ILLINOIS ROCSTAR

Contract Number DE-SC0018481

Rocstar

(Version 5.0.2)

User's Manual

Illinois Rocstar LLC
108 Hessel Boulevard, Unit 101
Champaign, Illinois 61820

Last Updated: January 5, 2021

Table of Contents

1.0 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. *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[1] 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. *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] Source code and manuals are available at https://github.com/IllinoisRocstar/IMPACT

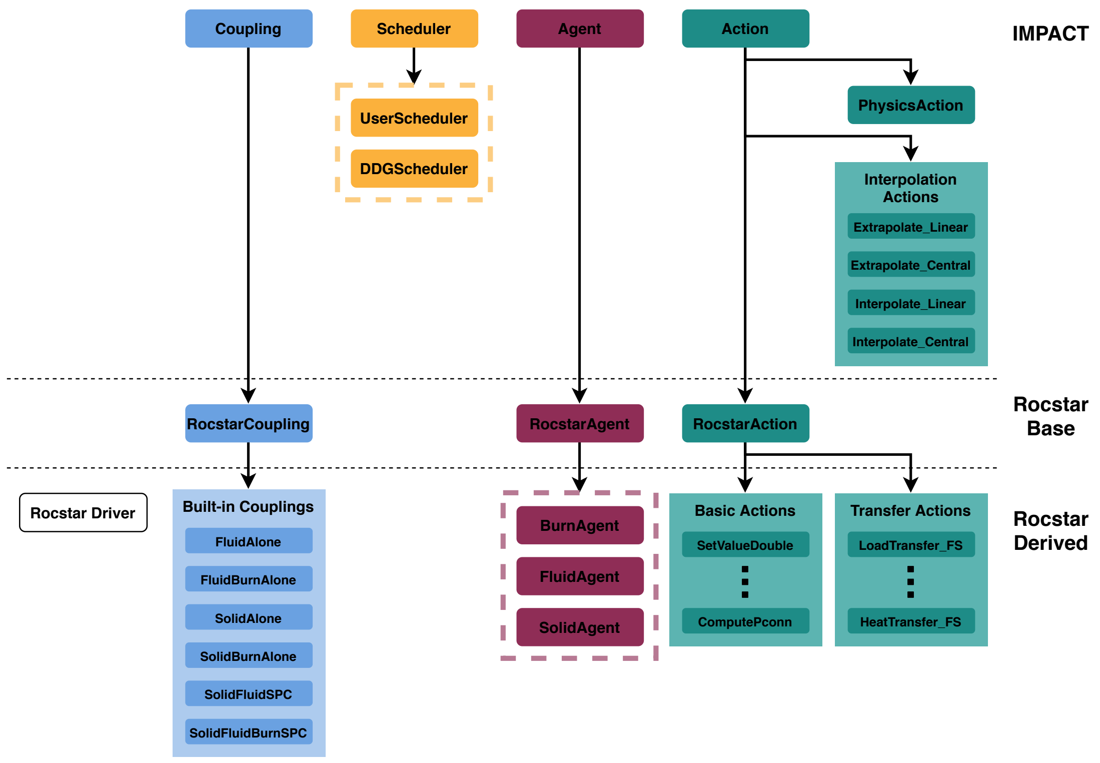

Figure 1. Architecture of the generalized SIM module (part of IMPACT library), Rocstar base and derived classes.

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.
    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 Component-side Client in *UserFoo's* library.
    a. Implement `UserFoo_load_module` and `UserFoo_unload_module` (see *COM* User's Guide).
    b. Using *COM* API, create *UserFoo*'s ComponentInterface and register *UserFoo*-native data and functions as needed.
3. Implement a driver (optional, for testing and development purposes only) making sure it does the following:
    a. Initializes MPI, if necessary
    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. *SIM* may be optionally used to create multiphysics drivers, or the driver can be entirely designed by the user. Again, the *COM* API is used to access integrated applications’ 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

1.4.1 *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.

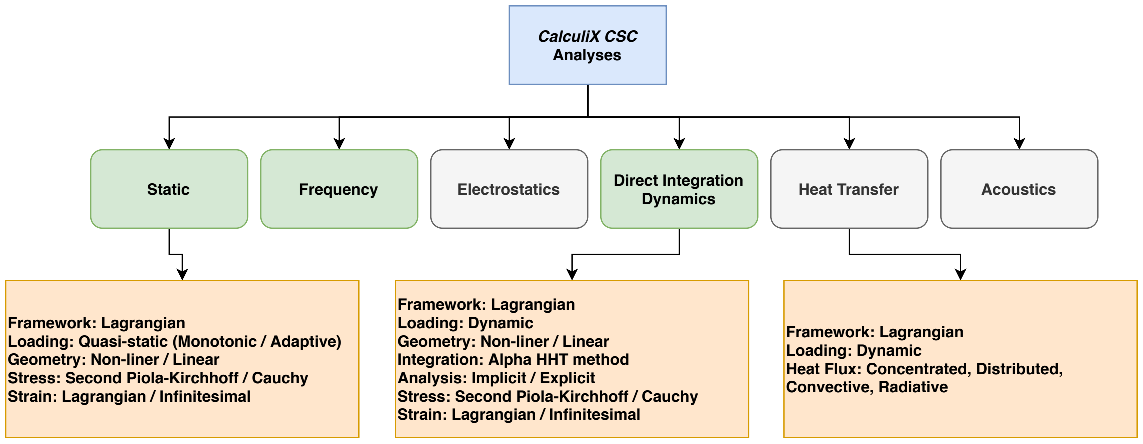
CalculiX CSC Analyses
Static
Frequency
Electrostatics
Direct Integration Dynamics
Heat Transfer
Acoustics
Framework: Lagrangian
Loading: Quasi-static (Monotonic / Adaptive)
Geometry: Non-liner / Linear
Stress: Second Piola-Kirchhoff / Cauchy
Strain: Lagrangian / Infinitesimal
Framework: Lagrangian
Loading: Dynamic
Geometry: Non-liner / Linear
Integration: Alpha HHT method
Analysis: Implicit / Explicit
Stress: Second Piola-Kirchhoff / Cauchy
Strain: Lagrangian / Infinitesimal
Framework: Lagrangian
Loading: Dynamic
Heat Flux: Concentrated, Distributed, Convective, Radiative

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* [24]. 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* can be used.

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 integrated into *Rocstar*. Two compressible flow solvers, *rocRhoCentral* and *rocRhoPimple*, currently comprise *rocfoam*.

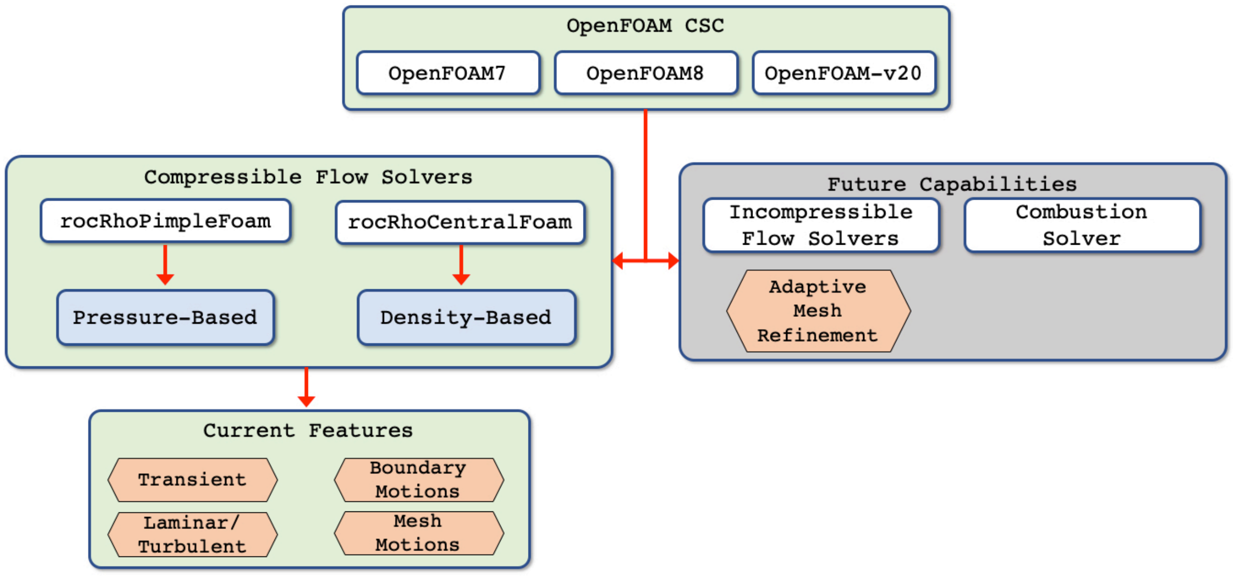

Figure 3. Architecture of the *rocfoam* module and its current/future functionality.

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

1.6.1 *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 below:

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

The flags used in the example command 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>`.

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 `-t <path>`
`-P`	Run module preptools 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>`	*surfdive* interface meshes, default infers 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 preptool binaries, default will use shell path
`-x`	ignore RocprepControl.txt control file

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

2.0 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 16.04 and *CentOS* version 7.5.1804.

Warning

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
- libsm-dev
- libice-dev
- gfortran
- libboost-all-dev
- libxt-dev
- zlib1g-dev
- tcl-dev
- tk-dev
- libxmu-dev
- python-dev

As a single command:

```
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 libsm-dev libice-dev gfortranlibboost-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* uses *METIS* Version 4.0.3[2]. The default build configuration described in the build instructions included in the package is sufficient.

2.1.2 Building *IMPACT*

IMPACT, the **I**llinois Rocstar **M**ultiphysics **A**pplication **C**oupling **T**oolkit, 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:

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

Executing the commands above will build all libraries and executables for *IMPACT*.

2.1.3 Installing *Rocstar Multiphysics*

The modernized *Rocstar Multiphysics* is 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

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`)

[2] http://glaros.dtc.umn.edu/gkhome/fetch/sw/metis/OLD/metis-4.0.3.tar.gz

The build process is started by executing:

```
$  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
```

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 repositories[3].

- 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)

As a single command:

```
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* uses *METIS* Version 4.0.3[4]. The default build configuration described in the build instructions included in the package is sufficient.

[3] https://fedoraproject.org/wiki/EPEL
[4] http://glaros.dtc.umn.edu/gkhome/fetch/sw/metis/OLD/metis-4.0.3.tar.gz

The *CGNS* library version 3.2.1 is required. It is available from GitHub[5] 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*

IMPACT, the **I**llinois Rocstar **M**ultiphysics **A**pplication **C**oupling **T**oolkit, is the software 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:

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

Executing the commands above will build all libraries and executables for *IMPACT*.

2.2.3 Installing *Rocstar Multiphysics*

The modernized *Rocstar Multiphysics* is 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

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

[5] https://github.com/CGNS/CGNS/releases/tag/v3.2.1

- `$METIS_LIB_PATH` is the path to the location of the *METIS* library (`/path/to/metis/libmetis.a`)
- `$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:

```
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
```

3.0 Setting Up an FSI Simulation

Currently two case studies will be 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) and is discussed in more detail in Section 4.0 The second will use a different fluid solver in a coupled fluid-solid simulation of a wind turbine that is discussed in more detail in Section 5.0

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

Currently, *Rocstar* requires that case files be set up in a specific file hierarchy, as 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, containing the simulation data and the associated grid data.

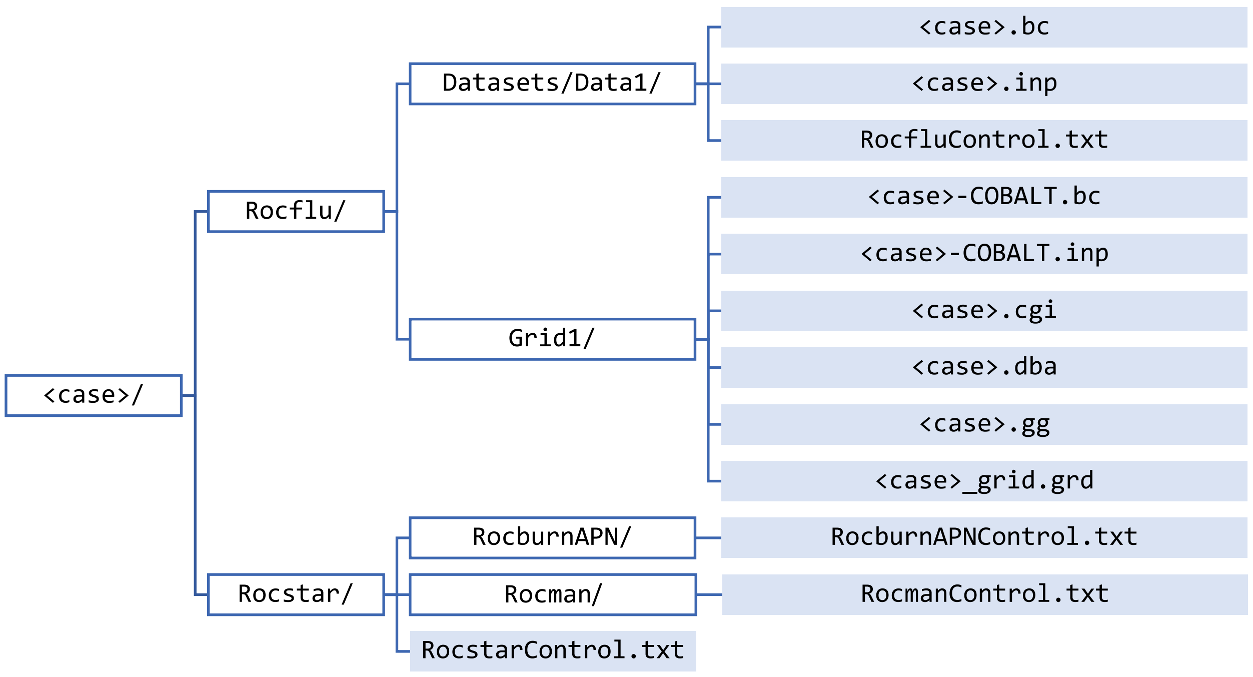

Figure 4. NDA for a FluidBurnAlone case named <case>, using *Rocflu* and *RocburnAPN*.

