User Guide for OpenHurricane

Version 2.0.0

Contents

1. Copyright

2. Introduction

This guide is released as the part of the OpenHurricane (Open parts of Highly Universal Rocket & Ramjet sImulation Codes for ANalysis and Evaluation project) version 2 program. It provides a brief description of the basic operation of OpenHurricane.

OpenHurricane is an open-source library written in C++ 17 and CUDA for computing supersonic reacting flows in the rocket and ramjet engines. The primary applications are computational fluid dynamics, but have been extended to chemically reacting flows, turbulent flows and multiphase flows, etc. In the current version, the OpenHurricane is able to compute chemistry source terms and gas physical properties on the CPU and GPU platforms alternatively. OpenHurricane is in its development phase, and there would be more features in it.

Further details of OpenHurricane, including physical models, numerical methods and operating instructions are described in remaining chapters.

Chapter 3 describes the compilation and installation method, including third-party dependencies in Linux and Windows OS.

Chapter 4 covers the detailed operation procedure of OpenHurricane, including all kinds of files needed in actual simulation cases.

Chapter 5 gives various tutorial examples with detail information about computational configures and meshes, boundary conditions, numerical settings, running status and post-processing of results.

Chapter 6 introduces the models and methods used in the current version of OpenHurricane.

3. Compilation and installation

OpenHurricane provides a main subroutine named *HurricaneMain* in the directory applications, which can be run from a terminal command line. This chapter briefly shows how to compile and install OpenHurricane applications.

3.1 The programming language used in OpenHurricane

Source codes of OpenHurricane is mainly written in C++ and in CUDA C. It's not necessary for users to have a deep understanding of C++ programming when using OpenHurricane. But some background C++ knowledge helps users understand the source codes of OpenHurricane.

The development of OpenHurricane is mainly based on the object-oriented programming (OOP) manner. OOP models complex things as simple structures and makes it more reusable. Its polymorphism and inheritance allow for class-specific behavior and make it easier to develop the main framework for the program.

3.2 Dependencies

Before compiling the applications of OpenHurricane, users should install the following third-party software packages in your operating systems or build them in the *thirdParty* directory of OpenHurricane. These third-party packages include:

(1) CGNS: A general, portable, and extensible standard for the storage and retrieval of CFD analysis data.

(2) HDF5: A set of file formats (HDF4, HDF5) designed to store and organize large amounts of data.

(3) Metis: A set of serial programs for partitioning graphs, partitioning finite element meshes.

(4) MPI: Massage Passing Interface. A standardized and portable message-passing standard for parallel computing. Users can use MS-MPI, Intel MPI from Intel oneAPI toolkits, MPICH and OpenMPI packages.

(5) Eigen: A C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms.

(6) CUDA: A parallel computing platform and programming model that makes using a GPU for general purpose computing simple and elegant.

The users can also use the prebuilt packages including CGNS, HDF5 and Metis in the *thirdParty* directory of OpenHurricane in both Windows and Linux systems.

3.3 Compiling and installing applications

OpenHurricane can be compiled and installed in both Windows and Linux systems.

3.3.1 Compiling in Windows systems

For Windows platform, the Visual Studio Community is recommended. Please use the CMake tool to configure the project of OpenHurricane. However, in-source builds are not allowed. There are three ways to configure OpenHurricane as follows:

(1) Using Visual Studio IDE

If the C++ CMake tools for Windows is installed with the Visual Studio, then users can use Visual Studio directly to run CMake configuration and build applications for OpenHurricane. To do this, run the "Visual Studio 2022.exe", which should be in your Start menu or on your desktop, and click "Open a local folder" to select the location of OpenHurricane's source directory. Then the project will be automatically configured by the Visual Studio IDE with the CMake. Click on "Open CMake Settings Editor," and Visual Studio will generate *CMakeSettings.json* to enter the CMake setting GUI. Then the compilation flags can be changed in this setting GUI.

(2) Using CMake GUI

To do this, run the cmake-gui.exe, which should be in your Start menu or on your desktop, and browse to the location of OpenHurricane's source with the "Browse Source" button. You can also change the binary directory, where the Visual Studio project files will be placed, with the "Browse Build" button. Click "Generate" to select the correct visual studio version and build the project files. Then the Visual Studio project will be generated.

When the Visual Studio project of OpenHurricane is generated, the applications of OpenHurricane can be built by Visual Studio IDE.

(3) Using command line

Open the command prompt and cd to the OpenHurricane source directory. Make a directory (e.g., *build*) where the resulting binaries should be placed:

```
mkdir build
cd build
```

Run

```
cmake --help
```

and look at the list of generators for the Visual Studio you want to build for. For example, the generator for Visual Studio 2022 is called "Visual Studio 17".

After you have found the appropriate generator, run

```
cmake .. -G "Visual Studio 17 2022" -DCMAKE_BUILD_TYPE=Release
```

to generate the project files. The project files will be placed in the directory *build*. The Visual Studio project will be called “OpenHurricane.sln”. Open it in Visual Studio. When the Visual Studio project of OpenHurricane is generated, the applications of OpenHurricane can be built by Visual Studio IDE.

3.3.2 Compiling in Linux systems

To compile OpenHurricane in Linux systems, users should use the CMake tool to configure the project and then use the GNU Make tool to control the generations of the applications. The following additional requirements must be met in your Linux systems:

(1) GCC: OpenHurricane has been only tested by using the GCC compiler in Linux systems. Therefore, the GCC compiler is recommended for Linux platforms to build OpenHurricane. Require version 9.0.0 or later. Check with command:

```
gcc --version
```

(2) CMake: Require version 3.18 or later. Check with command:

```
cmake --version
```

(3) GNU Make: Require version 3.82 or later. Check with command:

```
make --version
```

The source codes are already together with prebuilt third-party libraries. You can also build them by yourself. If users want to use the prebuilt third-party libraries in Linux

systems, then the below requirements should be met.

(1) GNU binutils: Require version 3.82 or later. Check with command:

```
ld --version
```

(2) ldd: Require version 2.17 or later. Check with command:

```
ldd --version
```

Then users can build OpenHurricane with CMake or CCMake tools. In-source builds are not allowed.

(1) Using CMake from the command line

The example of using the *cmake* command for building OpenHurricane object is

```
mkdir build
cd build
cmake .. -DCMAKE_BUILD_TYPE=Release
make && make install
```

(2) Using CCMake from the command line

If the CMake curses interface is supported in your Linux platform, then the *ccmake* command can be used to configure this project. The example of using the *ccmake* command for building OpenHurricane object is

```
mkdir build
cd build
ccmake ..
make && make install
```

4. Simulation setup

This chapter gives the descriptions of the setup of the simulation cases in the OpenHurricane. Normally, the user should assign a name to a case, and locate it in the directory specified by the user. All the output files for the given case by OpenHurricane are stored in this project directory by default.

The example cases that accompany the current version of OpenHurricane provide a useful way to learn the simulation setup for running the OpenHurricane. The example cases are located in the *example* directory.

4.1 Input file format

OpenHurricane needs to input several files to operate the simulation tasks. The OpenHurricane input file format is described as follows.

4.1.1 Controller file format

OpenHurricane uses a *controller* file to control the simulation process. A controller file consists of several elements that can be read by OpenHurricane by means of keywords. A controller file must be written using the XML language. XML (Extensible Markup Language) is a markup language similar to HTML.

The example about the structure of a controller file can be found in the file *Hurricane-XMLstandard.cont* in the root directory of OpenHurricane. The structure a controller file consists of:

(1) **XML declaration**: It is not a tag. It can be given as:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
```

(2) **XML comments**: The comments in the controller files can be given by the following format:

```
<!-- This is a comment. -->

<!--
    This is a comment.
-->
```

(3) **Root element**: It is the root element of the whole controller file which must be

named as *Config*. All other elements must be included within this root element.

```
<Config>
    (All other elements)
</Config>
```

The keyword elements can include the following types:

(1) Word control element: It consists of a string, for example:

```
<meshFile>OpenHurricaneMesh.h5</meshFile>    <!-- Specify the mesh file name. -->
<meshFormat>OpenHurricane</meshFormat>    <!-- Specify the mesh file format. -->
```

(2) Parameter control element: It is used to specify a real parameter or an integer parameter, for example:

```
<maxStep>200000</maxStep> <!-- Specify the maximum number of steps for simulation. -->
<totalTemperature>300.0</totalTemperature> <!-- Specify the total temperature is 300 K. -->
```

(3) Controller control element: It is used to specify a multi-level controller element, for example:

```
<meshSet>    <!-- meshSet is a controller control element. -->
    <meshFile>OpenHurricaneMesh.h5</meshFile>
    <meshFormat>OpenHurricane</meshFormat>
</meshSet>
```

(4) Text control element: It is used to specify a text to the program. It is usually to specify a list of a controller elements, for example:

```
<regions><![CDATA[shapeRegion, hexRegion]]></regions>
<shapeRegion>…</shapeRegion>
<hexRegion>…</hexRegion>
```

(5) Real array control element: It is used to specify an array of real parameters, for example:

```
<piecewiseLinear>
    <cp><![CDATA[904.0, 1004.0, 1104.0]]></cp>
    <!-- Specify an array of the specific heat cp for different temperature. -->
    <T><![CDATA[300.0, 1000.0, 2000.0]]></T>
    …
</piecewiseLinear>
```

4.1.2 Mesh file format

In the current version of OpenHurricane, three mesh formats are supported. The names of all face zones should be specified in order to set up the boundary conditions.

(1) **Fluent mesh**: the mesh format of the commercial software Ansys Fluent. The

details about the mesh file format can be found in the helper of Ansys Fluent.

(2) **CGNS unstructured mesh**: the mesh format of the CFD General Notation System (CGNS) unstructured mesh. The details about the mesh file format can be found in the website of CGNS.

(3) **Hurricane mesh**: the mesh format of the Hurricane and the OpenHurricane. The mesh file is stored in the HDF5 format. The structure of Hurricane mesh file is illustrated in Fig. 1. There are three element types in Hurricane mesh file, i.e., point, face and cell. And all of them are organized into several zones which are stored in the section named "grid". Therefore, the section named "elementZoneId" stores indices of all zones of type "element", i.e., point, face and cell. In Fig. 1., "pointZone8" denotes that the index of this point zone is 8. Then all grid points of this point zone are stored in the data section of "pointZone8". Besides, "cellZone9" stands for a cell zone of which the index is 9, and its data section stores the shape types of all the cells belonged to itself. Moreover, "faceZone10" represents a face zone with index 10, and in its data section, the point indices that forming faces and the adjacent cell indices of faces are stored. The section "cellOriginIndex9" stores the original indices of all cells in cell zone 9. The section "gridDecompose" stores the partition information for parallel computing.

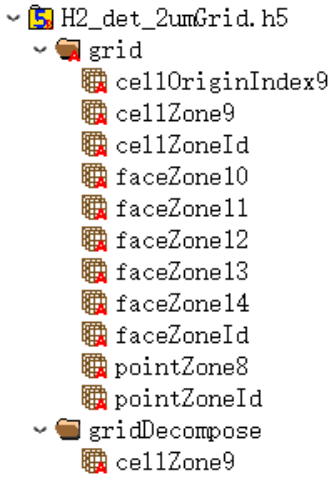

Fig. 1. The structure of an example of Hurricane mesh file.

4.1.3 Chemical mechanism file format

In order to calculate chemical reacting flows, the chemical mechanism must be given

to OpenHurricane. The current version of OpenHurricane only supports the chemical mechanism file in the format of Chemkin.

4.1.4 Thermodynamic file format

All gas-phase species contained in the chemical mechanism file must have associated thermodynamic data. OpenHurricane uses the format for thermodynamic data that used by Gordon and McBride [1] for the thermodynamic database in the NASA chemical equilibrium code. This format is also used in Chemkin.

4.1.5 Transport file format

All gas-phase species contained in the chemical mechanism file must have associated transport data in order to compute transport properties. The format of transport data supported by the current version of OpenHurricane is the same as that of Chemkin software.

4.2 Basic sections in controller file

The details about the controller file used by OpenHurricane to control the run of simulations can be found in the file *Hurricane-XMLstandard.cont* in the root directory of OpenHurricane. There are nine basic sections in the controller file as given below:

```
<Config>
    <!-- [1] Grid setting: grid source, grid pre-processing -->
    <meshSet>…</meshSet>
    <!-- [2] Iteration setting: iteration control, time scheme, monitor -->
    <iteration>…</iteration>
    <!-- [3] Spatial scheme setting: spatial scheme -->
    <spatialScheme>…</spatialScheme>
    <!-- [4] Flow setting: turbulence model, combustion model -->
    <flow>…</flow>
    <!-- [5] Boundary condition -->
    <boundaryCondition>…</boundaryCondition>
    <!-- [6] Reference setting -->
    <ref>…</ref>
    <!-- [7] Free stream setting -->
    <freeStream>…</freeStream>
    <!-- [8] Initialization setting -->
    <initialization>…</initialization>
```

```
    <!-- [9] Write control -->
    <writeControl>…</writeControl>
</Config>
```

4.2.1 Grid setting

This subsection describes how to set up the mesh information and pre-processing. The grid setting section can be specified as below:

```
< meshSet>
    <meshFile>meshFileName</meshFile ><!-- Specify the mesh source by given the mesh file name. It can be given in an absolute path, otherwise, the mesh file should be located in the same directory of the controller file. -->
    <meshFormat>OpenHurricane</meshFormat> <!-- The format can be fluent, CGNS or OpenHurricane. -->
    <unit>mm</unit> <!-- The length unit of the given mesh. The types supported include: mm for millimeter; cm for centimeter; dm for decimeter; m for meter; km for kilometer; um for micrometer. -->
    <reordering>RCMReordering</reordering><!-- Reordering methods: MetisReordering or RCMReordering. -->
    <scaleMesh>(x-factor, y-factor, z-factor)</scaleMesh><!-- To scale mesh in the x-direction, y-direction and z-direction respectively. -->
    <regions><![CDATA[regionName,…]]></regions><!-- To mark regions by given region name. -->
        <regionName>
            <id>0</id> <!-- Index -->
            <option>inside</option> <!-- Inside or outside. -->
            <shape>shapeType</shape><!-- The shape types of the marked region. The valid shape types include: hex, sphere, cylinder or plane. -->
            <!-- (1) For the type: hex, the below information must be given to define a hexahedral region. -->
                <xmax>0.0</xmax><xmin>0.0</xmin>
                <ymax>0.0</ymax><ymin>0.0</ymin>
                <zmax>0.0</zmax><zmin>0.0</zmin>
            <!-- (2) For the type: sphere, the below information must be given to define a sphere region. -->
                <center>(0.0,0.0,0.0)</center><radius>0.0</radius>
            <!-- (3) For the type: cylinder, the below information must be given to define a cylinder region. -->
                <Amax>(0.0,0.0,0.0)</Amax><Amin>(0.0,0.0,0.0)</Amin>
                <radius>0.0</radius>
            <!-- (4) For the type: plane, the below information must be given. -->
                <normal>(1.0,0.0,0.0)</normal> <!—It points to “outside” -->
                <point>(0.0,0.0,0.0)</point>
        </regionName>
```

```
</meshSet>
```

4.2.2 Iteration setting

This subsection gives the structure of the controller file that controls the iteration of the simulations. There are six parts in this controller of iteration setting.

(1) **Step setting**: to set up the maximum steps for the iteration and the step intervals for output of the intermediate solutions.

