My Project

Generated by Doxygen 1.8.11

Contents

Chapter 1

My Personal Index Page

1.1 Introduction

This is the C++ code to solve the high speed fluid flow. Currently, Euler flow is being solved but this code has been designed in moulder way so to solve the viscus flow additional viscus flux class can be added very easily. This code has been written to fulfill the requirement of the Dual Degree Project(DDP).

1.2 Installation & Use

To use the solver. Follow these simple steps.

- Download form here : https://github.com/singh-kuldeep/DDP2 or click here
- Go to the folder DDP2 and compile and run the file TVD.cpp (ex. g++ TVD.cpp && ./a.out)
- Nozzle has been set up as a default geometry but it can be changed from "run.h" file by uncommenting the header file
- Currently there are two different geometry options are available
    1. Curved wall high area ratio diverging nozzle
    2. Triangular bump inside straight duct

1.3 Brief about the solver

- 3D Cartesian (x,y,z)
- Roe scheme based
- C++
- Exact theory can be found here

1.4 Input to the solver

- Grid points
- Boundary condition
- Some initial condition

1.5 Output files.

Here are the list of files which will come as the output of the solver.

- Residual_Nozzle.csv : This file contains the all the residuals (Mass, Momentum, Energy).
- grids_Nozzle_2D.csv : This file contains the grid point (x,y) coordinates.
- 2D_parameters_B.csv : This file contains all the conserved parameters at the 2D plane.

1.6 Results & Plots

Same older contains the MATLAB script "plot_data.m". Once the simulation has started and the output files are generated, one can simply run the MATALB script and can see the plots which are listed below.

- Density Residual
- X Momentum Residual
- Y Momentum Residual
- Z Momentum Residual
- Energy Residual
- Mach Number
- Density
- Velocity
- Temperature
- Pressure
- Geometry 2D cross section /

Chapter 2

Bug List

File diffusionfluxinterface.h

Needs to explain the code little bit more.

Member diffusionfluxinterface::diffusionfluxinterface (vector< double > &ConservedVariableLeftMinus, vector< double > &ConservedVariableLeft, vector< double > &ConservedVariableRight, vector< double > &ConservedVariableRightPlus, vector< double > &FaceAreaVectorLeft, vector< double > &FaceAreaVectorRight, vector< double > &FaceAreaVectorRightPlus, double CellVolumeLeftMins, double CellVolumeLeft, double CellVolumeRight, double CellVolumeRightPlus, double DeltaT)

Here syntax needs to be changed for gvactor[i] calculation

Here I have doubt about "not equal to sign" because it can't be exactly equal to 0.00000 so most of the time we end up choosing theta i = 0.0

File eulerflux.h

Not all memory is freed when deleting an object of this class.

Not all memory is freed when deleting an object of this class.

Not all memory is freed when deleting an object of this class.

File local_time_step.h

Currently not using this, because grid() is not calculating ds value. So recheck this function as well after fixing the grid() function.

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 diffusionfluxAUSM Class Reference

Public Member Functions

- **diffusionfluxAUSM** (vector< double > &ConservedVariable, vector< double > &AreaVector)

Public Attributes

- double **Flux** [5]

The documentation for this class was generated from the following file:

- diffusionfluxAUSM.h

5.2 diffusionfluxinterface Class Reference

Public Member Functions

- diffusionfluxinterface (vector< double > &ConservedVariableLeftMinus, vector< double > &Conserved↩ VariableLeft, vector< double > &ConservedVariableRight, vector< double > &ConservedVariableRight↩ Plus, vector< double > &FaceAreaVectorLeft, vector< double > &FaceAreaVectorRight, vector< double > &FaceAreaVectorRightPlus, double CellVolumeLeftMins, double CellVolumeLeft, double CellVolumeRight, double CellVolumeRightPlus, double DeltaT)

Public Attributes

- double **DiffusionFluxVector** [5]