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

3.1.2 Control Files

Control files are typically short text files that specify simulation parameters to *Rocstar* and its solvers.

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*: FullyCoupled no burn with simple staggered scheme with predictor-corrector iterations.
  - *SolidFluidBurnSPC*: FullyCoupled with burn with simple staggered scheme with predictor-corrector iterations.
- Derived:
  - *SolidFluidISS*: FullyCoupled no burn with improved staggered scheme
  - *SolidFluidBurnEnergySPC*: FullyCoupled 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 below.

```
# SECTION_NAME
KEYWORD_1 VALUE_1
KEYWORD_2 VALUE_2
KEYWORD_3 VALUE_3
#
```

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)
  - `WRITIME`: 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`**[6]
  - `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
  - `WRITIME`: 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/m^3] (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)

[6] The recommended values for non-`MESQUITE` smoothing are `TYPE=1`, `NITER=4`, `SFACT=0.25` or `TYPE=2`, `NITER=10`, `SFACT=0.1`

  - `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 below.

```
# SECTION_NAME
PATCH PATCH_1 PATCH_2
KEYWORD_1 VALUE_1
KEYWORD_2 VALUE_2
#
```

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/m^{2}s]
  - `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, withing $xy/xz/yz$-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, withing $xy/xz/yz$-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, withing $xy/xz/yz$-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 5.0 which couples the new fluid solver *rocfoam* and the structural solver *CalculiX*, is used as an example.

3.2.1 Writing *rocfoam* Input Files

The fluid solver *rocfoam* is based on *rhoPimpleFoam* and follows the *OpenFOAM* input file structure. For a general guide on setting up *OpenFOAM* simulations, see the user guide[7]. 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 like 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 that contains the initial conditions. Running the simulation will result in the creation of additional time directories, each containing the field values for that time.

[7] https://www.openfoam.com/documentation/user-guide/

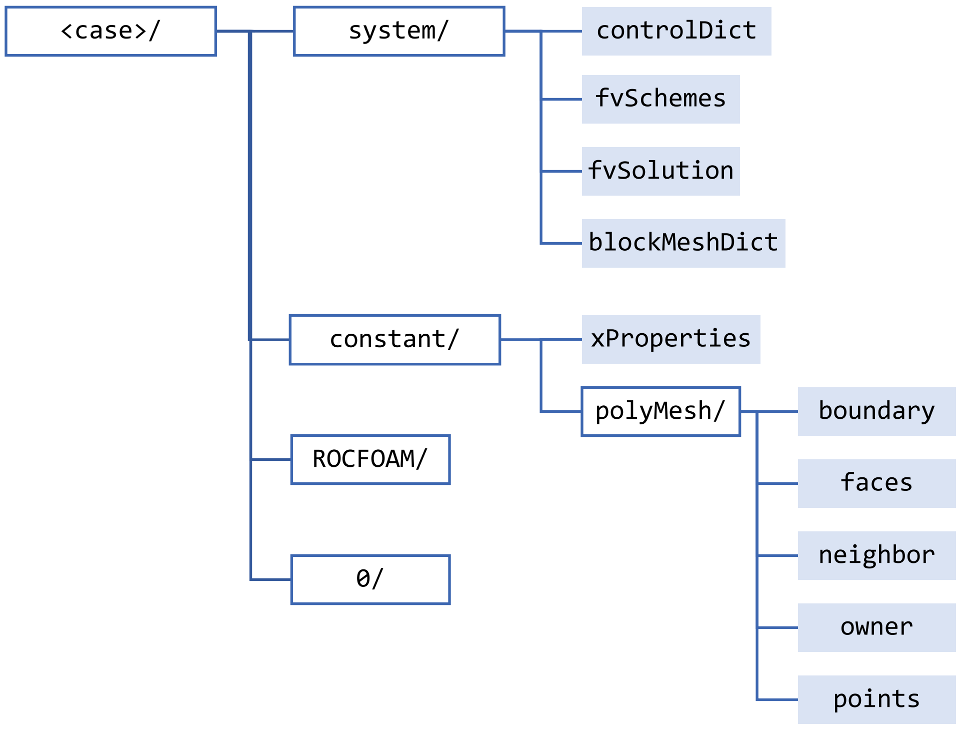

Figure 5. Initial *rocfoam* file hierarchy. Running the case will spawn additional time directories, each containing copies of the time dependent properties

An example of the format used in *OpenFOAM* input files is shown below.

Listing 1. Example controlDict file.

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  5                                     |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
application     simpleFoam;
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         100000;
deltaT          0.00001;
writeControl    adjustableRunTime;
writeInterval   0.001;
```

```
purgeWrite      0;
writeFormat     binary;
writePrecision  6;
writeCompression off;
timeFormat      general;
timePrecision   6;
runTimeModifiable true;
adjustTimeStep yes;
maxCo 0.9;
// ************************************************************************* //
```

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.

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 (°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 follows:

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

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

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 guide[8]. Although the example deck in Listing 2 strategically uses upper- and lower-case letters, this is done for readability—*CalculiX* input is case insensitive.

Listing 2. Example *CalculiX* input file, named turbine_metric.inp.

```
**
**   Structure: 1/3 wind turbine FSI model
**   Test objective: test FSI capabilities of the CSC module
**
*HEADING
Model: WINDTURBINE     Date: 17/06/2020
*INCLUDE, INPUT=hollow_solid_mesh.inp
```

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

```
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 following degrees of freedom are allowed:

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

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

The `*MATERIAL` card indicates the start of a material definition. The material definition block in Listing 2 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 following variables may be selected:

Table 4. Nodal variables that may be saved for solids.

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, and the `FREQUENCY` parameter specifies how often the data will be saved. The following element variables are available:

Table 5. Element variables that may be saved for solids.

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 Table 5.

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

4.0 Case Study: Attitude Control Motor Rocket Test

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.

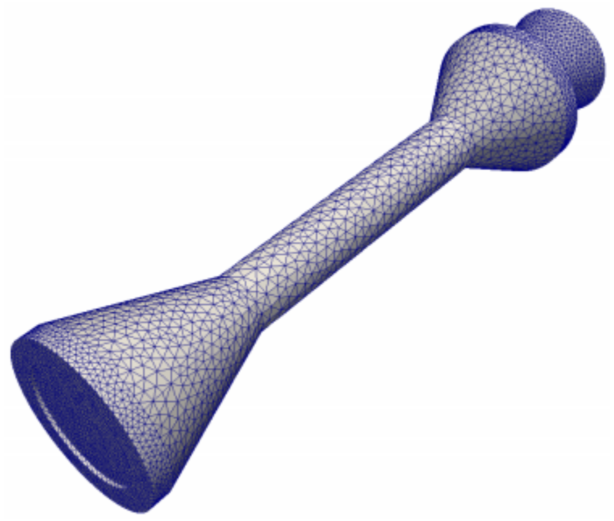

Figure 6. ACM model.

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. Simulation parameters are given in Table 6.

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
	ρ	Propellant density	1703.0 kg m^{-3}
Initial Conditions	P	Initial fluid pressure	1.0×10^5 Pa
	ρ	Initial fluid density	1.16 kg m^{-3}
	T_{init}	Initial fluid temperature	298.0 K
Boundary Conditions	$\dot{m_{inj}}$	Injection mass flow	5.7429 kg m^{-2} s^{-1}
	T_{inj}	Injection temperature	2855 K

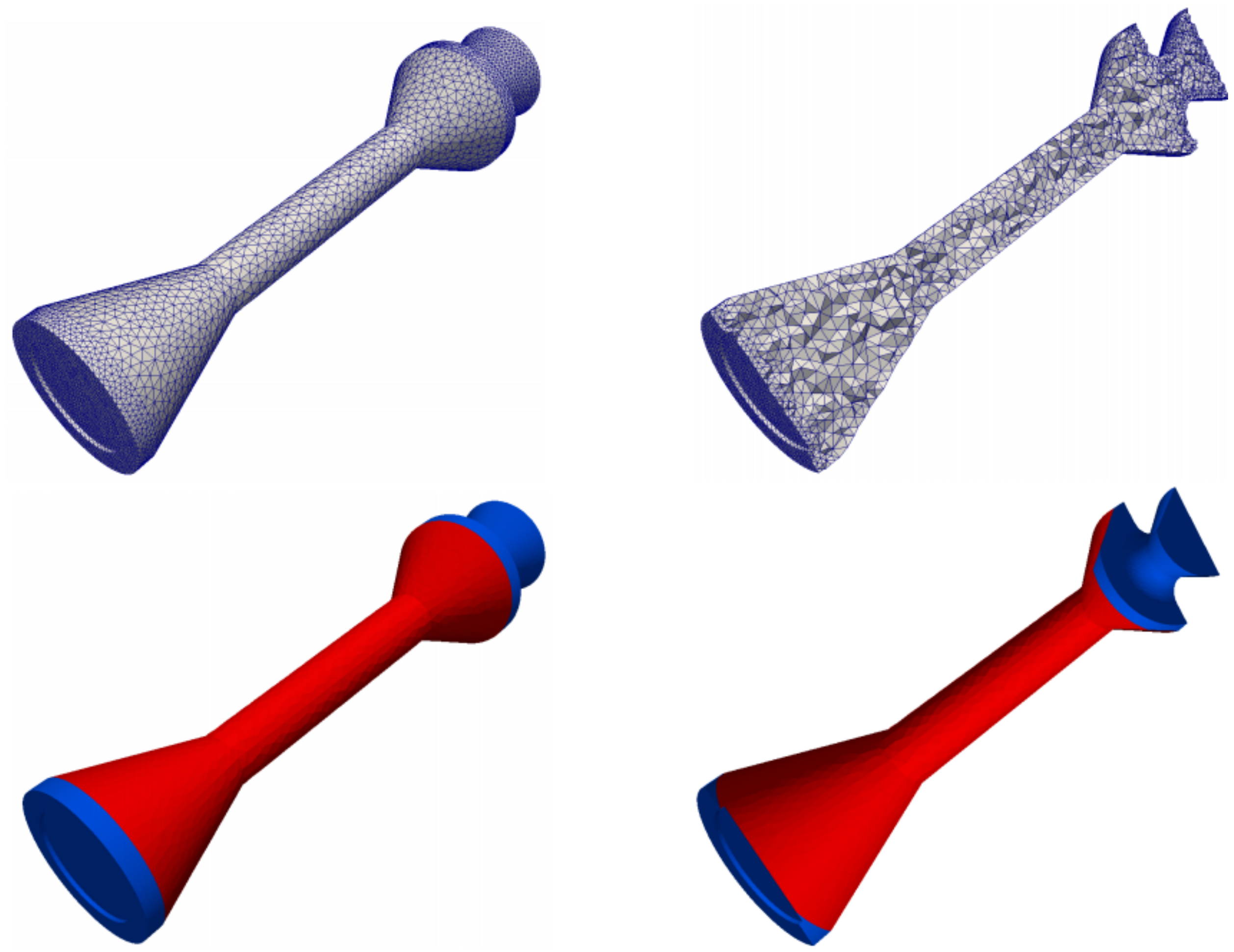

Figure 7. (TOP) ACM volumetric grid. Crinkle-cut highlights cells are tetrahedra. (BOTTOM) ACM surface grid. Burning surfaces are red, non-interacting faces are shown in blue.

4.2 Input Files

The ACM case uses *Rocflu* and *Rocburn* with the NDA file hierarchy, as 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.