```
<iteration>
…
    <maxStep>200000</maxStep>
    <writeOutputStep>1</writeOutputStep>
…
</iteration>
```

(2) Restart setting: to set up the simulation that restarted from another solutions when it is necessary.

```
<iteration>
…
    <restartFrom>OpenHurricane-1.h5</restartFrom><!-- Restart from a given solution. If it is a new start simulation, then this option must be commented out in the file. -->
    <relayType>originData</relayType><!-- If it is restarted from a given solution, then specify "originData" to continue the simulation or "interpolationData" to interpolate from the given solution. Default option is "originData". -->
…
</iteration>
```

(3) Time dependence setting: to set up steady or unsteady flow.

```
<iteration>
…
    <flowState>steady</flowState>        <!-- The flow state can be steady or unsteady. -->
…
</iteration>
```

(4) Pseudo time-step method setting: to set up the pseudo time-step method.

```
<iteration>
…
    <pseudoTime><!-- Pseudo time-step method. -->
        <timeStepMethod>cellBasedMethod</timeStepMethod>            <!-- It must be faceBasedMethod or cellBasedMethod. -->
        <CFLSetting>
            <cflMin>1.1</cflMin><!-- The minimum CFL number. Default is 0.1. -->
            <cflMax>10.0</cflMax><!-- The maximum CFL number. Default is 10.0. -->
            <stepForPrintCFL>20</stepForPrintCFL>
```

```
            <CFLType>linearCFL</CFLType><!-- The CFL type for adapting CFL number. It can be: linearCFL, expCFL or expertSystemAdaptCFL. If not specified, the linearCFL will be used. -->
            <!--(1) For "linearCFL". -->
                <linearCFL>
                    <cflConst>200</cflConst>
                    <cflLinear>400</cflLinear>
                </linearCFL>
            <!-- (2) For "expCFL" (exponent method). -->
                <expCFL>
                    <cflConst>200</cflConst>
                </expCFL>
            <!-- (3) For expertSystemAdaptCFL. -->
                <expertSystemAdaptCFL>
                    <CFLFactor0>0.01</CFLFactor0>
                    <breakdownFactor>0.5</breakdownFactor>
                    <divergenceFactor>0.8</divergenceFactor>
                    <residualJumpThreshold>0.5</residualJumpThreshold>
                    <updateFactor>2.0</updateFactor>
                    <interval>10</interval>
                </expertSystemAdaptCFL>
        </CFLSetting>
        <timeStepType>localTimeStep</timeStepType><!-- It can be "localTimeStep" or "globalTimeStep". Default is "localTimeStep". -->
        <CForTimeStep>2.0</CForTimeStep><!-- Range [1,4]-->
        <isStretchAc>on</isStretchAc><!-- On or off. Convergence acceleration for stretched meshes. Notice: Only works in implicit time scheme. Default is on. -->
        <StretchAc>
            <minStretchScale>1.0</minStretchScale>
            <cflRatioMax>5000000.0</cflRatioMax>
        </StretchAc>
    </pseudoTime>
…
</iteration>
```

(5) Time scheme setting: to set up temporal discretization schemes.

```
<iteration>
…
    <timeMethod>
        <timeMarching>LUSGS</timeMarching>          <!-- It can be LUSGS, MRK, unsteadyMRK, semiImplicitMRK, TVDRK, unsteadyTVDRK, TVDRK4, unsteadyTVDRK4, BDF123LUSGS or ESDIRKLUSGS. -->
        <!-- (1) For LUSGS -->
        <LUSGS>
```

```
            <omegaForLUSGS>1.050</omegaForLUSGS><!-- Range [1,2]-->
            <betaT>0.2</betaT>
        </LUSGS>
        <!-- (2) For MRK or unsteadyMRK. -->
        <MRK>
            <MRKStage>s5</MRKStage><!-- Stage : f3, f4, f5, s3, s4, s5 (f = first order; s = second order)-->
            <impResSmooth>on</impResSmooth><!-- On or off. Implicit residual smoothing for Multi-Stage Runge-Kutta method.-->
            <ep>0.5</ep><!-- Range [0.3,0.8]-->
            <maxJacStep>2</maxJacStep><!-- The maximum Jacobi iterations. Usually, two steps are sufficient. -->
            <cflRatioForImpResSmoo>2.0</cflRatioForImpResSmoo><!-- The ratio to increase the CFL number due to implicit residual smoothing. Usually, 2~5 times. -->
            <cfl>1.0</cfl>
        </MRK>
        <!-- (3) For semiImplicitMRK. -->
        <semiImplicitMRK>
            <MRKStage>s5</MRKStage><!-- Stage : f3, f4, f5, s3, s4, s5 (f = first order; s= second order)-->
            <impResSmooth>off</impResSmooth><!-- On or off. Implicit residual smoothing for Multi-Stage Runge-Kutta method. -->
            <ep>0.5</ep><!-- Range [0.3,0.8] -->
            <maxJacStep>2</maxJacStep><!-- The maximum Jacobi iterations. Usually, two step is sufficient. -->
            <cflRatioForImpResSmoo>2.0</cflRatioForImpResSmoo><!-- The ratio to increase the CFL number due to implicit residual smoothing. Usually, 2~5 times. -->
            <cfl>1.0</cfl>
        </semiImplicitMRK>
        <!-- (4) For TVDRK or unsteadyTVDRK. -->
        <TVDRK>
            <TVDRKOrder>TVDRK2</TVDRKOrder><!-- TVDRK2 – 2nd order; TVDRK3 – 3rd order. -->
            <cfl>1.0</cfl>
        </TVDRK>
        <!-- (5) For unsteady flow. -->
        <physicalTimes>
            <dynamicTimeStep>on</dynamicTimeStep><!-- On or off. Default is off. -->
            <dynamicCFL>0.8</dynamicCFL> <!-- If the dynamicTimeStep is used, then user can specify the CFL number. If it is not given, it will be 0.8 as default for BDF123LUSGS, or taken from that of explicit scheme, such as unsteadyTVDRK, unsteadyTVDRK4 or unsteadyMRK. -->
            <phyTimeStep>1e-7</phyTimeStep> <!-- Physical time step in seconds. When dynamic time-step is used, this parameter will be ignored. -->
```

```
            <maxPhyTime>0.1</maxPhyTime>
        </physicalTimes>
        <!-- (6) For dual time-stepping method. -->
        <dualTimeStep>
            <subIteration>
                <maxSubStep>30</maxSubStep>
                <minSubStep>20</minSubStep>
            </subIteration>
        </dualTimeStep>
        <!-- (7) For BDF method. -->
        <BDF123LUSGS>
            <BDF123>BDF2</BDF123><!-- BDF1, BDF2 or BDF3 -->
            <useLinearDtCFL>on</useLinearDtCFL><!-- If on, it is to use a linear dt or
CFL for the initial stage of calculation. Default is off. -->
            <linearDtCFL>
                <dtCFLConst>10</dtCFLConst>
                <dtCFLLinear>20</dtCFLLinear>
            </linearDtCFL>
        </BDF123LUSGS>
        <!-- (8) For ESDIRKLUSGS method. -->
        <ESDIRKLUSGS>
            <ESDIRK>ESDIRK3</ESDIRK><!-- ESDIRK3 for four-stages third-order.
ESDIRK4 for six-stages fourth-order. -->
        </ESDIRKLUSGS>
        <sourceTermTreating>explicitSource</sourceTermTreating><!-- It can be specified
as: explicitSource, diagImpliciSource or fullJacobian. Default is explicitSource. -->
    </timeMethod>
…
</iteration>
```

(6) Monitor setting: to set up monitors of the simulations when it is necessary.

```
<iteration>
…
    <!-- Monitor. -->
    <monitor>
        <monitorList><![CDATA[monitorName,…]]></monitorList>
        <monitorName>
            <monitorType>residuals</monitorType><!-- It can be residuals,
pointMonitors, faceMonitors or forceMonitors. -->
            <updateStep>1</updateStep>
            <writeToFile>off</writeToFile><!-- On or off. Should write the information to
the file. Default is off. -->
            <fileName>HurricaneResidual.dat</fileName>          <!-- Default is
case name + "_residuals.dat". -->
```

```
            …<!-- For more information, please refer to the file Hurricane-XMLstandard.cont in the root directory of OpenHurricane -->
        </monitorName>
    </monitor>
…
</iteration>
```

4.2.3 Spatial scheme setting

This subsection gives the details about the setting of spatial schemes. There are three types of methods should be specified: convective flux schemes, reconstruction approaches and gradient methods

```
<spatialScheme>
    <!-- [1] Convective flux schemes-->
        <spatialScheme>upwindSchemes</spatialScheme><!-- upwindSchemes, blendingSchemeTVDLimiter2, JST. -->
        <blendingSchemeTVDLimiter2>
            <upwindScheme>LDFSS2</upwindScheme> <!-- AUSMPWPlus, AUSMPlusUP, LDFSS2, HLLC, HLLC_HLL, AUSM, HLL, or vanLeer -->
            <minAlphaf>0.04</minAlphaf> <!-- [0~1]. The allowed minimum factor of upwind part of the flux. Default is 0.1.-->
            <maxAlphaf>1</maxAlphaf> <!-- [0~1]. The allowed maximum factor of upwind part of the flux. Default is 1.0. -->
            <temperatureSensorK>8e-5</temperatureSensorK>
            <THighLimitShock>100.0</THighLimitShock> <!--Default is 100.-->
            <checkTemperature>on</checkTemperature> <!--Default is on.-->
            <THighLimit>3000.0</THighLimit><!--Default is 3000.-->
            <TLowLimit>100.0</TLowLimit><!--Default is 100.-->
        </blendingSchemeTVDLimiter2>
        <upwindSchemes>
            <upwindScheme>HLLC</upwindScheme> <!-- AUSMPWPlus, AUSMPlusUP, LDFSS2, HLLC, HLLC_HLL, AUSM, HLL, or vanLeer. -->
            <AUSMPlusUP> <!--AUSMPlusUP scheme parameters-->
                <Kp>0.25</Kp>
                <Ku>0.75</Ku>
                <sigma>1.0</sigma>
                <beta>0.125</beta>
            </AUSMPlusUP>
        </upwindSchemes>
        <JST>
            <k2>0.5</k2>
            <k4>7.8125e-3</k4> <!-- [1/128 ~ 1/64] -->
            <useModifyFactor>off</useModifyFactor> <!-- Use a factor to modify
```

```
pressure sensor to reduce numerical dissipation. Default is off. -->
            </JST>
        <!-- [2] Reconstruction methods-->
            <!-- (1) For first-order reconstruction. -->
                <reconstruction>firstOrder</reconstruction>
            <!-- (2) for multi-dimensional linear method. -->
                <reconstruction>linear</reconstruction>
                <limitersForLinear>Venk</limitersForLinear> <!-- Barth or Venk. -->
                <K>5.0</K><!-- Venk parameters. If Mach number is large, it is recommended
to reduce K. -->
        <!-- [3] Gradient methods. -->
            <gradient>cellGaussGrad</gradient> <!-- It can be specified as: nodeGaussGrad,
leastSquareGrad or cellGaussGrad. -->
            <!--For "leastSquareGrad". -->
                <weightType>WLSQG</weightType> <!-- Least square gradient weight type:
WLSQ0, WLSQ1, WLSQ2, WLSQ3, WLSQG-->
                <cellNeighbourCell>FACENEIGHBOUR</cellNeighbourCell> <!-- It can be
specified as FACENEIGHBOUR, NODENEIGHBOUR or TWOFACENEIGHBOUR. -->
</spatialScheme>
```

4.2.4 Flow setting

This subsection gives the details on the setup of flow models in the controller files. In this subsection the user can learn about how to setup turbulence models and combustion models.

There are three flow models supported in the current version of OpenHurricane: "*EulerFlow*" denotes Euler inviscid flow, "*laminarFlow*" stands for laminar flow and "*eddyViscosity*" is eddy-viscosity hypothesis flow.

```
<flow>
    <flowModel>EulerFlow</flowModel><!-- Available flow model: EulerFlow,
laminarFlow or eddyViscosity. -->
…
</flow>
```

The current version of OpenHurricane uses the following setting to set up turbulence models.

```
<flow>
    <flowModel>eddyViscosity</flowModel><!—In order to use turbulence model, the
flowModel must be eddyViscosity. -->
    …
    <turbulence>
        <distanceMethod>searchProcedures</distanceMethod>
```

```
            <turbulenceModel>SST</turbulenceModel> <!-- It can be SpalartAllmaras or SST.
-->
            <coefTurb>0.05</coefTurb> <!-- The limit coefficient for RANS turbulence
parameter increase rate. Default is 0.05. -->
            …
        </turbulence>
…
</flow>
```

If users want to simulate reacting flows, then the below setup should be specified in the flow setting section. To use GPU to calculate chemical source terms, the option: "*CUDAChemistrySourceNoSolver*" should be specified in **chemistrySource** element.

