introduction

1.1 Rocstar

Rocstar Multiphysics (or interchangeably, *Rocstar*) is a multiphysics simulation application designed for coupled multiphysics simulations involving fluid-structure interaction (FSI) across moving, reacting interfaces. *Rocstar* programmatically couples multiple domain-specific simulation packages and disparately discretized domains and provides several simulation-supporting services including conservative and accurate data transfer, surface propagation, and parallel I/O (input/output). *Rocstar* is MPI-parallel.

Rocstar was conceived with funding from the US Department of Energy (DOE) and developed at the University of Illinois Center for Simulation of Advanced Rockets under the Accelerated Strategic Computing Initiative (ASCI). *Rocstar* uses the computational fluid dynamics (CFD) general notation system (CGNS) and hierarchical data format (HDF) as its primary I/O format. *Rocstar* relies on a few modules within the *Illinois Rocstar Multiphysics Application Coupling Toolkit* (*IMPACT*) for its I/O.

1.2 Coupling Framework

Multiphysics simulations in general, and FSI simulations in particular, require two or more physics solvers interacting with each other for data transfer and synchronization. To accomplish this purpose, Illinois Rocstar (IR) has developed the *IMPACT* suite, a standalone infrastructure for orchestrating multiphysics simulations, while serving as the core coupling framework for the Rocstar suite. *IMPACT* provides several features to facilitate multiphysics and multidisciplinary simulations, including **component object manager** (COM), which is the integration interface of *IMPACT* suite. COM is a component-based integration software that provides a systematic data-centric approach for inter-modular interaction. COM provides methods to keep track of and access data by simulation modules. Using the infrastructure constructs, a **component-side client** (CSC) associated with a particular module/solver creates a **component interface** (CI) and registers its datasets into CI instances called windows. With authorization from their base module, these datasets can later be retrieved via handles provided by COM to register datasets to a window. This approach is beneficial since it allows independent design and development of individual modules in a multiphysics suite. For detailed information, we refer to the COM user guide manual available in the *IMPACT* source repository.

Orchestration of a *Rocstar* simulation is managed by the **Simulation Integration Manager** (SIM) module of the *IMPACT* library, Figure 1. SIM organizes a simulation through its five key components: **Action**, **Scheduler**, **Agent**, **Coupling**, and **driver**. Combined with the CSC modules and the COM module, *Rocstar* orchestrates a multiphysics simulation.

1.3 IMPACT

IMPACT is not an application, but a software development infrastructure. It is designed to be used from the build directory and provides its capabilities in libraries that are linked by the user's applications. Assuming the user's software package is named "UserFoo," the following general steps are taken to integrate "UserFoo":

1. Prepare the application for integration
 - (a) Massage the "UserFoo" architecture so that it consists of a library and a driver, which links and drives the library.

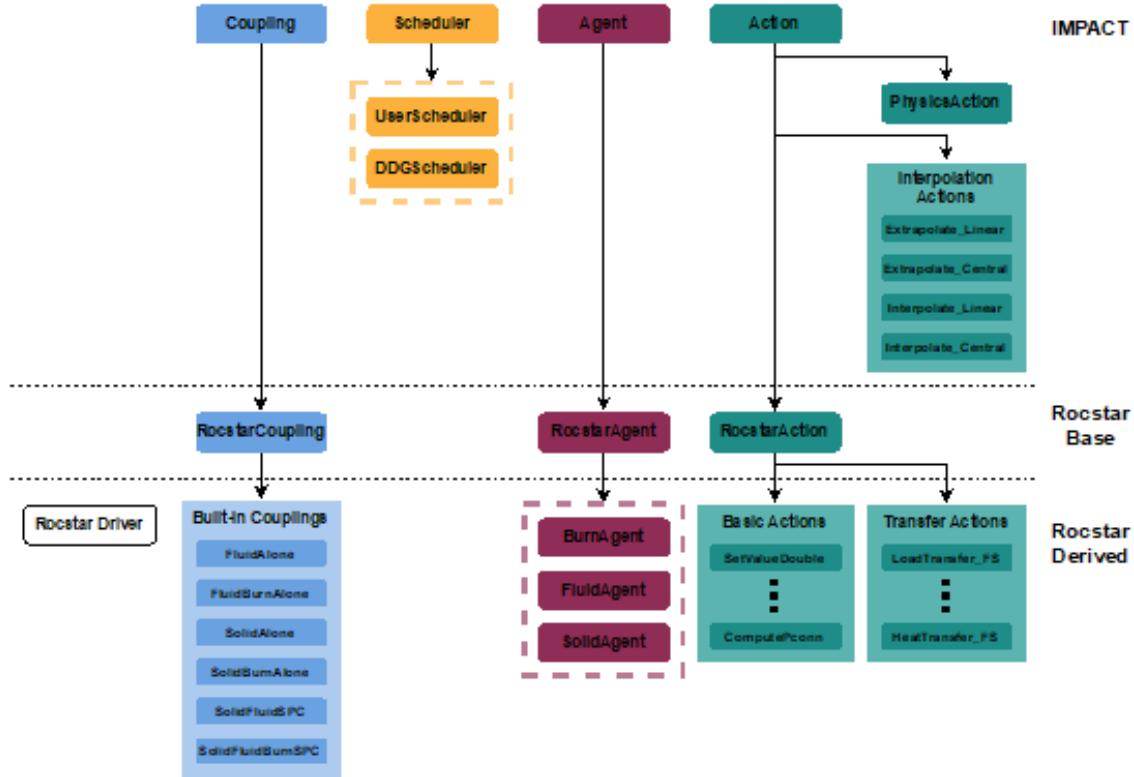


Figure 1: Architecture of the generalized SIM module (part of *IMPACT* library), *Rocstar* base and derived classes.

- (b) Make necessary changes to represent interacting interface surfaces as stand-alone, self-descriptive surface mesh.
 - (c) Make necessary changes to support externally supplied boundary solutions on interface meshes. The source of the solution and any transformations may be neglected.
2. Implement the CSC in the “UserFoo” library
- (a) Implement “UserFoo_load_module” and “UserFoo_unload_module” (see COM Users Guide).
 - (b) Using the COM API, create the “UserFoo” CI and register UserFoo-native data and functions, as needed.
3. Implement a driver and ensure it does the following (step optional, used for testing and development purposes only)
- (a) Initializes message passing interface (MPI).
 - (b) Initializes COM.
 - (c) Loads “UserFoo” with “COM_LOAD_STATIC_DYNAMIC(UserFoo, “userfoowindow_name”).
 - (d) Can access registered DataItems through the CI.
 - (e) Can call “UserFoo” functions through the CI.

The COM API is the most used part of *IMPACT* for preparing and integrating an existing application. This API is documented in the COM User Guide document). The SIM may be optionally



used to create multiphysics drivers or the driver can be entirely designed by the user. The COM API is used to access integrated application data and methods through the COM CI. A user's driver may load and use any of the service modules included in *IMPACT* in creating a multiphysics capability. Each package is described in its own documentation included in the *IMPACT* distribution in *IMPACT_SOURCE/Documentation*.

1.4 Structural Domain Solver (*CalculiX*)

CalculiX is an open-source package designed to use the finite element method to solve field problems. It is used within *Rocstar* to manage computational structural mechanics calculations, Figure 2.

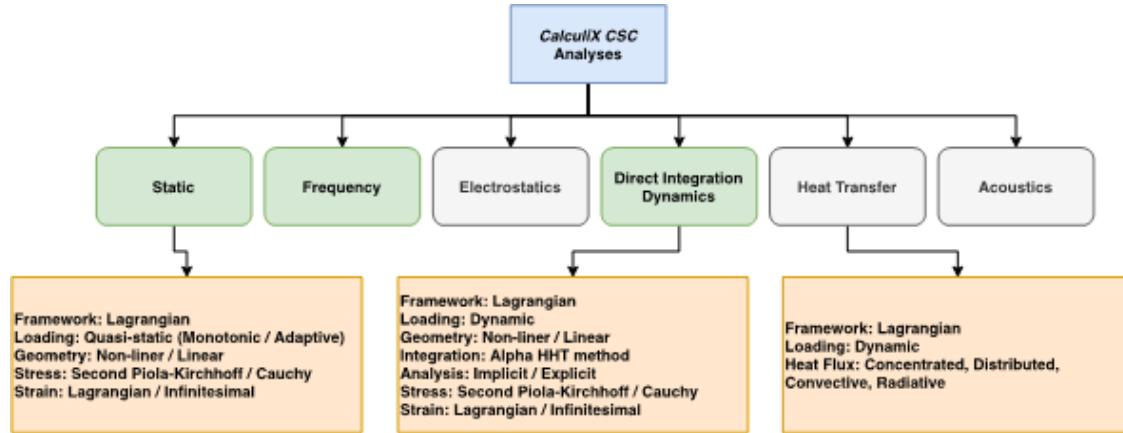


Figure 2: Analyses supported by *CalculiX* modules (green: functional, gray: under development).

The solid solver is implemented as an object-oriented extension module to the open-source *CalculiX*¹. The solver uses a finite element scheme to discretize the equation of motion. Solid structures are modeled using Hooke's law of linear elasticity as well as several other non-linear material models. Loads are applied to the structure using an outer incremental loop combined with an iterative Newton Raphson inner loop to address material and geometric nonlinearities. For frequency analyses, the lowest eigen frequencies/modes are computed using a generalized eigenvalue solution step. Before frequency analysis, the structure is constrained with boundary conditions, and loads are applied to account for the contributions of the displacements and stresses on the stiffness matrix. The direct integration dynamics module allows for the integration of the equation of equilibrium through time to resolve the time-domain structural dynamics of the system. The module is equipped with an implementation of the alpha-HHT integration method to allow for both implicit and explicit steps in time. The module supports material and geometric non-linearities. Proper measures of stress and strain tensors are used to address large deformations.

1.5 Fluid Domain Solvers

The IR team is currently developing *RocFOAM* as a replacement for the current flow solver, *Rocflu*. Until the functionality is fully reproduced by *RocFOAM*, *Rocflu* should be used.

¹<http://www.calculix.de/>



1.5.1 *Rocflu*

Rocflu is an unstructured, explicit finite-volume (FV) flow solver with support for Lagrangian and Eulerian multiphase flows. *Rocflu* solves the compressible Navier-Stokes equations on moving block-structured grids. It supports tetrahedral, hexahedral, triangular prism, pyramids, and mixed element meshes.

1.5.2 *RocFOAM*

RocFOAM is an *OpenFOAM* CSC module undergoing integration into *Rocstar*, Figure 3. *RocFOAM* is comprised of two compressible flow solvers, *rocRhoCentral* and *rocRhoPimple*.

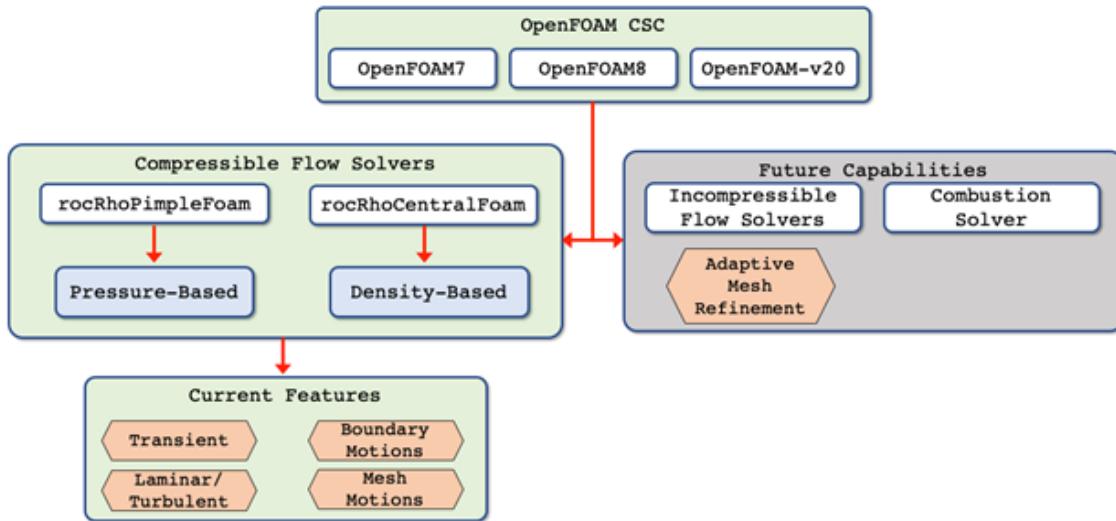


Figure 3: Architecture of the *RocFOAM* module and its current/future functionality..

The flow solver is implemented as an object-oriented module that relies on pre- and post-processing as well as the computational fluid dynamics (CFD) capabilities of the *Open-source Field Operation And Manipulation (OpenFOAM)* [21] library. The compressible flow regime is used to address the tip velocity of the blades at operating conditions that may signify the importance of compressibility. The *rhoPimpleFoam* solver is adopted based on the compressible *PIMPLE* algorithm, which is a combination of *PISO* (*Pressure Implicit with Splitting of Operator*) [22] and *SIMPLE* (*Semi-Implicit Method for Pressure-Linked Equations*) [23]. The *PIMPLE* algorithm is composed of multiple inner corrections iterations on the pressure field, and various outer correction iterations for the solution of a system of equations until the residuals reach a tolerance value or a maximum number of iterations before advancing to the next time step. The solution of the transport equations based on the outer and inner iterations continues to the end of the flow simulation.

1.6 Combustion Solver (*Rocburn*)

Rocburn is a zero and one-dimensional combustion solver designed to implement burn rate and material heating models. *Rocburn* is capable of simulating material heating, ignition, and regression rates for burning or otherwise reacting surfaces. Triangular and quadrilateral meshes are supported.



1.7 Services

1.7.1 *Rocprop*

Rocprop is a surface propagation service module providing Lagrangian surface tracking and propagation for *Rocstar*. *Rocprop* implements marker-particle and more advanced face-offsetting methods for surface propagation.

1.7.2 *Rocprep*

Rocprep is a preprocessing utility. It is run with a command similar to the one in Listing 1.

Listing 1: *Rocprep* run command.

```
$ {buildDir}/bin/rocprep -A -b -m -u 1 1 -n 4 -d <source/path> -t <target/path>
```

The flags used in Listing 1 tell *Rocprep* to use four processors to extract and preprocess the data in the Native Data Archive (NDA) specified with **-d <source/path>**, apply *Rocburn* specific preprocessing, apply *Rocflu* specific preprocessing on the NDA directories **Data1** and **Grid1**, and save the preprocessed data in the **<target/path>**. All *Rocprep* options are provided in Table 1.

Table 1: *Rocprep* Options.

Major Modes of Operation

- A Extract and preprocess
- C Check an existing dataset at -d <path>
- E Copy case files to target at -d <path>
- P Run module “pretools” on data at -d <path>

Physics & Service Module Selection

- u [m] [n] *Rocflu* preprocessing, optional NDA Data <m> & Grid <n> dirs
- b *Rocburn* preprocessing

Module-Specific Flags

- r <m> Specify <m> regions (*Rocflu* only), default is -n value

General Options

- i <o|u|f|s> *surfdrive* interface meshes, default inferred from physics options
- d <path> Path to source data, default is current working directory
- h Print a help message and terminate
- n <m> Specify <m> processors/partitions
- t <path> Target path for new *Rocstar* dataset
- p <path> Path to *pretool* binaries, default will use shell path
- x Ignore RocprepControl.txt control file

Future preprocessing will be handled with the Illinois Rocstar *NEMoSys* software, which offers automated meshing, remeshing, and adaptive mesh refinement and uses a clear, human-readable JSON input file.

2 Installation

Rocstar Multiphysics contains two primary software packages: the *IMPACT* coupling software and the *Rocstar Multiphysics* code itself. The following sections provide instructions on how to install



both software packages. Compilation and installation have been tested on both Ubuntu and CentOS Linux, specifically Ubuntu version 18.04 and CentOS version 7.5.1804.

Warning on instructions

Instructions given here may not be up to date. Please refer to the project `README` files for latest versions of the compile and build instructions.

2.1 Ubuntu

2.1.1 Third-Party Library Dependencies

To build the packages provided, the following third-party libraries must be installed using the `apt-get install` command.

- `build-essential`
- `cmake`
- `mpich`
- `libcgns-dev`
- `libhdf4-dev`
- `liblapack-dev`
- `libblas-dev`
- `libjpeg-dev`
- `libhdf5-dev`
- `libproj-dev`
- `libmetis-dev`
- `libfltk1.3-dev`
- `libgmp-dev`
- `libsml-dev`
- `libice-dev`
- `gfortran`
- `libboost-all-dev`
- `libxt-dev`
- `zlib1g-dev`
- `tcl-dev`
- `tk-dev`
- `libxmu-dev`
- `python-dev`

To install all of the above libraries in a single command, run the following:

```
apt-get install build-essential cmake mpich libcgns-dev libhdf4-dev liblapack
    -dev libblas-dev libjpeg-dev libhdf5-dev libproj-dev libmetis-dev libfltk1
    .3-dev libgmp-dev libsml-dev libice-dev gfortran libboost-all-dev libxt-dev
    zlib1g-dev tcl-dev tk-dev libxmu-dev python-dev
```

The *METIS* graph partitioning library is also required and must be built from source. *Rocstar Multiphysics* currently uses *METIS* Version 4.0.3². The default build configuration described in the build instructions included in the package is sufficient.

2.1.2 Building *IMPACT*

The *Illinois Rocstar Multiphysics Application Coupling Toolkit (IMPACT)*, is the software library that orchestrates the coupling between different solvers in *Rocstar Multiphysics*. As such, *IMPACT* must be installed to use *Rocstar Multiphysics*. For the following steps, we assume `$IMPACT_PROJECT_PATH` is the path to *IMPACT* and that `$IMPACT_INSTALL_PATH` is the desired installation location. To start the build process, execute:

²<http://glaros.dtc.umn.edu/gkhome/fsroot/sw/metis/OLD>

Listing 2: "Commands for building all *IMPACT* libraries and executables."

```
$ cd $IMPACT_PROJECT_PATH
$ mkdir build cd build
$ cmake -DCMAKE_INSTALL_PREFIX=$IMPACT_INSTALL_PATH ..
$ make -j$(nproc)
$ make install
```

Executing the commands in Listing 2 will build all libraries and executables for *IMPACT*.

2.1.3 Installing *Rocstar Multiphysics*

The modernized *Rocstar Multiphysics* currently under development and will support many new features including a more streamlined and robust I/O, updated physics solvers, integration of advanced meshing tools, and more generalized solver coupling capabilities. Further, ease-of-use features such as a graphical user interface, cloud-computing support, and a developer toolkit are in the process of being integrated into the new *Rocstar Multiphysics*.

Note on installing *Rocstar Multiphysics*.

Prior to compiling *Rocstar Multiphysics*, *IMPACT* must be installed.

For the following steps, we assume:

- `$ROCKSTAR_PROJECT_PATH` is the path to *Rocstar Multiphysics*
- `$ROCKSTAR_INSTALL_PATH` is the desired installation location
- `$IMPACT_INSTALL_PATH` is the path to the location where *IMPACT* project is installed
- `$METIS_LIB_PATH` is the path to the location of the *METIS* library (`/path/to/metis/libmetis.a`)
- `$METIS_INC_PATH` is the path to the location of the *METIS* headers (`/path/to/metis/Lib`)

The build process is started by executing:

Listing 3: "Commands for building *Rocstar Multiphysics*."

```
$ cd $ROCKSTAR_PROJECT_PATH
$ mkdir build cd build
$ IMPACT_DIR=$IMPACT_DIR cmake -DCMAKE_INSTALL_PREFIX=$ROCKSTAR_INSTALL_PATH -
    DMETIS_LIB=$METIS_LIB ..
$ make -j$(nproc)
$ make install
```

2.2 CentOS

2.2.1 Third-Party Library Dependencies

To build the packages provided, the following third-party libraries must be installed using the `yum install` command. Note that libraries marked with EPEL are located in Extra Packages for Enterprise Linux repositories³.

³<https://fedoraproject.org/wiki/EPEL>



- mpich-3.2
- mpich-3.2-autoload
- mpich-3.2-devel
- lapack-devel
- blas-devel
- hdf5-devel
- libjpeg-turbo-devel
- fltk-devel
- gmp-devel
- metis-devel (EPEL)
- netgen-mesher-devel (EPEL)
- hdf4 (EPEL)
- hdf4-devel (EPEL)

To install all of the above libraries in a single command, run the following:

```
yum install mpich-3.2 mpich-3.2-autoload mpich-3.2-devel lapack-devel blas-devel  
hdf5-devel libjpeg-turbo-devel fltk-devel gmp-devel metis-devel netgen-mesher  
-devel hdf4 hdf4-devel
```

Because of version limitations, incompatibility issues, and other considerations, any default system-provided libraries such as *CGNS* cannot be used. We recommend building these packages from source using MPI compilers.

The *METIS* graph partitioning library must be built from source. *Rocstar Multiphysics* currently uses *METIS* Version 4.0.3⁴. The default build configuration described in the build instructions included in the package is sufficient.

The *CGNS* library Version 3.2.1 is required. It is available from GitHub⁵ with build instructions included in the package. HDF5 and Scoping capabilities must be enabled during configuration. Since this library uses CMake, the following procedure can be used to build the package properly:

```
mkdir build cd build  
cmake -DCMAKE_INSTALL_PREFIX=$CGNS_INSTALL_PATH -DCGNS_ENABLE_SCOPING=on \  
-DCGNS_ENABLE_HDF5=on ..  
make -j  
make install
```

In the event that the default, system-provided *CGNS* library is installed, keep the current build in a separate location (such as /opt/ path) and point *Rocstar Multiphysics* and *IMPACT* to this build during build configuration.

2.2.2 Building *IMPACT*

The *Illinois Rocstar Multiphysics Application Coupling Toolkit (IMPACT)*, is the software library that orchestrates the coupling between different solvers in *Rocstar Multiphysics*. As such, *IMPACT* must be installed to use *Rocstar Multiphysics*. For the following steps, we assume \$IMPACT_PROJECT_PATH is the path to *IMPACT* and that \$IMPACT_INSTALL_PATH is the desired installation location. To start the build process, execute:

Listing 4: "Commands for building all *IMPACT* libraries and executables."

```
cd $IMPACT_PROJECT_PATH
```

⁴<http://glaros.dtc.umn.edu/gkhome/fsroot/sw/metis/OLD>

⁵<https://github.com/CGNS/CGNS/releases>



```
mkdir build cd build
cmake -DCMAKE_INSTALL_PREFIX=$IMPACT_INSTALL_PATH ..
make -j$(nproc)
make install
```

Executing the commands in Listing 4 will build all libraries and executables for *IMPACT*.

2.2.3 Installing *Rocstar Multiphysics*

The modernized *Rocstar Multiphysics* currently under development and will support many new features including a more streamlined and robust I/O, updated physics solvers, integration of advanced meshing tools, and more generalized solver coupling capabilities. Further, ease-of-use features such as a graphical user interface, cloud-computing support, and a developer toolkit are in the process of being integrated into the new *Rocstar Multiphysics*.

Note on installing *Rocstar Multiphysics*.