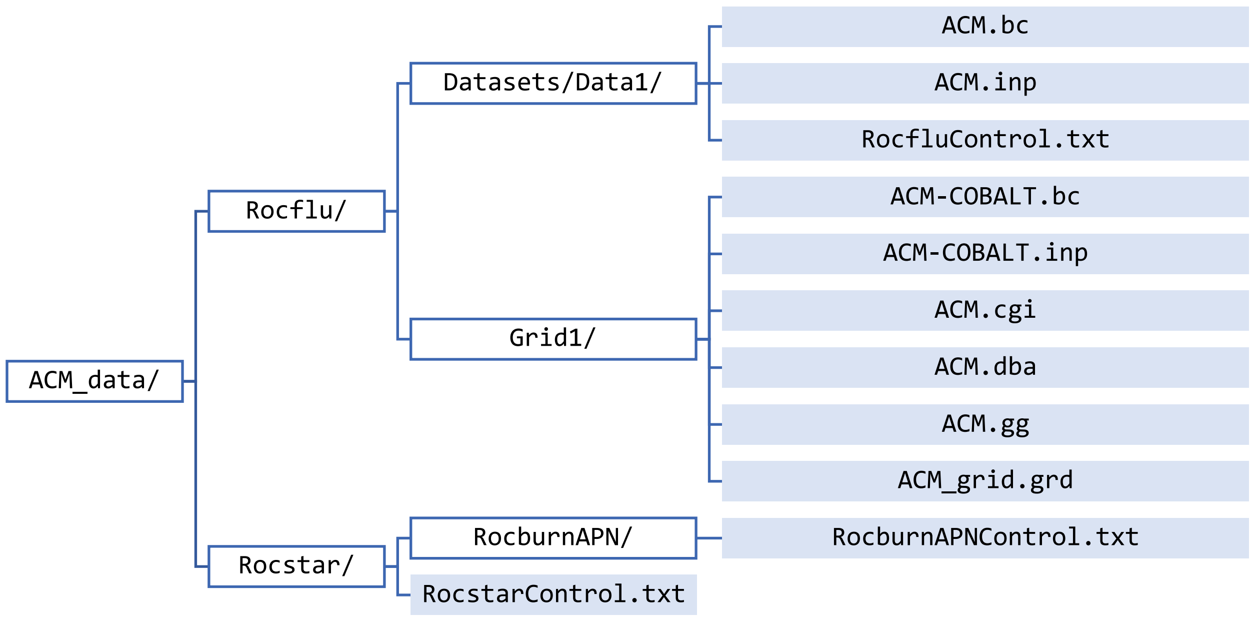

Figure 8. NDA File directory for ACM_data. Files for deprecated modules not included.

4.2.1 Control Files

Rocstar uses a text file located within the `Rocstar/` directory to specify FSI simulation parameters. Listing 3 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 3. RocstarControl.txt

```
CouplingScheme = FluidBurnAlone
FluidModule = Rocflu
BurnModule = RocburnAPN
OutputModule = Rocout
MaximumTime =  2.0e-6
ZoomFactor = 1.0
AutoRestart = F
CurrentTimeStep = 1.0e-07
OutputIntervalTime = 1.0E-07
MaxWallTime = 86300
ProfileDir = Rocman/Profiles
```

The RocfluControl.txt file 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 4. RocfluControl.txt

```
ACM
Rocflu/Modin
Rocflu/Modout
Rocflu/Rocin
1
1
```

The RocburnAPNControl.txt file 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 5. RocburnAPNControl.txt

```
0.77       a in rb=a*P^n,  rb in cm/sec and P in atm, a_p (cm/sec)
0.62       n in rb=a*P^n,  rb in cm/sec and P in atm, n_p
1          Maximum_number_of_spatial_nodes,_nxmax
2916.0     adiabatic flame temperature, Tf_adiabatic (K)
298.00     initial temperature        , To_read      (K)
Rocburn_2D_Output/Rocburn_APN location of the Rocout files

Solid Propellant Properties for ARC Attitude Control Motor (ACM) per
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 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're 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 6. RocmanControl.txt

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

Listing 7. ACM.inp

```
# FORMATS
GRID       0  ! 0 - ROCFLU ASCII, 1 - ROCFLU binary, 2 - ROCFLU HDF
SOLUTION   0  ! 0 - ROCFLU ASCII, 1 - ROCFLU binary, 2 - ROCFLU HDF
GRIDSRC    4  ! 0 - CENTAUR ASCII, 1 - VGRIDNS, 2 - MESH3D
#

# FLOWMODEL
MODEL    0    ! 0 - Euler, 1 - Navier-Stokes
MOVEGRID 1    ! 0 - static grid, 1 - moving grid
#

# NUMERICS
CFL      1.0 ! CFL number
DISCR    3      ! Type of space discretization (1 - Roe, 2 - MAPS)
ORDER    2      ! Order of accuracy (1 - first, 2 - second)
ENTROPY  0.05   ! Entropy correction coefficient (if DISCR=1)
#

# TIMESTEP
FLOWTYPE  1  ! 0 - steady flow, 1 - unsteady flow
TIMESTEP  0.000001 ! Max. physical time step
STARTTIME 0.0 ! Current iteration
MAXTIME   0.2 ! Maximum number of iterations
WRITIME   0.001 ! Offset between iterations to store solutions
PRNTIME   0.00000005 ! Offset between iterations to print convergence
DTMINLIMIT 5.0E-09 ! Timestep below which debug info will be printed
#

# GRIDMOTION
TYPE   3     ! 1 – smooth boundary disp., 2 – smooth coordinates, 3 - MESQUITE
NITER  10    ! Number of smoothing iterations (for TYPE 1 or TYPE 2 only)
SFACT  0.25  ! Smoothing coefficient (for TYPE 1 or TYPE 2 only)
#

# REFERENCE
GAMMA  1.2444  ! Ratio of specific heats
CP     1905.0  ! Specific heat coefficient at constant pressure in J/kg K
#

# PROBE
NUMBER 2   ! Number of probes
0.001 0.0 0.0   ! x y z coordinates of probe 1
0.0495735 0.0 0.0    ! x y z coordinates of probe 2
#
WRITIME 0.000005  ! Offset in seconds at which data is written to probe files
OPENCLOSE 1 ! Close and open probe file after writing? (0-no, 1-yes)
#
```

```

# INITFLOW
FLAG    1  ! Generate init. solution with: 1–keywords, 2–file, 3–hardcode
DENS   1.16 ! Density of initial solution in kg/m3 (FLAG 1 only)
VELX   0.0  ! x-component of velocity of initial solution in m/s (FLAG 1 only)
VELY   0.0  ! y-component of velocity of initial solution in m/s (FLAG 1 only)
VELZ   0.0  ! z-component of velocity of initial solution in m/s (FLAG 1 only)
PRESS  1.0E+5 ! static pressure of initial solution in Pa (FLAG 1 only)
#

# TRANSFORM
FLAG 1  ! Scale or rotate grid? (0-no, 1-yes)
SCALE_X 0.0254  ! Scaling factor for x-component of coordinates
SCALE_Y 0.0254  ! Scaling factor for y-component of coordinates
SCALE_Z 0.0254  ! Scaling factor for z-component of coordinates
#

# PREP
SURFFLAG 1   !  Set to 1 if rfluprep was not compiled as part of GENx
#

# POST
PLTTYPE    1  ! 1-Write only grid to output file, 2-Write grid and solution
PLTVOLFLAG 0  ! Should volume data be written to output file (0-no, 1-yes)
MERGEFLAG  0  ! Merge regions from parallel computation? (0-no, 1-yes)
#

# ROCKET
CASERAD .0066167   ! Cylindrical case constraint radius
#

# TIMEZOOMING
MAXPLANE     0.047625   ! Max coordinate to apply zooming
MINPLANE     0.00   ! Min coordinate to apply zooming
#
```

Listing 8. ACM.bc

```
# BC_INJECT
PATCH   1 1   !  Range of patches this section applies to (from 1 to 1)
NAME   InjectionWall   !  Name of boundary
MFRATE  5.7429   !  Injection mass flux in kg/m2s
TEMP  2855.0   ! Injection static temperature in K
RFVFU  0.0
RFVFV   0.0
RFVFW   0.0
COUPLED  1   !  Is the patch interacting? (1-yes, 2-no)
BFLAG  1   !  Is the patch burning at t = 0? (0-no, 1-yes)
MOVEDIR  7   !  Vertices on patch may move in xyz-space
#

# BC_SLIPW
PATCH  2 2   !  Range of patches this section applies to
NAME  AftFlatWall   !  Name of boundary
```

```
COUPLED  2  !  Is the patch interacting? (1-yes, 2-no)
MOVEDIR 0   !  Vertices on patch may not move in any direction
#

# BC_SLIPW
PATCH  3 3  ! Range of patches this section applies to
NAME   HeadEndSurface   !  Name of boundary
COUPLED  2  !  Is the patch interacting? (1-yes, 2-no)
MOVEDIR  0   !  Vertices on patch may not move in any direction
#

# BC_OUTFLOW
PATCH   4 4   !  Range of patches this section applies to
NAME   NozzleOutlet   !  Name of boundary
TYPE    0   !  Specifies type of outflow (0-supersonic, 1-subsonic, 2-mixed)
COUPLED 2   !  Is the patch interacting? (1-yes, 2-no)
MVPATCH 0   !  Is the patch moving? (0-no, 1-yes)
SMGRID  0   !  Should the grid on the patch be smoothed? (0-no, 1-yes)
CORR    0   !  Should point-vortex correction be applied? (0-no, 1-yes)
MOVEDIR 0   !  Vertices on patch may not move in any direction
#

# BC_SLIPW
PATCH  5 5  !  Range of patches this section applies to
NAME   HeadEndRing !  Name of boundary
COUPLED      2  !  Is the patch interacting? (1-yes, 2-no)
MOVEDIR 0   !  Vertices on patch may not move in any direction
#

# BC_SLIPW
PATCH  6 6  ! Range of patches this section applies to
NAME   AftEndRing   !  Name of boundary
COUPLED      2  !  Is the patch interacting? (1-yes, 2-no)
MOVEDIR 0   !  Vertices on patch may not move in any direction
#

# BC_SLIPW
PATCH   7 7  ! Range of patches this section applies to
NAME   NozzleSurface   !  Name of boundary
COUPLED  2  !  Is the patch interacting? (1-yes, 2-no)
MOVEDIR  0   !  Vertices on patch may not move in any direction
#