```
<flow>
    …
    <mixture>
        <equationOfState>perfectGas</equationOfState>
        <chemical>reaction</chemical>
        <thermo>
            <type>JANAF</type>
        </thermo>
        <transport>
            <type>kinetic</type>
        </transport>
        <mixtureFiles>
            <!-- To specify chemical mechanism file, thermodynamic data and transport
data. -->
            <chemFile>H2Air9sp_mechanism.che</chemFile>
            <thermoFile>thermo.dat</thermoFile>
            <tansportFile>transport.dat</tansportFile>
        </mixtureFiles>
        <reactions>
            <chemistryOption>coupled</chemistryOption>
            <!-- Chemistry options include coupled, strangSplitted and integrated.
                Default is coupled.
                (1) Option "coupled" is valid for laminar finite rate, PaSR.
                (2) Option "strangSplitted" is only valid for laminar finite rate.
                (3) Option "integrated" is valid for laminar finite rate, PaSR model.
            -->
            <combustionModel>
                <type>finiteRate</type> <!-- Turbulent combustion closure models:
finiteRate, PaSR-->
                <finiteRate> <!-- Laminar finite rate model. -->
                    <minDtFactor>0.1</minDtFactor> <!-- The minimum factor to
```

```
reduce the time step. Default is 0.1. -->
                    <tauIntFactor>0.1</tauIntFactor>
                </finiteRate>
                <PaSR> <!-- PaSR (Partially Stirred Reactor) model. -->
                    <mixingTimeScale>Mean</mixingTimeScale><!-- Mixing time
scale type: Kolmogorov, Integral, Mean or basedLocalRet. Default: Kolmogorov. -->
                    <cMix>0.3</cMix><!-- Mixing time-scale scale factor. For
Kolmogorov and Mean, it is recommended to be 1, and for Integral, to be 0.001~0.3. -->
                    <DRet>3.5</DRet><!-- The fraction dimension for computing
mixing time-scale based on local Reynolds number. Default is 4.0. D = 3.5 corresponds to the
Kolmogorov time-scale, whereas adopting D = 5 results in the integral time. -->
                    <chemicalTimeScale>
                        GivenSpeciesProductionRatesMin
                    </chemicalTimeScale>
                    <!-- It can be ForwardReactionRates, SpeciesFormationRates,
GivenSpeciesProductionRatesMax or GivenSpeciesProductionRatesMin. Default is
SpeciesFormationRates. -->
                    <GivenSpeciesProductionRates>
                        <![CDATA[CH4,H2,O2,H2O,CO2]]>
                    </GivenSpeciesProductionRates>
                    <!-- When chemicalTimeScale is GivenSpeciesProductionRatesMax
or GivenSpeciesProductionRatesMin this list must be specified.-->
                </PaSR>
            </combustionModel>
            <chemistrySource>
                <chemistrySource>chemistryODEs</chemistrySource>
                <!-- It can be chemistryODEs, chemistryNoSolver or
CUDAChemistrySourceNoSolver. -->
            </chemistrySource>
        </reactions>
        …
    </mixture>
…
</flow>
```

4.2.5 Boundary conditions

This subsection describes how to set up boundary conditions in the controller file. In OpenHurricane, the boundary parts are identified by names. Therefore, users should know the names of boundary parts of the corresponding mesh. There are several boundary conditions supported in OpenHurricane listed as follows:

(1) Riemann characteristic boundary or supersonic inlet

<userSpecifiedName>

<!-- [1] Boundary condition type -->

<bcType>pressureFarField</bcType> <!-- It can be *pressureFarField* for Riemann boundary or *supersonicInlet* for supersonic inlet boundary. -->

<!-- [2] Momentum condition -->

<momentumGivenBy>pressureAndMach</momentumGivenBy>

<!-- It can be *pressureAndMach*, *pressureAndVMag*, *ReynoldAndMach* or *ReynoldAndVMag*. Default is *pressureAndMach*. -->

<!-- (1) If element "**momentumGivenBy**" is specified as *pressureAndMach*, then the below parameters must be given. -->

101325.0 <!-- Pressure, unit: Pa. -->

<ma>0.2</ma><!-- Mach number. -->

<!-- (2) If element "**momentumGivenBy**" is specified as *pressureAndVMag*, then the below parameters must be given. -->

101325.0 <!-- Pressure, unit: Pa. -->

<v>20</v> <!-- Velocity magnitude, unit: m/s. -->

<!-- (3) If element "**momentumGivenBy**" is specified as *ReynoldAndMach*, then the below parameters must be given. -->

<Re>101325.0</Re> <!-- Reynold number based on a characteristic length of 1 meter. -->

<ma>0.2</ma> <!-- Mach number. -->

<!-- If element "**momentumGivenBy**" is specified as *ReynoldAndVMag,* then the below parameters must be given. -->

<Re>101325.0</Re> <!-- Reynold number based on a characteristic length of 1 meter. -->

<v>0.2</v><!-- Velocity magnitude, unit: m/s. -->

<!-- [3] Boundary direction -->

<directionType>directionVector</directionType> <!-- It can be *directionVector*, *normalToBoundary* or *angle*. -->

<!-- (1) If element "**directionType**" is specified as *directionVector,* then the below parameters must be given. -->

<direction>car(1,0,0)</direction> <!-- Which means the direction vector is (x = 1, y = 0, z = 0). -->

<!-- (2) If element "**directionType**" is specified as *angle,* then the below parameters must be given. -->

<alpha>0</alpha> <!-- Angle of attack. Given in degree. -->

<beta>0</beta> <!-- Angle of the sideslip. Given in degree. -->

<!-- [4] Turbulent condition -->

<specificationMethod>origianalTurbEquation</specificationMethod> <!-- It can be *intensityAndLength*, *intensityAndHydraulicD*, *viscosityRatio*, *origianalTurbEquation* or *givenDirectly*. -->

<!-- (1) If element "**specificationMethod**" is specified as *origianalTurbEquation,* then the below parameters must be given. -->
<kFactor>1.125</kFactor><!-- For SST model. Recommend range [3.0, 5.0]. -->
<wFactor>1.0</wFactor><!-- For SST model. Recommend range [1.0, 10.0]. -->
<comDomainLength>2.0</comDomainLength>
<nutFactor>1e-3</nutFactor><!-- For SpalartAllmaras model. Recommend range [1e-5,1e-2]-->
<!-- (2) If element "**specificationMethod**" is specified as *intensityAndLength,* then the below parameters must be given. -->
<intensity>5</intensity><!--. The turbulence intensity, which is defined as the ratio of the root-mean-square of the velocity fluctuation to the mean flow velocity. Unit: %. -->
<length-scale>1.0</length-scale><!--The turbulence length scale, which is a physical quantity related to the size of the large eddies that contain the energy in turbulent flows. -->
<!-- (3) If element "**specificationMethod**" is specified as *intensityAndHydraulicD,* then the below parameters must be given. -->
<intensity>5</intensity><!--. The turbulence intensity, which is defined as the ratio of the root-mean-square of the velocity fluctuation to the mean flow velocity. Unit: %. -->
<Hydraulic-Diameter>1.0</Hydraulic-Diameter><!-- The hydraulic diameter. Unit: m. -->
<!-- (4) If element "**specificationMethod**" is specified as *viscosityRatio,* then the below parameters must be given. -->
<intensity>5</intensity><!--. The turbulence intensity, which is defined as the ratio of the root-mean-square of the velocity fluctuation to the mean flow velocity. Unit: %. -->
<viscosity-ratio>10</viscosity-ratio><!-- The turbulent viscosity ratio. -->
<!-- (5) If element "**specificationMethod**" is specified as *givenDirectly,* then the below parameters must be given. -->
<k>5</k><!-- The turbulent kinetic energy for SST model. -->
<w>0.1</w><!-- The turbulent specific dissipation rate for SST model. -->
<nut>0.1</nut><!-- The modified turbulent viscosity for SpalartAllmaras model. -->

<!-- [5] Thermal condition -->
<T>300</T> <!-- Temperature. Unit: K. -->

<!-- [6] Species condition -->
<species>
<givenBy>massFraction</givenBy>
<!-- It can be *massFraction* or *moleFraction*. Default is *massFraction*.
[1] Note that you will explicitly set mass fractions only for the first N_s - 1 species. The solver will compute the mass fraction of the last species by subtracting the total of the specified mass fractions from 1. If you want to explicitly specify the mass fraction of the last species, you must reorder the species in the chemical mechanism file.
[2] Note that you should put the species with largest mass fractions at the end of the species list in the chemical mechanism file to reduce the numerical error.

```
            [3] Note that the default value of mass fractions for the first Ns -1 species is zero. Therefore, if not specified, it will be zero.
            -->
            <H2>0.03207</H2>
            <O2>0.25447</O2>
            <OH>0.0</OH>
        </species>

</userSpecifiedName>
```

(2) Subsonic inlet

```
<userSpecifiedName>
    <!-- [1] Boundary condition type -->
    <bcType>subsonicInlet</bcType> <!-- It can be subsonicInlet or pressureInlet. -->

    <!-- [2] Momentum condition -->
    <staticPressure>80000.0</staticPressure> <!-- Supersonic inlet static pressure or subsonic initial pressure. Unit: Pa. -->
    <totalPressure>101325.0</totalPressure> <!-- Unit: Pa. -->

    <!-- [3] Boundary direction -->
    <!-- See Riemann characteristic boundary or supersonic inlet. -->

    <!-- [4] Turbulent condition -->
    <!-- See Riemann characteristic boundary or supersonic inlet. -->

    <!-- [5] Thermal condition -->
    <totalTemperature>300</totalTemperature> <!-- Total temperature. Unit: K. -->

    <!-- [6] Species condition -->
    <!-- See Riemann characteristic boundary or supersonic inlet. -->
</userSpecifiedName>
```

(3) Jet inlet

```
<userSpecifiedName>
    <!-- [1] Boundary condition type -->
    <bcType>massFlowInlet</bcType> <!-- It should be massFlowInlet. -->

    <!-- [2] Momentum condition -->
    <p>80000.0</p> <!-- Supersonic inlet static pressure or subsonic initial pressure. Unit: Pa. -->
    <givenBy>massFlowRate</givenBy><!-- It can be massFlowRate or massFlux. -->
    <!-- (1) If element "givenBy" is specified as massFlowRate, then the below parameters must be given. -->
        <massFlowRate>1.0</massFlowRate> <!-- Mass flow rate. Unit: kg/s. -->
```

```
    <!-- (2) If element "givenBy" is specified as massFlux, then the below parameters must be given. -->
        <massFlux>1.0</massFlux> <!-- Mass flow flux. Unit: kg/(m^2 s). -->

    <!-- [3] Boundary direction -->
    <!-- See Riemann characteristic boundary or supersonic inlet. -->

    <!-- [4] Turbulent condition -->
    <!-- See Riemann characteristic boundary or supersonic inlet. -->

    <!-- [5] Thermal condition -->
    <totalTemperature>300</totalTemperature> <!-- Total temperature. Unit: K. -->

    <!-- [6] Species condition -->
    <!-- See Riemann characteristic boundary or supersonic inlet. -->
</userSpecifiedName>
```

(4) Outlet

```
<userSpecifiedName>
    <!-- [1] Boundary condition type -->
    <bcType>pressureOutlet</bcType> <!-- It should be pressureOutlet or outflow. -->

    <!-- [2] Momentum condition -->
    <p>1.01325e5</p> <!-- Note if the pressure is specified by this parameter, then the outlet pressure will be specified as the given value when the outlet face is subsonic. Unit: Pa. -->
</userSpecifiedName>
```

(5) Wall

```
<userSpecifiedName>
    <!-- [1] Boundary condition type -->
    <bcType>wall</bcType> <!-- It should be wall. -->

    <!-- [2] Momentum condition -->
    <momentum>
        <shearCondition>noSlip</shearCondition><!-- "noSlip" for no slip wall, "invSlip" for inviscid slip wall. -->
    </momentum>

    <!-- [3] Turbulent condition -->
    <wallFunction>off</wallFunction><!-- Wall function for SST model. -->

    <!-- [4] Thermal condition -->
    <thermal>
        <thermalCondition>isothermal</thermalCondition><!-- It can be isothermal, adiabatic. -->
```

```
            <T>300.0</T><!-- If "isothermal" is specified, then this parameter must be given. It is wall temperature. Unit: K. -->
        </thermal>

        <!-- [5] Species condition -->
        <!-- See Riemann characteristic boundary or supersonic inlet. -->
    </userSpecifiedName>
```

(6) Symmetry

```
    <userSpecifiedName>
        <!-- [1] Boundary condition type -->
        <bcType>symmetry</bcType> <!-- It should be symmetry. -->
    </userSpecifiedName>
```

(7) Periodic

```
    <userSpecifiedName>
        <!-- [1] Boundary condition type -->
        <bcType>periodic</bcType> <!-- It can be periodic. -->
    </userSpecifiedName>
```

4.2.6 Reference

Users can change the reference values that are used in the evaluation of several derived quantities and non-dimensional coefficients. These reference values are only used in monitoring or for post-processing.

```
    <ref>
        <givenBy>pFarInletFaceName</givenBy> <!-- If not specified, then the following value will be used. -->
        <length>1.0</length> <!-- The reference length [m], which is used to compute the moment coefficient. Default is 1. -->
        <area>1</area> <!-- The reference area, which is used to compute the force and the moment coefficients. [m^2]. Default is 1. -->
        <density>1.225</density> <!-- The reference density. [kg/m^3] -->
        <enthalpy>0.0</enthalpy> <!-- The enthalpy. [J] -->
        <pressure>0.0</pressure> <!-- The pressure. [Pa] -->
        <temperature>288.16</temperature><!-- The temperature. [K] -->
        <velocity>1</velocity><!-- The velocity. [m/s] -->
        <viscosity>1.7894e-5</viscosity><!-- The viscosity. [Pa s] -->
        <specificHeatRatio>1.4</specificHeatRatio>
        <gasConstantNumber>287.06</gasConstantNumber><!-- [J/(kg K)] -->
    </ref>
```

4.2.7 Initialization

Users can also define the way to initialize the simulations.