Prior to compiling *Rocstar Multiphysics*, *IMPACT* must be installed.

For the following steps, we assume:

- `$ROCSTAR_PROJECT_PATH` is the path to *Rocstar Multiphysics*
- `$ROCSTAR_INSTALL_PATH` is the desired installation location
- `$IMPACT_INSTALL_PATH` is the path to the location where *IMPACT* project is installed
- `$METIS_LIB_PATH` is the path to the location of the *METIS* library (`/path/to/metis/libmetis.a`)
- `$METIS_INC_PATH` is the path to the location of the *METIS* headers (`/path/to/metis/Lib`)
- `$CGNS_LIB_PATH` is the path to the location of the *CGNS* library

The build process is started by executing:

Listing 5: "Commands for building *Rocstar Multiphysics*."

```
cd $ROCSTAR_PROJECT_PATH
mkdir build cd build
IMPACT_DIR=$IMPACT_DIR cmake -DCMAKE_INSTALL_PREFIX=$ROCSTAR_INSTALL_PATH -
DMETIS_LIB=$METIS_LIB ..
make -j$(nproc)
make install
```

3 Setting Up an FSI Simulation

Two case studies are used to demonstrate how an FSI simulation can be run with *Rocstar*. The first will demonstrate a coupled fluid-burn simulation of an attitude control motor rocket (ACM), discussed in more detail within Section 4. The second will use a different fluid solver in a coupled fluid-solid simulation of a wind turbine, discussed in more detail within Section 5.

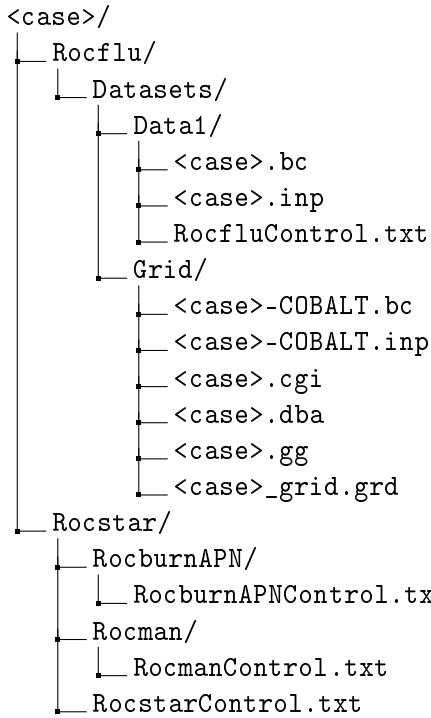


Figure 4: NDA for a `FluidBurnAlone` case named `<case>`, using *Rocflu* and *RocburnAPN*.

3.1 Fluid and Burning Coupling

The current implementation requires that the user specify input parameters for *Rocstar* and the solvers in input files placed in specific locations throughout the case directory. The following sections outline the structure of the directory and the format of the control and input files.

3.1.1 Native Data Archive

Rocstar requires that case files be set up in a specific file hierarchy, depicted in Figure 4. The parent directory is labeled with the case name and contains directories for the solvers used as well as a `Rocstar/` directory. The directory for the solver, in this case *Rocflu*, contains two directories, which contain the simulation data and the associated grid data.

Note that the directory for the combustion solver *RocburnAPN* is located within the `Rocstar/` directory.

3.1.2 Control Files

3.1.2.1 *Rocstar* Control

`RocstarControl.txt` contains:

- Coupling scheme
- Solver modules
- Method/location of output
- Maximum simulation time, timestep, frequency of output, and the maximum wall time the simulation will be run (all in units of seconds)



Supported coupling schemes:

- **Built in**

- **FluidAlone**: Fluid alone with no burn.
- **FluidBurnAlone**: Fluid alone with burn.
- **SolidAlone**: Solid alone with no burn.
- **SolidBurnAlone**: Solid alone with burn.
- **SolidFluidSPC**: Fully coupled no burn with simple staggered scheme with predictor-corrector iterations.
- **SolidFluidBurnSPC**: Fully coupled with burn with simple staggered scheme with predictor-corrector iterations.

- **Derived**

- **SolidFluidISS**: Fully coupled no burn with improved staggered scheme
- **SolidFluidBurnEnergySPC**: Fully coupled with burn energy

3.1.2.2 *Rocflu* Control

RocfluControl.txt contains:

- Case name
- Relative paths to the **Modin**, **Modout**, and **Rocin** directories
- Output verbosity (0-none, 1-low, or 2-high)
- Runtime checks (0-none, 1-low, or 2-high)

3.1.2.3 *RocburnAPN* Control

RocburnAPNControl.txt contains:

- a and n for the $rb = aP^n$ burn rate model, where rb is the burn rate, a and n are parameters specific to the propellant, and P is the pressure
- Maximum number of spatial nodes
- Adiabatic flame temperature (K)
- Initial temperature (K)
- The file path to the **Rocburn** output directory.



3.1.2.4 Rocman Control

`RocmanControl.txt` contains:

- Verbosity level (0-none, 1-low, or 2-high)
- Order of interpolation
- Traction mode
- Ambient pressure
- Solid density (fluid-alone mode)
- Pressure (solid-alone mode)
- Burn rate (solid-alone mode)
- Data transfer parameters
 - Verbosity level
 - Order of quadrature rules
 - Maximum iterations
 - Tolerance for the iterative solver
- Flag to determine if face-offsetting should be enabled
- Flags to determine whether to use asynchronous input and output

3.1.3 Data and Grid Files

Data files contain information specific to the solvers being used. In the simulation being used to demonstrate the current implementation, only the *Rocflu* solver requires data files besides the control files already described in the previous sections. *Rocflu* data is separated into Data and Grid information. The data files that must be included with a *Rocflu* simulation are the input file and boundary condition file, named `<case>.inp` and `<case>.bc`, respectively.

3.1.3.1 Input

The input file contains sections demarcated with # signs. Each section contains several lines made up of a keyword and a value, as shown in Listing 6.

Listing 6: *Rocflu* Data and Grid File Input.

```
1 # SECTION_NAME
2 KEYWORD_1 VALUE_1
3 KEYWORD_2 VALUE_2
4 KEYWORD_3 VALUE_3
5 #
```

Input sections include:

- FORMATS
 - GRID: grid file format (ASCII or binary)



- **SOLUTION**: flow file format (ASCII or binary)
- **GRIDSRC**: source grid format (CENTAUR ASCII, VGRIDns, MESH3D, TETMESH, Cobalt, GAMBIT, or CENTAUR binary)

- **FLOWMODEL**

- **MODEL**: flow equations to be solved (Euler or Navier-Stokes)
- **MOVEGRID**: whether grid motion will be active.

- **NUMERICS**

- **CFL**: CFL number to be used
- **DISCR**: discretization scheme to be used
- **ORDER**: order of accuracy of flux discretization
- **ENTROPY**: value of entropy correction coefficient

- **TIMESTEP**

- **FLOWTYPE**: steady or unsteady flow
- **TIMESTEP**: maximum timestep to be used in [s] (unsteady flow only)
- **STARTTIME**: time in [s] from which computation should be started (unsteady flow only)
- **MAXTIME**: time in [s] at which computation should be stopped (unsteady flow only)
- **WRITETIME**: offset in [s] at which flow files (and grids, if they are moving) are to be written (unsteady flow only)
- **PRNTIME**: offset in [s] at which convergence information is printed and written to the convergence file (unsteady flow only)
- **DTMINLIMIT**: minimum timestep below which information will be printed about the region and cell where the minimum timestep occurs (unsteady flow only)

- **GRIDMOTION**

- **TYPE**: type of grid motion (smoothing boundary displacements, smoothing coordinates, or with the MESQUITE package)
- **NITER**: number of smoothing iterations (for first two types only)
- **SFACT**: smoothing coefficient (for first two types only)

- **REFERENCE**

- **GAMMA**: ratio of specific heats
- **CP**: specific heat coefficient at constant pressure in [J/kg K]

- **PROBE**

- **NUMBER**: number of probes
- **WRITETIME**: offset in [s] at which data is written to probe files
- **OPENCLOSE**: whether probe files should be closed and opened after writing data

- **INITFLOW**



- FLAG: how an initial solution will be generated (with keywords, a data file, or a hardcode)
- DENS: density of the initial solution in [kg/m³] (for keyword only)
- VEL(X/Y/Z): (x/y/z) component of the initial velocity vector in [m/s] (for keyword only)
- PRESS: static pressure of the initial solution in [Pa] (for keyword only)

- **TRANSFORM**

- FLAG: whether the grid should be scaled and rotated
- SCALE_(X/Y/Z): scaling factor for the (x/y/z) component of coordinates

- **PREP**

- SURFLAG: whether rfluprep was compiled as part of GENx

- **POST**

- PLTTYPE: data to be written to output (grid only or grid and solution)
- PLTVOLFLAG: whether volume data should be written or just surface data
- MERGEFLAG: whether regions from parallel computation should be merged for postprocessing

- **ROCKET**

- CASERAD: radius of the cylindrical case

- **TIMEZOOMING**

- MAXPLANE: maximum coordinate to apply zooming
- MINPLANE: minimum coordinate to apply zooming

Note that this list is incomplete, and only covers sections and keywords used in the ACM case. For a more complete review of *Rocflu*, see the legacy user guide found in the `Docs/legacy/UG` directory.

3.1.3.2 Boundary Conditions

Boundary condition files follow a similar format as input files; the file contains sections with keywords and values as shown in Listing 7.

Listing 7: *Rocflu* Data and Grid File Boundary Conditions.

```
1 # SECTION_NAME
2 PATCH PATCH_1 PATCH_2
3 KEYWORD_1 VALUE_1
4 KEYWORD_2 VALUE_2
5 #
```

Each section assigns a boundary condition to a range of patches, starting with `PATCH_1` and ending with `PATCH_2`. To specify a single patch, set `PATCH_1` and `PATCH_2` to the same value. The patch types used in the ACM case include:

- **BC_INJECT**



- NAME: name of the boundary
- MFRATE: injection mass flux in [kg/m²s]
- TEMP: injection static temperature [K]
- RFVF(U/V/W):
- COUPLED: whether patch is interacting during a computation with GENx
- BFLAG: whether patch is burning at t=0
- MOVEDIR: which directions vertices on the patch can move (none, x/y/z coordinate directions, within xy/xz/yz-planes, any direction)

- **BC_SLIPW**

- NAME: name of the boundary
- COUPLED: whether patch is interacting during a computation with GENx
- MOVEDIR: which directions vertices on the patch can move (none, x/y/z coordinate directions, within xy/xz/yz-planes, any direction)

- **BC_OUTFLOW**

- NAME: name of the boundary
- TYPE: type of outflow (supersonic, subsonic, or mixed)
- COUPLED: whether patch is interacting during a computation with GENx
- MVPATCH: whether patch is moving
- SMGRID: whether grid on patch should be smoothed
- CORR: whether point-vortex correction should be applied
- MOVEDIR: which directions vertices on the patch can move (none, x/y/z coordinate directions, within xy/xz/yz-planes, any direction)

3.1.3.3 Grid Files

Rocflu can use VGRIDns, MESH3D, TETMESH, and Cobalt grid files. For a case, <case>, that uses the Cobalt file format generated by GRIDGEN, the files in the “Grid1” directory will include:

- <case>-COBALT.bc
- <case>-COBALT.inp
- <case>.cgi – patch-mapping file, which defines the mapping between the patches defined in the grid files and those specified in Rocflu input files.
- <case>.dba
- <case>.gg
- <case>_grid.grd

3.2 Fluid and Solid Coupling

The wind turbine case detailed in Section 5, which couples the new fluid solver rocfoam and the structural solver *CalculiX*, is used as an example.



```
<case>/  
  system/  
    controlDict  
    fvSchemes  
    fvSolution  
    blockMeshDict  
  constant/  
    xProperties  
    polyMesh/  
      boundary  
      faces  
      neighbor  
      owner  
      points  
  ROCFOAM/  
  0/
```

Figure 5: Initial *RocFOAM* file hierarchy. Running the case will spawn additional time directories, each containing copies of the time dependent properties.

3.2.1 Writing *RocFOAM* Input Files

The *RocFOAM* fluid solver is based on *rhoPimpleFoam* and follows the *OpenFOAM* input file structure, Figure 5. For a general guide on setting up *OpenFOAM* simulations, see the *OpenFOAM* user guide⁶. Inside the case file, the “constant” directory contains the full description of the mesh in the *polyMesh* subdirectory as well as files specifying physical properties such as “*transportProperties*,” “*thermophysicalProperties*,” and “*turbulenceProperties*.” The “*system*” directory contains *controlDict*, *fvSchemes*, and *fvSolution*, as well as other dict files that set parameters associated with the solution procedure. The time directories contain the field data at specified points in time. Simulations typically start with a “0” directory, which contains the initial conditions. Running the simulation will result in the creation of additional time directories, each containing the field values for that time.

An example of the format used in *OpenFOAM* input files is shown in Listing 8. *OpenFOAM* uses a vector made up of seven scalars delimited by square brackets to ensure that dimensions are kept consistent, and that meaningless operations (such as attempting to subtract a unit of time from a unit of length) are prevented. The base units are shown in Table 2.

Listing 8: Example *controlDict* file.

```
1 /*-----* C++ -----*/  
2 | ======  
3 | \\ / Field | OpenFOAM: The Open Source CFD Toolbox |  
4 | \\ / Operation | Version: 5 |  
5 | \\ / And | Web: www.OpenFOAM.org |  
6 | \\ / Manipulation |  
7 */  
8 FoamFile  
9 {
```

⁶<https://www.openfoam.com/documentation/user-guide/>



```
10    version      2.0;
11    format       ascii;
12    class        dictionary;
13    location     "system";
14    object       controlDict;
15 }
16 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
17 application    simpleFoam;
18 startFrom      startTime;
19 startTime       0;
20 stopAt         endTime;
21 endTime         100000;
22 deltaT         0.00001;
23 writeControl   adjustableRunTime;
24 writeInterval   0.001;
25 purgeWrite     0;
26 writeFormat    binary;
27 writePrecision 6;
28 writeCompression off;
29 timeFormat     general;
30 timePrecision  6;
31 runTimeModifiable true;
32 adjustTimeStep yes;
33 maxCo 0.9;
34 // *****
```

Table 2: Base units for SI and USCS.

No.	Property	SI Unit	USCS Unit
1	Mass	Kilogram [kg]	Pound-mass [lbm]
2	Length	Meter [m]	Foot [ft]
3	Time	Second [s]	Second [s]
4	Temperature	Kelvin [K]	Degree Rankine [$^{\circ}$ R]
5	Quantity	Kilogram-mole [kgmol]	Pound-mole [lbmol]
6	Current	Ampere [A]	Ampere [A]
7	Luminous intensity	Candela [cd]	Candela [cd]

Dimensioned types are thus specified within an input file as

```
1 nu [0 2 -1 0 0 0] 1;
```

where `nu` is the keyword, `[0 2 -1 0 0 0]` is the `dimensionSet`, and `1` is the scalar value. This entry represents $v = 1m^2/s$.

3.2.2 Writing *CalculiX* Input Files

A brief introduction to composing *CalculiX* input files (called input “decks”) is provided here. The input deck consists of keywords (or keyword “cards”) followed by the data required to fully specify



that keyword. For a general guide on keywords and setting up simulations, see the user guide⁷. Although the example deck in Listing 9 strategically uses upper- and lower-case letters, this is done for readability – *CalculiX* input is case insensitive.

Listing 9: Example *CalculiX* input file, named turbine_metric.inp.

```
1 **
2 ** Structure: 1/3 wind turbine FSI model
3 ** Test objective: test FSI capabilities of the CSC module
4 **
5 *HEADING
6 Model: WINDTURBINE Date: 17/06/2020
7 *INCLUDE, INPUT=hollow_solid_mesh.inp
8 *BOUNDARY
9 hub_interior,1,3
10 *MATERIAL,NAME=GFRP
11 *ELASTIC
12 25E9,.38
13 *DENSITY
14 1.90E3
15 *SOLID SECTION,MATERIAL=GFRP,ELSET=Solid
16 *STEP,INC=100000,NLGEOM
17 *DYNAMIC,DIRECT
18 *NODE PRINT,NSET=ALLNODES,FREQUENCY=1
19 U
20 *EL PRINT,ELSET=Solid,FREQUENCY=1
21 S
22 *NODE FILE
23 U, RF
24 *EL FILE
25 E, S
26 *END STEP
```

The ***HEADING** card provides basic information about the problem and is reproduced at the head of the output file. The ***INCLUDE** card provides a way to include part of the input deck in a separate file—in this case, the mesh input file. The ***BOUNDARY** card is used to prescribe boundary conditions. For solids, the degrees of freedom provided in Table 3 are allowed.

Table 3: Degrees of freedom for the ***BOUNDARY** card in solids for *CalculiX*.

Key	Description
1	Translation in the local x-direction
2	Translation in the local y-direction
3	Translation in the local z-direction
4	Rotation about the local x-axis
5	Rotation about the local y-axis
6	Rotation about the local z-axis
11	Temperature

⁷http://web.mit.edu/calculix_v2.7/CalculiX/ccx_2.7/doc/ccx/index.html



The ***MATERIAL** card indicates the start of a material definition. The material definition block in Listing 9 extends from Line 10 to Line 14, and specifies the name, elastic properties (modulus and Poisson's ratio in Line 12), and density of the material. Note that no units are included; the onus of maintaining a consistent system of units falls on the user.

The ***SOLID SECTION** card assigns material properties to 3D, plane stress, plane strain, axisymmetric element sets. “MATERIAL” and “ELSET” are required parameters. The ***STEP** card indicates the start of a new step. The optional parameters “INC” and “NLGEOM” are given to specify the maximum number of iterations and signal that geometrically nonlinear effects should be considered in the calculations.

The ***DYNAMIC** card specifies the procedure that calculates the response of a structure subjected to dynamic loading. The “DIRECT” parameter is added to indicate that the user-defined initial time increment should not be changed—if convergence does not occur, the calculation stops with an error message. The line after the ***DYNAMIC** keyword should include the user-specified timestep when the “DIRECT” parameter is used.

The ***NODE PRINT** card prints nodal variables into a .dat file named after the job. The “NSET” parameter specifies which nodes’ values should be printed, and the “FREQUENCY” parameter specifies how often the node data will be saved—in this case, FREQUENCY=1, which means that the results of every increment will be saved. Line 19 provides the nodal variable that will be saved. For solids, the variables in Table 4 may be selected.

Table 4: Nodal variables that may be saved for solids in a *CalculiX* input file.

Key	Description
U	Displacements
NT or TS	Structural temperatures
RF	External forces
RFL	External concentrated heat source

The ***EL PRINT** card prints element variables into a .dat file named after the job. The “ELSET” parameter is used to specify which elements values should be printed. The “FREQUENCY” parameter specifies how often the data will be saved. The element variables provided in Table 5 are available.

Table 5: Element variables that may be saved for solids in a *CalculiX* input file.

Key	Description
S	Cauchy stress
E	Total Lagrangian strain for (hyper)elastic materials
ME	Mechanical Lagrangian strain for (hyper)elastic materials
PEEQ	Equivalent plastic strain
ENER	Energy density
SDV	Internal state variables
ELSE	Internal energy
ELKE	Kinetic energy
EVOL	Volume

***NODE FILE** and ***EL FILE** are used to print nodal and element variables to an .exo output file named after the job for subsequent viewing. The variable names are the same as given in Table 4 and 5.

Finally, the ***END STEP** card concludes the definition of a step.



4 Case Study: Attitude Control Motor Rocket

The Attitude Control Motor (ACM) *Rocflu* test will be used as an example. The data for this test can be found in the `/testing/share/Testing/test_data/ACM_Rocflu/` directory. The geometry of the case is pictured in Figure 6.

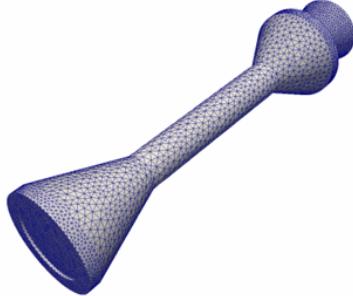


Figure 6: Attitude Control Model for the *Rocflu* test example.

4.1 Problem Setup

The ACM is a 2 inch motor provided by the Atlantic Research Corporation. The propellant grain has an angled approach and exit. The case is a titanium alloy and the nozzle is a phenolic composite. The motor burns out in approximately 12 ms. The volumetric and surface grids are provided in Figure 7. Simulation parameters are given in Table 6.

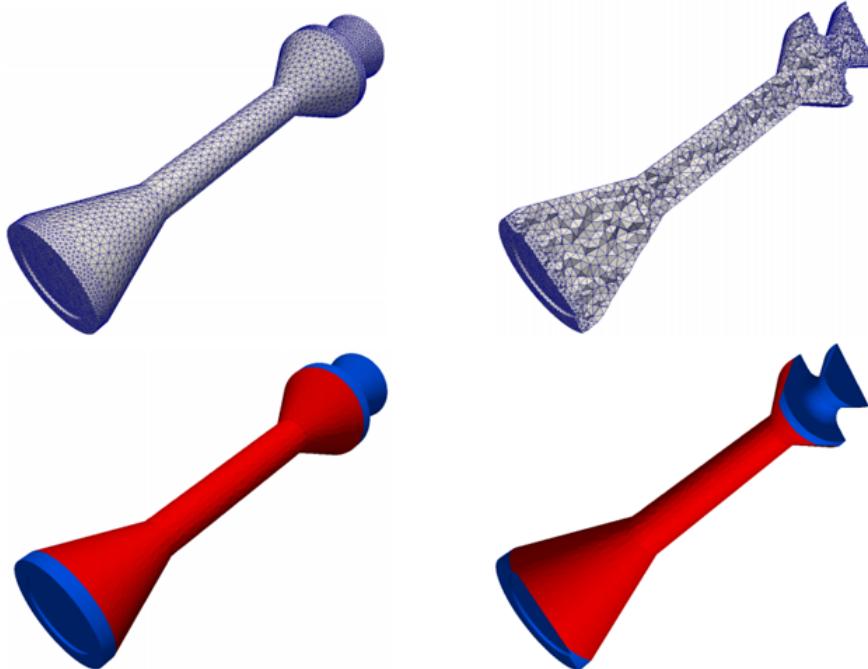


Figure 7: (TOP) ACM volumetric grid. Crinkle-cut highlights cells are tetrahedra. (BOTTOM) ACM surface grid. Burning surfaces are red, non-interacting faces are shown in blue.



Table 6: Simulation parameters used for the ACM model.

Simulation Parameter	Symbol	Parameter	Value
Physical Parameters	C_p	Fluid heat capacity (constant pressure)	1905.0 J kg ⁻¹ K ⁻¹
	γ	Fluid heat capacity ratio	1.2444
	r_b	Burn rate	aP^n
	a	a in APN model	0.77
	n	n in APN model	0.62
	T_{flame}	Adiabatic flame temperature	2916.0 K
Initial Conditions	ρ	Propellant density	1703.0 kg m ⁻³
	P	Initial fluid pressure	1.0×10^5 Pa
	ρ	Initial fluid density	1.16 kg m ⁻³
Boundary Conditions	T_{init}	Initial fluid temperature	298.0 K
	m_{inj}	Injection mass flow	5.7429 kg m ⁻² s ⁻¹
	T_{inj}	Injection temperature	2855 K

4.2 Input Files

The ACM case uses *Rocflu* and *Rocburn* with the NDA file hierarchy shown in Figure 8. The control files for *Rocstar*, *Rocflu*, and *RocburnAPN*, as well as the input and boundary condition input files for *Rocflu*, are reproduced here.