# END
```

4.3 Run Instructions

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.

4.4 Output Files

4.4.1 *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 × 10^{xx} × 10^{-9} s. For example, at a simulation time of t = 1.2 × 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, as well as the `fluid` filenames associated with them.
- `ifluid_in_xx.yyyyyy.txt`: *Rocflu* file that tells `Rocin` which panes belong to each process, as well as 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 `mmmmm` 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

4.4.2 *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.

5.0 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

5.1.1 Geometry

The blade is a 5.532 m long NREL S809 tapered and twisted airfoil[9]. 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).

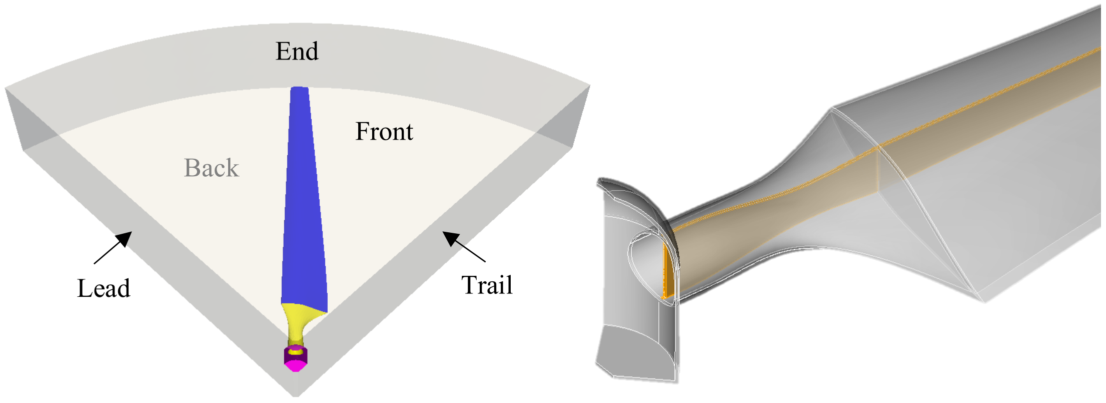

Figure 9. (left) 3D model of single S809 profile wind turbine blade. The fluid region is the transparent 120° pie-shaped domain. The active FSI zone of is shaded in blue, and the connector (blade-to-hub) surface is shaded in yellow. The hub surface is shaded in magenta. (right) Transparent view of solid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

5.1.2 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 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

[9] M.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.

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.

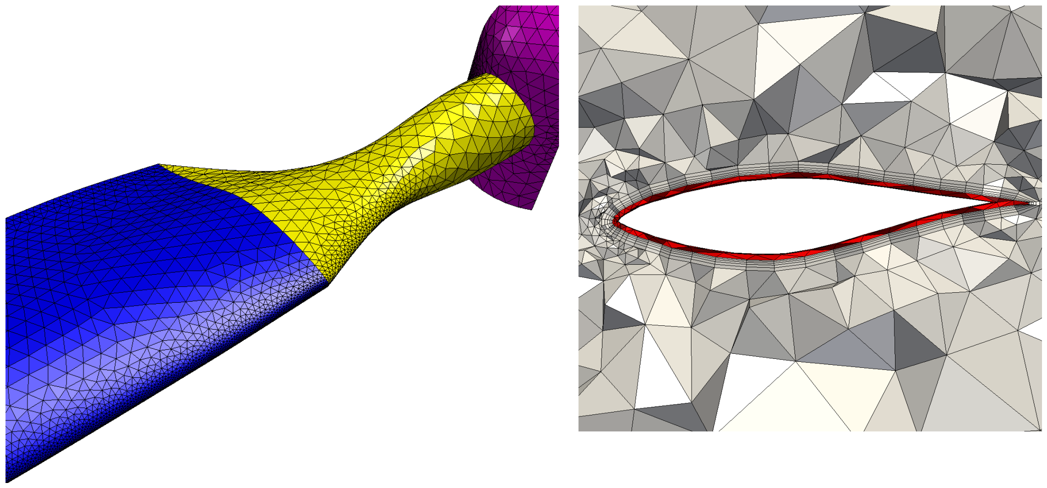

Figure 10. Surface mesh of the hub, blade and connector sections. The inset shows a crinkled slice of the volume mesh highlighting fluid (white) and solid (red) sections.

5.1.3 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
CFRP	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.

5.1.4 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 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 etc.	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

5.1.5 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.

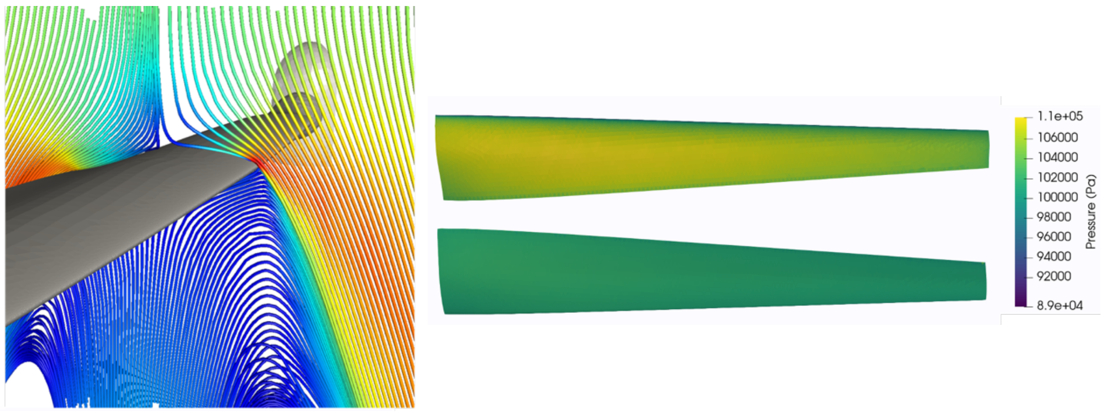

Figure 11. (left) Flow streamlines and (right) distribution of wind pressures for normal atmospheric conditions on the forward and backward surfaces of the blade.

5.2 Input Files

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

5.2.1 Fluid

Rocfoam is based on the *OpenFOAM* solver and uses a similar file structure.

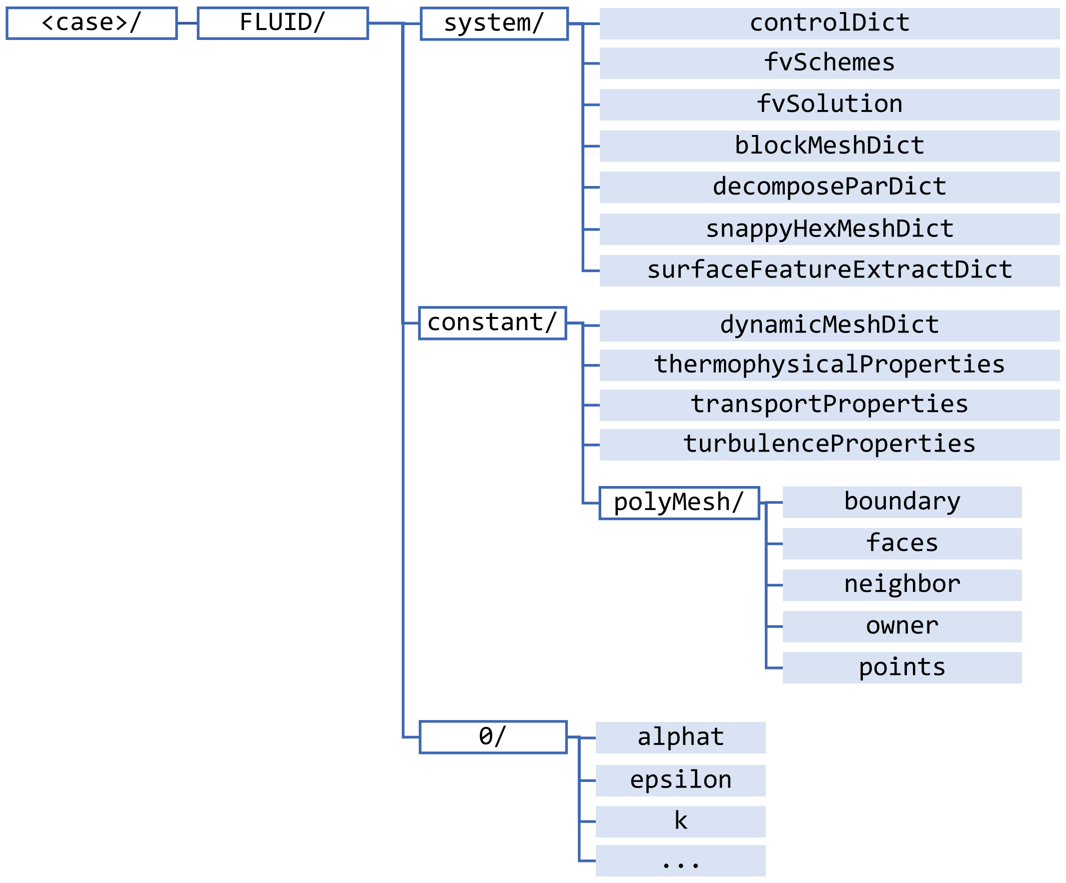

Figure 12. *Rocfoam* file structure. The system directory contains the dictionaries that specify solving methods, the constant directory contains the constant physical properties, and the initial time directory 0 contains the properties that evolve with time (truncated here to fit). As the problem runs,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

A selection of the files listed in Figure 12 will be explained below.

Listing 9. The file system/fvSchemes, which defines the finite volume solution methods.

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

```
ddtSchemes
{
    default Euler;
}
d2dt2Schemes
{
}
gradSchemes
{
    default         cellMDLimited Gauss linear 1.0;
    //grad(U)         leastSquares;
    //grad(U)         cellLimited Gauss linear 1;
    //grad(p)         cellLimited Gauss linear 1;
    //grad(h)         cellLimited Gauss linear 1;
    //grad(k)         cellLimited Gauss linear 1;
    //grad(omega)     cellLimited Gauss linear 1;
}
divSchemes
{
    default         none;
    div(phi,U)      Gauss linearUpwindV grad(U);
    div(phi,h)      Gauss upwind;
    div(phi,k)      Gauss upwind;
    div(phi,K)      Gauss upwind;
    div(phid,p)     Gauss upwind;
    div(phi,epsilon)  Gauss upwind;
    div(meshPhi,p)  Gauss upwind;
    div((phi|interpolate(rho)),p) Gauss upwind;
    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}
laplacianSchemes
{
    default         Gauss linear limited corrected 0.5;
}
interpolationSchemes
{
    default         linear;
}
snGradSchemes
{
    default         limited corrected 0.5;
}
wallDist
{
    method meshWave;
}
// ************************************************************************* //
```

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

```
FoamFile
{
```

```
3        version     2.0;
4        format      ascii;
5        class       dictionary;
6        location    "system";
7        object      fvSolution;
8    }
9    // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

This file, called `fvSolution`, 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. 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.

```
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.

```
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.

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

```
        tolerance       5e-4;
        relTol          0;
    }
```

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

```
    Phi
    {
        $p;
    }
```

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.

```
    "(rho|U|k|epsilon)"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-06;
        relTol          0.01;
        minIter         1;
    }
    cellDisplacement
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-06;
        relTol          0.1;
        minIter         1;
        maxIter        30;
    }
    h
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-06;
        relTol          0.01;
        minIter         1;
    }
    "(rho|U|h|k|epsilon)Final"
    {
        $U;
        relTol          0;
    }
    cellDisplacementFinal
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-06;
        relTol          0.0;
        minIter         1;
        maxIter         30;
    }
}
```

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.

```
potentialFlow
{
    nNonOrthogonalCorrectors 10;
}
```

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.

```
PIMPLE
{
    nOuterCorrectors    1;
    nCorrectors         1;
    nNonOrthogonalCorrectors 3;
    momentumPredictor   yes;
    transonic           no;
    consistent          no;
    simpleRho           no;
    pMax                250000;
    pMin                  7000;
    outerCorrectorResidualControl
    {
        "(U|k|epsilon|h)"
        {
             relTol    0.01;
             tolerance 1e-6;
        }
        p
        {
             relTol    0.01;
             tolerance 1e-6;
        }
    }
    turbOnFinalIterOnly no;
}
SIMPLE
{
    transonic           no;
    nNonOrthogonalCorrectors 3;
    consistent          no;
    simpleRho           no;
    pMax                150000;
    pMin                7000;
    pRefCell       0;
    pRefValue           101325;
    residualControl
    {
        "(U|k|epsilon|p|h)"  1e-6;
    }
}
relaxationFactors
{
    fields
    {
        "p.*"           0.5;
        "rho.*"         1.0;
        //"U.*"           0.9;
        //"h.*"           0.9;
        //"(k|epsilon).*"   0.9;
    }
    equations
    {
        "U.*"           0.5;
        "h.*"           0.5;
        "(k|epsilon).*"   0.5;
    }
}
// ********************************************************************** //
```

5.2.2 Solid

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 the following.

Listing 10. The file turbine_metric.inp, the *CalculiX* input file.

```
**
**   Structure: 1/3 wind turbine FSI model
**   Test objective: test FSI capabilities of the CSC module
**
*HEADING
Model: WINDTURBINE     Date: 08/11/2020
*INCLUDE, INPUT=blade_v4.inp
*BOUNDARY
hub_interior,1,3
*MATERIAL,NAME=GFRP
*ELASTIC
25E9,.38
*DENSITY
1.90E3
*SOLID SECTION,MATERIAL=GFRP,ELSET=Solid
*AMPLITUDE,NAME=A1
0.,0.,1.E-4,1.
1.1E-4,0.
*STEP,INC=100000,NLGEOM
```

```
*DYNAMIC,DIRECT
5e-5, 10, 1e-6, 1e-4
*DLOAD
FSI, P1, 0.0
*NODE PRINT,NSET=ALLNODES,FREQUENCY=1
U
*EL PRINT,ELSET=Solid,FREQUENCY=1
S
*NODE FILE
U, RF
*EL FILE
E, S
*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 of the case is shown in Figure 13.

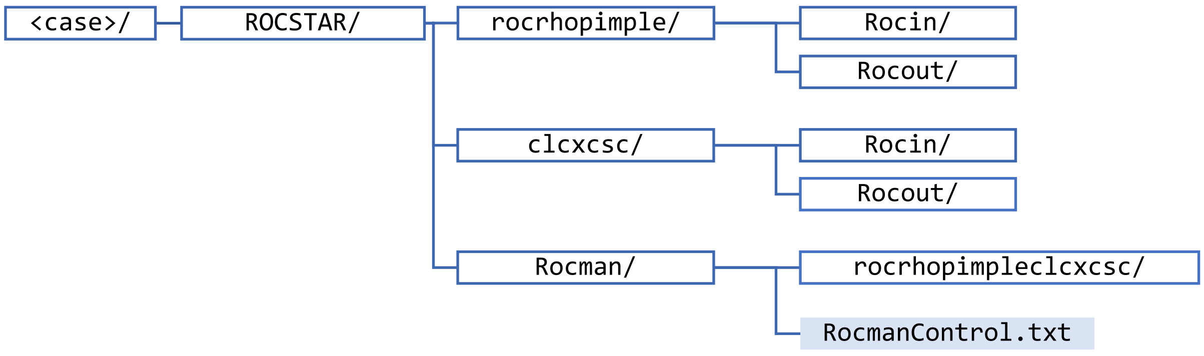

Figure 13. File directory for *Rocstar*. Note that the fluid and solid inputs have not been added to their respective Rocin directories.

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/`

It is also assumed 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 following commands, adjusting the paths to the specified directories.:

```
$  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 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.