```
<initialization>
    <!-- (1) Initialize from Specific Boundary-->
    <initFromBoundary>boundaryFaceName</initFromBoundary>
    <!-- (2) Initialize from Given Value. However, the above option initFromBoundary must be given.-->
    <initFromValue>
        <rho>1</rho> <!-- Density. Unit: kg/m^3. -->
        <v>(1, 1, 1)</v> <!-- Velocity. Unit: m/s. -->
        <p>1</p> <!-- Pressure. Unit: Pa. -->
        <T>1</T> <!-- Temperature. Unit: K. -->
     </initFromValue>
    <!-- (3) Patch Region. -->
    <initPatch><![CDATA[patch1]]></initPatch>
    <patch1>
        <region>region0</region><!-- The marked region must be defined in the grid setting. See 4.2.1 Grid setting-->
        <patchVar><![CDATA[p, v]]></patchVar>
        <p>1.0e5</p>
        <v>(100.0, 20.0, 0.0)</v> <!-- Velocity. Unit: m/s. -->
    </patch1>
</initialization>
```

4.2.8 Write control

Write control section of the controller file is to control the output of solutions. The details about write control section please see section 4.3.

4.3 Output file format

Output of data is usually important during the simulation. OpenHurricane would generate intermediate files at intervals of steps during the run to save the intermediate solutions. Besides, the solutions can be exported in the format used by CGNS or Tecplot software. The output solution format can be controlled in the controller file as given below:

```
<writeControl>
    <solutionWrite>
        <writeType>tecplot</writeType><!-- Solution file format: “tecplot” or “cgns”-->
        <solutionType>onlyInstantaneous</solutionType>
        <varList>
            <![CDATA[…]]>   <!-- Given the name list of variables to be exported. -->
        </varList>
```

```
        </solutionWrite>
    </writeControl>
```

4.4 Start simulations

This section shows how to start a simulation using the OpenHurricane application. Users can run the simulation in CPU serial computing or parallel computing. Users can also use GPU as accelerator to speed up the computation of chemical source terms and gas physical properties.

4.4.1 Running simulation in serial

The OpenHurricane application can be executed in a terminal window. Users can simply enter "./OpenHurricane -help" at the command line to find the usage of OpenHurricane application. The usage for the current version of OpenHurricane is:

```
Usage: ./OpenHurricane.exe [opts] <opts args> ...

Options
========================================
 -help       (-h)
             Print this help information on this executable program.
 -input      (-I)     <file name>
             Input the case file specified by <file name>. If not specify the case file name, the
             default case filename: "OpenHurricane.cont" would be used.
 -inputpath      (-IP)     <file path>
             Set the file path specified by <file path>. If not specify the file path, the case
             file and executable file should be included in the same directory.
 -output      (-O)     <file name>
             Output the data file specified by <file name>. If not specify the case file name,
             the default case filename: "OpenHurricane.h5" would be used.
 -outputpath      (-OP)     <file path>
             Set the file path specified by <file path>. If not specify the file path, the data
             file and executable file should be included in the same directory.
 -log       (-L)     <file name (optional)>
             Turn on the log file system on this executable program, and use the default log
             file: "OpenHurricane.log" if the file name is not specified.
 -version      (-V)
             Print version information on this executable program.
 -nGPU
             To set the number of GPUs per machine.
 -tecplotFileWriteOnMaster
```

Set the Tecplot file writing on master process. Default is false.

-noWriteAtEnd

To specify do not write the output and relay files at the end of the computation. Default is false.

-skewnessCorrection

To specify the implementation of a correction for the interpolated value at the face center when the face is skewed. Default is false.

-checkWriteSolution

To specify checking the solution for writing out is NaN or INF. Default is false.

Therefore, users can type

```
./OpenHurricane -i userSpecifiedController.cont
```

at the command line to start the simulation in serial computing with a controller named "*userSpecifiedController.cont*".

4.4.2 Running simulation in parallel

The parallel running uses the MPI (Message Passing Interface) package, including: Intel MPI, openMPI and MPICH2. To run the simulation in parallel, users can use the below command:

```
mpiexec -np 12 ./OpenHurricane -i userSpecifiedController.cont
```

The above command line uses 12 CPU processes, and users can change the number of processes to be used according to their computers or servers.

4.4.3 Running simulation using CPU and GPU

Users can also use GPU as accelerator to speed up the computation of chemical source terms and gas physical properties. The below command line can be used to start simulations with both CPU and GPU:

```
mpiexec -np 12 ./OpenHurricane -i userSpecifiedController.cont -nGPU 2
```

The above command line uses 2 GPUs with 12 CPU processes. In the current version of OpenHurricane, the number of GPUs cannot larger than the number of CPU processes.

5. OpenHurricane examples

This chapter deals with the information of OpenHurricane example cases.

5.1 Supersonic combustion

The Burrows-Kurkov supersonic combustion flow experiment is a scramjet experiment conducted by Burrows et al. at NASA in the 1970's[2]. The fuel used is hydrogen, and the experimental setup is shown in Fig. 2. The inlet height of the experimental device is 8.9 cm, the height of the hydrogen fuel orifice is 0.4 cm, and the thickness of the orifice lip is 0.076 cm. The lower wall of the experimental device is tilted from the back of the hydrogen fuel orifice, and the distance to the outlet is 35.6 cm, and the height of the outlet is 10.48 cm. The experimental device can be used as a two-dimensional combustion chamber for simulation.

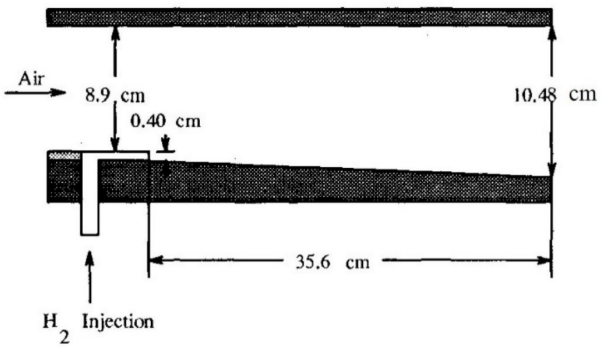

Fig. 2. Engine schematic[2].

5.1.1 Mesh information

The calculations are performed using a structured grid with the hydrogen fuel nozzle location at x = 0 m as shown in Fig. 3. The total number of grid cells is about 70,000, and the first layer of the grid is at a distance of 0.01mm from the wall.

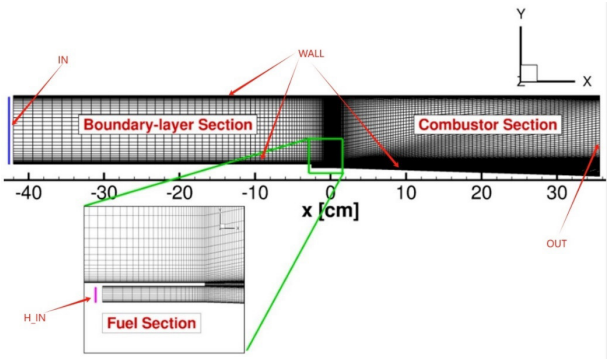

Fig. 3. Grid schematic.

5.1.2 Boundary and initial conditions

The example calculation is divided into two parts: mixing and combustion. For the mixing calculation, the air inlet (IN) parameter is Ma = 2.44, static pressure p = 101325Pa, static temperature T=1150K, and the mass fraction of each species is: O_2-0, H_2O-0.256, and N_2-0.744. The hydrogen inlet (H_IN) parameter is Ma = 1, static pressure p = 101325 Pa, static temperature T = 254K. For the combustion calculation, the static temperature of the air inlet was changed to T = 1270K, the mass fraction of each substance was changed to: O_2-0.258, H_2O-0.256, N_2-0.486. The rest of the parameters remained unchanged.

```
<IN>
    <bcType>pressureFarField</bcType>
    <momentumGivenBy>pressureAndMach</momentumGivenBy>
        <p>101325.0</p>
        <ma>2.44</ma>
    <directionType>directionVector</directionType>
        <direction>car(1,0,0)</direction>
    <specificationMethod>viscosityRatio</specificationMethod>
        <intensity>3</intensity>
        <viscosity-ratio>10</viscosity-ratio>
    <T>1150</T>
    <species>
        <givenBy>massFraction</givenBy>
        <H2O>0.256</H2O>
    </species>
</IN>
<H_IN>
    <bcType>pressureFarField</bcType>
    <momentumGivenBy>pressureAndMach</momentumGivenBy>
        <p>101325.0</p>
```

```
        <ma>1</ma>
    <directionType>directionVector</directionType>
        <direction>car(1,0,0)</direction>
    <specificationMethod>intensityAndHydraulicD</specificationMethod>
        <intensity>0.5</intensity>
        <Hydraulic-Diameter>4e-3</Hydraulic-Diameter>
    <T>254</T>
     <species>
        <givenBy>massFraction</givenBy>
        <H2>1.0</H2>
    </species>
</H_IN>
```

The outlet boundary (OUT) condition is set to outflow and the pressure parameter is 101325 Pa. The wall (WALL) is no-slip and isothermal, T_w = 298 K. Initialization is done using air inlet (IN) initialization.

```
<OUT>
    <bcType>outflow</bcType>
    <p>1.01325e5</p>
</OUT>
<WALL>
    <bcType>wall</bcType>
    <momentum>
        <shearCondition>noSlip</shearCondition>
    </momentum>
    <thermal>
        <thermalCondition>isothermal</thermalCondition>
        <T>298.0</T>
    </thermal>
</WALL>
```

5.1.3 Numerical setups

The algorithm is a steady state, turbulent flow where the fluid is an ideal gas and the inlet is injected with air and hydrogen respectively. The turbulence model is Standard-SST model, the turbulence Prandtl number Pr_t is taken as 0.9 and the turbulence Schmidt number Sc_t is taken as 0.5. The spatial scheme is used as HLLC and the temporal scheme is used as LUSGS. Laminar finite rate model and PaSR model were used for combustion modeling respectively.