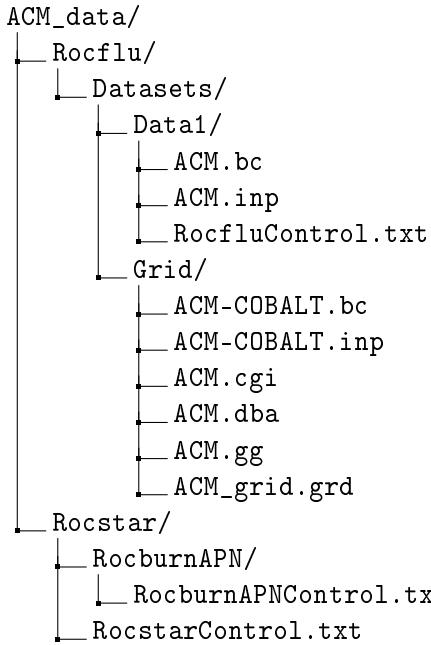


Figure 8: NDA File directory for ACM_data. Files for deprecated modules not included.

4.2.1 Control Files

Rocstar uses a text file located within the *Rocstar/* directory to specify FSI simulation parameters. Listing 10 shows an example *RocstarControl.txt* that specifies a fluid simulation with burning, using *Rocflu* for the fluid module and *RocburnAPN* for the burn module. Output files (in CGNS



format) will be written to a Rocout/ directory in the run directory. The maximum physical problem time is set to 2.0 milliseconds, and the timestep is set to 0.1 milliseconds. The output interval is also set to 0.1 milliseconds, meaning that output will be dumped for every timestep. The zoom factor, which is a means of accelerating the slowest time scale in rocket problems, is set to 1.0 to indicate normal burn-back with no acceleration. The simulation will continue until it is finished, or until the maximum wall clock time of 86,300 seconds (approximately 24 hours) has been reached.

Listing 10: Example RocstarControl.txt file.

```
1 CouplingScheme = FluidBurnAlone
2 FluidModule = Rocflu
3 BurnModule = RocburnAPN
4 OutputModule = Rocout
5 MaximumTime = 2.0e-6
6 ZoomFactor = 1.0
7 AutoRestart = F
8 CurrentTimeStep = 1.0e-07
9 OutputIntervalTime = 1.0E-07
10 MaxWallTime = 86300
11 ProfileDir = Rocman/Profiles
```

The RocfluControl.txt file, Listing 11, contains the case name (ACM), sets the relative paths to the Modin, Modout, and Rocin directories, and sets the output verbosity and level of runtime checks to low.

Listing 11: Example RocfluControl.txt file.

```
1 ACM
2 Rocflu/Modin
3 Rocflu/Modout
4 Rocflu/Rocin
5 1
6 1
```

The RocburnAPNControl.txt file, Listing 12, sets the burn rate model to $rb = 0.77P^{0.62}$, sets the maximum number of spatial nodes to 1, and sets the adiabatic flame temperature to 2916 K and the initial temperature to 298 K.

Listing 12: Example RocburnAPNControl.txt file.

```
1 0.77      !a in rb = a * Pn, rb in cm/sec and P in atm, a_p (cm/sec) !
2 0.62      !n in rb = a * Pn, rb in cm/sec and P in atm, n_p!
3 1         !Maximum_number_of_spatial_nodes, nxmax!
4 2916.0    !Adiabatic flame temperature, Tf_adiabatic (K) !
5 298.00    !Initial temperature , To_read (K) !
6 Rocburn_2D_Output/Rocburn_APN !Location of the Rocout files!
7
8 !Solid Propellant Properties for ARC Attitude Control Motor (ACM) per!
9 !AIAA UQ paper by Brandyberry from Summer 2006.! 
```

Note that *RocburnAPN* stops reading a line when it encounters the first non-number character.



The `RocmanControl.txt` file, Listing 13 sets the verbosity to silent, requests first order interpolation to compute interface quantities, and specifies that no shear forces should be transferred from the fluid to a solid (since this case does not involve a solid). The ambient pressure, which will be subtracted from the fluid pressure in computing the load, is set to 100 kPa. The RFC_ parameters are specific to data transfer; here they are set to a low level of verbosity, second order quadrature, a maximum of 100 iterations, and a tolerance of 1e-6. Face-offsetting is set to true, which is the recommended setting.

Listing 13: Example `RocmanControl.txt` file.

```
1 Verbose = 0           ! # Rocman verbosity!
2 InterpolationOrder = 1 ! # Order of interpolation!
3 TractionMode = 1     ! # 1 for no shear, 2 for shear!
4 P_ambient = 1.0E+05   ! # Ambient pressure!
5 Rhoc = 1703.0         ! # Solid density for fluid-alone mode!
6 Pressure = 6.8d6      ! # Pressure for solid-alone mode!
7 BurnRate = 0.01        ! # Burn-rate for solid-alone mode!
8 RFC_verb = 1          ! # Verbosity level of data transfer!
9 RFC_order = 2          ! # Order of quadrature rules!
10 RFC_iteration = 100    ! # Maximum number of iterations!
11 RFC_tolerance = 1.e-6  ! # Tolerance for iterative solver!
12 Face-offsetting = T    ! # Whether to enable face-offsetting (T/F) !
13 AsyncInput = F         ! # Whether to use asynchronous input (T/F) !
14 AsyncOutput = F        ! # Whether to use asynchronous output (T/F) !
```

4.2.2 Data Files

Several of the *Rocflu* input files used in the ACM case are reproduced in Listings 14 and 15. For more detailed information regarding these simulation options, please consult the legacy *Rocflu* documentation on the Illinois Rocstar GitHub page ([Rocstar-legacy/Docs/legacy/UG/rocflumpbook.pdf](#)).

Listing 14: `ACM.inn` file.

```
1 # FORMATS
2 GRID      0           ! 0 - ROCFLU ASCII, 1 - ROCFLU binary, 2 - ROCFLU HDF !
3 SOLUTION   0           ! 0 - ROCFLU ASCII, 1 - ROCFLU binary, 2 - ROCFLU HDF !
4 GRIDSRC   4           ! 0 - CENTAUR ASCII, 1 - VGRIDNS, 2 - MESH3D !
5 #
6
7 # FLOWMODEL
8 MODEL      0           ! 0 - Euler, 1 - Navier-Stokes !
9 MOVEGRID   1           ! 0 - static grid, 1 - moving grid !
10 #
11
12 # NUMERICS
13 CFL       1.0          ! CFL number!
14 DISCR     3           ! Type of space discretization (1 - Roe, 2 - MAPS) !
15 ORDER     2           ! Order of accuracy (1 - first, 2 - second) !
16 ENTROPY   0.05         ! Entropy correction coefficient (if DISCR=1) !
```



```
17 #
18
19 # TIMESTEP
20 FLOWTYPE    1      ! 0 - steady flow, 1 - unsteady flow !
21 TIMESTEP    0.000001 ! Max. physical time step !
22 STARTTIME   0.0      ! Current iteration !
23 MAXTIME    0.2      ! Maximum number of iterations !
24 WRITIME    0.001    ! Offset between iterations to store solutions !
25 PRNTIME    0.00000005 ! Offset between iterations to print convergence !
26 DTMINLIMIT 5.0E-09 ! Timestep below which debug info will be printed !
27 #
28
29 # GRIDMOTION
30 TYPE        3      ! 1-smooth boundary disp., 2-smooth coordinates, 3-MESQUITE !
31 NITER       10     ! Number of smoothing iterations (for TYPE 1 or TYPE 2 only) !
32 SFACT       0.25   ! Smoothing coefficient (for TYPE 1 or TYPE 2 only) !
33 #
34
35 # REFERENCE
36 GAMMA      1.2444 ! Ratio of specific heats !
37 CP          1905.0 ! Specific heat coefficient at constant pressure [J/kg K] !
38 #
39
40 # PROBE
41 NUMBER      2      ! Number of probes !
42 0.001 0.0 0.0      ! x y z coordinates of probe 1 !
43 0.0495735 0.0 0.0 ! x y z coordinates of probe 2 !
44 #
45 WRITIME    0.000005 ! Offset in [s] at which data is written to probe files !
46 OPENCLOSE   1      ! Close and open probe file after writing? (0-no, 1-yes)
!
47 #
48
49 # INITFLOW
50 FLAG        1      ! Generate init. solution with: 1-keywords, 2-file, 3-hardcode
!
51 DENS        1.16   ! Density of initial solution in kg/m3 (FLAG 1 only) !
52 VELX        0.0     ! x-component velocity of initial solution in m/s (FLAG 1 only)
!
53 VELY        0.0     ! y-component velocity of initial solution in m/s (FLAG 1 only)
!
54 VELZ        0.0     ! z-component velocity of initial solution in m/s (FLAG 1 only)
!
55 PRESS       1.0E+5 ! Static pressure of initial solution in Pa (FLAG 1 only) !
56 #
57
58 # TRANSFORM
59 FLAG        1      ! Scale or rotate grid? (0-no, 1-yes) !
60 SCALE_X     0.0254 ! Scaling factor for x-component of coordinates !
61 SCALE_Y     0.0254 ! Scaling factor for y-component of coordinates !
```



```
62 SCALE_Z      0.0254    ! Scaling factor for z-component of coordinates !
63 #
64
65 # PREP
66 SURFFLAG     1          ! Set to 1 if rfluprep was not compiled as part of GENx
67 !
68
69 # POST
70 PLTTYPE       1          ! 1-Write only grid to output file, 2-Write grid and solution
71 !
72 PLTVOLFLAG   0          ! Should volume data be written to output file (0-no, 1-yes) !
73 MERGEFLAG    0          ! Merge regions from parallel computation? (0-no, 1-yes) !
74 #
75 # ROCKET
76 CASERAD      .0066167  ! Cylindrical case constraint radius !
77 #
78
79 # TIMEZOOMING
80 MAXPLANE     0.047625  ! Max coordinate to apply zooming !
81 MINPLANE     0.00       ! Min coordinate to apply zooming !
82 #
```

Listing 15: ACM.inn file.

```
1 # BC_INJECT
2 PATCH  1 1      ! Range of patches this section applies to (from 1 to 1) !
3 NAME    InjectionWall ! Name of boundary !
4 MFRATE  5.7429 ! Injection mass flux in kg/m2s !
5 TEMP    2855.0 ! Injection static temperature in K !
6 RFVFU   0.0
7 RFVFV   0.0
8 RFVFW   0.0
9 COUPLED 1        ! Is the patch interacting? (1-yes, 2-no) !
10 BFLAG    1        ! Is the patch burning at t = 0? (0-no, 1-yes) !
11 MOVEDIR 7        ! Vertices on patch may move in xyz-space !
12 #
13
14 # BC_SLIPW
15 PATCH  2 2      ! Range of patches this section applies to !
16 NAME    AftFlatWall ! Name of boundary !
17 COUPLED 2        ! Is the patch interacting? (1-yes, 2-no) !
18 MOVEDIR 0        ! Vertices on patch may not move in any direction !
19 #
20
21 # BC_SLIPW
22 PATCH  3 3      ! Range of patches this section applies to !
23 NAME    HeadEndSurface ! Name of boundary !
```



```
24 COUPLED 2           ! Is the patch interacting? (1-yes, 2-no) !
25 MOVEDIR 0          ! Vertices on patch may not move in any direction !
26 #
27
28 # BC_OUTFLOW
29 PATCH   4 4      ! Range of patches this section applies to !
30 NAME     NozzleOutlet ! Name of boundary !
31 TYPE     0        ! Specify outflow type (0-supersonic, 1-subsonic, 2-mixed) !
32 COUPLED 2           ! Is the patch interacting? (1-yes, 2-no) !
33 MVPATCH 0          ! Is the patch moving? (0-no, 1-yes) !
34 SMGRID   0        ! Should the grid on the patch be smoothed? (0-no, 1-yes) !
35 CORR     0        ! Should point-vortex correction be applied? (0-no, 1-yes) !
36 MOVEDIR 0          ! Vertices on patch may not move in any direction !
37 #
38
39 # BC_SLIPW
40 PATCH   5 5      ! Range of patches this section applies to !
41 NAME     HeadEndRing ! Name of boundary !
42 COUPLED 2           ! Is the patch interacting? (1-yes, 2-no) !
43 MOVEDIR 0          ! Vertices on patch may not move in any direction !
44 #
45
46 # BC_SLIPW
47 PATCH   6 6      ! Range of patches this section applies to !
48 NAME     AftEndRing ! Name of boundary !
49 COUPLED 2           ! Is the patch interacting? (1-yes, 2-no) !
50 MOVEDIR 0          ! Vertices on patch may not move in any direction !
51 #
52
53 # BC_SLIPW
54 PATCH   7 7      ! Range of patches this section applies to !
55 NAME     NozzleSurface ! Name of boundary !
56 COUPLED 2           ! Is the patch interacting? (1-yes, 2-no) !
57 MOVEDIR 0          ! Vertices on patch may not move in any direction !
58 #
59
60 # END
```

To run the ACM case using the files provided in the testing directory included in the *Rocstar* distribution, first create an input directory within the build directory:

```
$ mkdir ACM_RocfluTest
```

Copy the folders within the ACM_Rocflu directory into the newly created ACM_RocfluTest directory:

```
$ cp -r ${sourceDir}/testing/share/Testing/tst_data/ACM_Rocflu/*.
```

Run *Rocprep* with the following flags:



```
$ ${buildDir}/bin/rocprep -A -b -m -u 1 1 -n 4 -d ./ACM_data -t ./ACM_4
```

Refer to Table 1 for more information on *Rocprep* flags.

Change the working directory to ACM_4, which should have been created by the previous command:

```
$ cd ACM_4
```

Run *Rocstar*:

```
$ mpirun -np 4 ${buildDir}/bin/Rocstar
```

This will generate output files, detailed in the following section.

4.3 ACM Output Files

4.3.1 *Rocflu* Output

Rocflu generates flow solution files to the “*Rocflu/Rocout*” directory. CGNS output files are formatted as “*file_type_xx.yyyyyy_zzzz.cgns*.” The time stamp is given in the *Rocstar* convention of “*xx.yyyyyy*,” where the simulation time is “*0.yyyyyy x 10xx x 10-9 s*.” For example, at a simulation time of $t = 1.2 \times 10^{-3}$ s, *Rocstar* generates the output files with a corresponding time stamp of “07.120000.” The process number is contained in the string “*zzzz*” and is “zero-indexed.” The .txt files below are formatted “*file_type_xx.yyyyyy.txt*,” following the same convention as the CGNS files.

- **fluid_xx.yyyyyy_zzzz.cgns**: *Rocflu* volumetric fluid output file. Contains information about the volumetric mesh and all associated field data.
- **ifluid_b_xx.yyyyyy_zzzz.cgns**: *Rocflu* burning surface fluid output file. Contains information about the surface mesh and all associated field data.
- **ifluid_nb_xx.yyyyyy_zzzz.cgns**: *Rocflu* interacting but non-burning surface fluid output file. Contains information about the surface mesh and all associated field data.
- **ifluid_ni_xx.yyyyyy_zzzz.cgns**: *Rocflu* non-interacting surface fluid output file. Contains information about the surface mesh and all associated field data.
- **fluid_in_xx.yyyyyy.txt**: *Rocflu* file that tells *Rocin* which panes belong to each process and the fluid filenames associated with them.
- **ifluid_in_xx.yyyyyy.txt**: *Rocflu* file that tells *Rocin* which panes belong to each process and the ifluid filenames associated with them.

If one or more probes is specified in the *Rocflu* input file’s #PROBE section, the output files will be written to files with names such as “*<case>.prb_mmmmm*,” where “*mmmmmm*” is the number of the probe. Each line of the probe file consists of seven columns:

- Column 1: iteration number (steady flow) or time (unsteady flow)
- Column 2: density
- Column 3: x-velocity component
- Column 4: y-velocity component



- Column 5: z-velocity component
- Column 6: static pressure
- Column 7: static pressure

4.3.2 Rocburn Output

Rocburn outputs CGNS files (`burn*.cgns`, `iburn_all*.cgns`) and files with processor distribution information (`burn_in*.txt`, `iburn_in*.txt`) to the `RocburnAPN/Rocout` directory.

- `burn_xx.yyyyyy_zzzz.cgns`: *RocburnAPN* burning surface fluid output file. Contains information about the surface mesh.
- `iburn_all_xx.yyyyyy_zzzz.cgns`: *RocburnAPN* burning surface fluid output file. Contains information about the surface mesh and all associated field data
- `burn_in_xx.yyyyyy.txt`: *RocburnAPN* file that tells `Rocin` which panes belong to each process, as well as the burn filenames associated with them.
- `iburn_in_xx.yyyyyy.txt`: *RocburnAPN* file that tells `Rocin` which panes belong to each process, as well as the iburn filenames associated with them.

5 Case Study: Wind Turbine

This case study demonstrates the application of *Rocstar* to high fidelity modeling and simulation of FSI phenomena in a wind turbine. We treat the problem in coupled time-domain mode while resolving critical structural characteristics of composite blades. The focus of our modeling effort is on resolving the critical features of the blade in response to a free stream with a limited element count for the computational mesh so that the solver and coupling scheme can be tested hastily.

5.1 Problem Setup

The blade is a 5.532 m long NREL S809 tapered and twisted airfoil⁸. The trailing edge of the blade is slightly blunted to improve the mesh quality. The thickness of the blades is selected to be 7 mm. The hub section is arbitrarily modeled, and the blade-to-hub connector section is generated by lofting an initially cylindrical geometry to the first blade area. Since only one non-rotating blade is being modeled, the fluid domain is selected to be a pie-shaped section, as shown in Figure 9 (left). For all simulations, the blade is assumed to be in a parked (locked) position subject to flow normal to the rotation axis. For some of the simulations, a 7 mm thick shear web is also introduced centered at 30% chord spanning the entire blade length, as shown in Figure 9 (right).

5.1.1 Mesh

Solid and fluid regions are discretized with a matching mesh at the FSI interface. The surface mesh is limited to triangular elements. The sizes of surface mesh elements are adapted to the geometric features of the blade. This results in maximum surface element size to be 40 mm in length with

⁸M.M. Hand, D.A. Simms, L.J. Fingersh, D.W. Jager, J.R. Cotrell, S. Schreck, S.M. Larwood. 2001. "Unsteady Aerodynamics Experiment Phase VI: Wind Tunnel Test Configurations and Available Data Campaigns," Report Number TP-500-29955, National Renewable Energy Laboratory.

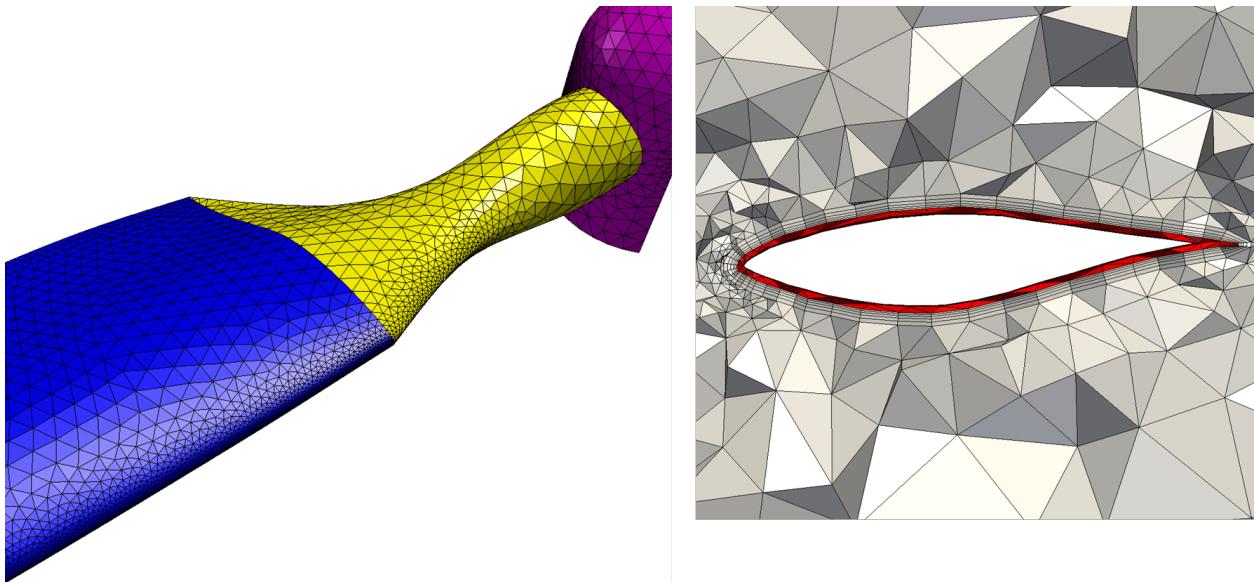


Figure 9: (Left) 3D model of single S809 profile wind turbine blade. The fluid region is the transparent 120° pie-shaped domain. The active FSI zone of is shaded in blue, and the connector (blade-to-hub) surface is shaded in yellow. The hub surface is shaded in magenta. (Right) Transparent view of solid section of the blade with shear webs.

the deviation angle of 11 degrees and size gradation of 1.3. The generated surface mesh is shown in Figure 10. The computational volume is subsequently discretized using tetrahedral elements for both the solid and fluid regions. For the fluid domain, boundary layer meshes are obtained by extruding surface elements from the blade walls, as shown in the inset in Figure 10.

5.1.2 Materials

The properties of glass fiber polymer composites (GFRP) and carbon fiber polymer composites (CFRP) are given in Table 7. For the sake of simplicity, the modulus of elasticity is averaged over the lengthwise and crosswise directions, and homogenized values for Poisson's ratios and densities for typical CFRP and GFRP composites are used in the simulations. Furthermore, materials are assumed to be linear and isotropic.

Table 7: Properties of composite materials used in the simulations.

Material	E [GPa]	ν	ρ [kg/m ³]
GFRP	25.0	0.38	1,900
GFRP	50.0	0.30	1,600

GFRP are traditionally used as a structural material for wind turbine blades, but CFRP are a promising alternative since they allow for a stiffer and lighter blade.