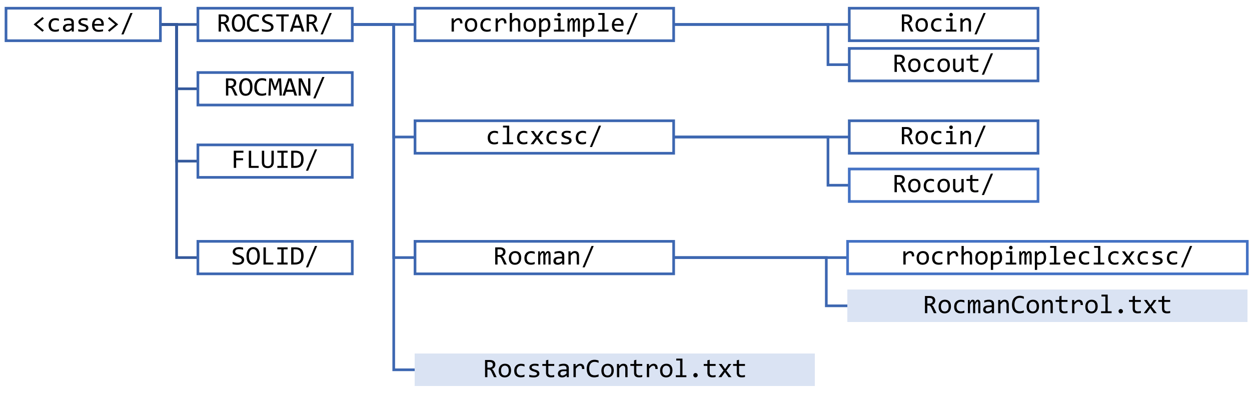

Figure 14. Full file hierarchy showing the directories and files that the user must manually create.

5.3.2 Preprocessing Fluid Input Files

This tutorial assumes the user has already generated the appropriate fluid meshes and variable fields as 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`. As shown in the following listing, this domain will be divided into four parts.

Listing 11. The system/decomposeParDict file, which specifies the number and method for decomposing a domain for parallel runs.

```
FoamFile
```

```
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
numberOfSubdomains 4;
method          scotch;
// ************************************************************************* //
```

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 containing 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 15 shows the file hierarchy *Rocstar* expects.

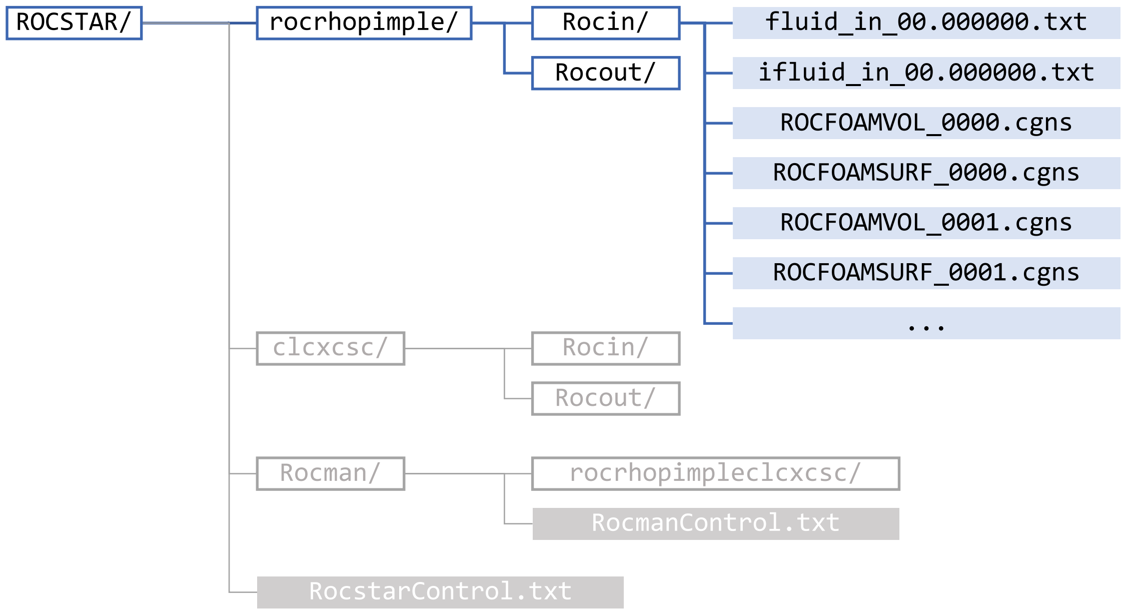

Figure 15. Preprocessed fluid solver input files (truncated for space) that have been copied from the FLUID/ROCFOAM folder into the correct Rocstar input directory.

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 10) 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 also 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/clcxcsc/Rocin
```

Figure 16 shows the file hierarchy *Rocstar* expects.

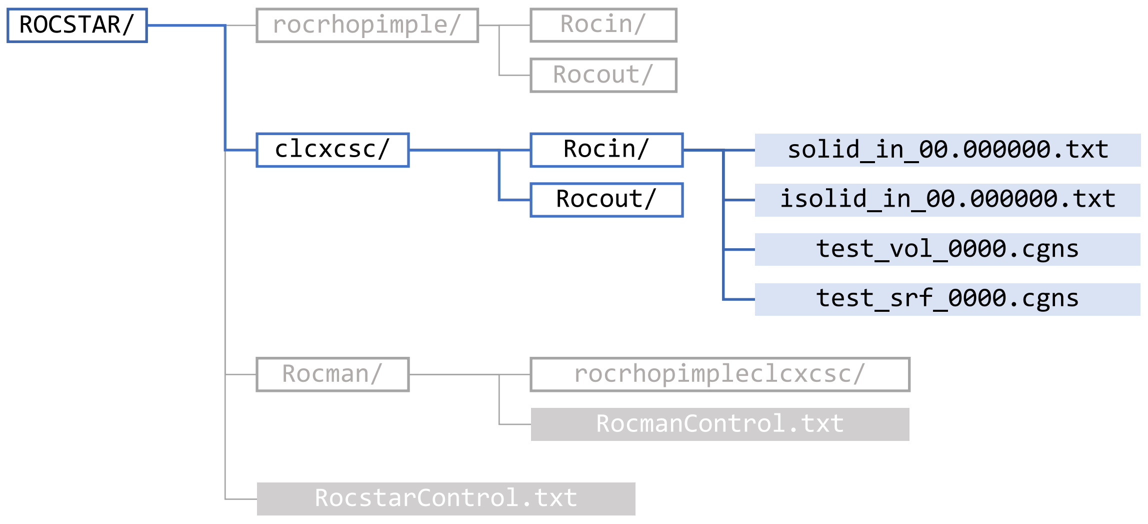

Figure 16. Preprocessed solid solver input files that have been copied form the SOLID folder into the correct Rocstar input directory.

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}/ROCSTAR/rocrhopimple/Rocin/ifluid_in_00.000000.txt \
   ${CASE_DIR_PATH}/ROCSTAR/clcxcsc/Rocin/isolid_in_00.000000.txt \
   ${CASE_DIR_PATH}/ROCSTAR/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 `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 17.

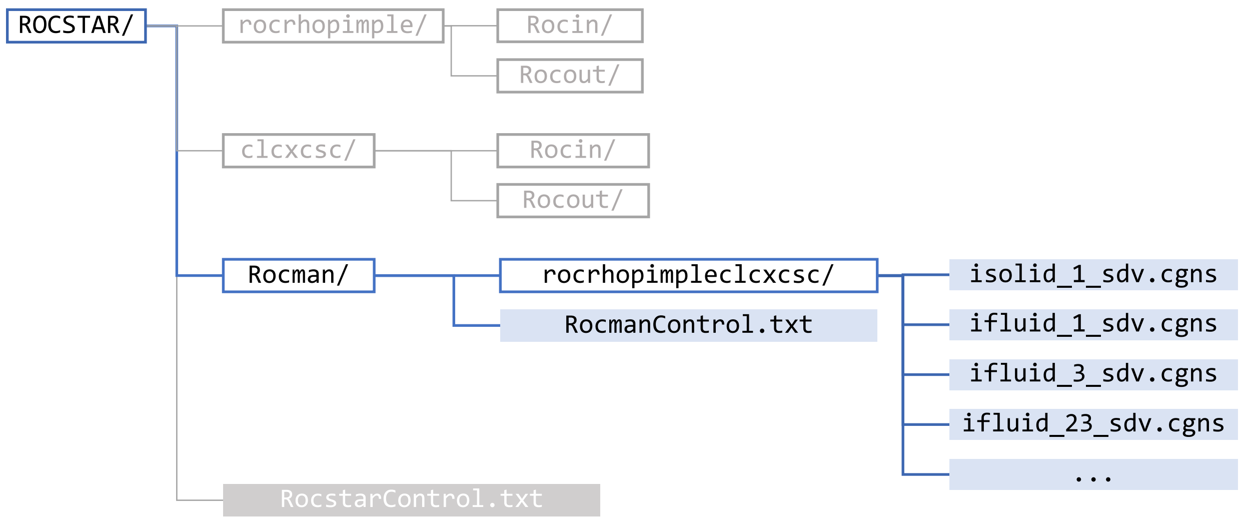

Figure 17. File hierarchy for the Rocman directory after the common refinement files have been generated.

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 12. RocstarControl.txt

```
CouplingScheme = SolidFluidSPC
SolidModule = clcxcsc
FluidModule = rocrhopimple
OutputModule = Rocout
MaximumTime = 10
AutoRestart = F
CurrentTimeStep = 5.0e-05
OutputIntervalTime = 1.0E-01
MaxWallTime = 1000000
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 18.

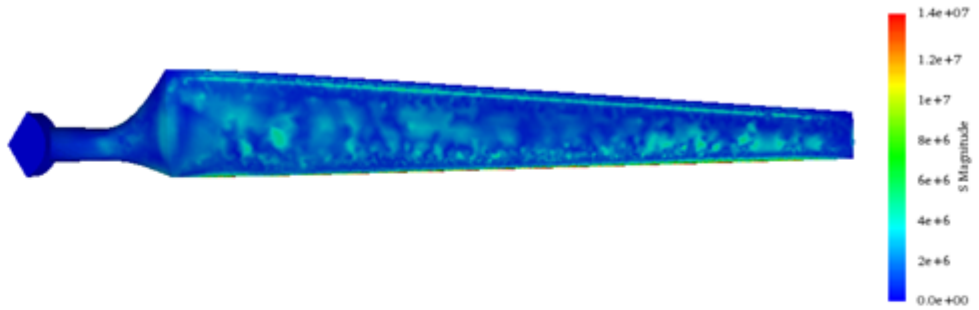

Figure 18. *Paraview* visualization of the turbine_metric.exo file. The magnitude of the stress at t=5e-4 s is shown.