5.2.1 Constructor & Destructor Documentation

5.2.1.1 diffusionfluxinterface::diffusionfluxinterface (vector< double > & *ConservedVariableLeftMinus,* vector< double > & *ConservedVariableLeft,* vector< double > & *ConservedVariableRight,* vector< double > & *ConservedVariableRightPlus,* vector< double > & *FaceAreaVectorLeft,* vector< double > & *FaceAreaVectorRight,* vector< double > & *FaceAreaVectorRightPlus,* double *CellVolumeLeftMins,* double *CellVolumeLeft,* double *CellVolumeRight,* double *CellVolumeRightPlus,* double *DeltaT*) [inline]

Bug Here syntax needs to be changed for gvactor[i] calculation

Bug Here I have doubt about "not equal to sign" because it can't be exactly equal to 0.00000 so most of the time we end up choosing theta i = 0.0

The documentation for this class was generated from the following file:

- diffusionfluxinterface.h

5.3 eulerflux Class Reference

Public Member Functions

- **eulerflux** (vector< double > ConservedVariable)

Public Attributes

- double **EulerFluxX** [5]
- double **EulerFluxY** [5]
- double **EulerFluxZ** [5]

The documentation for this class was generated from the following file:

- eulerflux.h

5.4 eulerfluxAUSM Class Reference

Public Member Functions

- **eulerfluxAUSM** (vector< double > ConservedVariable, vector< double > AreaVector, string gamma, double SpecificHeatRatio)

Public Attributes

- double **Flux** [5]
- double **MachPlus**
- double **MachMinus**
- double **PressurePlus**
- double **PressureMinus**
- double **Mach**

The documentation for this class was generated from the following file:

- eulerfluxAUSM.h

5.5 interface Class Reference

Public Member Functions

- **interface** (vector< double > &ConservedVariableLeft, vector< double > &ConservedVariableRight, vector< double > &FaceAreaVectorInterface, double CellVolumeLeft, double CellVolumeRight, double DeltaT)

Public Attributes

- double **DensityInterface**
- double **VelocityXInterface**
- double **VelocityYInterface**
- double **VelocityZInterface**
- double **EnthalpyInterface**
- double **VectorJumpInterface** [5]
- double **EigenValue** [5]
- double **EigenVectorMatrix** [5][5]
- double **EigenVectorMatrixInverse** [5][5]
- double **AlphaVectorInterface** [5]
- double **MuVectorInterface** [5]
- double **ZVectorInterface** [5]
- double **PshiVectorInterface** [5]
- double **GVectorInterface** [5]

The documentation for this class was generated from the following file:

- interface.h

5.6 netfluxAUSM Class Reference

Public Member Functions

- **netfluxAUSM** (vector< double > LeftConservedVariable, vector< double > RightConservedVariable, vector< double > AreaVector, string gamma, double SpecificHeatRatio)

Public Attributes

- double **NetFlux** [5]

The documentation for this class was generated from the following file:

- netfluxAUSM.h

5.7 netfluxBase Struct Reference

Public Attributes

- double **NetFlux** [5]

The documentation for this struct was generated from the following file:

- netfluxBase.h

5.8 netfluxRoe Class Reference

Public Member Functions

- netfluxRoe (vector< double > &ConservedVariableLeftMinus, vector< double > &ConservedVariableLeft, vector< double > &ConservedVariableRight, vector< double > &ConservedVariableRightPlus, vector< double > &FaceAreaLeft, vector< double > &FaceAreaVectorRight, vector< double > &FaceArea↩ VectorRightplus, double CellVolumeLeftMins, double CellVolumeLeft, double CellVolumeRight, double Cell↩ VolumeRightPlus, double DeltaT)

Public Attributes

- double **NetFlux** [5]