```
        <timeMethod>
            <timeMarching>LUSGS</timeMarching>
```

```
                <LUSGS>
                    <omegaForLUSGS>1.050</omegaForLUSGS>
                    <betaT>0.2</betaT>
                </LUSGS>
                <sourceTermTreating>diagImpliciSource</sourceTermTreating>
            </timeMethod>
<spatialScheme>
    <spatialScheme>upwindSchemes</spatialScheme>
                <upwindSchemes>
                    <upwindScheme>HLLC</upwindScheme>
                    <HLLC>
                        <lowMachCorrected>off</lowMachCorrected>
                        <enhancedShockStability>off</enhancedShockStability>
                    </HLLC>
                </upwindSchemes>
</spatialScheme>
```

5.1.4 Running the simulations

The mixing calculations are straightforward from scratch, while the combustion calculations need to be continued with the results of the mixing calculations and the ignition operation. A rectangular area with a patch temperature of 2000 K is selected to simulate the ignition process.

5.1.5 Simulations results

Fig. 3. shows how the component distributions at the outlet cross section of the doped condition compare with the experimental data and the results calculated by the NASA PNS program. It can be seen that the Hurricane calculations are in good agreement with the experimental and UPS calculations, and are closer to the experimental results than the UPS calculations near the wall.

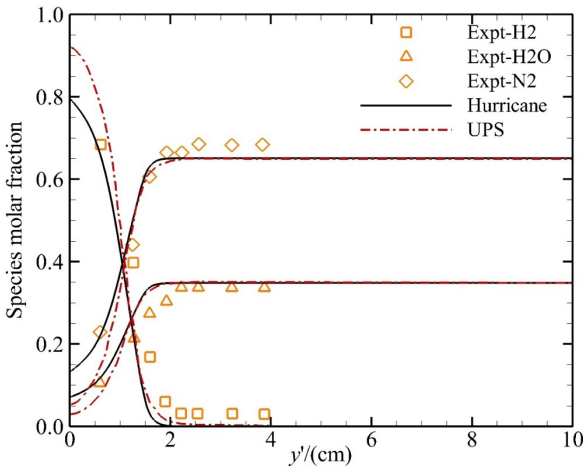

Fig. 4. Mixing condition.

The molar concentration distributions of the components at the exit of the combustion condition are shown in Fig. 4 and Fig. 5, respectively. The Mach number and total temperature distributions at the exit of the combustion condition are shown in Fig. 6 and Fig. 7, respectively.

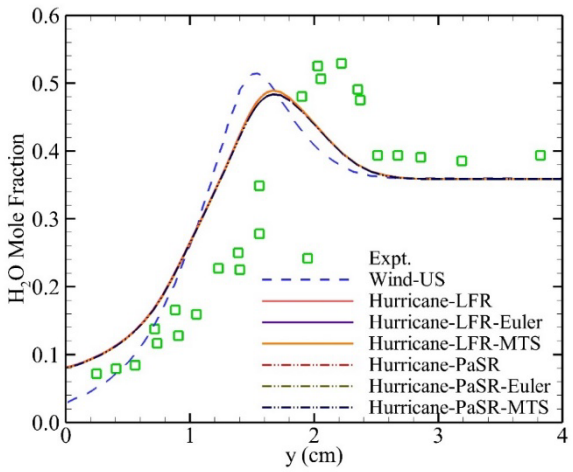

Fig. 5. H_2O molar fraction.

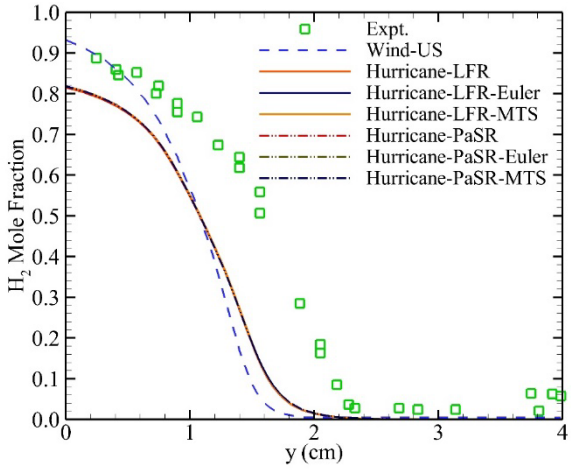

Fig. 6. H_2 molar fraction.

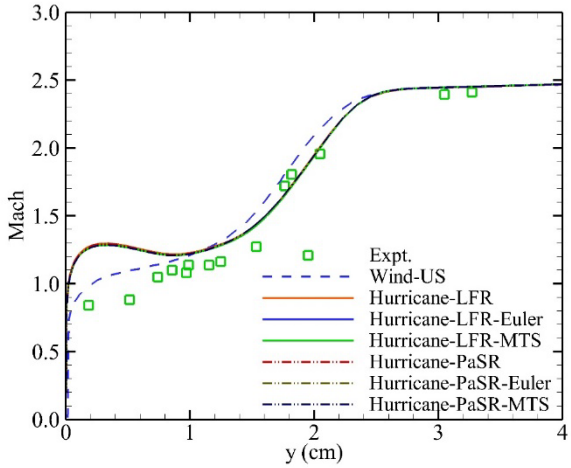

Fig. 7. Mach number distribution.

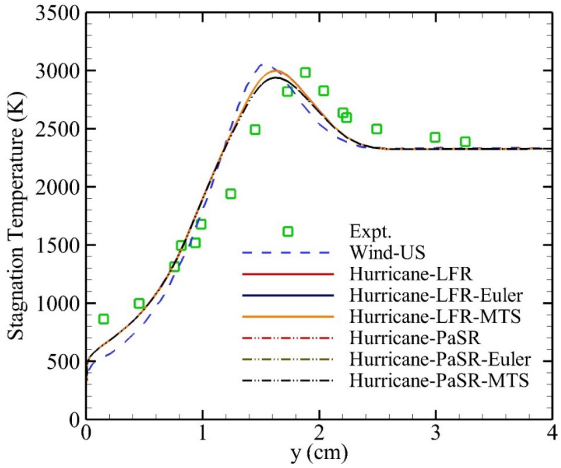

Fig. 8. Stagnation temperature distribution.

5.2 Rotating detonation engine

The combustion products may remain near the inlet of the combustion chamber of

Rotating detonation engine (RDE) and mix with fresh reactants, resulting in a discontinuous distribution of reactants in the inlet triangle and an inhomogeneous wavefront fraction. RDE usually adopts annular combustion chamber, and since the radial width of the combustion chamber is much smaller compared with the circumferential length, the effect of the combustion chamber width can be neglected, and the two-dimensional simplified structure obtained by unfolding the combustion chamber along its axis is used as the computational domain for the numerical simulation work.

5.2.1 Mesh information

Along the *x*-direction is a uniform mesh of size 0.2 mm, and along the y-direction the mesh gradually increases from 0.1 mm to 1 mm with a growth rate of 1.01, for a total of 0.3528 million mesh globally.

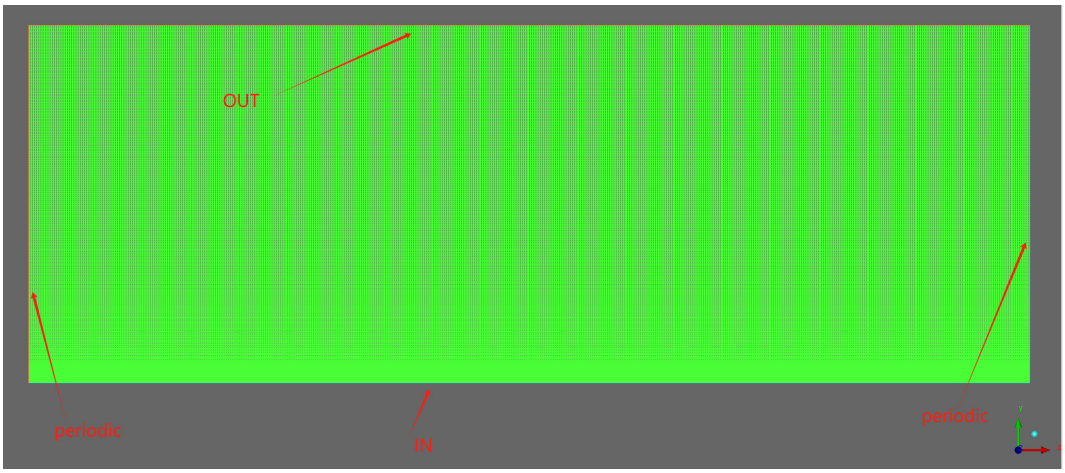

Fig. 9. Grid schematic

5.2.2 Boundary and initial conditions

The combustion chamber has a fuel and oxidizer inlet (INLET) at the bottom, an outlet (OUT) for the detonation products at the top, and periodic boundaries at the left and right. Chemically stoichiometric premixed ethylene/air is injected into the combustion chamber from the bottom with total injection temperature and pressure of 300 K and 10 atm.

```
<INLET>
    <bcType>detonationInlet</bcType>
    <totalTemperature>300.0</totalTemperature>
```

```
        <totalPressure>1013250</totalPressure>

        <directionType>normalToBoundary</directionType>

        <species>
            <givenBy>massFraction</givenBy>
            <CO2>0.0</CO2>
            <C2H4>0.06364</C2H4>
            <H2O>0.0</H2O>
            <O>0.0</O>
            <O2>0.21818</O2>
            <CO>0.0</CO>
            <N2>0.71818</N2>
        </species>
    </INLET>

    <OUT>
        <bcType>outflow</bcType>
        <p>101325</p>
    </OUT>

    <PRIOID1>
        <bcType>periodic</bcType>
    </PRIOID1>

    <periodic_sh>
        <bcType>periodic</bcType>
    </periodic_sh>
```

5.2.3 Numerical setups

Laminar flow was used for the simulation calculations. BDF123LUSGS was used for the temporal format. HLLC was used for the spatial format. And the finite rate method was used for the combustion modeling.

```
    <timeMethod>
        <timeMarching>BDF123LUSGS</timeMarching>

        <LUSGS>
            <omegaForLUSGS>1.05</omegaForLUSGS>
            <betaT>0.2</betaT>
        </LUSGS>
```

```
            <BDF123LUSGS>
                <BDF123>BDF3</BDF123>
            </BDF123LUSGS>

            <!-- For unsteady flow. -->
            <physicalTimes>
                <dynamicTimeStep>off</dynamicTimeStep>
                <dynamicCFL>0.1</dynamicCFL>
                <phyTimeStep>5e-9</phyTimeStep>
                <maxPhyTime>0.1</maxPhyTime>
                <totalPhyTime>0.0</totalPhyTime>
            </physicalTimes>
            <dualTimeStep>
                <subIteration>
                    <maxSubStep>45</maxSubStep>
                    <minSubStep>40</minSubStep>
                </subIteration>
                <iterationMethod>Newton</iterationMethod>
                <newTimeStepInitialize>lastTime</newTimeStepInitialize>
                <beginStep>5</beginStep>
            </dualTimeStep>
            <sourceTermTreating>diagImpliciSource</sourceTermTreating>
        </timeMethod>
<spatialScheme>
  <spatialScheme>upwindSchemes</spatialScheme>
        <upwindSchemes>
            <upwindScheme>HLLC</upwindScheme>
            <HLLC>
                <lowMachCorrected>off</lowMachCorrected>
                <enhancedShockStability>off</enhancedShockStability>
            </HLLC>
        </upwindSchemes>
</spatialScheme>
```

5.2.4 Running the simulations

RDE needs to be ignited for detonation. A high-temperature, high-pressure, high-velocity zone (2000 K, 3 MPa, 1800 m/s) was set up at the lower left corner of the combustion chamber (x = 10~15 mm, y = 0~20 mm) as the ignition zone. To the right of the ignition zone was a pre-filled zone (x = 15~235 mm, y = 0~20 mm) consisting of pre-mixed ethylene/air with an equivalence ratio of 1, whereas the initial components in the rest of the combustion chamber were combustion products consisting of water

and carbon dioxide. In order to prevent the formation of two burst waves with opposite propagation directions after ignition, a buffer zone (x = 0~10mm, y = 0~20mm) was set to the left of the ignition zone, and the temperature, pressure and velocity in the buffer zone were linearly reduced to the initial filling state of the combustion chamber.

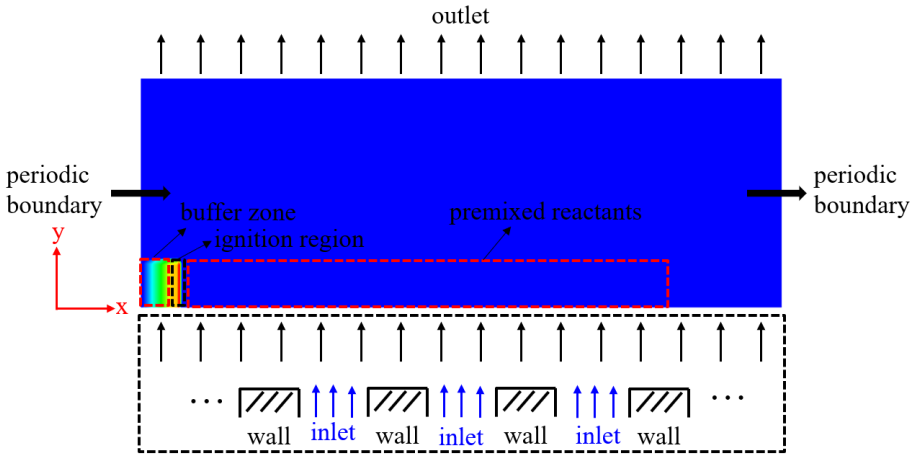

Fig. 10. Schematic diagram of the 2D computational domain with boundary conditions

5.2.5 Simulations results

The numerical simulation of RDE is very non-stationary, requiring hundreds of thousands of steps of computation to obtain a roughly stable rotating detonation wave. The temperature cloud of the rotating detonation wave is shown in Fig. 11.

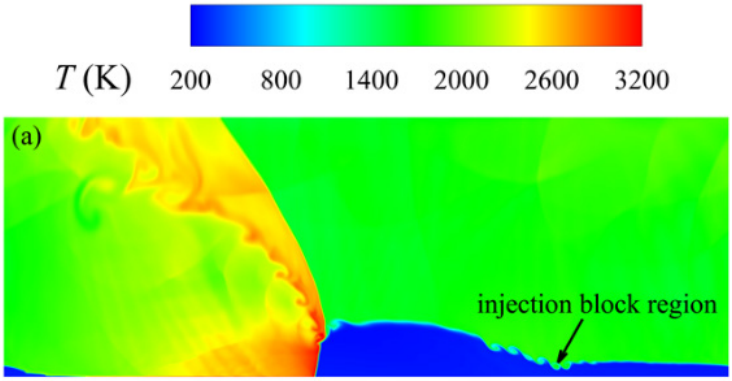

Fig. 11. Instantaneous contour plots of temperature

5.3 Nozzle for a scramjet engine

The calculation example is for the nozzle flow of a scramjet engine[3]. The configuration used for the calculation is shown in Fig. 12.

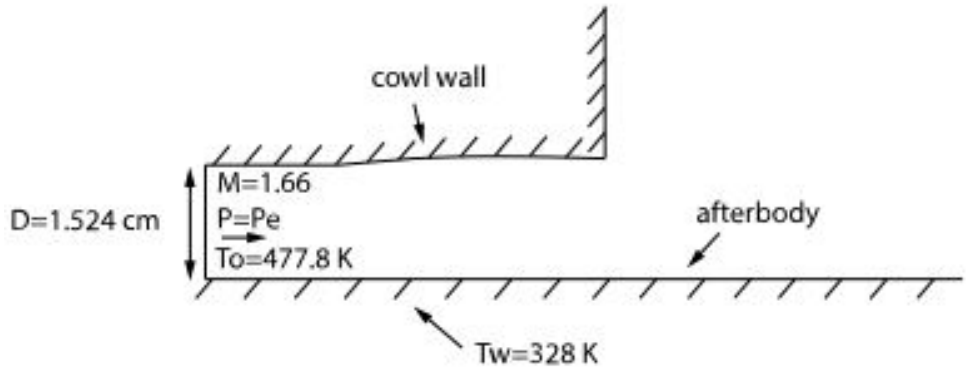

Fig. 12. Scramjet nozzle configuration[3]

5.3.1 Mesh information

The calculations were performed using the grid shown in Fig. 13.

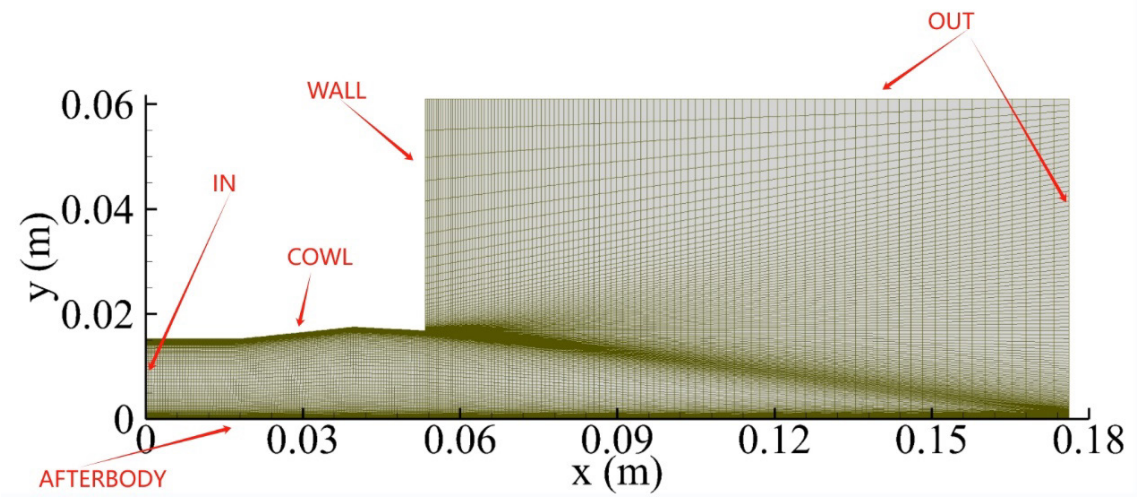

Fig. 13. Nozzle grid.

5.3.2 Boundary and initial conditions

The inlet(IN) conditions are: total pressure p_t =551600 Pa, static pressure p = 127100 Pa, total temperature T_t = 477.8 K and turbulence intensity = 2%. The outlet (OUT) back pressure p = 2780 Pa. The walls are isothermal with T_w = 328 K, where WALL is inviscid and the rest are non-slip walls.