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_name>.foam
```

This will create an empty .foam file that can be opened with *Paraview.* A screenshot is shown in Figure 19.

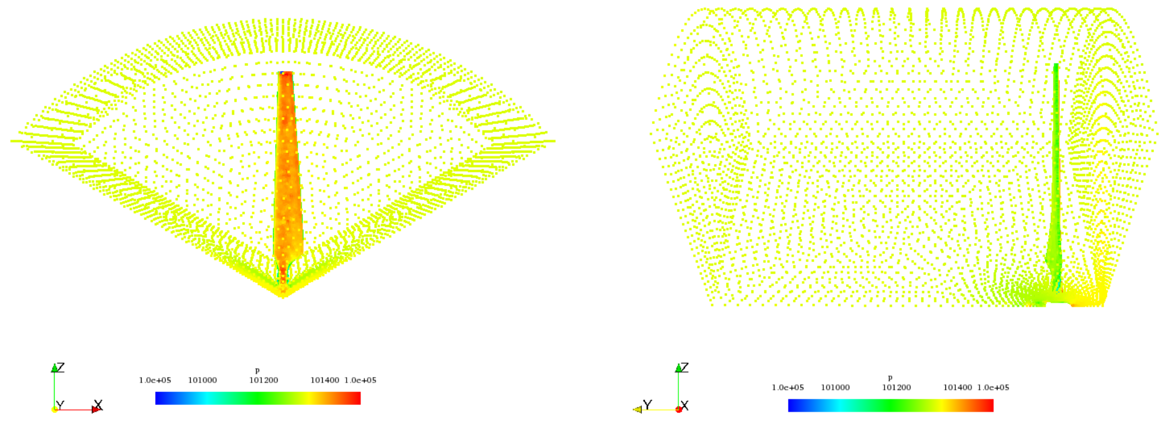

Figure 19. *Paraview* visualization of the fluid solution. Two views of the pressure at t=5e-4 s are shown.

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 20 and Figure 21.

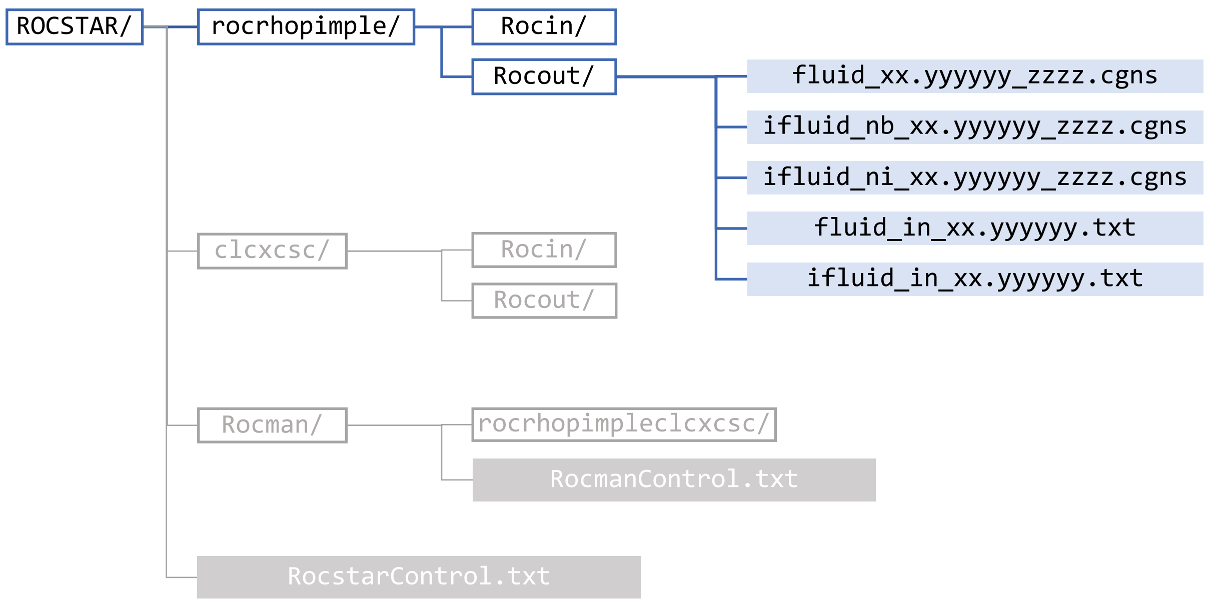

Figure 20. Files generated in rocrhopimple/Rocout after *Rocstar* has been run.

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 × 10xx × 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.

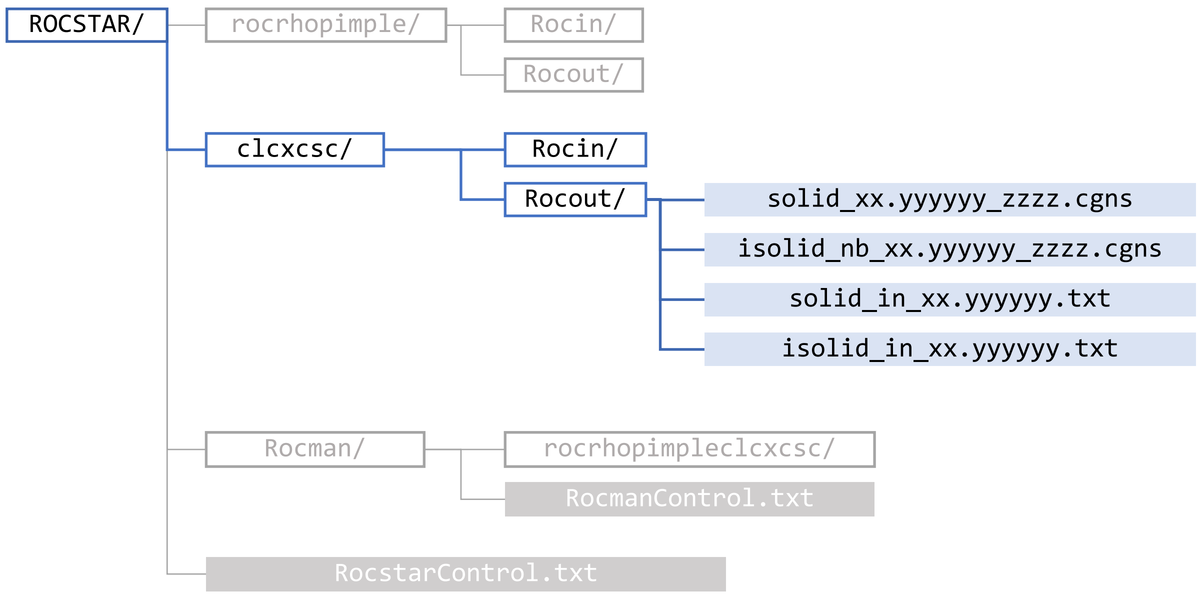

Figure 21. Files generated in clcxcsc/Rocout after *Rocstar* has been run.

6.0 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 as shown in Figure 22.

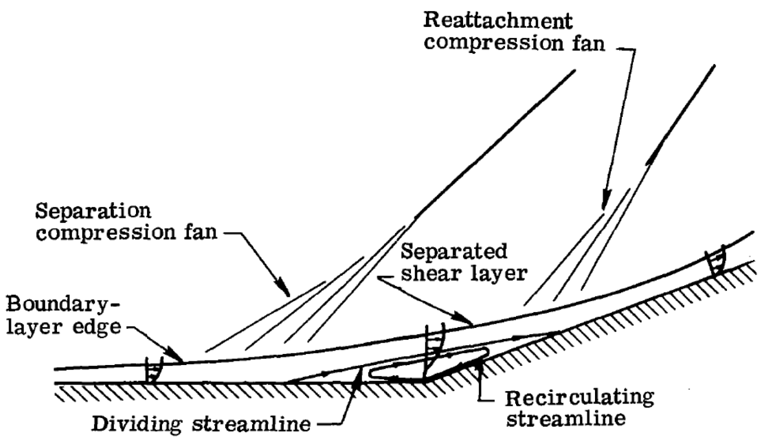

Figure 22. Various phenomena resulting from hypersonic flow over a ramped plate.

6.1 Problem Setup

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

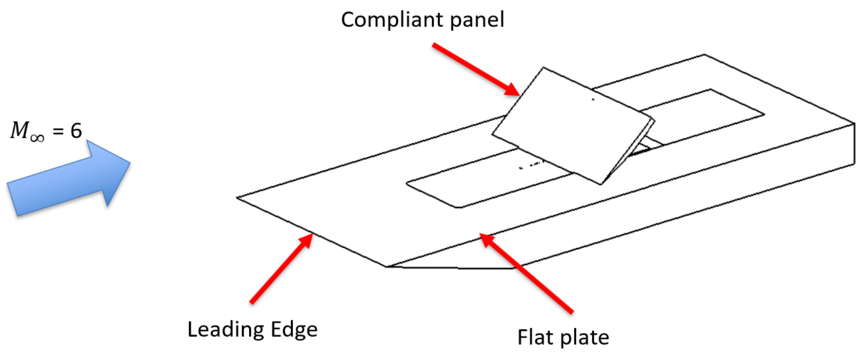

Figure 23. 3D representation of the problem setup for the ramped plate in a hypersonic flow.

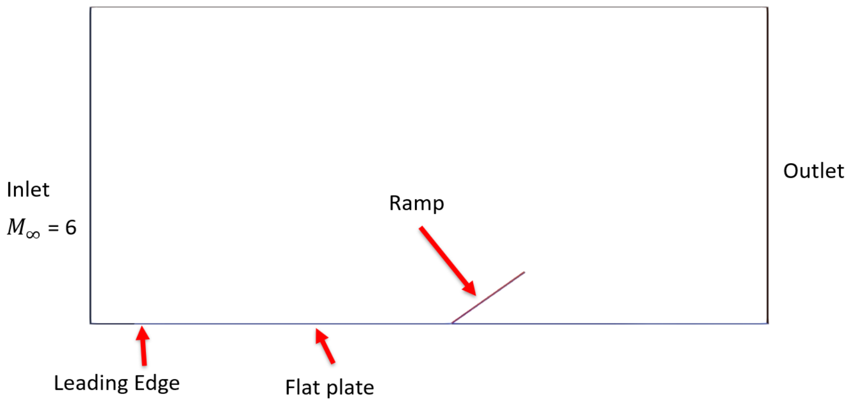

Figure 24. 2D representation of the problem setup for the ramped plate in a hypersonic flow.

6.1.1 Geometry

6.1.2 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.

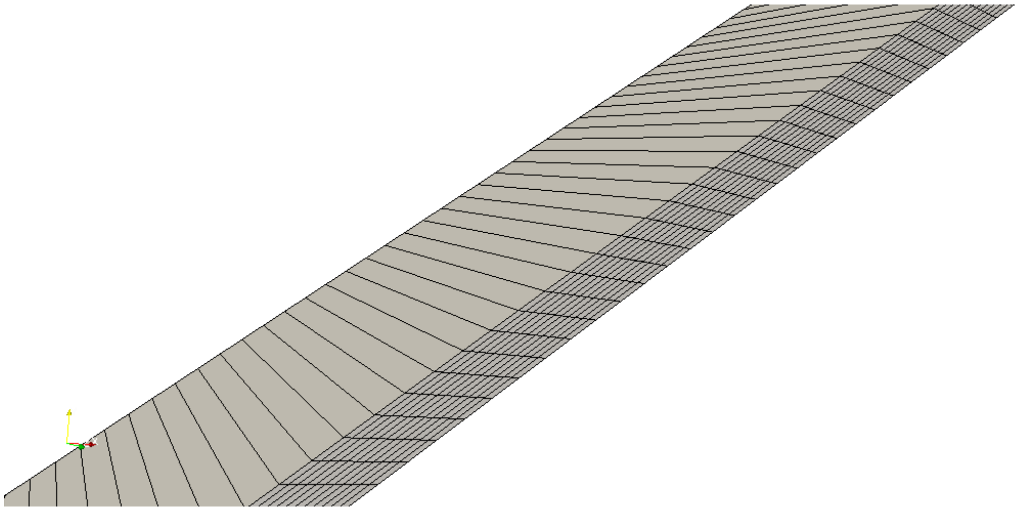

Figure 25. Zoomed view of the solid plate mesh.

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, and Figure 27 shows the surface mesh surrounded by the internal mesh in cyan.

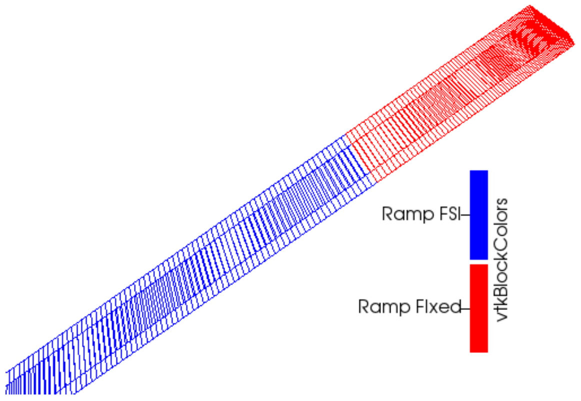

Figure 26. Zoomed in view of the surface mesh surrounding the ramp structure.

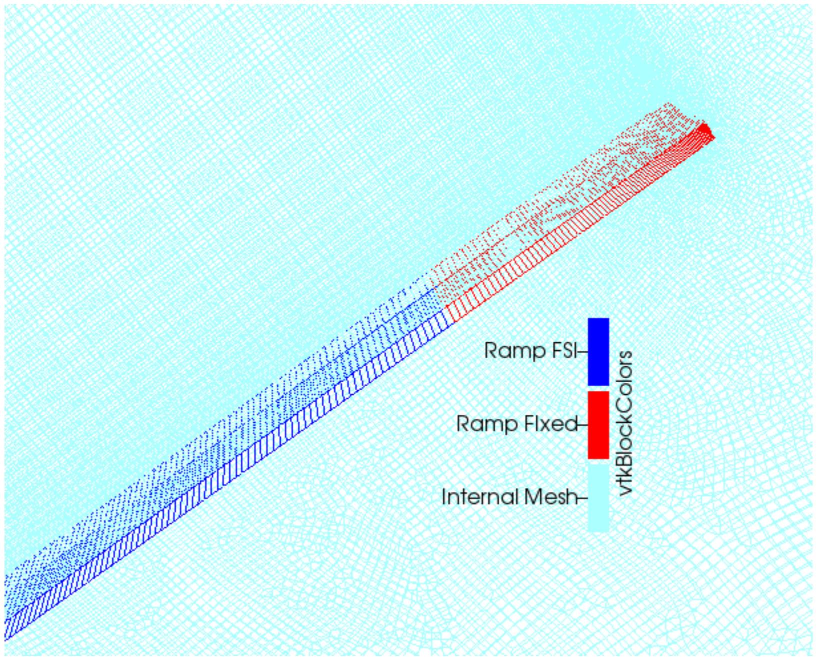

Figure 27. Zoomed in view of the surface mesh and the internal mesh surrounding the ramp structure.

6.1.3 Materials

The plate is made from AISI 4140 alloy steel[10]. The relevant material properties are given in Table 9.

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

Material	E (GPa)	ν	ρ (kg/m^3)
AISI 4140	210.0	0.27	7,850

The fluid is air, with properties shown in Table 10.

[10] Properties from https://www.azom.com/article.aspx?ArticleID=6769

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

Fluid	Molecular Weight (g/mol)	c_p (kJ/kgK)	μ (kg/m s)	Pr
Air	28.9647	1004.9	1.837e-05	0.7

6.1.4 Boundary Conditions

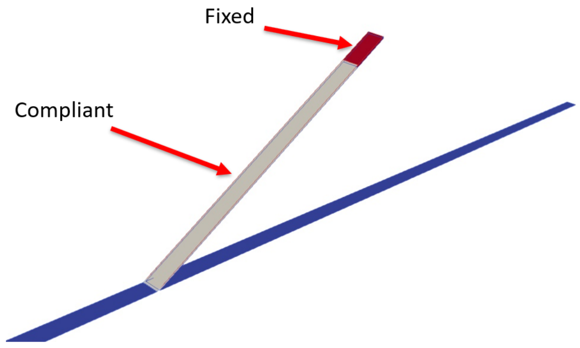

Figure 28. Boundary conditions on the ramp.

6.2 Input Files

6.2.1 Fluid

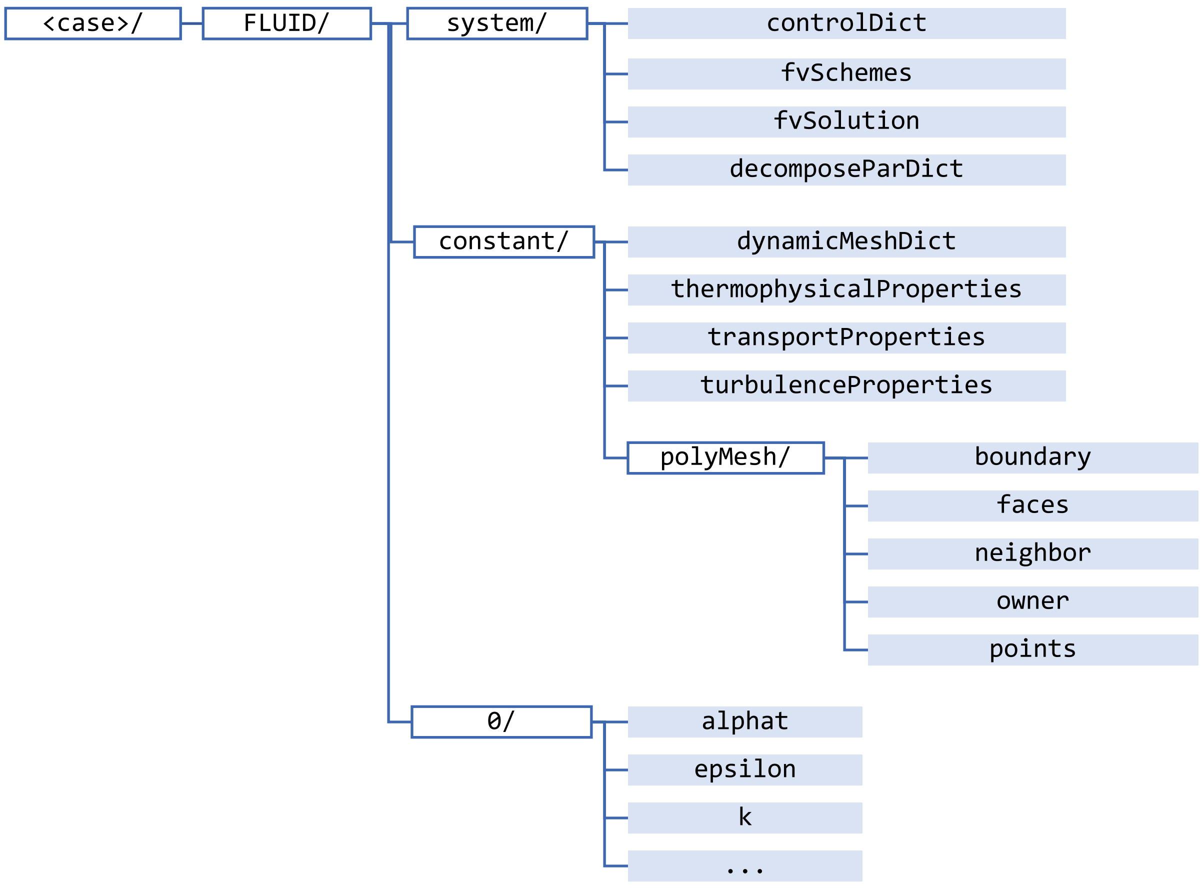

Figure 29. *Rocfoam* file structure. The system directory contains the dictionaries that specify solving methods, the constant directory contains the constant physical properties, and the initial time directory 0 contains the properties that evolve with time (truncated here to fit). As the problem runs,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

The thermodynamic and physical properties of air, which is the fluid being used in the case, are specified via the thermophysicalProperties file, as shown in Listing 13. 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 13. The file constant/thermophysicalProperties, which specifies the thermodynamic and physical properties of the fluid.

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  6
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
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;
21      transport       const;
22      thermo          hConst;
23      equationOfState perfectGas;
24      specie          specie;
25      energy          sensibleEnthalpy;
26  //    energy          sensibleInternalEnergy;
27  }
```