5.8.1 Constructor & Destructor Documentation

5.8.1.1 netfluxRoe::netfluxRoe (vector< double > & *ConservedVariableLeftMinus,* vector< double > & *ConservedVariableLeft,* vector< double > & *ConservedVariableRight,* vector< double > & *ConservedVariableRightPlus,* vector< double > & *FaceAreaLeft,* vector< double > & *FaceAreaVectorRight,* vector< double > & *FaceAreaVectorRightplus,* double *CellVolumeLeftMins,* double *CellVolumeLeft,* double *CellVolumeRight,* double *CellVolumeRightPlus,* double *DeltaT*) [inline]

Parameters

CellVolumeInterface	Average of left and right cell volume

See also

diffusionfluxinterface()
eulerflux()

The documentation for this class was generated from the following file:

- netfluxRoe.h

Chapter 6

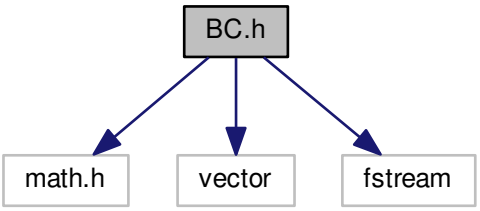
File Documentation

6.1 BC.h File Reference

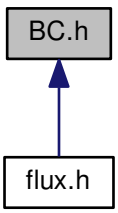
This header file implements all three boundary conditions.

```
#include "math.h"
#include <vector>
#include <fstream>
```

Include dependency graph for BC.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define **RealGasConstant** 287.17

Functions

- void getNormal (vector< double > &UnitNormal, vector< double > areaVector)

  Changes the input vector into the unit normal vector.
- double **getMachfromPressureRatio** (double Pressure, double TotalPressure, double SpecificHeatRatio)
- void WallBC (vector< double > &GhostCellConservedVariables, vector< double > LiveCellConserved↩ Variables, vector< double > AreaVectors, double SpecificHeatRatio)

  This function implements the wall boundary condition.
- void **SubSonicInletBC** (vector< double > &GhostCellConservedVariables, vector< double > LiveCell↩ ConservedVariables, double InletTotalPressure, double InletTotalTemperature, double SpecificHeatRatio)
- void **SubSonicExitBC** (vector< double > &GhostCellConservedVariables, vector< double > LiveCell↩ ConservedVariables, double ExitPressure, double SpecificHeatRatio)
- void **SuperSonicExitBC** (vector< double > &GhostCellConservedVariables, vector< double > LiveCell↩ ConservedVariables)
- void **SuperSonicInletBC** (vector< double > &GhostCellConservedVariables, double InletTotalPressure, double InletTotalTemperature, double InletMach, double SpecificHeatRatio)
- void BC (vector< vector< vector< vector< double > > > > ConservedVariables, vector< vector< vector< vector< double > > > > iFaceAreaVector, vector< vector< vector< vector< double > > > > jFace↩ AreaVector, vector< vector< vector< vector< double > > > > kFaceAreaVector, vector< vector< vector< vector< double > > > > &i0GhostConservedVariable, vector< vector< vector< vector< double > > > > &j0GhostConservedVariable, vector< vector< vector< vector< double > > > > &k0GhostConserved↩ Variable, vector< vector< vector< vector< double > > > > &iNiGhostConservedVariable, vector< vector< vector< vector< double > > > > &jNjGhostConservedVariable, vector< vector< vector< vector< double > > > > &kNkGhostConservedVariable, int Ni, int Nj, int Nk, double SpecificHeatRatio)

  Function BC() implements the boundary condition. Here two ghost cell are used to implement the boundary condition. In simple words this function calculates the conserved variables for all ghost cells. For inlet it uses the stagnation parameters, for exit it simply uses the live cell parameters and copies them into the ghost cells, and for wall boundary it uses the fact that flow should be parallel to the wall.

6.1.1 Detailed Description

This header file implements all three boundary conditions.