```
<INLET>
    <!-- (1) Boundary type -->
    <bcType>pressureInlet</bcType>
    <!-- (2) Momentum -->
        <staticPressure>127100</staticPressure>
        <totalPressure>551600</totalPressure>
```

```
        <!-- (3) Direction -->
            <directionType>directionVector</directionType>
            <direction>car(1,0,0)</direction>
        <!-- (4) Turbulent Inlet Condition-->

            <specificationMethod>viscosityRatio</specificationMethod>
                <intensity>2</intensity>
                <viscosity-ratio>10</viscosity-ratio>
        <!-- (5) Thermal -->
            <totalTemperature>477.8</totalTemperature>
    </INLET>

    <OUTLET>
        <bcType>outflow</bcType>
        <p>2780</p>
    </OUTLET>


    <WALL>
        <!-- (1) Basic Condition -->
        <bcType>wall</bcType>
        <!-- (2) Momentum -->
        <momentum>
            <shearCondition>invSlip</shearCondition>
        </momentum>
        <thermal>
            <thermalCondition>isothermal</thermalCondition>
            <T>328.0</T>
        </thermal>
    </WALL>

    <COWL>
        <!-- (1) Basic Condition -->
        <bcType>wall</bcType>
        <!-- (2) Momentum -->
        <momentum>
            <shearCondition>noSlip</shearCondition>
        </momentum>
        <thermal>
            <thermalCondition>isothermal</thermalCondition>
            <T>328.0</T>
        </thermal>
    </COWL>
    <AFTERBODY>
```

```
            <!-- (1) Basic Condition -->
            <bcType>wall</bcType>
            <!-- (2) Momentum -->
            <momentum>
                <shearCondition>noSlip</shearCondition>
            </momentum>
            <thermal>
                <thermalCondition>isothermal</thermalCondition>
                <T>328.0</T>
            </thermal>
        </AFTERBODY>
```

5.3.3 Numerical setups

The calculation example uses a turbulence model. The gas is an ideal gas with molecular weight: 113.2, viscous coefficient: 1.7894e-5 kg/(m s), thermal conductivity: 0.0242W/(m K), and a segmented linear distribution of specific heat as shown in the table below.

Table 1. The segmented linear distribution of specific heat

T(K)	c_p (J/(kg K))
205.6	423.2
438.9	543.65
533.3	572.1

The spatial format is in HLLC and the temporal format is in LUSGS. The initialization method is inlet boundary initialization.

```
        <timeMethod>
            <timeMarching>LUSGS</timeMarching>
            <LUSGS>
                <omegaForLUSGS>1.050</omegaForLUSGS>
                <betaT>0.2</betaT>
            </LUSGS>

            <sourceTermTreating>diagImpliciSource</sourceTermTreating>
        </timeMethod>

<spatialScheme>
    <spatialScheme>upwindSchemes</spatialScheme>
            <upwindSchemes>
                <upwindScheme>HLLC</upwindScheme>
                <HLLC>
```

```
                        <lowMachCorrected>off</lowMachCorrected>
                        <enhancedShockStability>off</enhancedShockStability>
                    </HLLC>
                </upwindSchemes>
</spatialScheme>
```

5.3.4 Running the simulations

Simulation results can be obtained after 3300 steps of computation.

5.3.5 Simulations results

Mach number and pressure distributions are shown in Fig. 14 and Fig. 15, respectively. The lower wall pressure and heat flow distributions are compared with the experimental counterparts shown in Fig. 16.

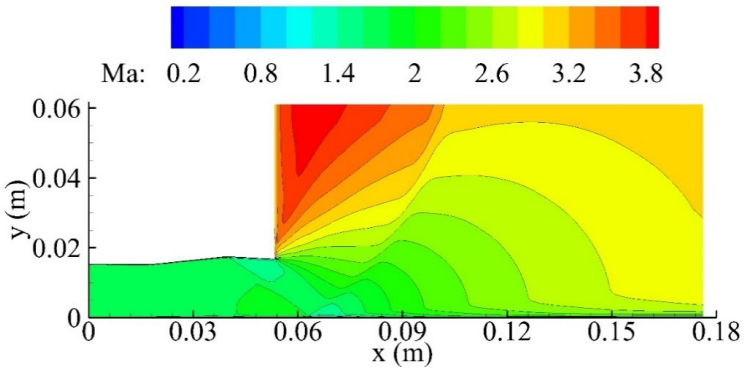

Fig. 14. Mach number distribution

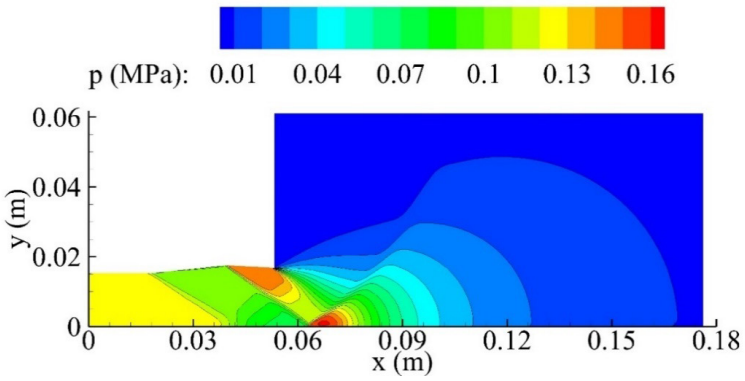

Fig. 15. Pressure distribution

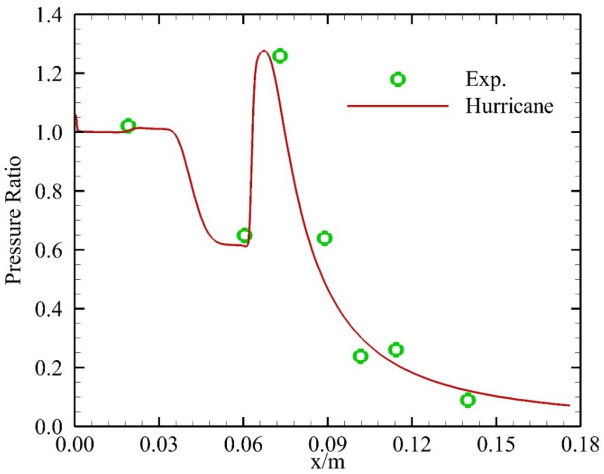

Fig. 16. Lower wall pressure distribution

5.4 Sod shock tube

The test case is an unsteady flow in a shock tube. Initially, the left side of the tube is a high pressure gas and the right side is a low pressure gas, and a separator membrane is used in the middle of the tube to separate the two states of the gas. As shown in Fig. 17.

With the sudden breakdown of the diaphragm, normal shock waves, contact discontinuities and center sparse waves are generated to propagate inside the tube. Fig. 18 shows the propagation of these waves inside the tube.

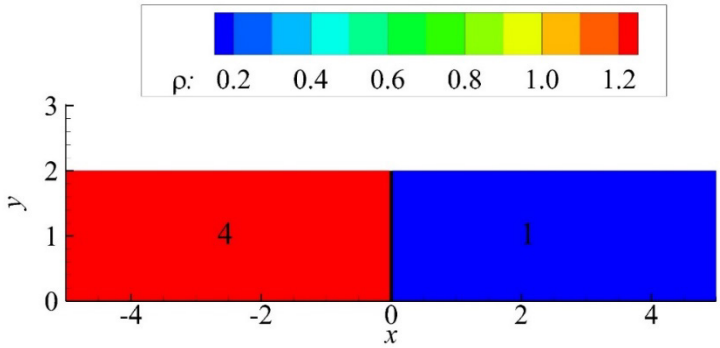

Fig. 17. Initial state of gas in the tube.

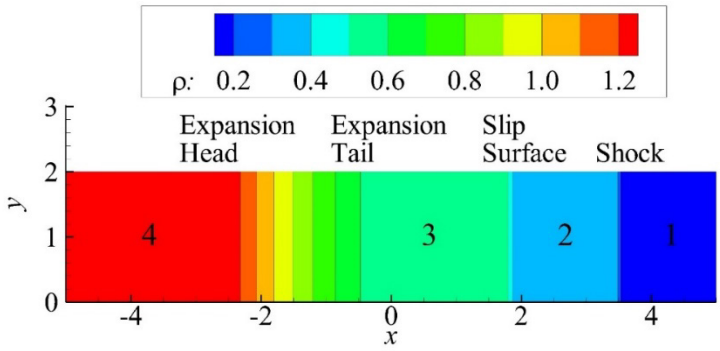

Fig. 18. Gas density distribution in the tube after a period of time

5.4.1 Mesh information

Homogeneous H-mesh, the number of cells in the x-direction is 400.

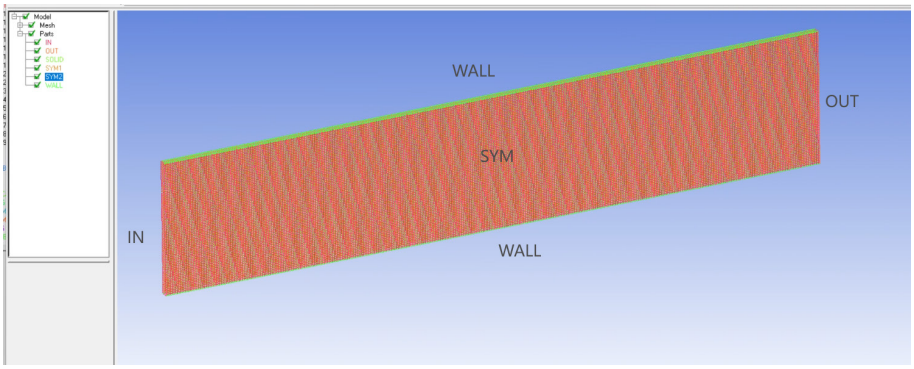

Fig. 19. Calculation mesh of the Sod shock tube

5.4.2 Boundary and initial conditions

The computational domain is quasi-two-dimensional, with no-slip walls all around and symmetry plane boundary conditions on both sides.

Specify the initialization area and parameters such as temperature and pressure within the area.

```
<boundaryCondition>

        <IN>
            <bcType>wall</bcType>
            <momentum>
                <shearCondition>noSlip</shearCondition>
            </momentum>
            <thermal>
                <thermalCondition>adiabatic</thermalCondition>
            </thermal>
        </IN>
```

```
            <OUT>
                <bcType>wall</bcType>
                <momentum>
                    <shearCondition>noSlip</shearCondition>
                </momentum>
                <thermal>
                    <thermalCondition>adiabatic</thermalCondition>
                </thermal>
            </OUT>
            <WALL>
                <bcType>wall</bcType>
                <momentum>
                    <shearCondition>noSlip</shearCondition>
                </momentum>
                <thermal>
                    <thermalCondition>adiabatic</thermalCondition>
                </thermal>
            </WALL>
            <SYM1>
                <bcType>symmetry</bcType>
            </SYM1>
            <SYM2>
                <bcType>symmetry</bcType>
            </SYM2>
        </boundaryCondition>
```

Initial patching:

```
            <region3>
                <id>3</id>
                <option>inside</option>
                <shape>plane</shape>
                <normal>(-1.0,0.0,0.0)</normal>
                <point>(0.0,1.0,0.0)</point>
            </region3>

    <initialization>
            <initFromBoundary>IN</initFromBoundary>
            <!--Initialize from Given Value-->
            <initFromValue>
                <rho>1.225</rho>
                <v>(0,0,0)</v>
                <p>105494.55</p>
                <T>300</T>
```

```
            <E>215082</E>
        </initFromValue>

        <!-- 【6.2】Patch Region-->
        <initPatch>
            <![CDATA[patch2]]>
        </initPatch>
        <patch2>
            <region>region3</region>
            <type>uniform</type>
            <!-- type : uniform or distributed -->
            <patchVar>
                <![CDATA[T,rho,p,E,v]]>
            </patchVar>
            <p>10549.455</p>
            <v>(0.0,0.0,0.0)</v>
            <rho>0.153125</rho>
            <T>240</T>
            <E>173068.5714</E>
        </patch2>
    </initialization>
```

5.4.3 Numerical setups

The Sod tube is a transient problem. The time format is the TVD third-order R-K method, and the spatial format is the upwind HLLC.