5.1.3 Boundary Conditions

Fluid domain boundary tags are shown in Figure 9, where the “Front” boundary is the inlet, and the “Back” boundary is the outlet. Fluid domain boundary conditions are set up for uniform flow in the direction of the wind turbine (along rotation axis) with a prescribed 10 m/s and 61 m/s flow

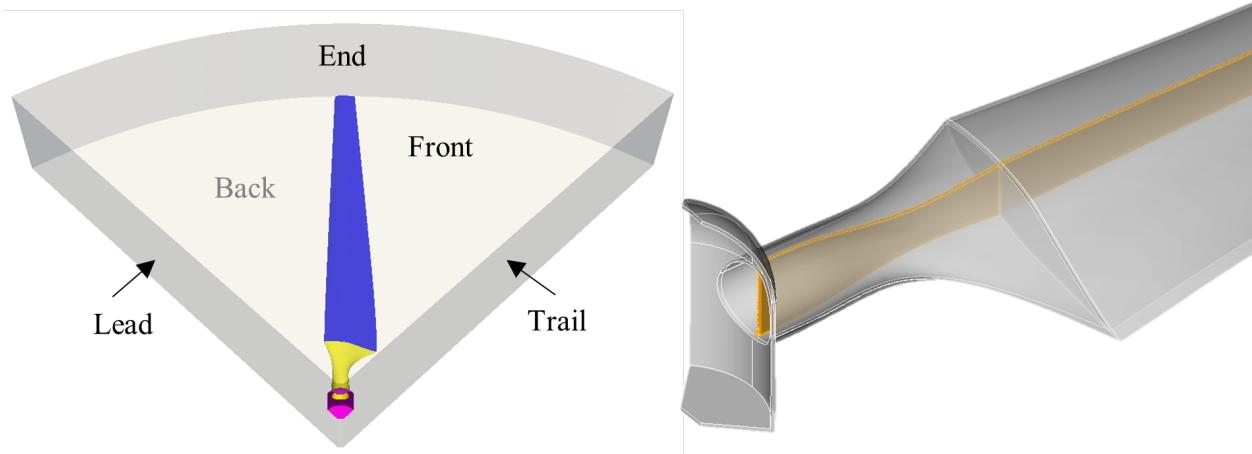


Figure 10: Surface mesh of the hub, blade and connector sections. The inset shows a crinkled slice of the volume mesh highlighting fluid (white) and solid (red) sections.

velocity for the normal and harsh conditions, respectively. All wall boundaries, such as the blade FSI surface, hub-to-blade connection, and hub, are set to no-slip conditions. The pressure is given a zero-gradient condition on all surfaces with a reference pressure of 1 atm. The inlet temperature is set to 300K. Table 8 lists details of boundary conditions applied to each surface. Wall functions are used at all no-slip walls.

Table 8: List of fluid solver boundary conditions.

Quantity	Blade	Front	Back	End	Lead	Trail
U [m/s]	no-slip	(0 10 0)	zeroGrad	zeroGrad	zeroGrad	zeroGrad
p [Pa]	zeroGrad	zeroGrad	zeroGrad	zeroGrad	zeroGrad	zeroGrad
T [K]	zeroGrad	300	zeroGrad	zeroGrad	zeroGrad	zeroGrad
k [m^2/s^2]	kqRWallFunc	0.1	zeroGrad	zeroGrad	zeroGrad	zeroGrad
ϵ [m^2/s^3]	epsilonWallFunc	200	zeroGrad	zeroGrad	zeroGrad	zeroGrad
ν_T [m^2/s]	nutkWallFunc	0	calculated	calculated	calculated	calculated
α_T [kg/m s]	alphatWallFunc	0.0001	calculated	calculated	calculated	calculated

5.1.4 Loads

Wind loads for atmospheric conditions are computed by the flow solver and applied to the structure. The normal component of the traction vector (pressure) is computed and passed to the structural solver as distributed loads applied to the center of the element. The distribution of pressures applied to the structure under these conditions is shown in Figure 11.

5.2 Input Files

This simulation was run with the fluid solver *RocFOAM* and the structural solver *CalculiX*.

5.2.1 Fluid Input Files

RocFOAM is based on the *OpenFOAM* solver and uses a similar file structure, Figure 12.

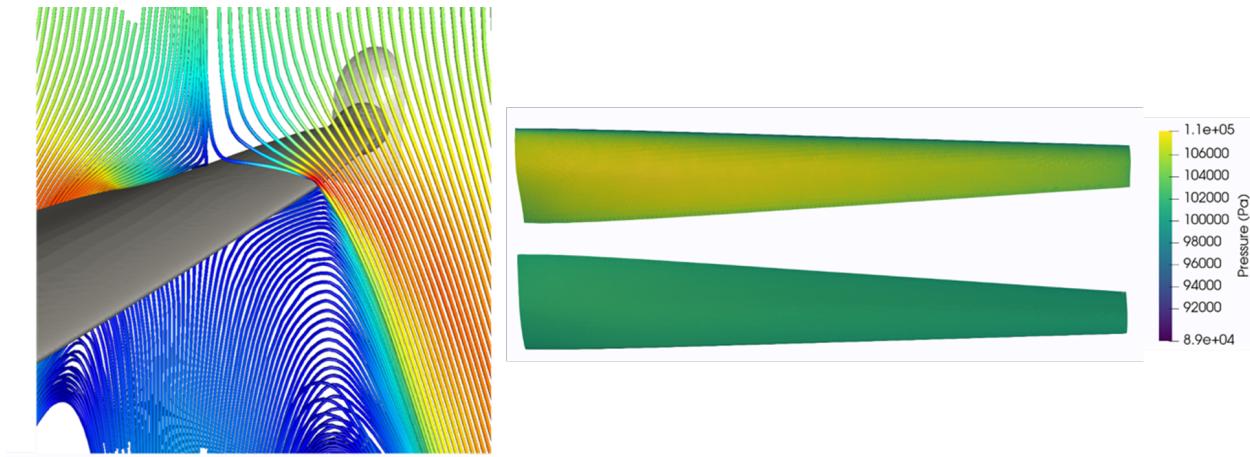


Figure 11: (Left) Flow streamlines and (Right) distribution of wind pressures for normal atmospheric conditions on the forward and backward surfaces of the blade.

A selection of the files listed in Figure 12 are explained in Listing 16 below.

Listing 16: The file `system/fvSchemes` that define the finite volume solution methods.



```
<case>
  FLUID/
    system/
      controlDict
      fvSchemes
      fvSolution
      blockMeshDict
      decomposeParDict
      snappyHexMeshDict
      surfaceFeatureExtractDict
    constant/
      dynamicMeshDict
      thermophysicalProperties
      transportProperties
      turbulenceProperties
      polyMesh/
        boundary
        faces
        neighbor
        owner
        points
    0/
      alphat
      epsilon
      k
      ...
    ...
```

Figure 12: *RocFOAM* file structure. The system directory contains the dictionaries that specify solving methods, the constant directory contains the constant physical properties, and the initial time directory 0 contains the properties that evolve with time (truncated here to fit). As the problem runs, more directories are added with the updated properties for the associated time.

```
29 //grad(p)           cellLimited Gauss linear 1;
30 //grad(h)           cellLimited Gauss linear 1;
31 //grad(k)           cellLimited Gauss linear 1;
32 //grad(omega)       cellLimited Gauss linear 1;
33 }
34 divSchemes
35 {
36   default            none;
37   div(phi,U)         Gauss linearUpwindV grad(U);
38   div(phi,h)         Gauss upwind;
39   div(phi,k)         Gauss upwind;
40   div(phi,K)         Gauss upwind;
41   div(phid,p)        Gauss upwind;
42   div(phi,epsilon)   Gauss upwind;
43   div(meshPhi,p)    Gauss upwind;
```

```

44     div((phi|interpolate(rho)),p) Gauss upwind;
45     div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
46     div((nuEff*dev2(T(grad(U))))) Gauss linear;
47 }
48 laplacianSchemes
49 {
50     default          Gauss linear limited corrected 0.5;
51 }
52 interpolationSchemes
53 {
54     default          linear;
55 }
56 snGradSchemes
57 {
58     default          limited corrected 0.5;
59 }
60 wallDist
61 {
62     method meshWave;
63 }
64 // ****

```

Each file contains a heading that gives some basic information, such as the version, format, class, location, and object.

This `fvSolution` file, Listings 17–24, contains a set of subdictionaries that specify the solvers, tolerances, and algorithms to be used. Each `solvers` subdictionary lists the variable or variables being solved, the solver, and the parameters that the solver uses.

Listing 17: The fvSolution FoamFile.

```
1 FoamFile
2 {
3     version 2.0;
4     format ascii;
5     class dictionary;
6     location "system";
7     object fvSolution;
8 }
```

Listing 18 Lines 12–20 below specify the method that will be used to solve for pressure, p . The generalized geometric-algebraic multi-grid (GAMG) solver will be used, which will iterate until the residual is below $1e-5$ or the ratio of current to initial residuals falls below 0.05. The GAMG solver will start with the user defined mesh and refine or coarsen (down to a minimum of 1000 cells) it in stages. Symmetric Gauss-Seidel is specified as the solver’s smoother.

Listing 18: The `fvSolution` `FoamFile`, continued.

```
10 solvers  
11 {  
12     p
```



```
13  {
14      solver GAMG;
15      tolerance 1e-5;
16      relTol 0.05;
17      smoother symGaussSeidel;
18      nCellsInCoarsestLevel 1000;
19      minIter 1;
20  }
```

The PIMPLE algorithm solves for pressure multiple times within a time step. Different settings will be used the final time pressure is solved, as specified within the `pFinal` subdictionary. The relative tolerance for the last solution is set to zero.

Listing 19: The `fvSolution` `FoamFile`, continued.

```
21  pFinal
22  {
23      $p;
24      relTol 0;
25  }
```

The solver requires an entry for a pressure corrector `pcorr`, which results from mesh motion as specified in the “`constant/dynamicMeshDict`” file. A `pcorrFinal` subdictionary is also specified with a relative tolerance of zero.

Listing 20: The `fvSolution` `FoamFile`, continued.

```
26  pcorr
27  {
28      $p;
29      tolerance 5e-4;
30      relTol 0.1;
31  }
32  pcorrFinal
33  {
34      $p;
35      tolerance 5e-4;
36      relTol 0;
37  }
```

The velocity potential `Phi` is set to be the same as `p`.

Listing 21: The `fvSolution` `FoamFile`, continued.

```
38  Phi
39  {
40      $p;
41  }
```

Solvers for density (`rho`), the velocity field (`U`), enthalpy (`h`), turbulent kinetic energy (`k`), and the turbulent kinetic energy dissipation rate (`epsilon`) are specified in a similar manner as pressure, using `smoothSolver` rather than `GAMG`. Minimum and maximum numbers of iterations are specified for some solvers. The `cellDisplacement` solver is specified here because the mesh is dynamic.

Listing 22: The `fvSolution` `FoamFile`, continued.

```
42   "(rho|U|k|epsilon)"
43 {
44     solver smoothSolver;
45     smoother symGaussSeidel;
46     tolerance 1e-06;
47     relTol 0.01;
48     minIter 1;
49 }
50 cellDisplacement
51 {
52   solver smoothSolver;
53   smoother symGaussSeidel;
54   tolerance 1e-06;
55   relTol 0.1;
56   minIter 1;
57   maxIter 30;
58 }
59 h
60 {
61   solver smoothSolver;
62   smoother symGaussSeidel;
63   tolerance 1e-06;
64   relTol 0.01;
65   minIter 1;
66 }
67 "(rho|U|h|k|epsilon)Final"
68 {
69   $U;
70   relTol 0;
71 }
72 cellDisplacementFinal
73 {
74   solver smoothSolver;
75   smoother symGaussSeidel;
76   tolerance 1e-06;
77   relTol 0.0;
78   minIter 1;
79   maxIter 30;
80 }
81 }
```

The `potentialFlow` solver solves for the velocity potential `Phi` to calculate the volumetric face-flux from which the velocity field `U` is obtained. The number of non-orthogonal correctors, which is the number of times the pressure equation is solved, is set to 10.

Listing 23: The `fvSolution` `FoamFile`, continued.

```
82 potentialFlow
```



```
83 {
84     nNonOrthogonalCorrectors 10;
85 }
```

The PIMPLE algorithm is a combination of the pressure-implicit-split-operator (PISO) and semi-implicit method for pressure linked-equations (SIMPLE). The number of outer correctors is set to one to indicate that the entire system of equations will be solved only once within a time step. Similarly, the number of correctors is set to one, indicating that the pressure equation and momentum corrector will also only be solved once per time step. The number of non-orthogonal correctors (described above in the previous code block) is set to three. The `momentumPredictor` switch is set to on, indicating that each step will begin by solving the momentum equation.

Listing 24: The `fvSolution` `FoamFile`, continued.

```
86 PIMPLE
87 {
88     nOuterCorrectors 1;
89     nCorrectors 1;
90     nNonOrthogonalCorrectors 3;
91     momentumPredictor yes;
92     transonic no;
93     consistent no;
94     simpleRho no;
95     pMax 250000;
96     pMin 7000;
97     outerCorrectorResidualControl
98     {
99         "(U|k|epsilon|h)"
100        {
101            relTol 0.01;
102            tolerance 1e-6;
103        }
104        p
105        {
106            relTol 0.01;
107            tolerance 1e-6;
108        }
109    }
110    turbOnFinalIterOnly no;
111 }
112 SIMPLE
113 {
114     transonic no;
115     nNonOrthogonalCorrectors 3;
116     consistent no;
117     simpleRho no;
118     pMax 150000;
119     pMin 7000;
120     pRefCell 0;
```



```
121     pRefValue 101325;
122     residualControl
123     {
124         "(U|k|epsilon|p|h)" 1e-6;
125     }
126 }
127 relaxationFactors
128 {
129     fields
130     {
131         "p.*" 0.5;
132         "rho.*" 1.0;
133         //"U.*" 0.9;
134         //"h.*" 0.9;
135         //"(k|epsilon).*" 0.9;
136     }
137     equations
138     {
139         "U.*" 0.5;
140         "h.*" 0.5;
141         "(k|epsilon).*" 0.5;
142     }
143 }
144 // ****
```

5.2.2 Solid Input Files

For this case, there are two input files: “`turbine_metric.inp`,” which contains material properties and load definitions, and “`blade_v4.inp`,” which contains the mesh nodes. The “`turbine_metric.inp`” file is shown in Listing 25.

Listing 25: The `turbine_metric.inp` *CalculiX* input file.

```
1 **
2 ** Structure: 1/3 wind turbine FSI model
3 ** Test objective: test FSI capabilities of the CSC module
4 **
5 *HEADING
6 Model: WINDTURBINE Date: 08/11/2020
7 *INCLUDE, INPUT=blade_v4.inp
8 *BOUNDARY
9 hub_interior,1,3
10 *MATERIAL,NAME=GFRP
11 *ELASTIC
12 25E9,.38
13 *DENSITY
14 1.90E3
15 *SOLID SECTION,MATERIAL=GFRP,ELSET=Solid
16 *AMPLITUDE,NAME=A1
```



```
17 0.,0.,1.E-4,1.  
18 1.1E-4,0.  
19 *STEP,INC=100000,NLGEOM  
20 *DYNAMIC,DIRECT  
21 5e-5, 10, 1e-6, 1e-4  
22 *DLOAD  
23 FSI, P1, 0.0  
24 *NODE PRINT,NSET=ALLNODES,FREQUENCY=1  
25 U  
26 *EL PRINT,ELSET=Solid,FREQUENCY=1  
27 S  
28 *NODE FILE  
29 U, RF  
30 *EL FILE  
31 E, S  
32 *END STEP
```

Section 3.2.2 gives more detail on the keywords used in CalculiX input files.

5.2.3 Rocstar

The file directory structure for the wind turbine case is shown in Figure 13.

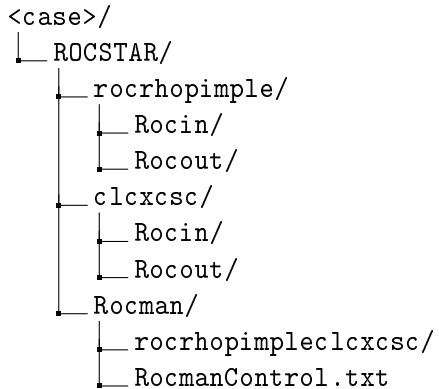


Figure 13: File directory for *Rocstar*. Note that the fluid and solid inputs have not been added to their respective *Rocin* directories.

5.3 Running a *Rocstar* Case

The process of running a *Rocstar* case involves five major steps:

1. Configuring the file structure
2. Preprocessing fluid input files
3. Preprocessing solid input files
4. Computing common refinement files



5. Executing *Rocstar*

For this case study, we assume that `$CASE_DIR_PATH = absolute/path/to/case/directory/`. We further assume that the system `PATH` contains the paths to the *RocFOAM*, *CalculiX-csc*, *surfdiver*, and *Rocstar* executables.

To use the commands in the following sections directly, it is recommended that the user execute the commands in Listing 42, adjusting the paths shown in red to the specified directories:

Listing 26: Commands for running *Rocstar* with *RocFOAM* and *CalculiX*.

```
$ export LD_LIBRARY_PATH=/path/to/IMPACT-install/lib/:${LD_LIBRARY_PATH}
$ export LD_LIBRARY_PATH=/path/to/rocfoam-install/lib/:${LD_LIBRARY_PATH}
$ export LD_LIBRARY_PATH=/path/to/calculix-csc-install/lib/:${LD_LIBRARY_PATH}
$ export CASE_DIR_PATH=/path/to/Rocstar-install/<case>
$ export PATH=/path/to/rocfoam-install/bin/:$PATH
$ export PATH=/path/to/calculix-csc-install/bin/:$PATH
$ export PATH=/path/to/IMPACT-install/bin/:$PATH
$ export PATH=/path/to/Rocstar-install/bin/:$PATH
```

5.3.1 Configuring the *Rocstar* File Structure

Currently, the onus is on the user to organize the required directories. Before beginning to preprocess, the user should construct the file hierarchy as shown in Figure 14.

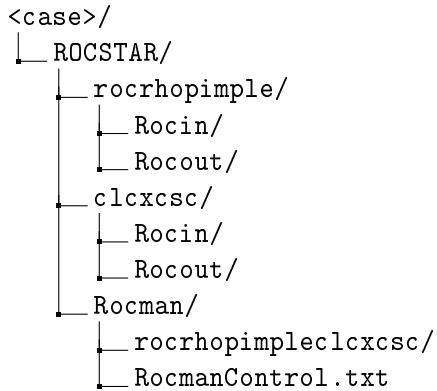


Figure 14: Full file hierarchy showing the directories and files that the user must manually create.

5.3.2 Preprocessing Fluid Input Files

This tutorial assumes the user has already generated the appropriate fluid meshes and variable fields shown in Figure 12. Before beginning, the user should copy their `system/`, `constant/`, and `0/` directories into the `FLUID` directory. If the user has not already set the *OpenFOAM* environment, that must be done first. On a default installation, this can be done with:

```
$ . /usr/lib/openfoam/openfoam2006/etc/bashrc
```

From within the `FLUID` directory, the user should execute:

```
$ decomposePar
```



This will decompose the fluid domain (mesh and fields) into the number of subdivisions specified by the user in `FLUID/system/decomposeParDict`. This domain will be divided into four parts, as shown in Listing 39.

Listing 27: The `system/decomposeParDict` file, which specifies the number and method for decomposing a domain for parallel runs.

```
1 FoamFile
2 {
3     version 2.0;
4     format ascii;
5     class dictionary;
6     location "system";
7     object decomposeParDict;
8 }
9 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
10 number0fSubdomains 4;
11 method scotch;
12 // *****
```

Running the `decomposePar` command will spawn as many new directories as was specified in the `number0fSubdomains` field. These directories, named `processor<n>` (where `<n>` goes from 0 to `number0fSubdomains-1`), each contain a `0/` and `constant/` directory, which consist of the data for each subdomain.

Warning

The number of desired subdomains given in `system/decomposeParDict` sets the number of processors that must be used for the remainder of the *Rocstar* run. Attempting to run with any other number of processors will result in an error.

Once the domain has been divided, the preprocessing command can be run:

```
$ mpirun -np 4 rocRhoPimple -preprocess -parallel
```

Note that this command assumes the paths to the *IMPACT* and *RocFOAM* libraries, as well as the path to the *RocFOAM bin* directory, have been exported. Without exporting those paths, the command resembles:

```
$ LD_LIBRARY_PATH=${HOME}/Rocstar/IMPACT-install/lib/:${HOME}/Rocstar/rocfoam-
install/lib:$LD_LIBRARY_PATH mpirun -np 4 ${HOME}/Rocstar/rocfoam-install/bin/
rocRhoPimple -preprocess -parallel
```

Executing the preprocessing command will create a new folder within the `FLUID` directory called “`ROCFOAM`.” This folder will contain generated volume and surface CGNS files named `ROCFOAMVOL_000<n>.cgns` and `ROCFOAMSURF_000<n>.cgns`, respectively, where `<n>` refers the processor ID. Two `.txt` files will also be generated. All the files within the `ROCFOAM` folder must be copied over into the `<case>/ROCKSTAR/rocRhoPimple/Rocin` directory with the following command:

```
$ cp ${CASE_DIR_PATH}/FLUID/ROCFOAM/* ${CASE_DIR_PATH}/ROCKSTAR/rocRhoPimple/Rocin
```

Figure 15 shows the file hierarchy *Rocstar* expects.



```
<case>/  
  ROCSTAR/  
    rocrhopimple/  
      Rocin/  
        fluid_in_00.000000.txt  
        ifluid_in_00.000000.txt  
        ROCFOAMVOL_0000.cgns  
        ROCFOAMSURF_0000.cgns  
        ROCFOAMVOL_0001.cgns  
        ROCFOAMSURF_0001.cgns  
      ...
```

Figure 15: Preprocessed fluid solver input files (truncated for space) that have been copied from the `FLUID/ROCFOM` folder into the correct `Rocstar` input directory.

5.3.3 Preprocessing Solid Input Files

This case study assumes the user has already generated (or otherwise obtained) the *CalculiX* mesh input as well as the input file (shown in Listing 25) and that these files should be present within the `SOLID` directory. To preprocess the solid files, run the following command (assuming paths have been exported):

```
$ mpirun -np 4 clcx_drv -i turbine_metric -p rocstar
```

Executing this command will create files within the `SOLID` directory. Two txt files, similar to the ones created after the fluid preprocessing step, will be generated, along with two CGNS files named “`test_vol_0000.cgns`” and “`test_srf_0000.cgns`.” There will be several intermediate files named after the input file, with various file extensions, which can generally be ignored. The two CGNS files and the two txt files must be copied over into the “`<case>/ROCFOM/clcxcs/Rocin`” directory. This can be done with the following command:

```
$ cp ${CASE_DIR_PATH}/SOLID/*.{txt,cgns} ${CASE_DIR_PATH}/ROCFOM/clcxcs/Rocin
```

Figure 16 shows the file hierarchy `Rocstar` expects.

```
<case>/  
  ROCSTAR/  
    clcxcs/  
      Rocin/  
        solid_in_00.000000.txt  
        isolid_in_00.000000.txt  
        test_vol_0000.cgns  
        test_srf_0000.cgns  
      Rocout/
```

Figure 16: Preprocessed solid solver input files that have been copied form the `SOLID` folder into the correct `Rocstar` input directory.