- Inlet
- Exit and
- Wall boundary

Author

Kuldeep Singh

Date

2017

6.1.2 Function Documentation

6.1.2.1 void BC (vector< vector< vector< vector< double > > > > *ConservedVariables*, vector< vector< vector< vector< double > > > > *iFaceAreaVector*, vector< vector< vector< vector< double > > > > *jFaceAreaVector*, vector< vector< vector< vector< double > > > > *kFaceAreaVector*, vector< vector< vector< vector< double > > > > & *i0GhostConservedVariable*, vector< vector< vector< vector< double > > > > & *j0GhostConservedVariable*, vector< vector< vector< vector< double > > > > & *k0GhostConservedVariable*, vector< vector< vector< vector< double > > > > & *iNiGhostConservedVariable*, vector< vector< vector< vector< double > > > > & *jNjGhostConservedVariable*, vector< vector< vector< vector< double > > > > & *kNkGhostConservedVariable*, int *Ni*, int *Nj*, int *Nk*, double *SpecificHeatRatio*)

Function BC() implements the boundary condition. Here two ghost cell are used to implement the boundary condition. In simple words this function calculates the conserved variables for all ghost cells. For inlet it uses the stagnation parameters, for exit it simply uses the live cell parameters and copies them into the ghost cells, and for wall boundary it uses the fact that flow should be parallel to the wall.

Parameters

in	*ConservedVariables*	This is the pointer to the 4D vector where all the conserved variables of previous time step are stored.
in	*&iFaceAreaVector*	This is a pointer to the 4D vector which has the area vector of all faces which are in "i" direction.
in	*&jFaceAreaVector*	This is a pointer to the 4D vector which has the area vector of all faces which are in "j" direction.
in	*&kFaceAreaVector*	This is a pointer to the 4D vector which has the area vector of all faces which are in "k" direction.
in	*Ni*	Number of cells in in "i" direction.
in	*Nj*	Number of cells in in "j" direction.
in	*Nk*	Number of cells in in "k" direction.

Returns

void

6.1.2.2 void getNormal (vector< double > & *UnitNormal*, vector< double > *areaVector*)

Changes the input vector into the unit normal vector.

Parameters

areaVector	A 3D vector.
vectorMagnitude	Magnitude of the 3D vector.

Returns

void

6.1.2.3 void WallBC (vector< double > & *GhostCellConservedVariables*, vector< double > *LiveCellConservedVariables*, vector< double > *AreaVectors*, double *SpecificHeatRatio*)

This function implements the wall boundary condition.

Parameters

AreaVectors	Surface faces area vectors.
LiveCellConservedVariables	Conserved variables array for the live cell.
GhostCellConservedVariables	Conserved variables array for the ghost cell.

Returns

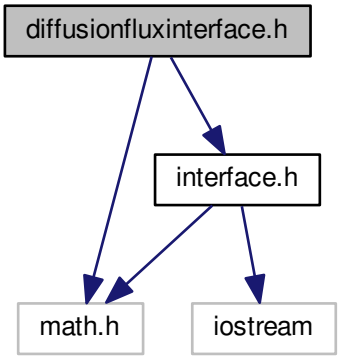
void

6.2 diffusionfluxinterface.h File Reference

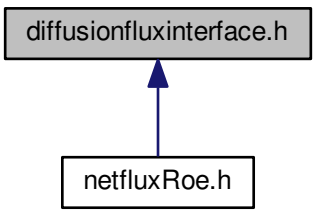
This class calculates the numerical diffusion flux.

```
#include "math.h"
#include "interface.h"
```

Include dependency graph for diffusionfluxinterface.h:



This graph shows which files directly or indirectly include this file:



Classes

- class diffusionfluxinterface

6.2.1 Detailed Description

This class calculates the numerical diffusion flux.

Author

Kuldeep Singh

Date

2017

Copyright

GNU Public License.