```
    <timeMethod>
            <timeMarching>unsteadyTVDRK</timeMarching>
            <TVDRK>
                <TVDRKOrder>TVDRK3</TVDRKOrder>
                <cfl>1.0</cfl>
            </TVDRK>
            <physicalTimes>
                <phyTimeStep>1.36176e-5</phyTimeStep>
                <maxPhyTime>0.0068088</maxPhyTime>
                <totalPhyTime>0</totalPhyTime>
            </physicalTimes>

            <sourceTermTreating>explicitSource</sourceTermTreating>

        </timeMethod>

    <spatialScheme>
```

```
        <spatialScheme>upwindSchemes</spatialScheme>
        <upwindSchemes>
            <upwindScheme>HLLC</upwindScheme>
        </upwindSchemes>

        <isLowMachCorrection>off</isLowMachCorrection>

        <reconstruction>linear</reconstruction>
        <limitersForLinear>Venk</limitersForLinear>
        <K>0.010</K>
        <gradient>leastSquareGrad</gradient>
        <weightType>WLSQG</weightType>
        <cellNeighbourCell>NODENEIGHBOUR</cellNeighbourCell>
    </spatialScheme>
```

5.4.4 Running the simulations

Place the mesh file, the control file, and the OpenHurricane calculation program in the same folder. Right-click in the current folder to open a powershell terminal and enter the parallel computing code: mpiexec -np xx OpenHurricane.exe, where "xx" is the number of parallel cores, e.g., "mpiexec -np 10 OpenHurricane.exe".

5.4.5 Simulations results

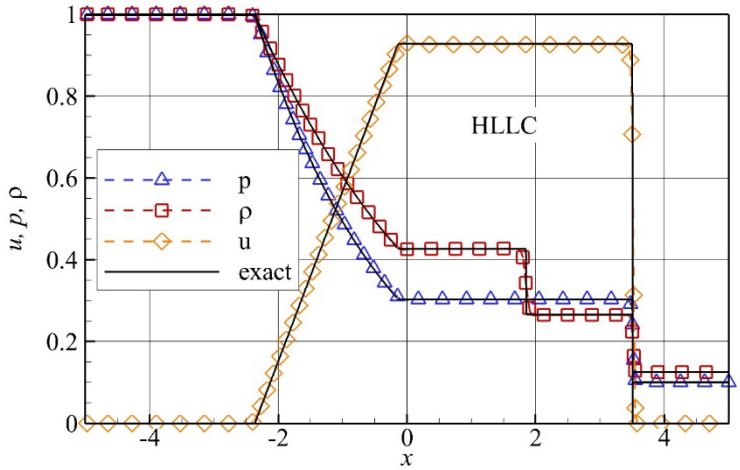

Fig. 20. Numerical results and theoretical solutions for Sod shock tube pressure, density and velocity

5.5 Supersonic laminar flow plate boundary layer

This test case is a supersonic laminar flow flat plate case[4] to show the ability of the

program to discriminate the boundary layer of supersonic laminar flow. The boundary conditions and mesh used for its calculation are shown in Fig. 21.

5.5.1 Mesh information

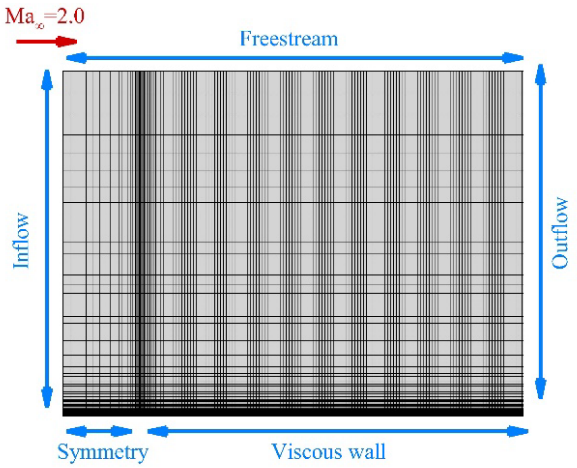

Fig. 21. Schematic of the boundary conditions and mesh used for the calculation

5.5.2 Boundary and initial conditions

The left and upper sides are pressure far-field boundary conditions with free incoming flow directed horizontally to the right. The incoming flow is Mach 2, the pressure is 2534 Pa, and the temperature is 221.6 K.

```
<boundaryCondition>

    <!--【4.1】 pressure-far-field or supersonic inlet boundary condition. -->
    <IN>
        <!-- Boundary type -->
        <bcType>pressureFarField</bcType>
        <momentumGivenBy>pressureAndMach</momentumGivenBy>
        <ma>2</ma>
        <p>2534.0</p>

        <directionType>directionVector</directionType>
        <direction>car(1,0,0)</direction>
        <T>221.6</T>
    </IN>
    <FAR>
        <bcType>pressureFarField</bcType>
        <momentumGivenBy>pressureAndMach</momentumGivenBy>
        <p>2534</p>
        <ma>2</ma>
```

```
            <directionType>directionVector</directionType>
            <direction>car(1,0,0)</direction>
            <T>221.6</T>
        </FAR>

        <OUT>
            <bcType>pressureOutlet</bcType>
            <p>2534</p>
        </OUT>

        <WALL>
            <bcType>wall</bcType>
            <momentum>
                <shearCondition>noSlip</shearCondition>

            </momentum>
            <thermal>
                <thermalCondition>adiabatic</thermalCondition>
                <!-- isothermal, adiabatic-->
            </thermal>
            <wallFunction>off</wallFunction>
        </WALL>


        <!--【4.7】 Symmetry -->
        <SYM1>
            <bcType>symmetry</bcType>
        </SYM1>

        <SYM2>
            <bcType>symmetry</bcType>
        </SYM2>
        <SYM3>
            <bcType>symmetry</bcType>
        </SYM3>
    </boundaryCondition>
```

5.5.3 Numerical setups

This example is a steady-state calculation. The temporal format is implicit LUSGS and the spatial format is upwind HLLC.

```
<spatialScheme>
    <spatialScheme>upwindSchemes</spatialScheme>
    <upwindSchemes>
        <upwindScheme>HLLC</upwindScheme>
    </upwindSchemes>
    <isLowMachCorrection>off</isLowMachCorrection>

    <reconstruction>linear</reconstruction>
    <limitersForLinear>Venk</limitersForLinear>
    <K>0.010</K>
    <gradient>cellGaussGrad</gradient>
    <cellNeighbourCell>FACENEIGHBOUR</cellNeighbourCell>
</spatialScheme>

    <timeMethod>
        <timeMarching>LUSGS</timeMarching>
        <LUSGS>
            <omegaForLUSGS>1.10</omegaForLUSGS>
            <betaT>0.2</betaT>
        </LUSGS>
        <sourceTermTreating>diagImpliciSource</sourceTermTreating>
    </timeMethod>
```

5.5.4 Running the simulations

Place the mesh file, the control file, and the OpenHurricane calculation program in the same folder. Right-click in the current folder to open a powershell terminal and enter the parallel computing code: mpiexec -np xx OpenHurricane.exe, where "xx" is the number of parallel cores, e.g., "mpiexec -np 10 OpenHurricane.exe".

5.5.5 Simulations results

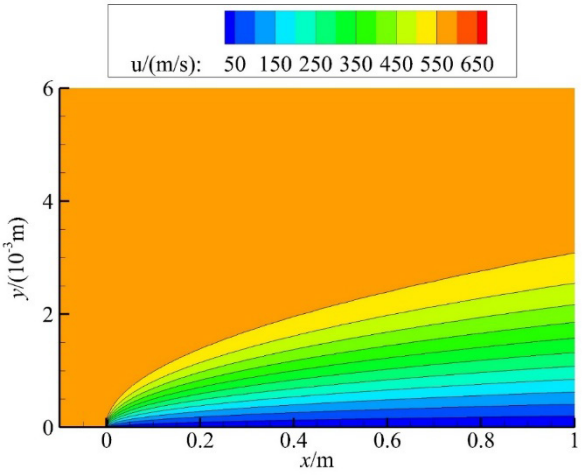

Fig. 22. Velocity distribution in the boundary layer

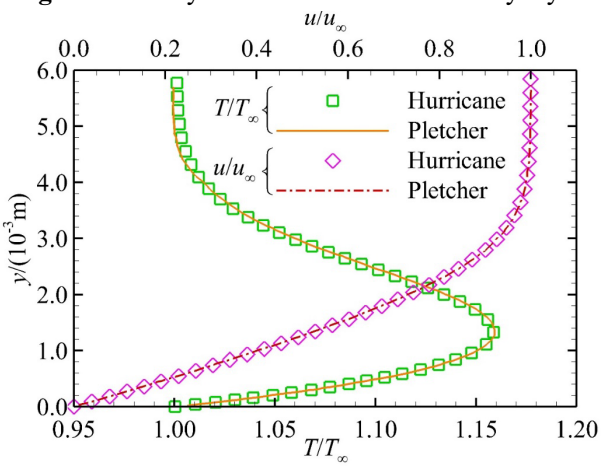

Fig. 23. Comparison of Hurricane calculations with Pletcher results

5.6 Oblique detonation

When the premixed gas of fuel and oxidizer passes through an oblique shock wave, the high temperature and high pressure generated after the wave forms a combustion wave, and the combustion wave and the shock wave couple with each other to form an oblique shock wave, which realizes high efficiency combustion within a very short time and space distance. It can be applied to hypersonic power systems flying at high Mach numbers.

5.6.1 Mesh information

A quasi-two-dimensional uniform mesh is used, and the boundary layer is encrypted near the wall surface.

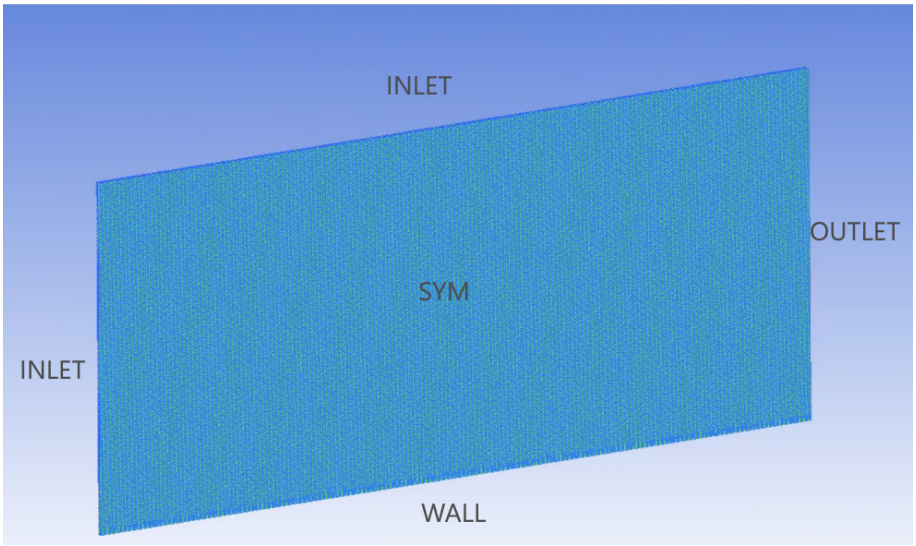

Fig. 24. Oblique detonation calculation mesh

5.6.2 Boundary and initial conditions

The inlet (INLET) uses a pressure far-field boundary condition to blow the wall at an angle. This calculation example is from reference[5] case 9. The incoming flow velocity is 42,500 Pa, the temperature is 851.5 K, the speed is 2,473.4 m/s, the direction is 19 ° with the wedge, and the chemical fraction is the mole fraction of the components at the time of combustion in the ratio of hydrogen to oxygen equivalent. The lower boundary (WALL) is non-slip, adiabatic wall. The outlet (OUTLET) boundary condition is a supersonic outlet.

```
<boundaryCondition>

    <INLET>
        <bcType>pressureFarField</bcType>
        <momentumGivenBy>pressureAndVMag</momentumGivenBy>
        <p>42500.0</p>
        <v>2473.4</v>
        <directionType>directionVector</directionType>
        <direction>car(0.9455186,-0.325568,0)</direction>
                                        <!-- wedge angle 19° -->
        <T>851.5</T>
        <species>
            <givenBy>moleFraction</givenBy>
            <H2>0.295858</H2>
            <O2>0.147929</O2>
            <N2>0.556213</N2>
        </species>
```

```
        </INLET>

        <OUTLET>
            <bcType>outflow</bcType>
            <p>42500.0</p>
        </OUTLET>

        <WALL>
            <bcType>wall</bcType>
            <momentum>
                <shearCondition>noSlip</shearCondition>
            </momentum>
            <thermal>
                <thermalCondition>adiabatic</thermalCondition>
            </thermal>
        </WALL>

        <SYM>
            <bcType>symmetry</bcType>
        </SYM>

    </boundaryCondition>
```

5.6.3 Numerical setups

This example is a steady-state calculation. The temporal format is implicit LUSGS and the spatial format is windward HLLC-HLL.

```
        <timeMethod>
            <timeMarching>LUSGS</timeMarching>
            <LUSGS>
                <omegaForLUSGS>1.10</omegaForLUSGS>
                <betaT>0.2</betaT>
            </LUSGS>
            <sourceTermTreating>diagImpliciSource</sourceTermTreating>
        </timeMethod>

                <omegaForLUSGS>1.10</omegaForLUSGS>
                <betaT>0.2</betaT>
            </LUSGS>
            <sourceTermTreating>diagImpliciSource</sourceTermTreating>
        </timeMethod>
```