5.3.4 Computing Common Refinement Files

The *IMPACT* utility *surfdiver* will be used to compute the common refinement files with the following command:

```
$ surfdiver  
${CASE_DIR_PATH}/ROCKSTAR/rocrhopimple/Rocin/ifluid_in_00.000000.txt ${CASE_DIR_  
PATH}/ROCKSTAR/clcxcsc/Rocin/isolid_in_00.000000.txt ${CASE_DIR_PATH}/ROCKSTAR/  
Rocman/rocrhopimpleclcxcsc
```

The syntax is somewhat obfuscated by the long file paths in the command above, but the usage is:

```
$ surfdiver <fluid interface txt file> <solid interface txt file> <destination folder>
```

The destination folder in this case is *rocrhopimpleclcxcsc*, which is just a concatenation of the names of the fluid and solid solvers. Running the *surfdiver* command will create .sdv files in the *rocrhopimpleclcxcsc* folder. These files will have names of the form “*ifluid_1_sdv.cgns*” and “*isolid_1_sdv.cgns*.” Before *Rocstar* can be executed, a *RocmanControl.txt* file must be added to the *ROCKSTAR/Rocman/* directory. This file can be empty, as long as it is present. After the common refinement files have been computed, the *Rocman* directory will appear as in Figure 33.

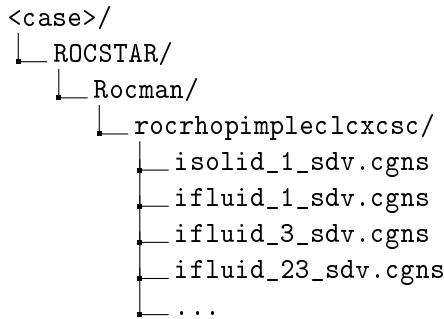


Figure 17: File hierarchy for the *Rocman* directory after the common refinement files have been generated.

5.3.5 Executing *Rocstar*

The final step before executing *Rocstar* is to edit the *RocstarControl.txt* file. This file specifies the coupling scheme to be used; names the solid and fluid solvers and output module; and sets the maximum time, time step, and output interval time (in seconds).

Listing 28: *RocstarControl.txt* File.

```
1 CouplingScheme = SolidFluidSPC  
2 SolidModule = clcxcsc  
3 FluidModule = rocrhopimple  
4 OutputModule = Rocout  
5 MaximumTime = 10  
6 AutoRestart = F  
7 CurrentTimeStep = 5.0e-05  
8 OutputIntervalTime = 1.0E-01  
9 MaxWallTime = 1000000  
10 ProfileDir = Rocman/Profiles
```



Warning

The turbine case detailed here was originally run on a cluster using 20 cores. For a personal desktop machine, this case will be intractable for a runtime of 10 seconds with a timestep of 50 microseconds. Changing the runtime to 500 microseconds (`MaximumTime = 5.0e-04`) to permit just 10 timesteps will allow the user to verify that the case is running as expected.

To run *Rocstar*, the following command is executed from within the `<case>/ROCSTAR` directory:

```
$ OMP_NUM_THREADS=4 mpirun -np 4 rocstar
```

Executing this command will generate a directory called `fluidTmp`, which will be populated with the timestep directories from the fluid solver. A set of files named after the *CalculiX* input file will also be generated to contain the solid solutions. Additionally, restart files and profiling data will be saved. To visualize the solid portion of the solution, use *Paraview* or your favorite visualization tool to display the `.exo` file named after the *CalculiX* input file. A screenshot is shown in Figure 34.

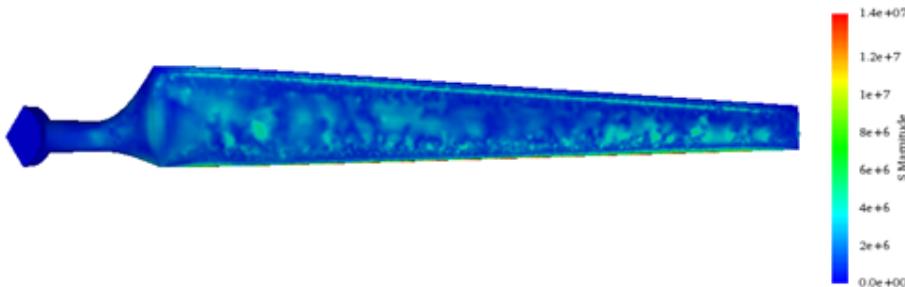


Figure 18: *Paraview* visualization of the `turbine_metric.exo` file. The magnitude of the stress at $t=5\text{e-}4$ s is shown.

To visualize the fluid portion of the solution, use *Paraview* or your favorite visualization tool to display the *OpenFOAM* files in the `fluidTmp` directory. *Paraview* requires a `.foam` file to be present in the `fluidTmp` directory. This can be created by executing the following command:

```
$ touch <case>.foam
```

This will create an empty `.foam` file that can be opened with *Paraview*. A screenshot is shown in Figure 35

Note that to reconstruct the decomposed fields, the `polyMesh` directory from the original `FLUID/constant` directory must be copied into the `fluidTmp/constant` directory before the `reconstructPar` command is run. The `Rocout` directories within the fluid and solid solvers will be populated with `.txt` and `CGNS` files, as shown in Figure 47 and Figure 48.

`CGNS` output files are formatted as `file_type_xx.yyyyyy_zzzz.cgns`. The time stamp is given in the *Rocstar* convention of `xx.yyyyyy`, where the simulation time is $0.yyyyyy \times 10^{xx} \times 10^{-9}$ s. For example, at a simulation time of $t = 1.2 \times 10^{-3}$ s, *Rocstar* generates the output files with a corresponding time stamp of `07.120000`. The process number is contained in the string `zzzz` and is “zero-indexed.” The `.txt` files are formatted `file_type_xx.yyyyyy.txt`, following the same convention as the `CGNS` files.

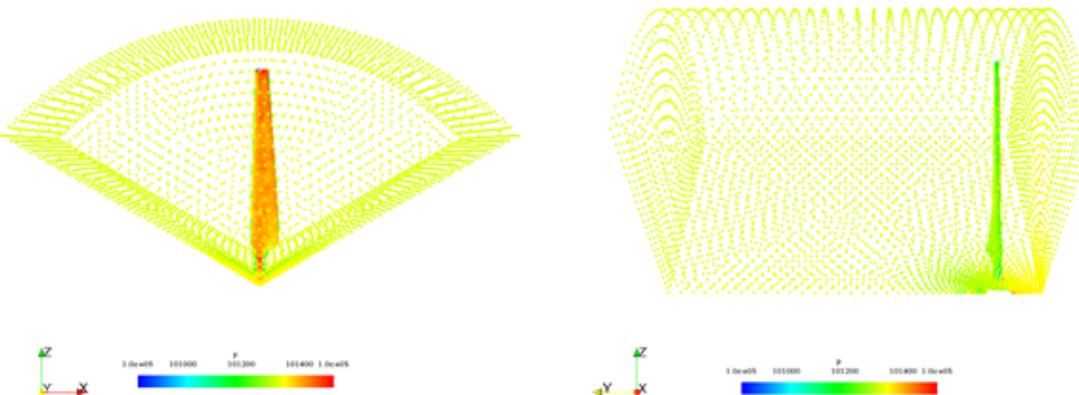


Figure 19: *Paraview* visualization of the fluid solution. Two views of the pressure at $t=5\text{e-}4$ s are shown.

```
<case>/  
  └── ROCSTAR/  
    └── rocrhopimple/  
      ├── Rocin/  
      └── Rocout/  
        ├── fluid_xx.yyyyyy_zzzz.cgns  
        ├── ifluid_nb_xx.yyyyyy_zzzz.cgns  
        ├── ifluid_ni_xx.yyyyyy_zzzz.cgns  
        ├── fluid_in_xx.yyyyyy.txt  
        └── ifluid_in_xx.yyyyyy.txt
```

Figure 20: Files generated in `rocrhopimple/Rocout` after *Rocstar* has been run.

6 Case Study: Hypersonic Plate

This case study features an angled compliant plate in a hypersonic flow. The goal is to demonstrate the capability of *Rocstar* to run FSI simulations of hypersonic flows and to simulate the physical phenomena shown in Figure 22.

6.1 Problem Setup

The case is made up of a ramped panel atop a flat plate, with an oncoming laminar Mach 6 flow, shown in Figure 23 and Figure 24.

6.1.1 Mesh

Solid and fluid regions are discretized with a matching mesh at the FSI interface. The plate is meshed with 2805 hexahedral elements ($255 \times 1 \times 11$). A close view of the solid plate mesh is shown in Figure 25.

The fluid mesh is shown in Figure 26 and Figure 27. Figure 26 shows the surface mesh that matches the solid mesh shown in Figure 25. Figure 27 shows the surface mesh surrounded by the internal mesh in cyan.



```
<case>/  
  ROCSTAR/  
    clcxsc/  
      Rocin/  
      Rocout/  
        solid_xx.yyyyyy_zzzz.cgns  
        isolid_nb_xx.yyyyyy_zzzz.cgns  
        solid_in_xx.yyyyyy.txt  
        isolid_in_xx.yyyyyy.txt
```

Figure 21: Files generated in `clcxsc/Rocout` after *Rocstar* has been run.

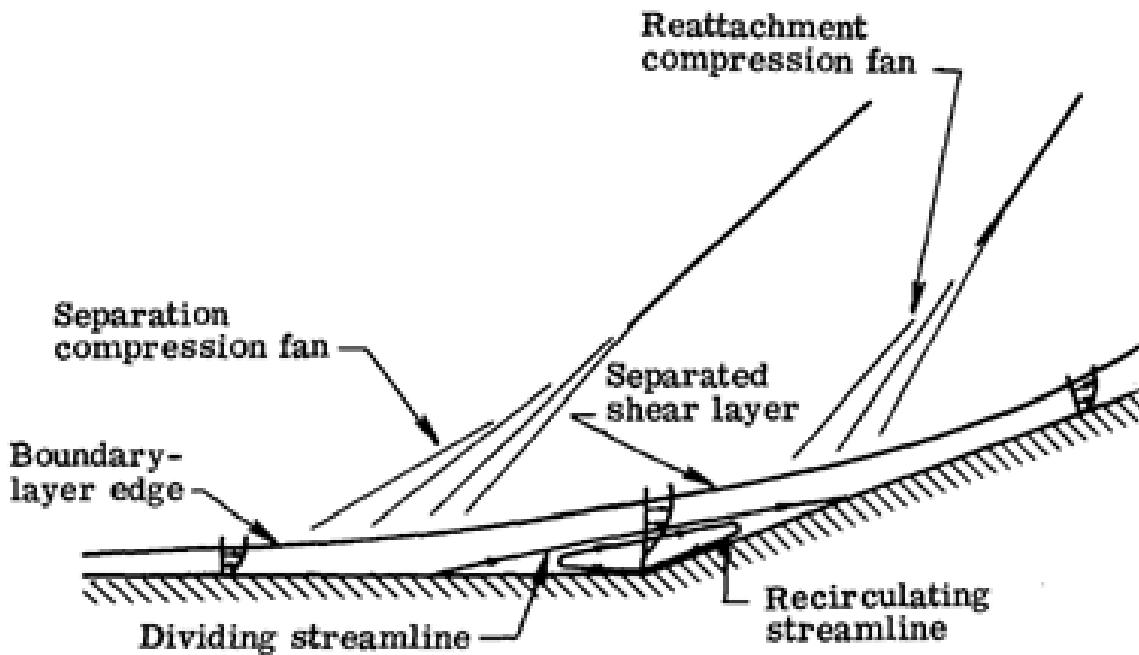


Figure 22: Various phenomena resulting from hypersonic flow over a ramped plate.

6.1.2 Materials

The plate is made from AISI 4140 alloy steel⁹. The relevant material properties are given in Table 9. The fluid is air, with properties shown in Table 10.

6.1.3 Boundary Conditions

6.2 Input Files

This simulation was run with the fluid solver *RocFOAM* and the structural solver *CalculiX*.

6.2.1 Fluid Input Files

RocFOAM is based on the *OpenFOAM* solver and uses a similar file structure, Figure 29.

⁹Properties from <https://www.azom.com/article.aspx?ArticleID=6769>

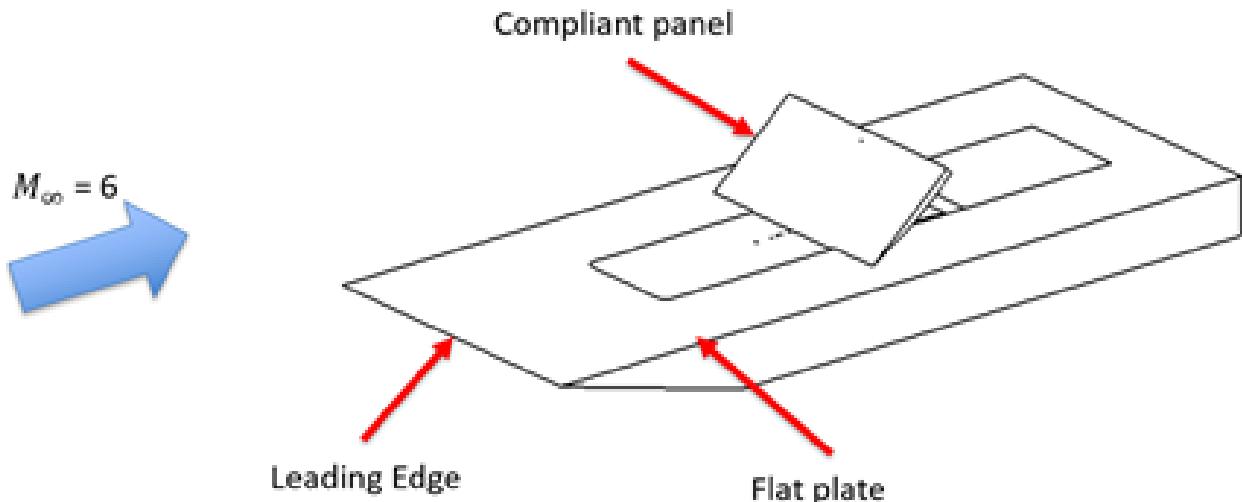


Figure 23: 3D representation of the problem setup for the ramped plate in a hypersonic flow.

Table 9: Properties of steel alloy used in the hypersonic plate simulation.

Material	E [GPa]	v	ρ [kg/m ³]
AISI	210.0	0.27	7,850

The thermodynamic and physical properties of air, the fluid used in this case, are specified via the `thermophysicalProperties` file, as shown in Listing 29. Note that the properties listed in Table 10 are present here, as well as the *OpenFOAM* specifications for the thermophysical models to be used.

Listing 29: The file `constant/thermophysicalProperties`, which specifies the thermodynamic and physical properties of the fluid.

```
1 /*-----* C++ -----*/
2 ====== |
3 \\\ / Field | OpenFOAM: The Open Source CFD Toolbox
4 \\\ / Operation | Website: https://openfoam.org
5 \\\ / And | Version: 6
6 \\\/ Manipulation |
7 /*-----*/
8 FoamFile
9 {
10     version      2.0;
11     format       ascii;
12     class        dictionary;
13     location     "constant";
14     object       thermophysicalProperties;
15 }
16 // * * * * *
17 thermoType
18 {
19     type          hePsiThermo;
20     mixture       pureMixture;
```

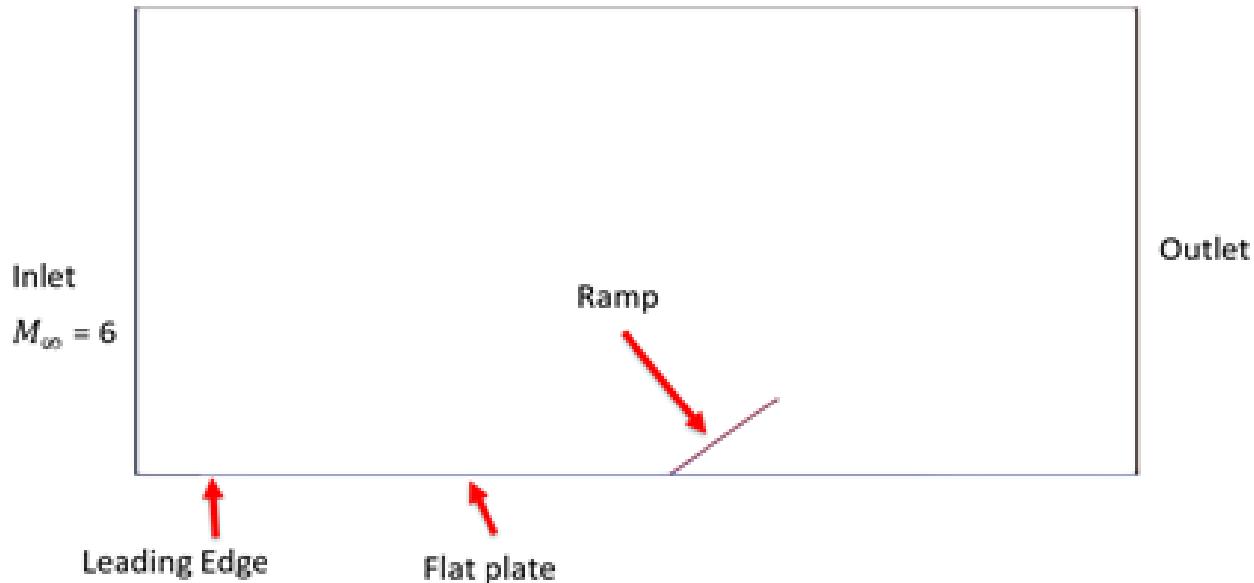


Figure 24: 2D representation of the problem setup for the ramped plate in a hypersonic flow.

Table 10: Properties of air used in the hypersonic plate simulation.

Fluid	Molecular Weight [g/mol]	c_p [kJ/kg K]	μ [kg/m s]	Pr
Air	28.9647	1004.9	1.837e-05	0.7

```
21     transport      const;
22     thermo         hConst;
23     equationOfState perfectGas;
24     specie         specie;
25     energy          sensibleEnthalpy;
26 //   energy         sensibleInternalEnergy;
27 }
```

The thermoType section (Listing 29, Lines 17–27) specifies the models that rocfoam will use. This case uses the hePsiThermo type which assumes a compressible fixed composition. Because the mixture is fixed, the mixture keyword is set with pureMixture. A constant transport model and a constant thermodynamic mode are used, which means the dynamic viscosity, Prandtl number, specific heat, and heat of formation specified in the mixture section (Listing 30 Lines 28–44) will be constant as well. The fluid is treated as a perfect gas, meaning that $\rho = \frac{1}{RT}p$ is the equation of state, and the energy model is sensibleEnthalpy, meaning that heat of formation is not included.

Listing 30: The `constant/thermophysicalProperties` file, continued.

```
28 Mixture
29 {
30     specie
31     {
32         molWeight    28.9647;
33     }
34     thermodynamics
```

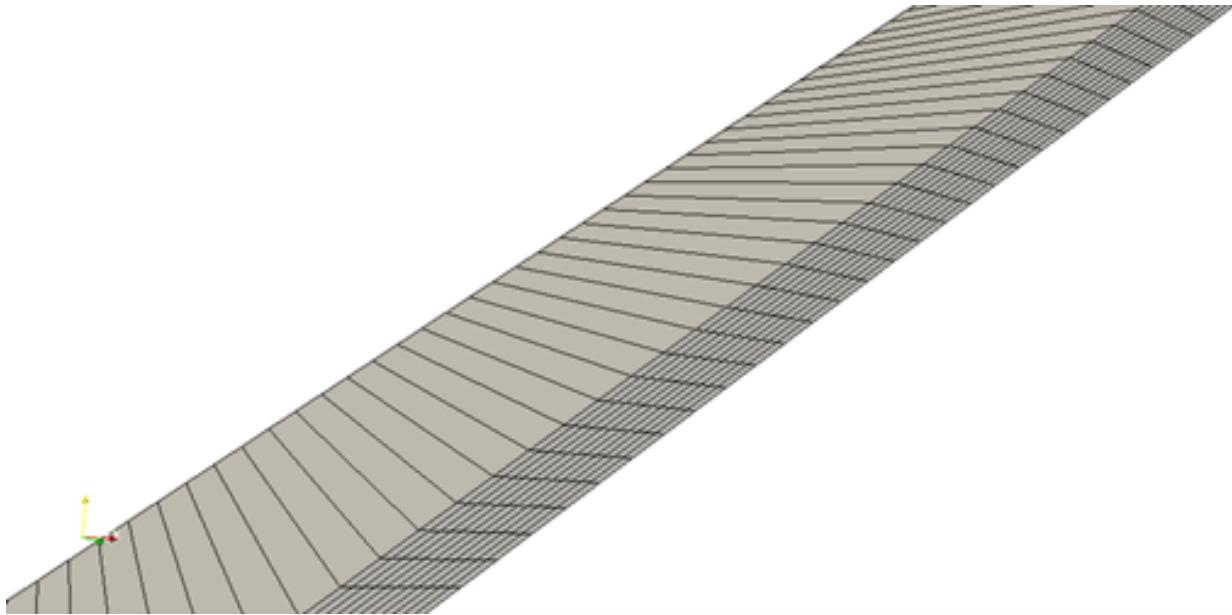


Figure 25: Zoomed view of the solid plate mesh.

```
35 {  
36     Cp          1004.9;  
37     Hf          2.544e+06;  
38 }  
39 transport  
40 {  
41     mu         1.837e-05;  
42     Pr         0.7;  
43 }  
44 }  
45 // **** //
```

Note that even though the heat of formation H_f will not be used, it still must be specified. For more examples of *RocFOAM* input files, see the wind turbine case study in Section 5.2.1.

6.2.2 Solid Input Files

The solid input files consist of the *CalculiX* input specification file, shown in Listing 31, as well as the mesh file referenced in Line 7 of the `plate1.inp` input file.