The thermoType section (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 (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.

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

The solid input files consist of the *CalculiX* input specification file, shown in Listing 14, as well as the mesh file referenced in line 7 of the input file.

Listing 14. The file plate1.inp, the *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
8   *BOUNDARY
9   bot,1,3
10  top,1,3
11  front,3,3
12  back,3,3
13  *MATERIAL,NAME=S4041
14  *ELASTIC
15  210E9,.27
16  *DENSITY
17  7.85E3
18  *SOLID SECTION,MATERIAL=S4041,ELSET=EB1
19  *STEP,INC=100000,NLGEOM
20  *DYNAMIC,DIRECT
21  1E-7, 1.0, 0.5E-9, 1.0E-1
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.

6.3 Running a Rocstar Case

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

6.3.1 Configuring the 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.

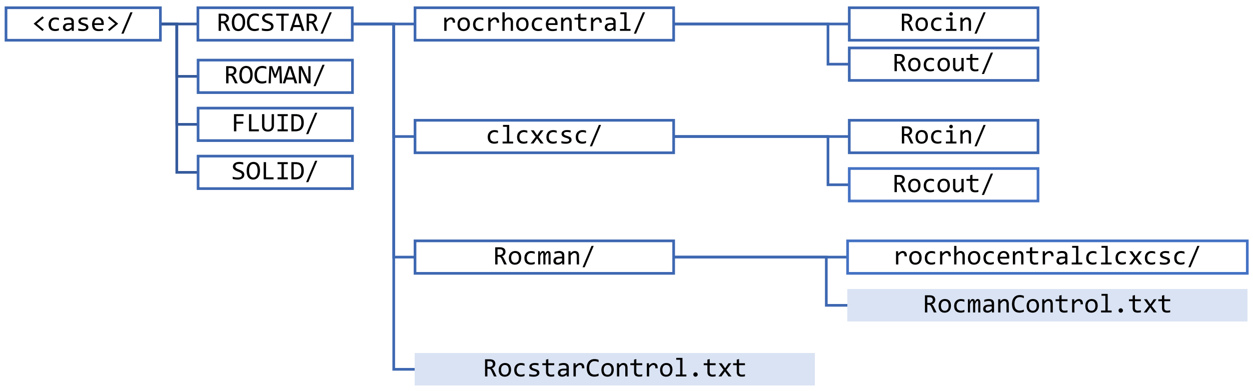

Figure 30. Native data archive for the hypersonic plate.

6.3.2 Preprocessing Fluid Input Files

This tutorial assumes the user has already generated the appropriate fluid meshes and variable fields as 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`. As shown in the following listing, this domain will be divided into four parts.

Listing 15. The system/decomposeParDict file, which specifies the number and method for decomposing a domain for parallel runs.

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
numberOfSubdomains 4;
method          scotch;
// ************************************************************************* //
```

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 containing 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 rocRhoCentral -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/rocRhoCentral -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/rocrhocentral/Rocin` directory with the following command:

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

Figure 31 shows the file hierarchy *Rocstar* expects.

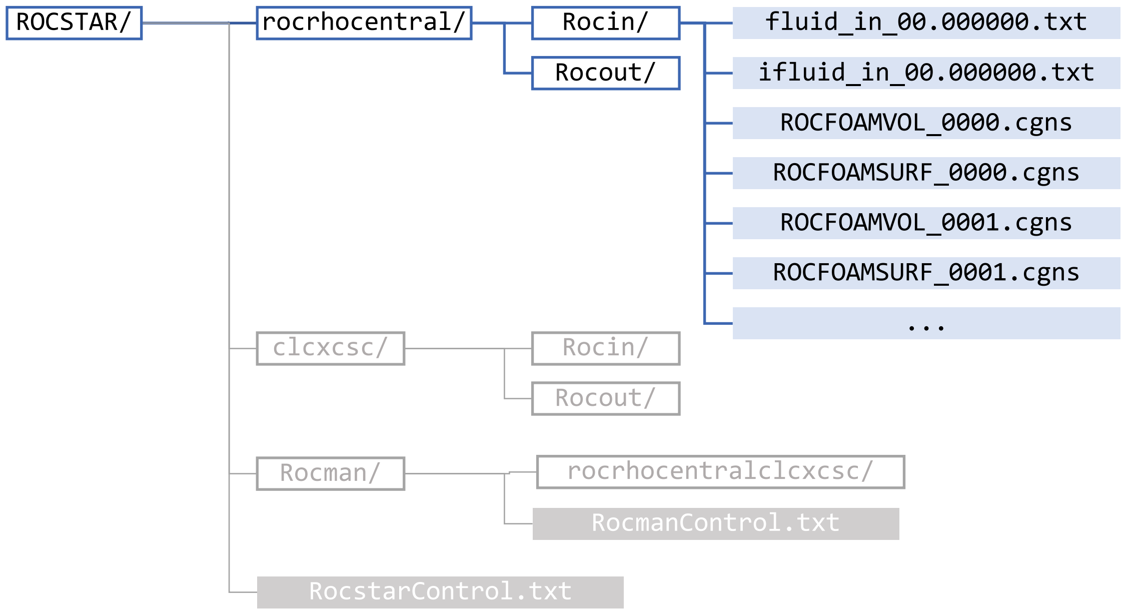

Figure 31. Preprocessed fluid solver input files (truncated for space) that have been copied from the FLUID/ROCFOAM folder into the correct Rocstar input directory.

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 10) 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 plate1 -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 also 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/clcxcsc/Rocin
```

Figure 32 shows the file hierarchy *Rocstar* expects.

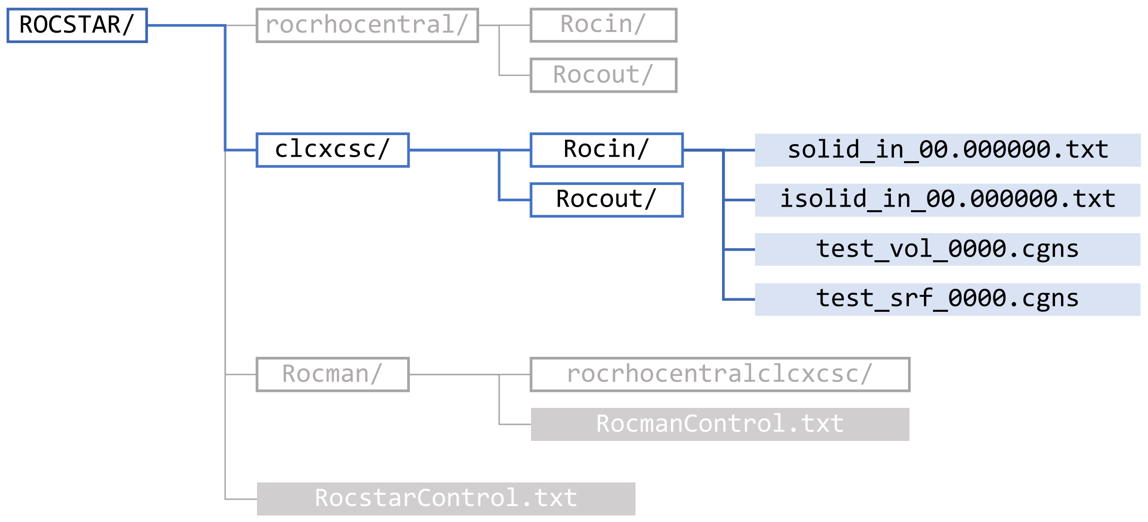

Figure 32. Preprocessed solid solver input files that have been copied form the SOLID folder into the correct Rocstar input directory.