```
<spatialScheme>
    <spatialScheme>upwindSchemes</spatialScheme>
    <upwindSchemes>
        <upwindScheme>HLLC_HLL</upwindScheme>
    </upwindSchemes>
    <isLowMachCorrection>off</isLowMachCorrection>
    <reconstruction>linear</reconstruction>
    <limitersForLinear>Venk</limitersForLinear>
    <K>0.010</K>
    <gradient>cellGaussGrad</gradient>
    <cellNeighbourCell>FACENEIGHBOUR</cellNeighbourCell>
</spatialScheme>
```

5.6.4 Running the simulations

Place the mesh file, the control file, the Chemical Reaction Mechanisms Folder, and the OpenHurricane calculation program in the same folder. Remember to set the name and path of the reaction mechanism in the control file.

Right-click in the current folder to open a powershell terminal and enter the parallel computing code: mpiexec -np xx OpenHurricane.exe, where "xx" is the number of parallel cores, e.g., "mpiexec -np 10 OpenHurricane.exe".

5.6.5 Simulations results

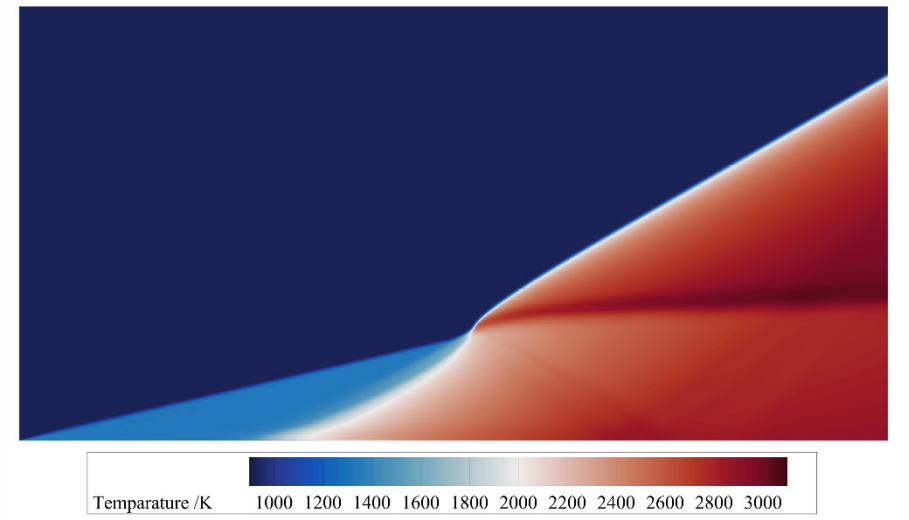

Fig. 25. Oblique detonation temperature distribution

6. Numerical methods and physical models

The OpenHurricane is designed to simulate flows especially for supersonic reacting flows. The governing equations for the basic solvers provided in the OpenHurricane are based on the classical Navier-Stokes equations. This chapter gives the details about the governing equations, numerical methods and common physical models implemented in the OpenHurricane.

6.1 Governing equations

In the OpenHurricane, the flow field is simulated by solving the Navier-Stokes equations which in integral form on the control volume V can be written as

$$\frac{\partial}{\partial t}\oint_V \boldsymbol{U}\mathrm{d}V+\oint_{\partial V}\left(\boldsymbol{F}_c-\boldsymbol{F}_v\right)\cdot\ \boldsymbol{n}\mathrm{d}S=\oint_V \boldsymbol{Q}\mathrm{d}V, \tag{6.1}$$

where t is time, S is the surface area of the control volume, and $\boldsymbol{U}$, $\boldsymbol{F}_c$, $\boldsymbol{F}_v$ and $\boldsymbol{Q}$ are the unknown vector of conservative variables, the convective flux and the diffusion flux as well as the source term, respectively. And $\boldsymbol{n}$ is the normal vector of the surface of the control volume. In Eq. (6.1) for a mixture of N_s species, the vector $\boldsymbol{U}$, $\boldsymbol{F}_c$, $\boldsymbol{F}_v$ and $\boldsymbol{Q}$ can be written as

$$\begin{gathered}\boldsymbol{U}=\begin{bmatrix}\rho\\ \rho\boldsymbol{v}\\ \rho E\\ \rho Y_1\\ \vdots\\ \rho Y_{N_s-1}\end{bmatrix},\quad \boldsymbol{F}_c=\begin{bmatrix}\rho\boldsymbol{v}\\ \rho\boldsymbol{v}\boldsymbol{v}+p\boldsymbol{I}\\ \left(\rho E+p\right)\boldsymbol{v}\\ \rho\boldsymbol{v}Y_1\\ \vdots\\ \rho\boldsymbol{v}Y_{N_s-1}\end{bmatrix},\\ \boldsymbol{F}_v=\begin{bmatrix}0\\ \boldsymbol{\tau}\\ \boldsymbol{\tau}\cdot\boldsymbol{v}+\kappa_{eff}\nabla T+\rho\sum_{i=1}^{N_s}h_i\boldsymbol{J}_i\\ \boldsymbol{J}_1\\ \vdots\\ \boldsymbol{J}_{N_s-1}\end{bmatrix},\quad \boldsymbol{Q}=\begin{bmatrix}0\\ 0\\ 0\\ W_1\bar{\dot{\omega}}_1\\ \vdots\\ W_{N_s-1}\bar{\dot{\omega}}_{N_s-1}\end{bmatrix},\end{gathered} \tag{6.2}$$

where ρ is the density of the gas mixture, $\boldsymbol{v}$ is the velocity vector with components v_x,

v_y, and v_z in cartesian directions x, y and z, E is the total energy per unit mass, T is the temperature, and Y_i is the mass fraction of species i (i = 1,…,N_s) and N_s is the number of species. The mass fraction of last species is determined through conservation of mass:

$$Y_{N_s} = 1 - \sum_{m=1}^{N_s - 1} Y_m. \tag{6.3}$$

6.2 Spatial schemes

In the OpenHurricane, the spatial schemes for convective flux include central scheme, upwind scheme and blending scheme.

6.2.1 Central schemes for convective flux

The central schemes available in the current version of OpenHurricane is the JST scheme:

(1) **JST**: JST scheme. (Because of the names of the authors: Jameson A, Schmidt W, Turkel E.)

6.2.2 Upwind schemes for convective flux

The upwind schemes available in the current version of OpenHurricane include:

(1) **AUSM**: The advection upstream splitting method (AUSM).

(2) **AUSMPlusUP**: The AUSM$^+$ up scheme.

(3) **AUSMPWPlus**: The AUSMPW$^+$ scheme.

(4) **vanLeer**: The van Leer flux-vector splitting scheme.

(5) **HLL**: The HLL-Riemann scheme. (Because of the names: Harten, Lax, van Leer).

(6) **HLLC**: The HLLC approximate Riemann solver. The scheme is based on the HLL Riemann solver.

(7) **LDFSS2**: The low-diffusion flux splitting scheme.

6.2.3 Blending schemes

The blending schemes available in the current version of OpenHurricane is:

(1) **blendingSchemeTVDLimiter2**: A mix of centered and upwind-biased Riemann flux scheme with a TVD limiter.

6.2.4 Reconstruction methods

In the OpenHurricane, the reconstruction methods for solution reconstruction include

(1) **firstOrder**: The left and right state are simply the flow variables computed for the left and the right control volume. This leads to a spatial discretization which is only first-order accurate.

(2) **linear**: Piecewise linear reconstruction method. It is formally second-order accurate on regular grids.

6.2.5 Gradient schemes

In the OpenHurricane, the methods for evaluating gradients include

(1) **cellGaussGrad**: Green-Gauss approach based on cell center value.

(2) **nodeGaussGrad**: Green-Gauss approach based on nodal value.

(3) **leastSquareGrad**: Least-squares approach. Weight type for the least-squares approach includes: WLSQ0, WLSQ1, WLSQ2, WLSQ3, WLSQG.

6.3 Temporal schemes

6.3.1 Temporal schemes for steady simulations

Temporal schemes for steady simulations of the current version of the OpenHurricane include:

(1) **LUSGS**: LU-SGS implicit scheme.

(2) **MRK**: Multi-stage Runge-Kutta explicit scheme.

(3) **semiImplicitMRK**: Semi-implicit multi-stage Runge-Kutta explicit scheme.

(4) **TVDRK**: TVD Runge-Kutta method. TVDRK2 for second-order time accurate version of the TVD Runge-Kutta method; TVDRK3 for third-order time accurate version of the TVD Runge-Kutta method.

(5) **TVDRK4**: Fourth-order time accurate version of TVD Runge-Kutta method.

6.3.2 Temporal schemes for unsteady simulations

Temporal schemes for unsteady simulations of the current version of the OpenHurricane include explicit schemes and implicit methods associated with the dual

time-stepping approach. In the current version of OpenHurricane, the explicit schemes for unsteady flows include:

(1) **unsteadyMRK**: Multi-stage Runge-Kutta explicit scheme for unsteady flows.

(2) **unsteadyTVDRK**: TVD Runge-Kutta method for unsteady flows. TVDRK2 for second-order time accurate version of the TVD Runge-Kutta method; TVDRK3 for third-order time accurate version of the TVD Runge-Kutta method.

(3) **unsteadyTVDRK4**: Fourth-order time accurate version of TVD Runge-Kutta method for unsteady flows.

The implicit methods associated with the dual time-stepping approach include:

(1) **BDF123LUSGS**: Backward-difference formular for unsteady flows. BDF1 for first-order, BDF2 for second-order and BDF3 for third-order time accurate. The LU-SGS implicit time-marching scheme is employed for the solution of the inner iteration in the pseudo time.

(2) ESDIRKLUSGS: First-stage Explicit, Single Diagonal coefficient, Implicit Runge-Kutta method for unsteady flows. ESDIRK3 for four stages third order ESDIRK method. ESDIRK4 for six stages fourth order ESDIRK method. The LU-SGS implicit time-marching scheme is employed for the solution of the inner iteration in the pseudo time.

6.4 Gas physical properties

6.4.1 Equation of state

The equation of state available in the current version of OpenHurricane only includes the perfect gas model:

$$p = \rho RT, \tag{6.4}$$

where R is the gas constant. The mixture pressure p is also determined through the equation of state of ideal gas as expressed follows:

$$p = \rho R_u T \sum_{i=1}^{N_s} \frac{Y_i}{W_i}, \tag{6.5}$$

where R_u is the universal gas constant.

6.4.2 Thermo properties

Thermo models are concerned with evaluating the specific heat, enthalpy h and other derived thermo properties. The current thermo models include:

(1) **constCp**: which assumes that the specific heat capacity at constant pressure c_p is a constant. The chemical enthalpy must be given explicitly.

(2) **constCv**: which assumes that the specific heat capacity at constant volume c_v is a constant. The chemical enthalpy must be given explicitly.

(3) **piecewiseLinear**: which evaluates the specific heat capacity at constant pressure c_p use a piecewise linear expression by interpolating from a given table of several pairs (T, c_p). The chemical enthalpy must be given explicitly.

(4) **JANAF**: which evaluates the specific heat capacity at constant pressure c_p, enthalpy h and entropy s using seven-coefficients polynomial expressions:

$$\begin{aligned}
\frac{c_p}{R} &= a_1 + a_2 T + a_3 T^2 + a_4 T^3 + a_5 T^4, \\
\frac{h}{RT} &= a_1 + \frac{a_2}{2} T + \frac{a_3}{3} T^2 + \frac{a_4}{4} T^3 + \frac{a_5}{5} T^4 + \frac{a_6}{T}, \\
\frac{s}{R} &= a_1 \ln T + a_2 T + \frac{a_3}{2} T^2 + \frac{a_4}{3} T^3 + \frac{a_5}{4} T^4 + a_7.
\end{aligned} \tag{6.6}$$

6.4.3 Transport properties

Transport models are concerned with evaluating the molecular viscosity μ, the thermal conductivity κ. The current transport models include:

(1) **sutherlandThree**: which calculates the viscosity using the Sutherland's law with three parameters. Sutherland's law can be expressed as:

$$\mu = \mu_{ref} \left(\frac{T}{T_{ref}} \right)^{3/2} \frac{T_{ref} + S}{T + S}, \tag{6.7}$$

where T_{ref} is a reference temperature, μ_{ref} is the viscosity at the T_{ref} reference temperature and S is the Sutherland constant. The thermal conductivity κ is calculated by given the Prandtl number:

$$\kappa = \frac{\mu c_p}{Pr}. \tag{6.8}$$

(2) **sutherlandTwo**: which calculates the viscosity using the Sutherland's law with three parameters. The thermal conductivity κ is also calculated by given the Prandtl number. Sutherland's law can also be written as:

$$\mu = \frac{C_1 T^{3/2}}{T + S}, \tag{6.9}$$

where S is the Sutherland constant and the constant C_1 is given by:

$$C_1 = \frac{\mu_{ref}}{T_{ref}^{3/2}}\left(T_{ref} + S\right). \tag{6.10}$$

(3) **constMuKappa**: assumes a constant molecular viscosity μ and a constant thermal conductivity κ.

(4) **variableHardSphereModel**: which calculates the molecular viscosity μ using the variable hard sphere model:

$$\mu = \mu_{ref}\left(\frac{T}{T_{ref}}\right)^{\omega}, \tag{6.11}$$

where T_{ref} is a reference temperature, ω is the temperature exponent and μ_{ref} is the viscosity at the T_{ref} reference temperature:

$$\mu_{ref} = \frac{15\sqrt{\pi m k_B T_{ref}}}{2\pi d_{ref}^2 \left(5 - 2\omega\right)\left(7 - 2\omega\right)}, \tag{6.12}$$

The thermal conductivity κ is also calculated by given the Prandtl number.

(5) **kinetic**: which evaluates the molecular viscosity μ_i and the thermal conductivity κ_i of species i by the standard kinetic theory expression:

$$\begin{aligned} \mu_i &= 2.6693\times10^{-6}\frac{\sqrt{W_i T}}{\sigma_i^2 \Omega_{\mu_i}}, \\ \kappa_i &= \frac{\mu_i R_u}{W_i}\left(c_{pi}\frac{W_i}{R_u} + \frac{5}{4}\right). \end{aligned} \tag{6.13}$$

where σ_i is the Lennard-Jones collision diameter and Ω_{μ_i} is the collision integral.

6.5 Turbulence models

The turbulence models available in the current version of OpenHurricane include:

(1) **SpalartAllmaras**: Spalart-Allmaras one equation RANS model.

(2) **SST**: the shear-stress transport (SST) k-ω RANS model.

6.6 Flow models

The flow models available in the current version of OpenHurricane include:

(1) **EulerFlow**: flow for Euler equations.

(2) **laminarFlow**: flow for laminar NS equations.

(3) **eddyViscosity**: flow for eddy-viscosity hypothesis.

7. User feedback

If you have any problems in building or running the code or any suggestions, please do not hesitate to contact. Email: xuxu_1521@126.com

Reference

[1] Gordon S. and McBride B. J., Computer Program for Calculation of Complex Chemical Equilibrium Compositions, Rocket Performance, Incident and Reflected Shocks and Chapman-Jouguet Detonations, NASA Report SP-273, 1971.

[2] Burrows, M. C. and Kurkov, A. P., "Analytical and Experimental Study of Supersonic Combustion of Hydrogen in a Vitiated Airstream," NASA-TM-X-2828, Sep. 1973.

[3] H.B. Hopkins, W. Konopka, J. Leng, “Validation of scramjet exhaust simulation technique at Mach 6”, NASA Contractor Report 3003, 1979.

[4] Lawrence S.L., Application of an Upwind Algorithm to the Parabolized Navier-Stokes Equations[D]. Iowa State University, 1987.

[5] Teng H, Tian C, Zhang Y, Zhou L, Ng HD. Morphology of oblique detonation waves in a stoichiometric hydrogen–air mixture. Journal of Fluid Mechanics. 2021; 913: A1. doi:10.1017/jfm.2020.1131