Parameters

	DiffusionFluxVector	Numerical diffusion flux vector at the interface
in	*ConservedVariable*	Conserved variable vector ([Density , x-momentum, y-momentum, z-momentum, Energy])
in	*CellVulume*	Pointer to the cell volume vector
in	*LeftMinus*	Cell just previous to the left
in	*RightPlus*	Cell just Next to the right
in	*DeltaT*	Time step

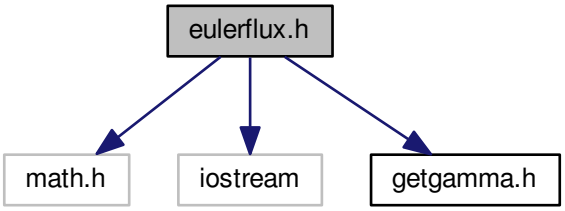
Bug Needs to explain the code little bit more.

6.3 eulerflux.h File Reference

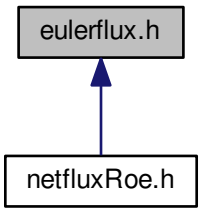
This class calculates the euler flux vectors(Ee,Fe,Ge) at the interface.

```
#include "math.h"
#include "iostream"
#include "getgamma.h"
```

Include dependency graph for eulerflux.h:



This graph shows which files directly or indirectly include this file:



Classes

- class eulerflux

6.3.1 Detailed Description

This class calculates the euler flux vectors(Ee,Fe,Ge) at the interface.

Author

Kuldeep Singh

Date

2017

Bug Not all memory is freed when deleting an object of this class.

Copyright

GNU Public License.

Parameters

	EulerFlux	Euler flux vector at interface
in	*AreaVector*	Interface area vector
in	*ConservedVariable*	Conserved variable vector ([Density , x-momentum, y-momentum, z-momentum, Energy])
	Pressure	Satic pressure (p)

Author

Kuldeep Singh

Date

2017

Bug Not all memory is freed when deleting an object of this class.

Copyright

GNU Public License.

Parameters

	EulerFluxX	x direction euler flux vector (Ee) at interface
	EulerFluxY	y direction euler flux vector (Fe) at interface
	EulerFluxZ	z direction euler flux vector (Ge) at interface
in	*ConservedVariable*	Conserved variable vector ([Density , x-momentum, y-momentum, z-momentum, Energy])
	Pressure	Satic pressure (p)

Author

Kuldeep Singh

Date

2017

Bug Not all memory is freed when deleting an object of this class.

Copyright

GNU Public License.

Parameters

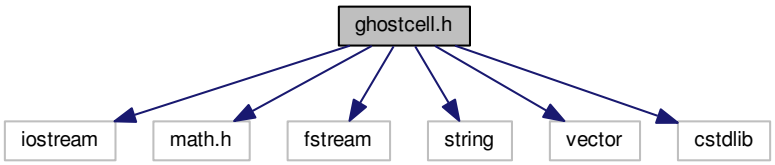
	Flux	Euler flux vector at interface
in	*AreaVectorNormal*	Interface area vector
in	*ConservedVariable*	Conserved variable vector ([Density , x-momentum, y-momentum, z-momentum, Energy])
	Pressure	Satic pressure (p)

6.4 ghostcell.h File Reference

This header file functions find the ghost cells area vectors and ghost cell volumes.

```
#include <iostream>
#include "math.h"
#include <fstream>
#include <string>
#include <vector>
#include <cstdlib>
```

Include dependency graph for ghostcell.h:



Functions

- void **ghostcell** (vector< vector< vector< vector< double > > > > Coordinates, vector< vector< vector< vector< double > > > > iFaceAreaVector, vector< vector< vector< vector< double > > > > jFaceAreaVector, vector< vector< vector< vector< double > > > > kFaceAreaVector, vector< vector< vector< double > > > CellVolume, vector< vector< vector< double > > > &i0GhostCellVolume, vector< vector< vector< double > > > &j0GhostCellVolume, vector< vector< vector< double > > > &k0GhostCellVolume, vector< vector< vector< double > > > &iNiGhostCellVolume, vector< vector< vector< double > > > &jNjGhostCellVolume, vector< vector< vector< double > > > &kNkGhostCellVolume, int Ni, int Nj, int Nk)

6.4.1 Detailed Description

This header file functions find the ghost cells area vectors and ghost cell volumes.