Listing 31: The `plate1.inp` *CalculiX* input file.

```
1 **  
2 ** Structure: Plate  
3 ** Test objective: test hypersonic FSI capabilities  
4 **  
5 *HEADING  
6 Model: Hex meshed plate  
7 *INCLUDE, INPUT=plate1_mesh.inp
```

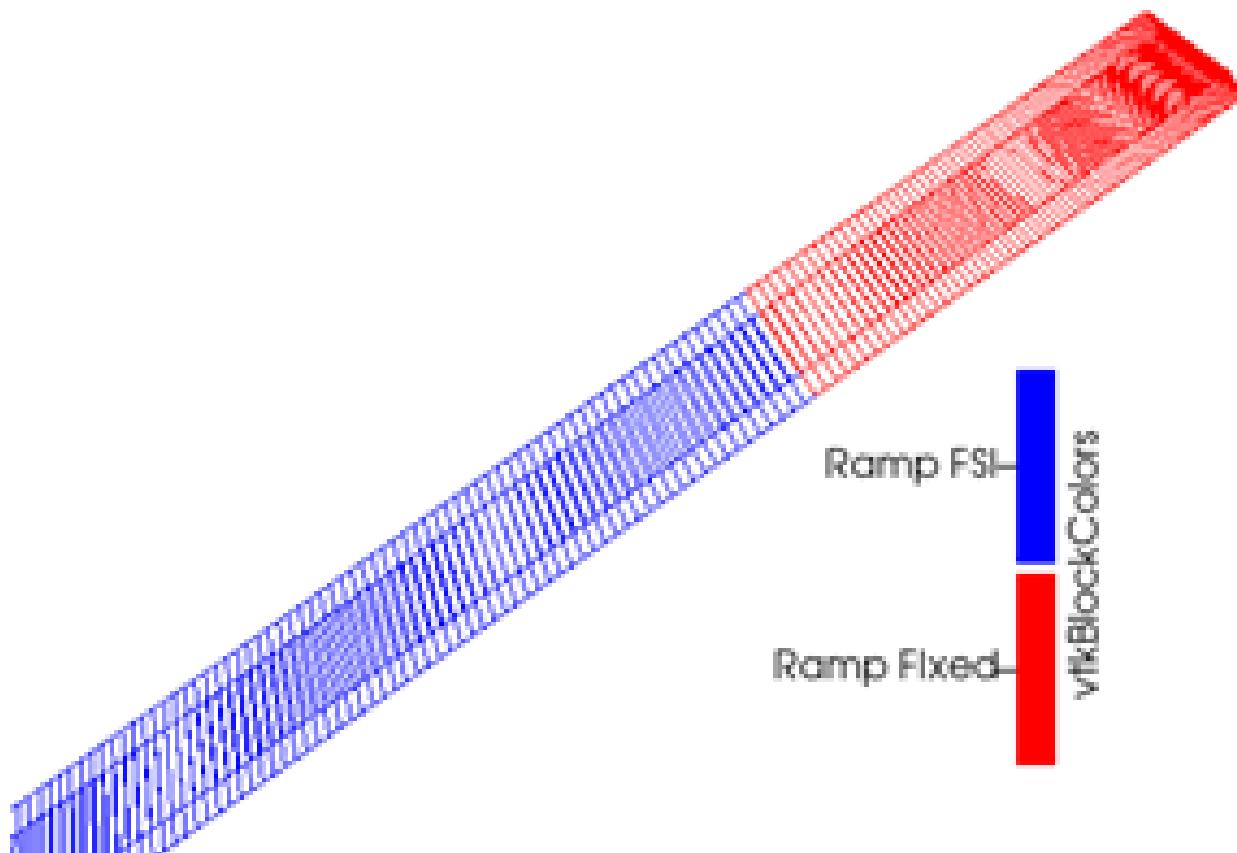


Figure 26: Enlarged view of the surface mesh surrounding the ramp structure.

```
8 *BOUNDARY
9 bot,1,3
10
11 top,1,3
12 front,3,3
13 back,3,3
14 *MATERIAL,NAME=S4041
15 *ELASTIC
16 210E9,.27
17 *DENSITY
18 7.85E3
19 *SOLID SECTION,MATERIAL=S4041,ELSET=EB1
20 *STEP,INC=100000,NLGEOM
21 *DYNAMIC,DIRECT
22 1E-7, 1.0, 0.5E-9, 1.0E-1
23 *DLOAD
24 FSI, P1, 0.0
25 *NODE PRINT,NSET=ALLNODES,FREQUENCY=1
26 U
27 *EL PRINT,ELSET=Solid,FREQUENCY=1
28 S
```

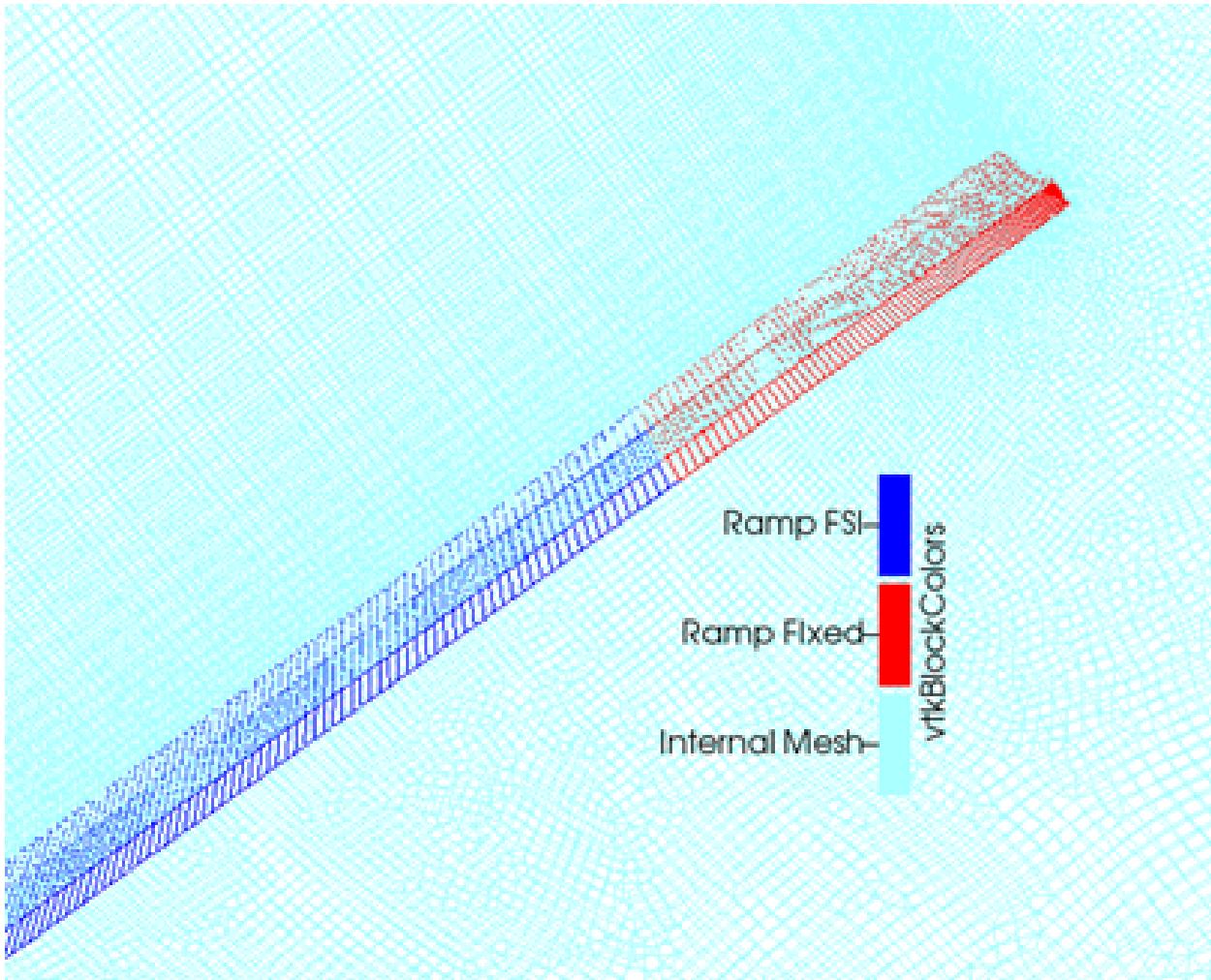


Figure 27: Enlarged view of the surface mesh and the internal mesh surrounding the ramp structure.

```
29 *NODE FILE
30 U, RF
31 *EL FILE
32 E, S
33 *END STEP
```

Section 3.2.2 gives more detail on the keywords used in CalculiX input files.

6.3 Running a *Rocstar* Case

Running the hypersonic plate case study follows the same procedure as the wind turbine case study, Section 5.3.

The process of running a *Rocstar* case involves five major steps:

1. Configuring the file structure
2. Preprocessing fluid input files
3. Preprocessing solid input files

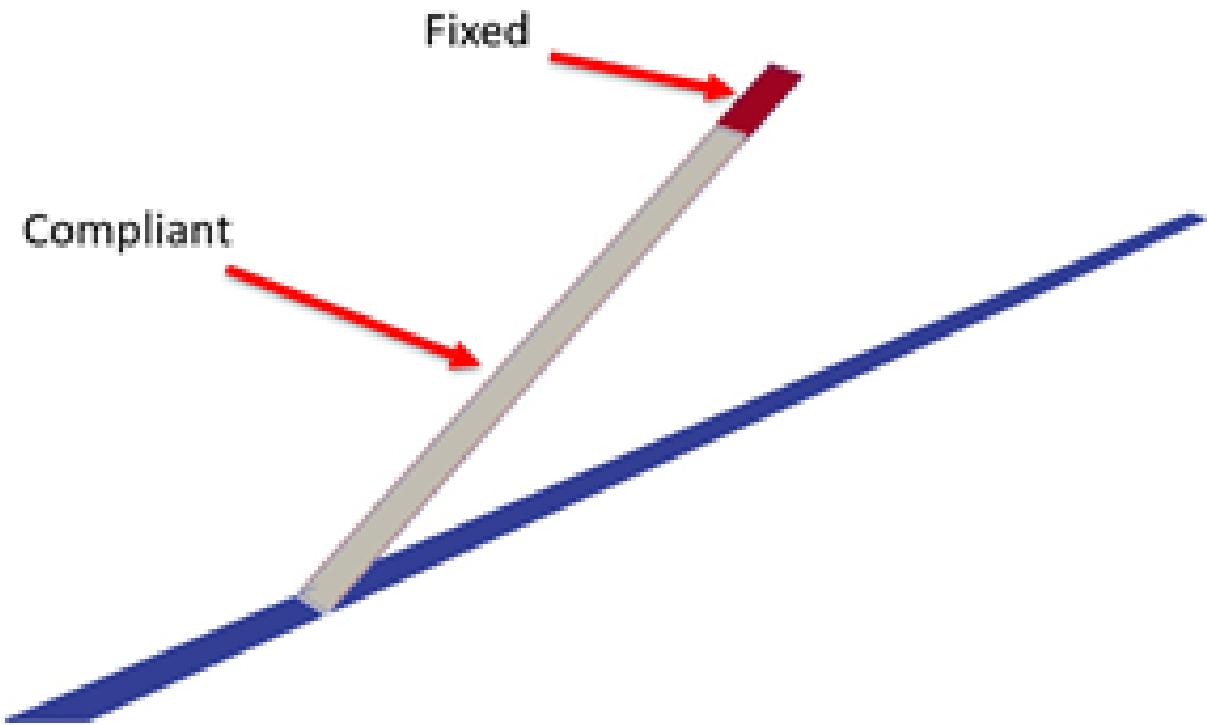


Figure 28: Boundary conditions on the ramp.

4. Computing common refinement files
5. Executing *Rocstar*

For this case study, we assume that `$CASE_DIR_PATH = absolute/path/to/case/directory/`. We further assume that the system `PATH` contains the paths to the *RocFOAM*, *CalculiX-csc*, *surfdiver*, and *Rocstar* executables.

To use the commands in the following sections directly, it is recommended that the user execute the commands in Listing 42, adjusting the paths shown in red to the specified directories:

Listing 32: Commands for running *Rocstar* with *RocFOAM* and *CalculiX*.

```
$ export LD_LIBRARY_PATH=/path/to/IMPACT-install/lib/:${LD_LIBRARY_PATH}
$ export LD_LIBRARY_PATH=/path/to/rocfoam-install/lib/:${LD_LIBRARY_PATH}
$ export LD_LIBRARY_PATH=/path/to/calculix-csc-install/lib/:${LD_LIBRARY_PATH}
$ export CASE_DIR_PATH=/path/to/Rocstar-install/<case>
$ export PATH=/path/to/rocfoam-install/bin/:$PATH
$ export PATH=/path/to/calculix-csc-install/bin/:$PATH
$ export PATH=/path/to/IMPACT-install/bin/:$PATH
$ export PATH=/path/to/Rocstar-install/bin/:$PATH
```

6.3.1 Configuring the *Rocstar* File Structure

Currently, the onus is on the user to organize the required directories. Before beginning to preprocess, the user should construct the file hierarchy as shown in Figure 30.

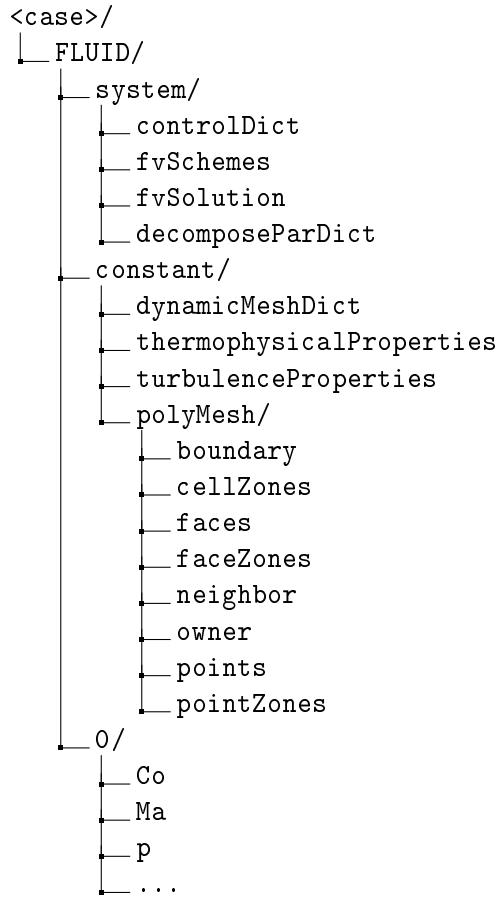


Figure 29: *RocFOAM* file structure. The `system` directory contains the dictionaries that specify solving methods, the `constant` directory contains the constant physical properties, and the initial time directory `0` contains the properties that evolve with time (truncated here to fit). As the problem runs, more directories are added with the updated properties for the associated time.

6.3.2 Preprocessing Fluid Input Files

This tutorial assumes the user has already generated the appropriate fluid meshes and variable fields shown in Figure 12. Before beginning, the user should copy their `system/`, `constant/`, and `0/` directories into the `FLUID` directory. If the user has not already set the *OpenFOAM* environment, that must be done first. On a default installation, this can be done with:

```
$ . /usr/lib/openfoam/openfoam2006/etc/bashrc
```

From within the `FLUID` directory, the user should execute:

```
$ decomposePar
```

This will decompose the fluid domain (mesh and fields) into the number of subdivisions specified by the user in `FLUID/system/decomposeParDict`. This domain will be divided into four parts, as shown in Listing 33.

Listing 33: The `system/decomposeParDict` file, which specifies the number and method for decomposing a domain for parallel runs.

```
<case>/  
    ROCSTAR/  
        rocrhocentral/  
            Rocin/  
            Rocout/  
        clcxsc/  
            Rocin/  
            Rocout/  
        Rocman/  
            rocrhocentralclcxsc/  
            RocmanControl.txt  
        RocstarControl.txt  
    RCMAN/  
    FLUID/  
    SOLID/
```

Figure 30: Native data archive for the hypersonic plate.

Running the `decomposePar` command will spawn as many new directories as was specified in the `numberOfSubdomains` field. These directories, named `processor<n>` (where `<n>` goes from 0 to `numberOfSubdomains-1`), each contain a `0/` and `constant/` directory, which consist of the data for each subdomain.

Warning

The number of desired subdomains given in `system/decomposeParDict` sets the number of processors that must be used for the remainder of the *Rocstar* run. Attempting to run with any other number of processors will result in an error.

Once the domain has been divided, the preprocessing command can be run:

```
$ mpirun -np 4 rocRhoPimple -preprocess -parallel
```

Note that this command assumes the paths to the *IMPACT* and *RocFOAM* libraries, as well as the path to the *RocFOAM bin* directory, have been exported. Without exporting those paths, the command resembles:



```
$ LD_LIBRARY_PATH=${HOME}/Rocstar/IMPACT-install/lib/:${HOME}/Rocstar/rocfoam-
install/lib:${LD_LIBRARY_PATH} mpirun -np 4 ${HOME}/Rocstar/rocfoam-install/bin/
rocRhoPimple -preprocess -parallel
```

Executing the preprocessing command will create a new folder within the FLUID directory called “ROCFOAM.” This folder will contain generated volume and surface CGNS files named `ROCFOAMVOL_000<n>.cgns` and `ROCFOAMSURF_000<n>.cgns`, respectively, where `<n>` refers the processor ID. Two txt files will also be generated. All the files within the ROCFOAM folder must be copied over into the `<case>/ROCSTAR/rocrhopimple/Rocin` directory with the following command:

```
$ cp ${CASE_DIR_PATH}/FLUID/ROCFOAM/* \\ ${CASE_DIR_PATH}/ROCSTAR/rocrhopimple/Rocin
```

Figure 31 shows the file hierarchy *Rocstar* expects.

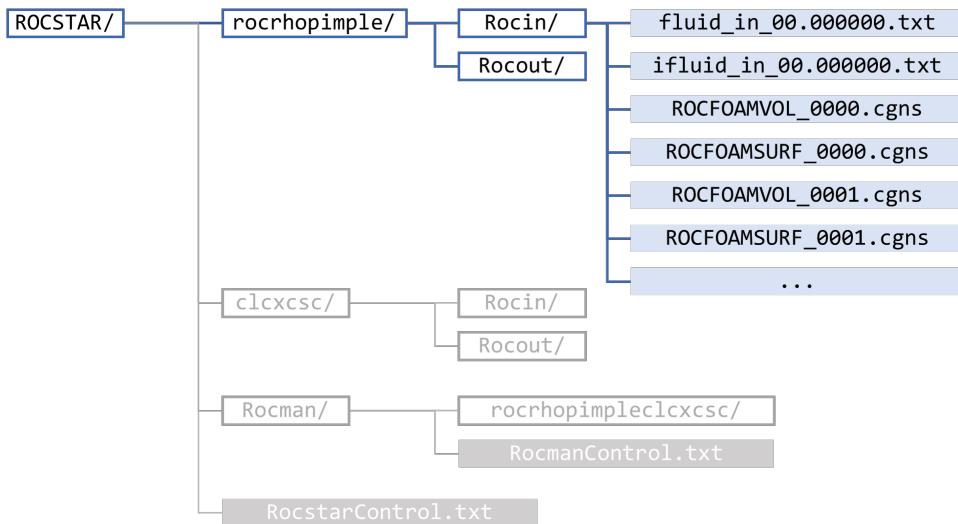


Figure 31: Preprocessed fluid solver input files (truncated for space) that have been copied from the `FLUID/ROCFOAM` folder into the correct *Rocstar* input directory.

6.3.3 Preprocessing Solid Input Files

This case study assumes the user has already generated (or otherwise obtained) the *CalculiX* mesh input as well as the input file (shown in Listing 25) and that these files should be present within the `SOLID` directory. To preprocess the solid files, run the following command (assuming paths have been exported):

```
$ mpirun -np 4 clcx_drv -i turbine_metric -p rocstar
```

Executing this command will create files within the `SOLID` directory. Two txt files, similar to the ones created after the fluid preprocessing step, will be generated, along with two CGNS files named “`test_vol_0000.cgns`” and “`test_srf_0000.cgns`.” There will be several intermediate files named after the input file, with various file extensions, which can generally be ignored. The two CGNS files and the two txt files must be copied over into the “`<case>/ROCSTAR/clcxcs/Rocin`” directory. This can be done with the following command:

```
$ cp ${CASE_DIR_PATH}/SOLID/*.{txt,cgns} \\ ${CASE_DIR_PATH}/ROCSTAR/clcxsc/Rocin
```

Figure 32 shows the file hierarchy *Rocstar* expects.

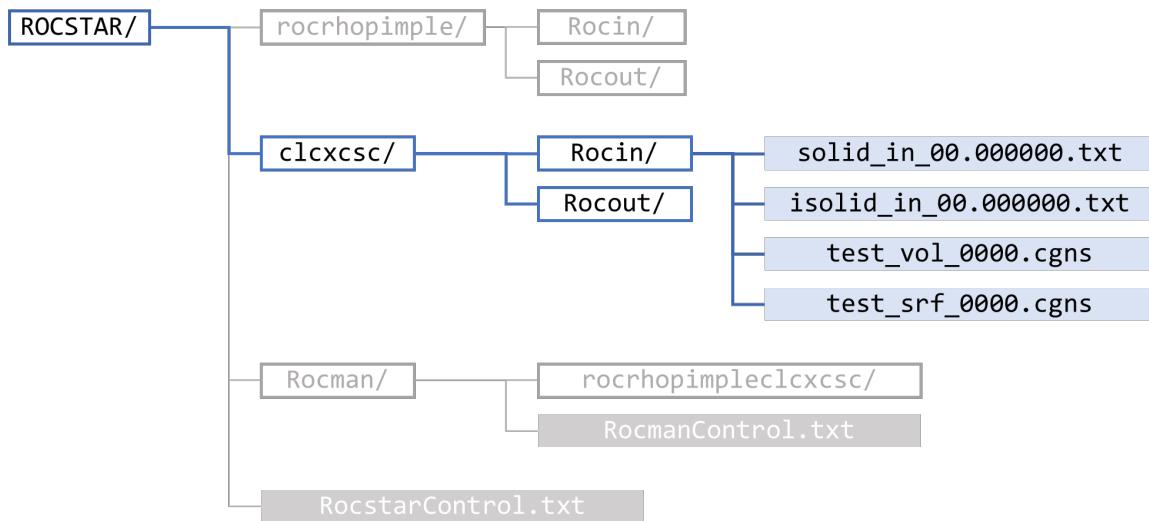


Figure 32: Preprocessed solid solver input files that have been copied from the `SOLID` folder into the correct `Rocstar` input directory.

6.3.4 Computing Common Refinement Files

The *IMPACT* utility *surfdiver* will be used to compute the common refinement files with the following command:

```
$ surfdiver \
${CASE_DIR_PATH}/ROCSTAR/rocrhopimple/Rocin/ifluid_in_00.000000.txt \ ${CASE_DIR_PATH}/ROCSTAR/clcxsc/Rocin/isolid_in_00.000000.txt \ ${CASE_DIR_PATH}/ROCSTAR/Rocman/rocrhopimpleclcxsc
```

The syntax is somewhat obfuscated by the long file paths in the command above, but the usage is:

```
$ surfdiver <fluid interface txt file> <solid interface txt file> <destination folder>
```

The destination folder in this case is `rocrhopimpleclcxsc`, which is just a concatenation of the names of the fluid and solid solvers. Running the *surfdiver* command will create `.sdv` files in the `rocrhopimpleclcxsc` folder. These files will have names of the form “`ifluid_1_sdv.cgns`” and “`isolid_1_sdv.cgns`.” Before *Rocstar* can be executed, a `RocmanControl.txt` file must be added to the `ROCSTAR/Rocman/` directory. This file can be empty, as long as it is present. After the common refinement files have been computed, the `Rocman` directory will appear as in Figure 33.

6.3.5 Executing *Rocstar*

The final step before executing *Rocstar* is to edit the `RocstarControl.txt` file. This file specifies the coupling scheme to be used; names the solid and fluid solvers and output module; and sets the maximum time, time step, and output interval time (in seconds).