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/rocrhocentral/Rocin/ifluid_in_00.000000.txt \
   ${CASE_DIR_PATH}/ROCSTAR/clcxcsc/Rocin/isolid_in_00.000000.txt \
   ${CASE_DIR_PATH}/ROCSTAR/Rocman/rocrhocentralclcxcsc
```

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 `rocrhocentralclcxcsc`, which is just a concatenation of the names of the fluid and solid solvers.

Running the surfdiver command will create sdv files in the `rocrhocentralclcxcsc` 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.

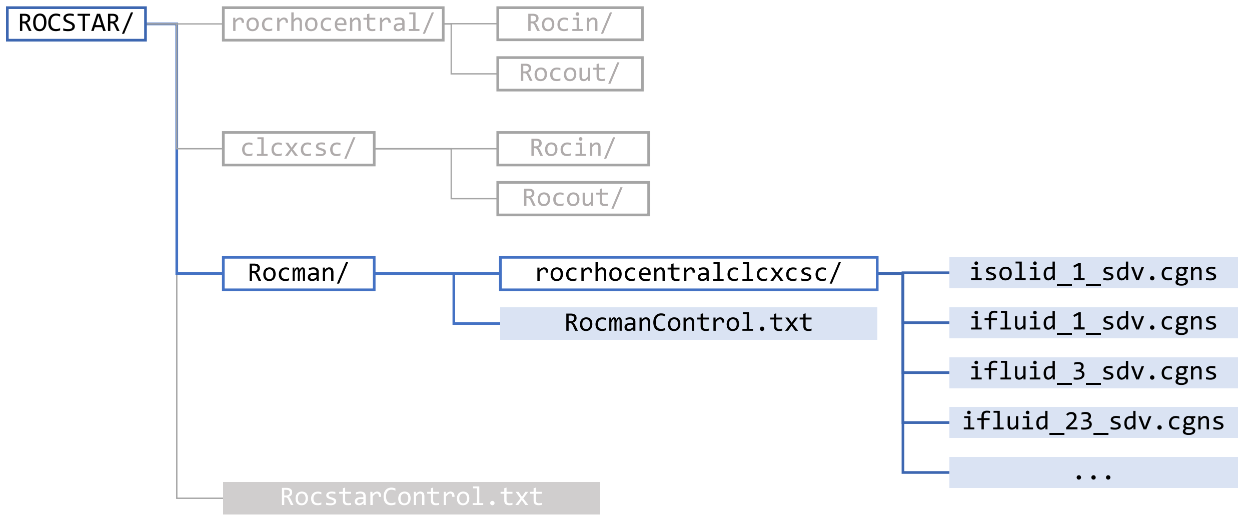

Figure 33. File hierarchy for the Rocman directory after the common refinement files have been generated.

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 16. RocstarControl.txt

```
CouplingScheme = SolidFluidSPC
SolidModule = clcxcsc
FluidModule = rocrhocentral
OutputModule = Rocout
MaximumTime = 1
AutoRestart = F
CurrentTimeStep = 1.0e-07
OutputIntervalTime = 1.0E-03
MaxWallTime = 1000000
ProfileDir = Rocman/Profiles
```

Warning

The hypersonic plate case detailed here was originally run on a cluster using 20 cores. For a personal desktop machine, this case may be intractable for a runtime of one second 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.

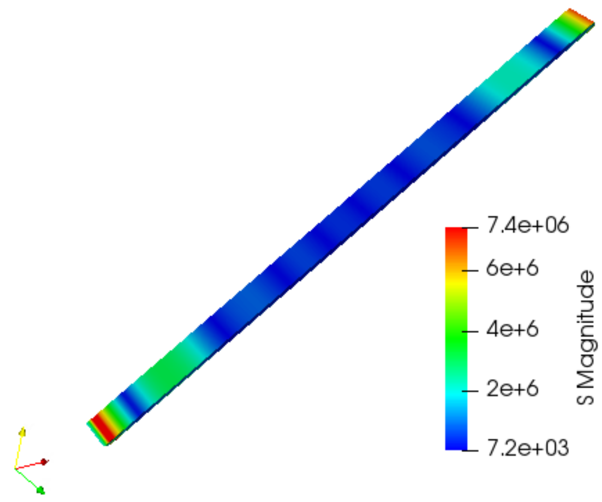

Figure 34.Paraview visualization of the plate1.exo file. The magnitude of the stress at t=5e-05 s is shown.

Figure 35 and Figure 36 show exaggerated deformation of the ramped plate at two different timesteps.

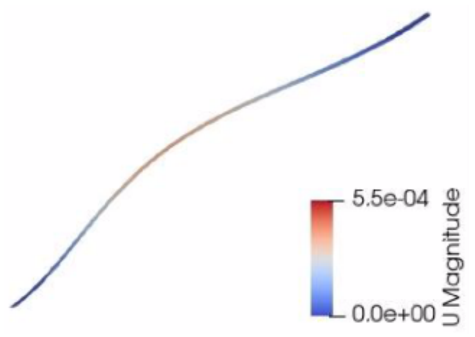

Figure 35. Exaggerated plate deformation at t=0.050.

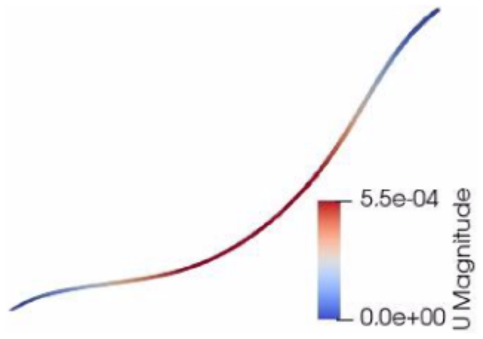

Figure 36. Exaggerated plate deformation at t=0.1.

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_name>.foam
```

This will create an empty .foam file that can be opened with *Paraview.* Figure 37 shows three screenshots of various timesteps.

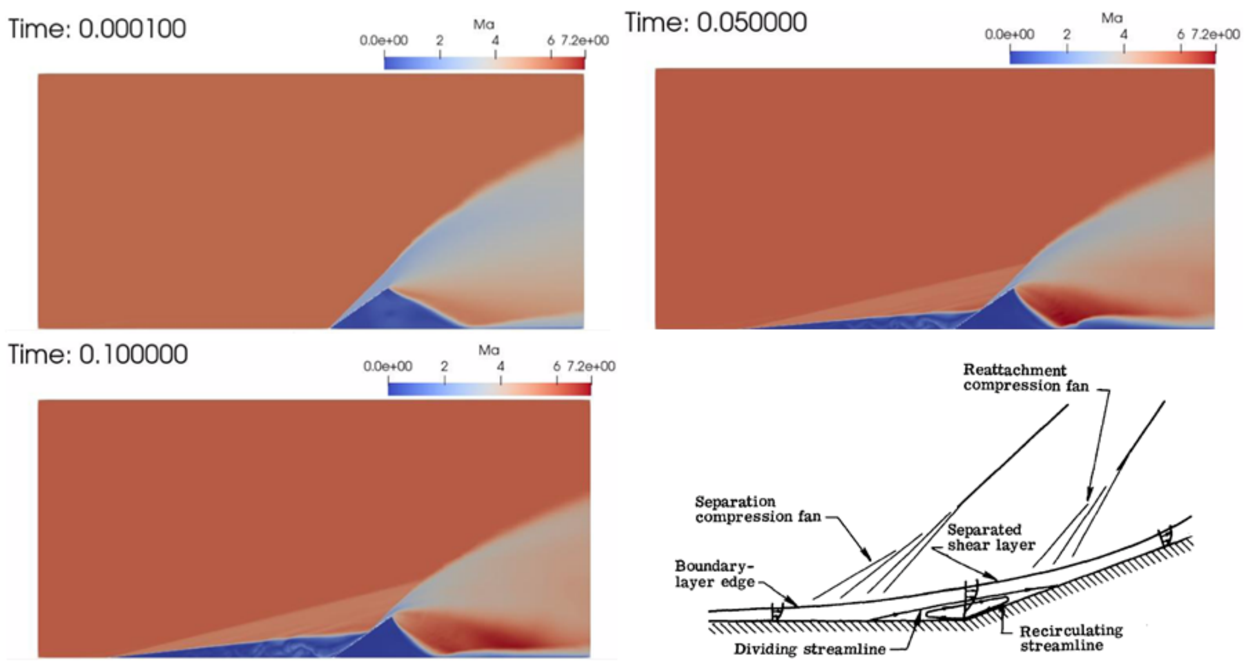

Figure 37. *Paraview* visualization of the fluid solution. Three different timesteps are shown, along with the diagram of the physical phenomena expected.

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 20 and Figure 21.

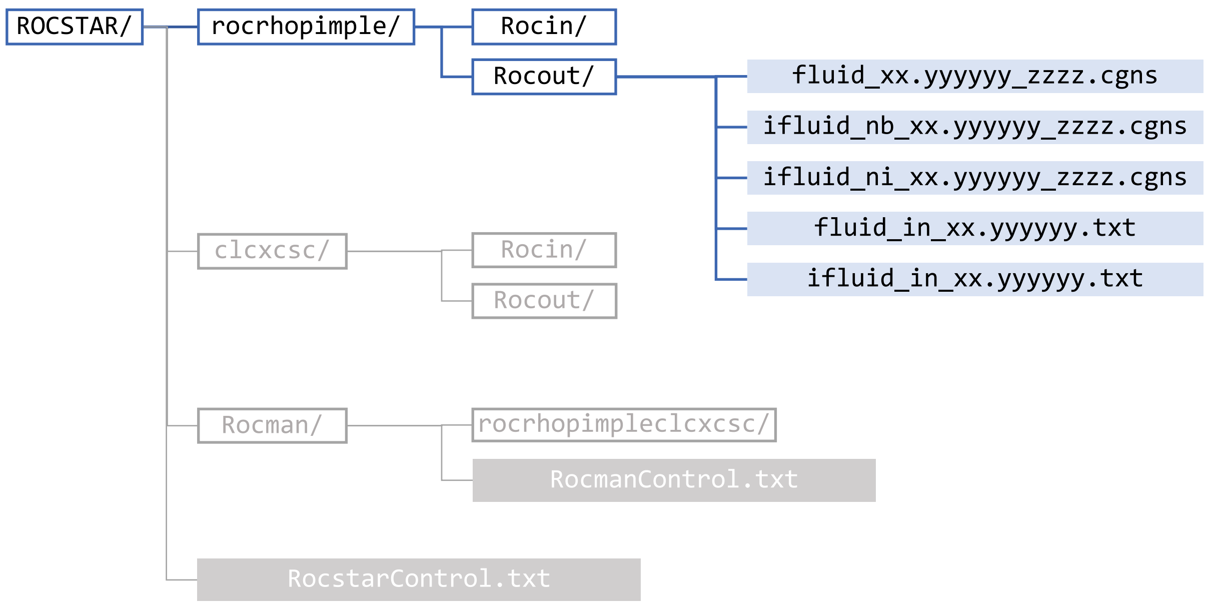

Figure 38. Files generated in rocrhopimple/Rocout after *Rocstar* has been run.

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 × 10xx × 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.

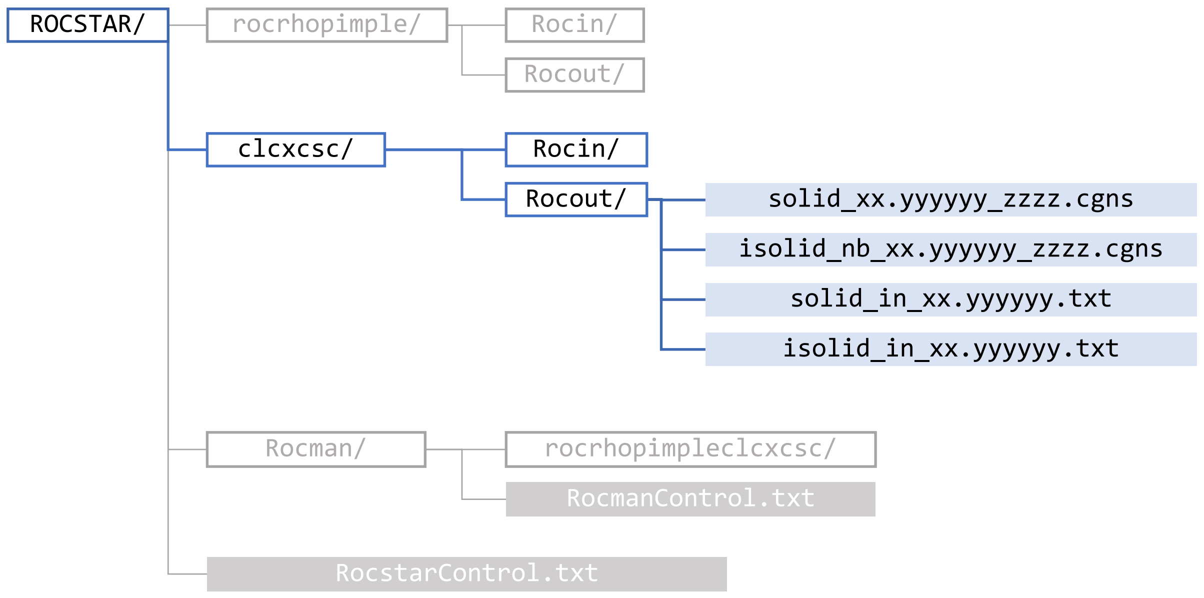

Figure 39. Files generated in clcxcsc/Rocout after *Rocstar* has been run.

7.0 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.

Support Team:

tech@illinoisrocstar.com

Principal Investigator:

Dr. Mohammad Mehrabadi
mehrabadi@illinoisrocstar.com

Address:

Attn: Rocmod Development Team
Illinois Rocstar LLC
108 Hessel Blvd, Suite 101
Champaign, IL 61820