Author

Kuldeep Singh

Date

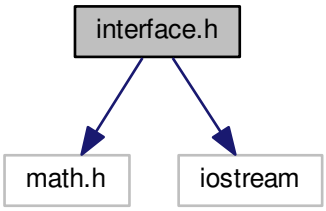
2017

6.5 interface.h File Reference

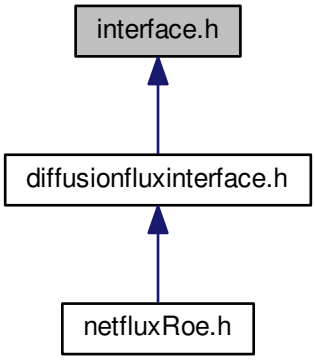
This class calculates the interface parameters using Reo scheme flux.

```
#include "math.h"
#include "iostream"
```

Include dependency graph for interface.h:



This graph shows which files directly or indirectly include this file:



Classes

- class interface

Macros

- #define SpecificHeatRatio 1.4

6.5.1 Detailed Description

This class calculates the interface parameters using Reo scheme flux.

Author

Kuldeep Singh

Date

2017

Copyright

GNU Public License.

Parameters

	DensityInterface	Roe density at interface
	VelocityXInterface	x velocity at interface
	VelocityYInterface	y velocity at interface
	VelocityZInterface	z velocity at interface
	EnthalpyInterface	Enthalpy at interface
	EnthalpyInterface	Enthalpy at interface
	VectorJumpInterface	Change in the conserved parameters at the interface
	EigenValue	Eigenvalue of the Jacobian matrix
	EigenVectorMatrix	Eigenvector of the Jacobian matrix
	EigenVectorMatrixInverse	Inverse of the Jacobian matrix
	AlphaVectorInterface[5]	EigenVectorMatrixInverse[5][5]∗VectorJumpInterface
	MuVectorInterface	= delta t ∗ EigenValue
	ZVectorInterface	This is same as MuVectorInterface
in	*ConservedVariables*	This is the pointer to the 4D vector where all the conserved variables of previous time step are stored.
in	*FaceAreaVectorInterface*	This is the pointer to the area vector the cell interface
	CellVolume	3D vector which has the cell volume of all cells inside the domain

6.5.2 Macro Definition Documentation

6.5.2.1 #define SpecificHeatRatio 1.4

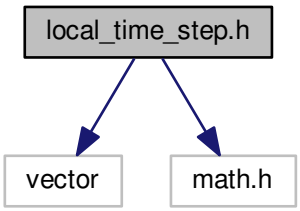
This is gas constant (Gamma). For air at room temperature it is almost equal to 1.4. If you are using some other gas at some other temperature then change it

6.6 local_time_step.h File Reference

This header file conditions the function TimeStep() which calculate the local time step for each cell at every iteration.

```
#include <vector>
#include <math.h>
```

Include dependency graph for local_time_step.h:



Functions

- double **TimeStep** (int i, int j, int k, vector< vector< vector< double > > > delta_s, vector< vector< vector< vector< double > > > > ConservedVariables)

6.6.1 Detailed Description

This header file conditions the function TimeStep() which calculate the local time step for each cell at every iteration.

Author

Kuldeep Singh

Date

2016

See also

grid()

Bug Currently not using this, because grid() is not calculating ds value. So recheck this function as well after fixing the grid() function.

Parameters

in	*i,j,k*	Cell location for which TimeStep is to be calculated
in	*delta_s*	ds value of the cell for which TimeStep is to be calculated
	[IN]	ConservedVariables Conserved variables vector
	CFL	Courant–Friedrichs–Lewy number
	Pressure	Static Pressure
	VelocityMagnitude	Magnitude of the velocity
	VelocitySound	Speed of sound
out	*TimeStep*	Time step (dt)

Returns

double

Index