Listing 34: `RocstarControl.txt` File.

```
1 CouplingScheme = SolidFluidSPC
2 SolidModule = clcxsc
```

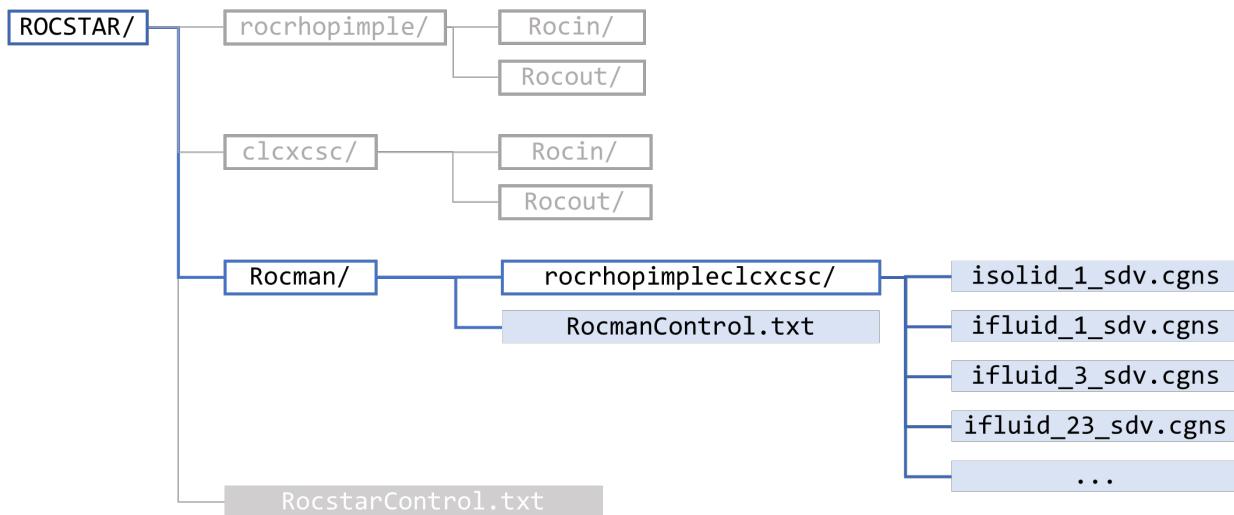


Figure 33: File hierarchy for the *Rocman* directory after the common refinement files have been generated.

```
3 FluidModule = rocrhopimple
4 OutputModule = Rocout
5 MaximumTime = 10
6 AutoRestart = F
7 CurrentTimeStep = 5.0e-05
8 OutputIntervalTime = 1.0E-01
9 MaxWallTime = 1000000
10 ProfileDir = Rocman/Profiles
```

Warning

The turbine case detailed here was originally run on a cluster using 20 cores. For a personal desktop machine, this case will be intractable for a runtime of 10 seconds with a timestep of 50 microseconds. Changing the runtime to 500 microseconds (`MaximumTime = 5.0e-04`) to permit just 10 timesteps will allow the user to verify that the case is running as expected.

To run *Rocstar*, the following command is executed from within the `<case>/ROCSTAR` directory:

```
$ OMP_NUM_THREADS=4 mpirun -np 4 rocstar
```

Executing this command will generate a directory called `fluidTmp`, which will be populated with the timestep directories from the fluid solver. A set of files named after the *CalculiX* input file will also be generated to contain the solid solutions. Additionally, restart files and profiling data will be saved. To visualize the solid portion of the solution, use *Paraview* or your favorite visualization tool to display the `.exo` file named after the *CalculiX* input file. A screenshot is shown in Figure 34. To visualize the fluid portion of the solution, use *Paraview* or your favorite visualization tool to display the *OpenFOAM* files in the `fluidTmp` directory. *Paraview* requires a `.foam` file to be present in the `fluidTmp` directory. This can be created by executing the following command:

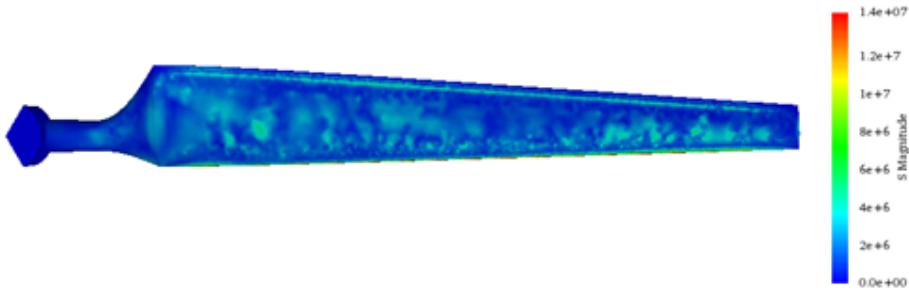


Figure 34: *Paraview* visualization of the `turbine_metric.exo` file. The magnitude of the stress at $t=5\text{e-}4$ s is shown.

```
$ touch <case_name>.foam
```

This will create an empty .foam file that can be opened with *Paraview*. A screenshot is shown in Figure 35

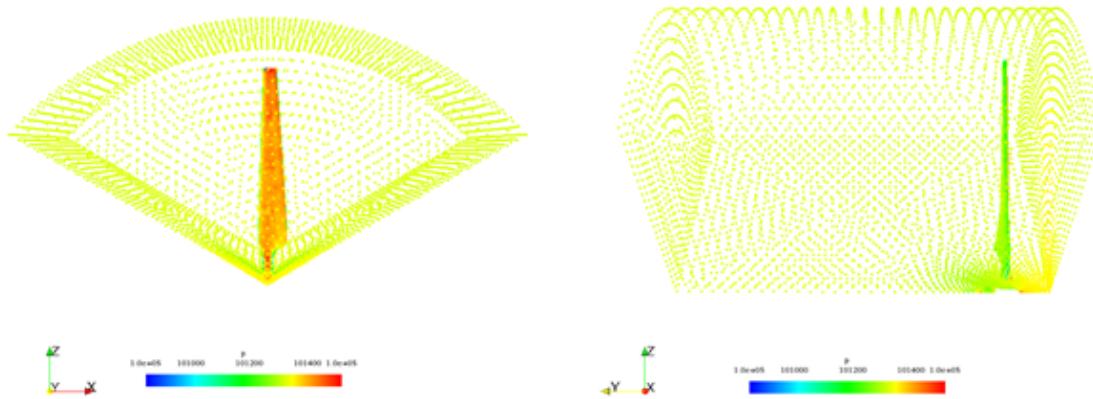


Figure 35: *Paraview* visualization of the fluid solution. Two views of the pressure at $t=5\text{e-}4$ s are shown.

Note that to reconstruct the decomposed fields, the `polyMesh` directory from the original `FLUID/constant` directory must be copied into the `fluidTmp/constant` directory before the `reconstructPar` command is run. The `Rocout` directories within the fluid and solid solvers will be populated with txt and CGNS files, as shown in Figure 47 and Figure 48.

CGNS output files are formatted as `file_type_xx.yyyyyy_zzzz.cgns`. The time stamp is given in the *Rocstar* convention of `xx.yyyyyy`, where the simulation time is $0.yyyyyy \times 10^{xx} \times 10^{-9}$ s. For example, at a simulation time of $t = 1.2 \times 10^{-3}$ s, *Rocstar* generates the output files with a corresponding time stamp of 07.120000. The process number is contained in the string `zzzz` and is “zero-indexed.” The .txt files are formatted `file_type_xx.yyyyyy.txt`, following the same convention as the CGNS files.

7 Case Study: Simple Mixer

The wind turbine case in Section 5 features an interaction between a fluid and a structure in which wind loads are applied to the structure, causing it to move. This case describes the opposite situation: the motion of the structure induces movement in the fluid.

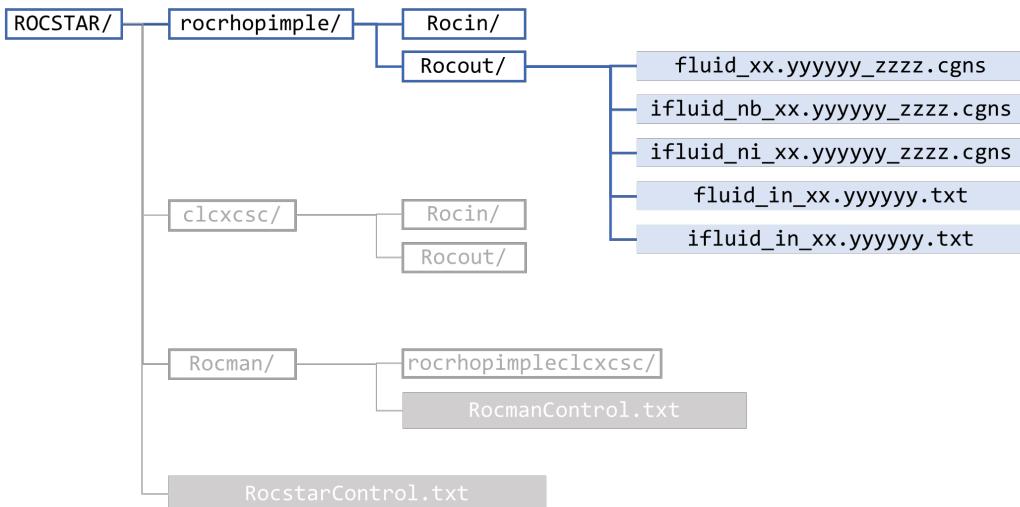


Figure 36: Files generated in `rocrhopimple/Rocout` after *Rocstar* has been run.

7.1 Problem Setup

The mixer is placed within a fluid-filled domain, surrounded by a circular baffle.

7.1.1 Mesh

Solid and fluid regions are discretized with a matching mesh at the FSI interface. Note the inner circle within the fluid domain—this is demarcated with a baffle, which turns internal faces into boundary faces.

7.2 Input Files

This simulation was run with the fluid solver *RocFOAM* and the structural solver *CalculiX*.

7.2.1 Fluid Input Files

RocFOAM is based on the *OpenFOAM* solver and uses a similar file structure, as shown in Figure 41. The `dynamicMeshDict` file in the `constant` directory describes how the fluid mesh must move to maintain the interface with the solid mixer.

Listing 35: The `dynamicMeshDict` file, which specifies mesh motion.

```
1 FoamFile
2 {
3     version 2.0;
4     format ascii;
5     class dictionary;
6     location "constant";
7     object dynamicMeshDict;
8 }
9 // * * * * *
10 dynamicFvMesh dynamicMultiMotionSolverFvMesh;
```

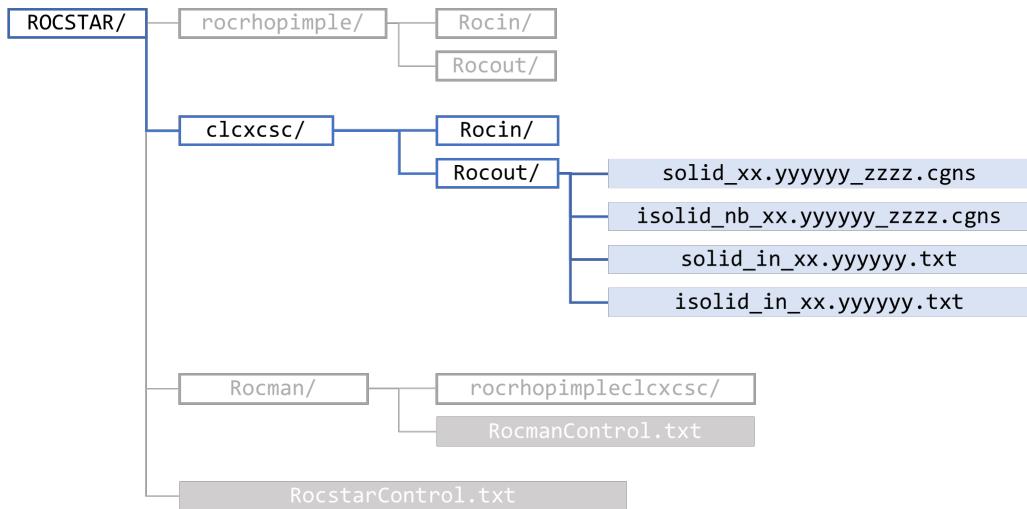


Figure 37: Files generated in `clcxscsc/Rocout` after *Rocstar* has been run.

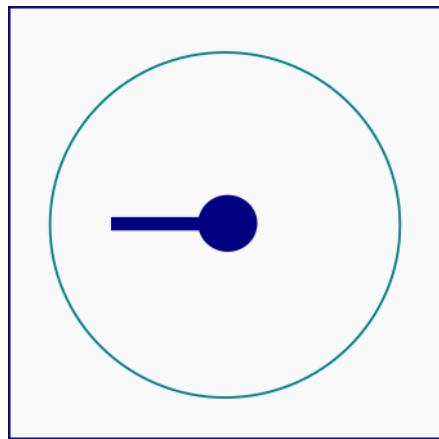


Figure 38: Cartoon of the simple mixer problem.

```
11 rotation1
12 {
13     solidBodyMotionFunction rotatingMotion;
14     rotatingMotionCoeffs
15     {
16         origin (0 0 0);
17         axis (0 0 1);
18         omega 10; // rad/s
19     }
20 }
21 dynamicMultiMotionSolverFvMeshCoeffs
22 {
23     blades
24     {
25         cellZone Rotating;
```

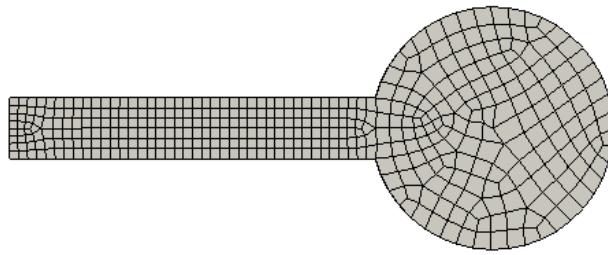


Figure 39: View of the solid mesh.

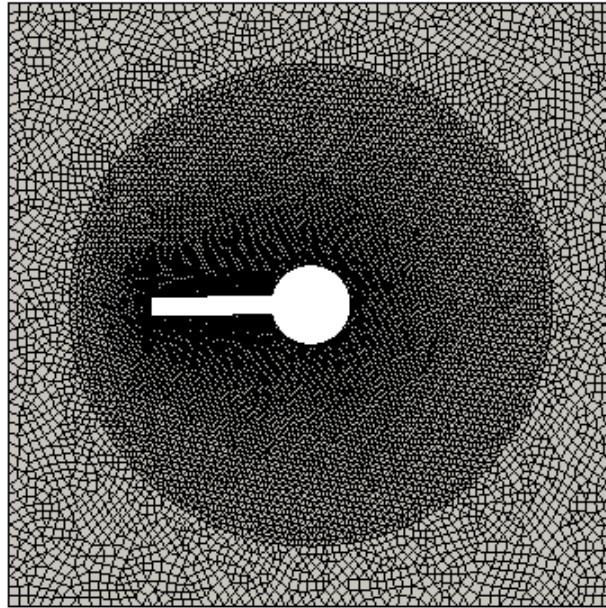


Figure 40: View of the fluid mesh.

```
26 // Solve displacement on top of solid-body rotation
27 solver solidBodyDisplacementLaplacian;
28 solidBodyDisplacementLaplacianCoeffs
29 {
30     solidBodyMotionFunction multiMotion;
31     multiMotionCoeffs
32     {
33         motion_1
34         {
35             $rotation1;
36         }
37     }
38     diffusivity inverseDistance (fsi);
39 }
40 }
41 }
42 // **** //
```

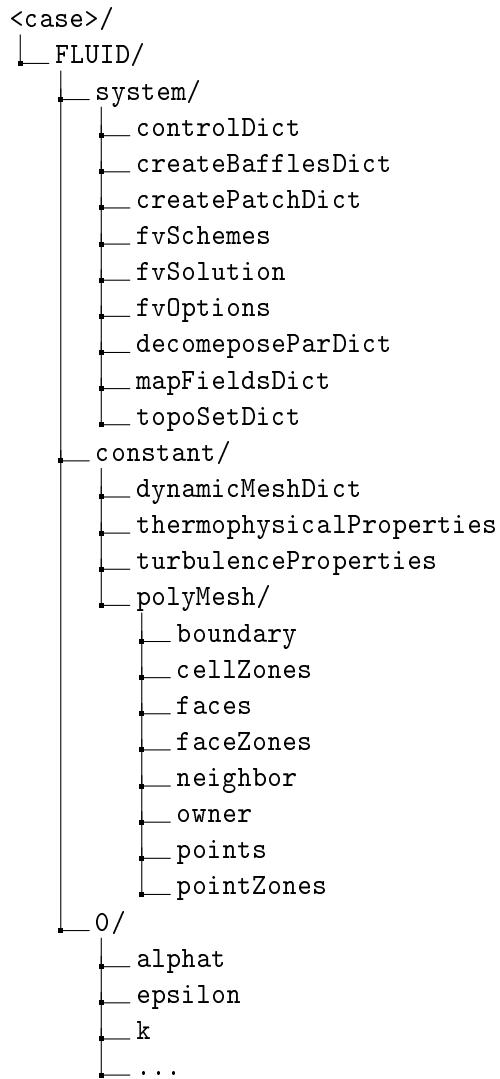


Figure 41: File structure of the fluid input files for the simple mixer case.

Lines 11–20 describe the solid body motion of the mixer—a rotation about (0,0,0) through the axis (0,0,1) at a rate of 10 rad/second.

7.2.2 Solid Input Files

The solid input files consist of the *CalculiX* input specification file, shown in Listing 36, as well as the mesh file referenced in Line 6 of the `single_blade.inp` input file.

Listing 36: The `single_blade.inp` *CalculiX* input file.

```
1 **  
2 ** Test objective: test FSI capabilities of the CSC module  
3 **  
4 *HEADING  
5 Model: Single blade Date: 11/23/2020  
6 *INCLUDE, INPUT=single_blade_mesh.inp
```



```
7 *BOUNDARY
8 top,1,3
9 bot,1,3
10 *BOUNDARY
11 symm,3,3
12 *MATERIAL,NAME=EL
13 *ELASTIC
14 5e9,.3
15 *DENSITY
16 3.e3
17 *SOLID SECTION,MATERIAL=EL,ELSET=EB1
18 *STEP,INC=1000000,NLGEOM
19 *DYNAMIC,DIRECT
20 1E-3, 1.0e1, 1E-5,1.0E-1
21 *DLOAD
22 EB1,Centrif,100,0,0,0,0,0,0,1
23 *DLOAD
24 FSI, P1, 0
25 *NODE FILE, FREQUENCY=1
26 U, RF
27 *EL FILE, FREQUENCY=1
28 E, S
29 *END STEP
```

Lines 21 and 22 of Listing 36 specify a centrifugal load with $\omega^2 = 100$ about the point $(0, 0, 0)$ around the axis $(0, 0, 1)$ on all the elements. Note that this matches the specification given in Listing 35. Section 3.2.2 gives more detail on the keywords used in *CalculiX* input files.

7.2.3 Rocstar

The *Rocstar* input file *RocstarControl.txt* for the simple mixer case is shown in Listing 37

Listing 37: The *RocstarControl.txt* input file.

```
1 CouplingScheme = SolidFluidSPC
2 SolidModule = clcxsc
3 FluidModule = rocrhopimple
4 OutputModule = Rocout
5 MaximumTime = 1
6 AutoRestart = F
7 CurrentTimeStep = 0.001
8 OutputIntervalTime = 0.1
9 MaxWallTime = 10000000
10 ProfileDir = Rocman/Profiles
```

7.3 Running a *Rocstar* Case

The process of running a *Rocstar* case involves five major steps:

1. Configuring the file structure



2. Preprocessing fluid input files
3. Preprocessing solid input files
4. Computing common refinement files
5. Executing *Rocstar*

For this case study, we assume that `$CASE_DIR_PATH = absolute/path/to/case/directory/`. We further assume that the system PATH contains the paths to the *RocFOAM*, *CalculiX-csc*, *surfdiver*, and *Rocstar* executables.

To use the commands in the following sections directly, it is recommended that the user execute the commands in Listing 42, adjusting the paths shown in red to the specified directories:

Listing 38: Commands for running *Rocstar* with *RocFOAM* and *CalculiX*.

```
$ export LD_LIBRARY_PATH=/path/to/IMPACT-install/lib/:${LD_LIBRARY_PATH}
$ export LD_LIBRARY_PATH=/path/to/rocfoam-install/lib/:${LD_LIBRARY_PATH}
$ export LD_LIBRARY_PATH=/path/to/calculix-csc-install/lib/:${LD_LIBRARY_PATH}
$ export CASE_DIR_PATH=/path/to/Rocstar-install/<case>
$ export PATH=/path/to/rocfoam-install/bin/:$PATH
$ export PATH=/path/to/calculix-csc-install/bin/:$PATH
$ export PATH=/path/to/IMPACT-install/bin/:$PATH
$ export PATH=/path/to/Rocstar-install/bin/:$PATH
```

7.3.1 Configuring the *Rocstar* File Structure

Currently, the onus is on the user to organize the required directories. Before beginning to preprocess, the user should construct the file hierarchy as shown in Figure 14.

This tutorial assumes the user has already generated the appropriate fluid meshes and variable fields shown in Figure 12. Before beginning, the user should copy their `system/`, `constant/`, and `0/` directories into the `FLUID` directory. If the user has not already set the *OpenFOAM* environment, that must be done first. On a default installation, this can be done with:

```
$ . /usr/lib/openfoam/openfoam2006/etc/bashrc
```

From within the `FLUID` directory, the user should execute:

```
$ decomposePar
```

This will decompose the fluid domain (mesh and fields) into the number of subdivisions specified by the user in `FLUID/system/decomposeParDict`. This domain will be divided into six parts, as shown in Listing 39.

Listing 39: The `system/decomposeParDict` file, which specifies the number and method for decomposing a domain for parallel runs.

```
1 FoamFile
2 {
3     version 2.0;
4     format ascii;
5     class dictionary;
6     location "system";
```



```
7     object decomposeParDict;
8 }
9 // * * * * *
10 numberOfSubdomains 6;
11 method scotch;
12 // *****
```

Running the `decomposePar` command will spawn as many new directories as was specified in the `numberOfSubdomains` field. These directories, named `processor<n>` (where `<n>` goes from 0 to `numberOfSubdomains-1`), each contain a `0/` and `constant/` directory, which consist of the data for each subdomain.

Warning

The number of desired subdomains given in `system/decomposeParDict` sets the number of processors that must be used for the remainder of the *Rocstar* run. Attempting to run with any other number of processors will result in an error.

Once the domain has been divided, the preprocessing command can be run:

```
$ mpirun -np 6 rocRhoPimple -preprocess -parallel
```

Note that this command assumes the paths to the *IMPACT* and *RocFOAM* libraries, as well as the path to the *RocFOAM bin* directory, have been exported. Without exporting those paths, the command resembles:

```
$ LD_LIBRARY_PATH=${HOME}/Rocstar/IMPACT-install/lib/:${HOME}/Rocstar/rocfoam-
install/lib:${LD_LIBRARY_PATH} mpirun -np 6 ${HOME}/Rocstar/rocfoam-install/bin/
rocRhoPimple -preprocess -parallel
```

Executing the preprocessing command will create a new folder within the `FLUID` directory called “`ROCFOMAM`.” This folder will contain generated volume and surface CGNS files named `ROCFOMAMVOL_000<n>.cgns` and `ROCFOMAMSURF_000<n>.cgns`, respectively, where `<n>` refers the processor ID. Two `.txt` files will also be generated. All the files within the `ROCFOMAM` folder must be copied over into the `<case>/ROCSTAR/rocrhopimple/Rocin` directory with the following command:

```
$ cp ${CASE_DIR_PATH}/FLUID/ROCFOMAM/* ${CASE_DIR_PATH}/ROCSTAR/rocrhopimple/Rocin
```

Figure 42 shows the file hierarchy *Rocstar* expects.

7.3.2 Preprocessing Solid Input Files

This case study assumes the user has already generated (or otherwise obtained) the *CalculiX* mesh input as well as the input file (shown in Listing 25) and that these files should be present within the `SOLID` directory. To preprocess the solid files, run the following command (assuming paths have been exported):

```
$ mpirun -np 6 clcx_drv -i turbine_metric -p rocstar
```



```
<case>/  
  ROCSTAR/  
    rocrhopimple/  
      Rocin/  
        fluid_in_00.000000.txt  
        ifluid_in_00.000000.txt  
        ROCFOAMVOL_0000.cgns  
        ROCFOAMSURF_0000.cgns  
        ROCFOAMVOL_0001.cgns  
        ROCFOAMSURF_0001.cgns  
        ...
```

Figure 42: Preprocessed fluid solver input files (truncated for space) that have been copied from the FLUID/ROCFOM folder into the correct *Rocstar* input directory.

Executing this command will create files within the SOLID directory. Two txt files, similar to the ones created after the fluid preprocessing step, will be generated, along with two CGNS files named “test_vol_0000.cgns” and “test_srf_0000.cgns.” There will be several intermediate files named after the input file, with various file extensions, which can generally be ignored. The two CGNS files and the two txt files must be copied over into the “<case>/ROCSTAR/clcxcs/Rocin” directory. This can be done with the following command:

```
$ cp ${CASE_DIR_PATH}/SOLID/*.{txt,cgns} ${CASE_DIR_PATH}/ROCSTAR/clcxsc/Rocin
```

Figure 43 shows the file hierarchy *Rocstar* expects.

```
<case>/  
  ROCSTAR/  
    clcxsc/  
      Rocin/  
        solid_in_00.000000.txt  
        isolid_in_00.000000.txt  
        test_vol_0000.cgns  
        test_srf_0000.cgns  
      Rocout/
```

Figure 43: Preprocessed solid solver input files that have been copied form the SOLID folder into the correct *Rocstar* input directory.

7.3.3 Computing Common Refinement Files

The *IMPACT* utility *surfdiver* will be used to compute the common refinement files with the following command:

```
$ surfdiver \  
 ${CASE_DIR_PATH}/ROCSTAR/rocrhopimple/Rocin/ifluid_in_00.000000.txt \ ${CASE_DIR_  
 PATH}/ROCSTAR/clcxsc/Rocin/isolid_in_00.000000.txt \ ${CASE_DIR_PATH}/ROCSTAR/  
 Rocman/rocrhopimpleclcxsc
```



The syntax is somewhat obfuscated by the long file paths in the command above, but the usage is:

```
$ surfdiver <fluid interface txt file> <solid interface txt file> <destination folder>
```

The destination folder in this case is `rocrhopimpleclcxsc`, which is just a concatenation of the names of the fluid and solid solvers. Running the `surfdiver` command will create .sdv files in the `rocrhopimpleclcxsc` folder. These files will have names of the form “`ifluid_1_sdv.cgns`” and “`isolid_1_sdv.cgns`.” Before *Rocstar* can be executed, a `RocmanControl.txt` file must be added to the `ROCKSTAR/Rocman/` directory. This file can be empty, as long as it is present. After the common refinement files have been computed, the *Rocman* directory will appear as in Figure 44.

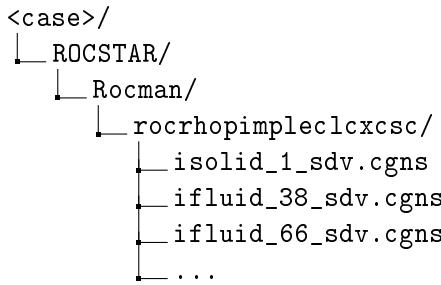


Figure 44: File hierarchy for the *Rocman* directory after the common refinement files have been generated.

7.3.4 Executing *Rocstar*

The final step before executing *Rocstar* is to edit the `RocstarControl.txt` file. This file specifies the coupling scheme to be used; names the solid and fluid solvers and output module; and sets the maximum time, time step, and output interval time (in seconds).

Listing 40: `RocstarControl.txt` File.

```
1 CouplingScheme = SolidFluidSPC
2 SolidModule = clcxsc
3 FluidModule = rocrhopimple
4 OutputModule = Rocout
5 MaximumTime = 1
6 AutoRestart = F
7 CurrentTimeStep = 0.001
8 OutputIntervalTime = 0.1
9 MaxWallTime = 10000000
10 ProfileDir = Rocman/Profiles
```

To run *Rocstar*, the following command is executed from within the `<case>/ROCSTAR` directory:

```
$ OMP_NUM_THREADS=6 mpirun -np 6 rocstar
```

Executing this command will generate a directory called `fluidTmp`, which will be populated with the timestep directories from the fluid solver. A set of files named after the *CalculiX* input file will also be generated to contain the solid solutions. Additionally, restart files and profiling data will be saved. To visualize the solid portion of the solution, use *Paraview* or your favorite visualization tool to display the .exo file named after the *CalculiX* input file. A screenshot is shown in Figure 45.

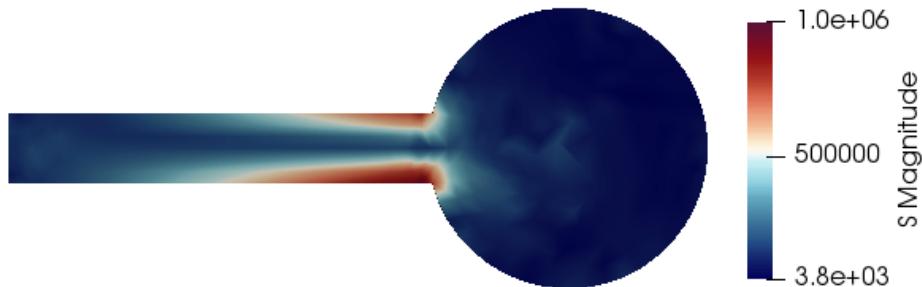


Figure 45: The stress in the mixer at the final timestep.

To visualize the fluid portion of the solution, use *Paraview* or your favorite visualization tool to display the *OpenFOAM* files in the `fluidTmp` directory. *Paraview* requires a `.foam` file to be present in the `fluidTmp` directory. This can be created by executing the following command:

```
$ touch <case>.foam
```

This will create an empty `.foam` file that can be opened with *Paraview*. A screenshot of the *Paraview* visualization is shown in Figure 46

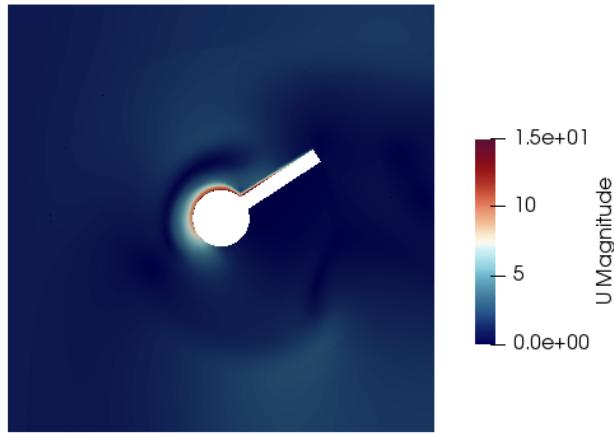


Figure 46: The displacement in the fluid at the final timestep.

Note that to reconstruct the decomposed fields, the `polyMesh` directory from the original `FLUID/constant` directory must be copied into the `fluidTmp/constant` directory before the `reconstructPar` command is run. The `Rocout` directories within the fluid and solid solvers will be populated with `txt` and `CGNS` files, as shown in Figure 47 and Figure 48.

`CGNS` output files are formatted as `file_type_xx.yyyyyy_zzzz.cgns`. The time stamp is given in the *Rocstar* convention of `xx.yyyyyy`, where the simulation time is $0.yyyyyy \times 10^{xx} \times 10^{-9}$ s. For example, at a simulation time of $t = 1.2 \times 10^{-3}$ s, *Rocstar* generates the output files with a corresponding time stamp of `07.120000`. The process number is contained in the string `zzzz` and is “zero-indexed.” The `.txt` files are formatted `file_type_xx.yyyyyy.txt`, following the same convention as the `CGNS` files.



```
<case>/  
  ROCSTAR/  
    rocRhopyimple/  
      Rocin/  
      Rocout/  
        fluid_xx.yyyyyy_zzzz.cgns  
        ifluid_nb_xx.yyyyyy_zzzz.cgns  
        ifluid_ni_xx.yyyyyy_zzzz.cgns  
        fluid_in_xx.yyyyyy.txt  
        ifluid_in_xx.yyyyyy.txt
```

Figure 47: Files generated in `rocRhopyimple/Rocout` after *Rocstar* has been run.

```
<case>/  
  ROCSTAR/  
    clcxsc/  
      Rocin/  
      Rocout/  
        solid_xx.yyyyyy_zzzz.cgns  
        isolid_nb_xx.yyyyyy_zzzz.cgns  
        solid_in_xx.yyyyyy.txt  
        isolid_in_xx.yyyyyy.txt
```

Figure 48: Files generated in `clcxsc/Rocout` after *Rocstar* has been run.

8 Case Study: Wave Rider

To further demonstrate the capability of *Rocstar* to simulate hypersonic flow, a 2D cold flow simulation of a waverider vehicle at Mach 7 was performed. The geometry of the Brazilian 14-X B waverider was used, based on the wind tunnel testing of [1] and computational studies of [2]. The simulation results from *Rocstar* obtained using the *rocRhoCentral* solver, show complex interactions between shock waves and with the viscous boundary layer. Future iterations of the case will include turbulence modeling, chemical species transport and reactions, and ablation of the leading edges as well as FSI.

8.1 Problem Setup

The waverider geometry is shown in Figure 49. The angles of the first and second compression ramps were 5.5° and 20° , and the angles of the first and second expansion ramps were 4.27° and 15° (relative to V_∞). The geometry was adapted to have $r = 1$ mm and $r = 2$ mm rounded leading edges on the main body and cowl, respectively. The radius tangent internal method was used for rounding so as not to change the angles of the geometry and thus location of the shocks. As exact dimensions were not available, the cowl was chosen to be 5 mm thick with a 5.5° ramp on the leading edge, similar to the main body.

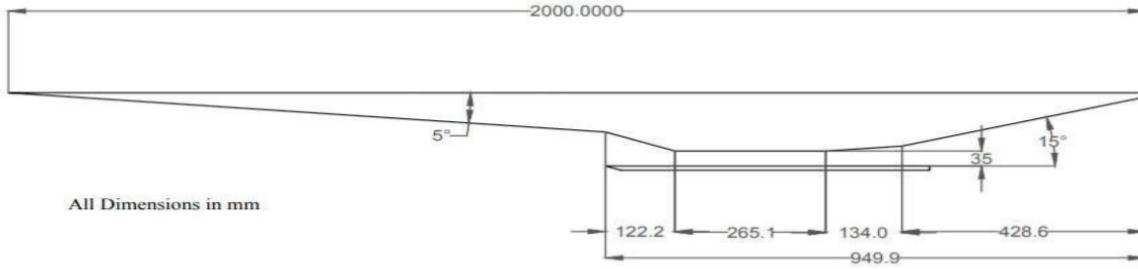


Figure 49: Geometry of the 14-X B waverider.

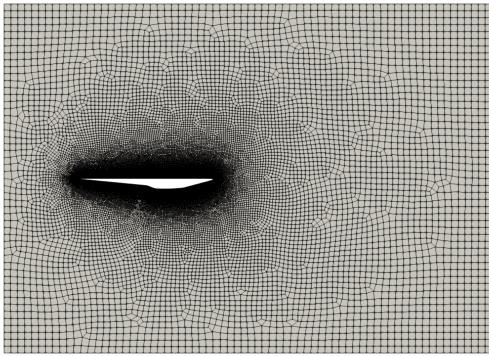


Figure 50: View of the fluid mesh.

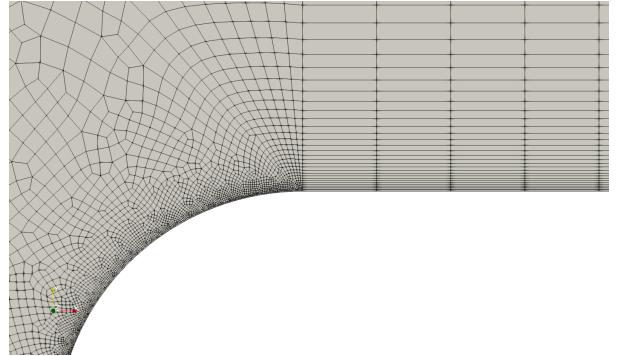


Figure 51: Detail of mesh at leading edge.

8.1.1 Mesh

The entire mesh is shown in Figure 50. The boundaries of the mesh extended 0.5 vehicle lengths in front of the waverider, 3 lengths behind, and 1.25 lengths above and below. As shown in Figure 51, the mesh was graded refined near the surface in order to capture the viscous boundary layer.

8.1.2 Materials

While the solid mechanics (and thus the solid material properties) were not of interest in this simulation, for future FSI analysis the materials were specified. The external surface of the waverider is composed of carbon-carbon with Inconel 718 used for the cowl and combustion chamber. In wind-tunnel testing, a stainless steel model was used. The waverider is intended to operate at Mach 7 and at 30 km altitude, yielding the air properties shown in Table 11.

Table 11: Properties of freestream air used in the waverider simulation.

Fluid	Mol. Weight [g/mol]	c_p [kJ/kg K]	U [m/s]	p [Pa]	T [K]
Air	28.9647	1004.9	2111.9	1197	226.5

For the initial inviscid simulation, viscosity was set to $\mu = 0$ kg/m s. Once the inviscid simulation was stable at the correct inlet velocity, viscosity was calculated via Sutherland's law:

$$\mu = \frac{A_s T^{3/2}}{T + T_s}, \quad (1)$$

with $A_s = 1.458e - 6$ and $T_s = 110.4$.



Table 12: List of fluid solver boundary conditions.

Quantity	Body	Inlet	Far	Front/Back
U [m/s]	noSlip	(2111.9 0 0)	zeroGradient	Empty
p [Pa]	zeroGradient	1197	zeroGradient	Empty
T [K]	zeroGradient	226.5	zeroGradient	Empty

8.1.3 Boundary Conditions

Boundary conditions are shown in Table 12. The “Body” boundary includes both the main body of the waverider and the cowl, and the “Far” boundary includes the top, bottom, and outlet of the computational domain. The *OpenFOAM* “empty” boundary condition was applied to the front and back of the domain, as this was a 2D simulation.

8.2 Input Files

The simulation was performed using the *rocRhoCentral* fluid solver.

8.2.1 Fluid Input Files

The basis *OpenFOAM* file structure is used in this case, including `0`, `constant`, and `system` directories.

8.2.2 Rocstar

The *Rocstar* input file `RocstarControl.txt` for the wave rider case is shown in Listing 41. Note that the `FluidAlone` coupling scheme was used, as there was no FSI. Additionally, even though `Rocburn` is specified as the `BurnModule`, the combustion solver is not used and therefore neither a `RocburnAPN/` directory or `RocburnAPNControl.txt` is required.

Listing 41: The `RocstarControl.txt` input file.

```
1 CouplingScheme = FluidAlone
2 FluidModule = rocrhocentral
3 BurnModule = RocburnAPN
4 OutputModule = Rocout
5 InitialTime = 0.0
6 MaximumTime = 0.001
7 ZoomFactor = 1.0
8 AutoRestart = F
9 CurrentTimeStep = 1.0e-06
10 OutputIntervalTime = 1.0e-5
11 MaxWallTime = 863000
12 ProfileDir = Rocman/Profiles
```

8.3 Running a *Rocstar* Case

As a convenience to the user, an `Allrun` bash script is provided with the case files to automatically preprocess the input files and execute the simulation. The script is configured to run the simulation in parallel on 8 cores, but this can be adjusted by changing the parameters `np` in `Allrun` and



numberOfSubdomains in FLUID/system/decomposeParDict.

As an alternative to running the bash script, the user can manually perform the simulation by following the steps below:

1. Preprocessing fluid input files
2. Configuring the file structure
3. Executing *Rocstar*

For this case study, we assume that \$CASE_DIR_PATH = absolute/path/to/case/directory/. We further assume that the system PATH contains the paths to the *RocFOAM* and *Rocstar* executables. To use the commands in the following sections directly, it is recommended that the user execute the commands in Listing 42, adjusting the paths shown in red to the specified directories:

Listing 42: Commands for running *Rocstar* with *RocFOAM*.

```
$ export LD_LIBRARY_PATH=/path/to/IMPACT-install/lib/:${LD_LIBRARY_PATH}
$ export LD_LIBRARY_PATH=/path/to/rocfoam-install/lib/:${LD_LIBRARY_PATH}
$ export CASE_DIR_PATH=/path/to/Rocstar-install/<case>
$ export PATH=/path/to/rocfoam-install/bin/:$PATH
$ export PATH=/path/to/IMPACT-install/bin/:$PATH
$ export PATH=/path/to/Rocstar-install/bin/:$PATH
```

8.3.1 Preprocessing Fluid Input Files

The files for the waverider case are provided in the correct structure. The user must first set the *OpenFOAM* environment:

```
$ . /usr/lib/openfoam/openfoam2006/etc/bashrc
```

From within the FLUID directory, the user should execute:

```
$ decomposePar
```

Note that, as mentioned in the previous section, this case is configured to run on 8 processors in parallel. Once the domain has been divided, the preprocessing command can be run:

```
$ mpirun -np 8 rocRhoCentral -preprocess -parallel
```

8.3.2 Configuring the *Rocstar* File Structure

The next step is to copy the contents of FLUID/ROCFOAM, which contain the *Rocstar* input CGNS and text files, to the proper location within the *Rocstar* hierarchy. Assuming that the user has a typical file structure (such as in Figure 15), the commands are as follows:

```
$ cd ../ROCSTAR/rocrhocentral/Rocin
$ cp ../../FLUID/ROCFOAM/* .
$ cd ../..
```



8.3.3 Executing *Rocstar*

To run *Rocstar*, the following command is executed from within the `<case>/ROCSTAR` directory:

```
$ mpirun -np 8 rocstar
```

Even though the simulation was initialized with a good solution, it may take some time to reach a quasi-steady state. While running, the simulation progress can be viewed by using *Paraview* or your favorite visualization tool to display the *OpenFOAM* files in the `fluidTmp` directory. *Paraview* requires a `.foam` file to be present in the `fluidTmp` directory. This can be created by executing the following command:

```
$ touch <case>.foam
```

To reconstruct the decomposed domain, copy the `polyMesh` directory from `FLUID/constant` to the `ROCSTAR/fluidTmp/constant` directory and execute from within the `fluidTmp` directory:

```
$ reconstructPar
```

CGNS files are written to the `rocrhocentral/Rocout/` directory. A different version of *Paraview*, such as v5.7, may be required to view these files. Visualizations of the simulation results are shown in Figure 52 and Figure 53. The Mach number was computed using:

$$M = |U|/\sqrt{\gamma RT}, \quad (2)$$

where $\gamma = 1.4$ and $R = 287 \text{ J/kg K}$. Another possibility is to calculate the density gradient maxima:

$$\frac{d\rho}{dn} = \nabla\rho \frac{U}{|U|}, \quad (3)$$

as suggested by [3]. This allows the user to visualize the shock lines around the waverider.

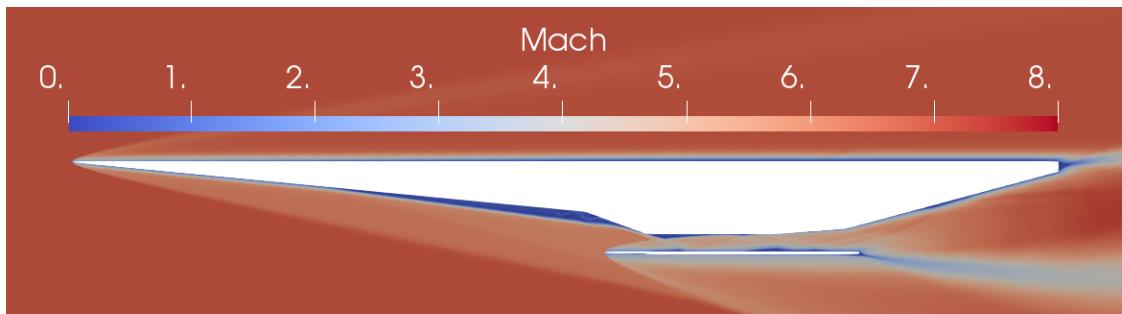


Figure 52: Mach contours.

9 Questions and Feedback

The software released under this contract is at its early alpha-release stage. Issues and bugs are expected during the operation. The IR team is constantly improving the software product and will periodically release it to several users. We highly appreciate your comments, questions, and requests for additional features. We encourage users to contact us using following contact information.

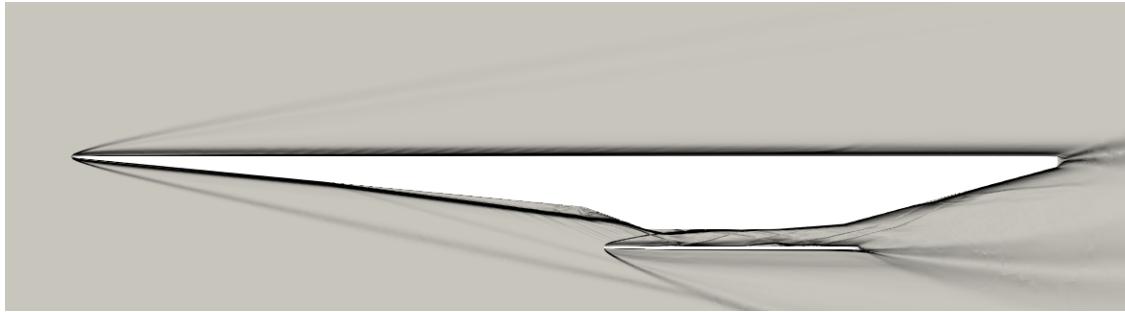


Figure 53: Density gradient representing shock lines.

9.1 Support Team

rocstar@illinoisrocstar.com

9.2 Company Contact Information

ATTN: Rocstar Development Team
Illinois Rocstar LLC
108 Hessel Blvd
Champaign, IL 61820

References

- [1] Paulo G. P. Toro, Marco A. Minucci, Tiago Rolim, Roberto Follador, Alberto Santos, Giannino Camillo, and Luiz Barreta. *Brazilian 14-X Hypersonic Aerospace Vehicle Project*.
- [2] S. N. P. LAITON, D. ROMANELLI PINTO, A. C. OLIVEIRA, and P.G.P. TORO. Numerical Study of Flow Characteristics of the 14-X B Hypersonic Aerospace Vehicle Inlet with Blunted Leading Edge. In *European Conference for Aeronautics and Space Sciences*, Milan, 2018.
- [3] Ziniu Wu, Yizhe Xu, Wenbin Wang, and Ruifeng Hu. Review of shock wave detection method in cfd post-processing. *Chinese Journal of Aeronautics*, 26(3):501–513, 2013.