2DNS Documentation

Release 0.0.1

Sayop Kim

September 25, 2014

Contents

Contents:

CHAPTER 1

Project description

1.1 Given task

In this project, 2-D explicit Navier-Stokes solver (Hereafter called 2DNS code) has been developed. The 2DNS code was verified in this project by applying supersonic flow over a 10 deg. diamond airfoil. The test cases were employed in terms of inviscid and viscous conditions.

1.2 Governing Equations

Here, the 2-D, unsteady Navier-Stokes equations will be solved. The equations will be marched forward in time until a steady state solution is achieved. The transformed conservative form of 2-D Navier-Stokes equations can be written:

$$\frac{\partial \left(\bar{U}/J\right)}{\partial t} + \frac{\partial \vec{F}'}{\partial \xi} + \frac{\partial \vec{G}'}{\partial \eta} = 0$$

where the transformed state and inviscid flux vectors are

$$\vec{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ E_t \end{bmatrix}$$

and transformed flux vector can be represented by:

$$\vec{F}' = \frac{1}{J}\left[\xi_x(\vec{F}_I - \vec{F}_V) + \xi_y(\vec{G}_I - \vec{G}_V)\right], \quad \vec{G}' = \frac{1}{J}\left[\eta_x(\vec{F}_I - \vec{F}_V) + \eta_y(\vec{G}_I - \vec{G}_V)\right]$$

Here $\vec{F}_I$ and $\vec{F}_V$ indicate the inviscid and viscous flux terms in x-direction, respectively. The same notation is employed to flux vector $\vec{G}$. The total energy per unit volume and stagnation enthalpy per unit mass are defined by followings:

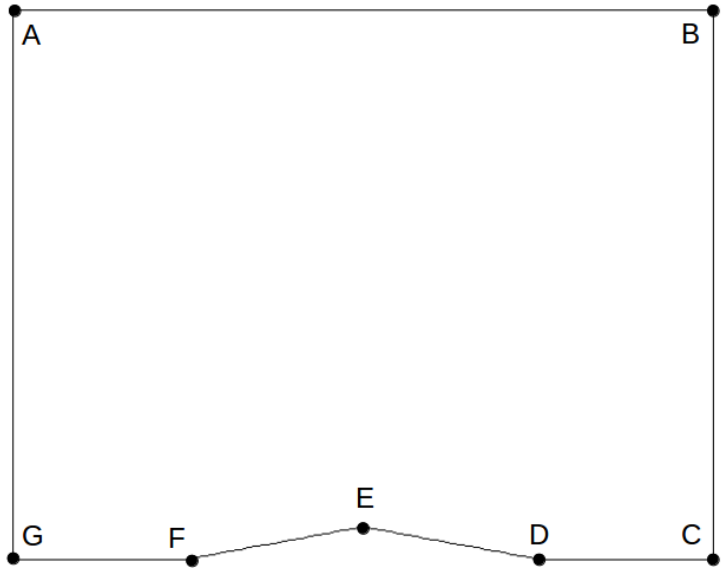
$$E_t = \frac{p}{\gamma - 1} + \frac{\rho}{2}\left(u^2 + v^2\right)$$

$$h_0 = \frac{p\gamma}{\rho(\gamma - 1)} + \frac{1}{2}\left(u^2 + v^2\right)$$

Detailed process for solving the Navier-Stokes equations will be repeated in the following sections.

1.3 Computational Domain

This project analyze the top half of a 10 deg. diamond airfoil so the location of point E is (x,y) = (0.5, 0.0882). Each grid point can be describd by (x,y) location or (i, j) location where the i index is in the ξ direction and the j index is in the η direction. The grid will consist of 65 points in the "i" direction and 57 points in the "j" direction. The inverse grid metrics must be evaluated at every grid point in the computational domain (including the boundaries). Use 2nd order accurate, central differences for interior points and 2nd order accurate, one-sided differences for boundary points. After the inverse metrics are computed, the grid Jacobian and grid metrics must be computed and stored at every location (including the boundaries)



CHAPTER 2

Code development

The present project is aimed to develop a computer program for solving 2-D unsteady Navier-Stokes equations for a supersonic problem. Hereafter, the program developed here in this project is called '2DNS'.

2.1 2DNS Code summary

The source code contains two directories, 'io', and 'main', for input/output related sources and main solver routines, respectively. 'CMakeLists.txt' file is also included for cmake compiling.

```
$ cd 2DNS/CODEdev/src/
$ ls
$ CMakeLists.txt  io  main
```

The **io** folder has **io.F90** and **ReadGrid.F90** files which contains subroutines for reading input/output data and grid info. It also includes **input** directory which contains a default **input.dat** file.

The **main** folder is only used for calculating essential subroutines required to solve the '2DNS' equation by using 'AUSMPW+' scheme for solving inviscid flux reconstruction and independent viscous flux calculator. The main routine is run by **main.F90** which calls important subroutines from **main** folder itself and **io** folder when needed.

CHAPTER 3

Numerical Method

During each time-integration step, the code calculate the fluxes $\vec{F'}_{i+1/2,j}$ at every "i" half-point locations, and $\vec{G'}_{i,j+1/2}$ at every "j" half-point locations. In order to obtain the flux terms properly treated with consideration of characteristics of wave propagation, MUSCL differencing should first be used to extrapolate the state vectors to every half point locations. After then AUSMPW+ scheme applies to those points for solving the inviscid flux terms. In addition, to evaluate the viscous flux terms, shear stress and heat flux terms should then be calculated at every half-points.

3.1 Flux vector evaluation

Here, the description and formulation of AUSMPW+ scheme are not repeated. The viscous flux vectors in the generalized coordinates can then be evaluated by solving the following forms

$$\vec{F}'_V = \frac{1}{J}\left(\xi_x \vec{F}_V + \xi_y \vec{G}_V\right)$$
$$\vec{G}'_V = \frac{1}{J}\left(\eta_x \vec{F}_V + \eta_y \vec{G}_V\right)$$

The above froms can be rearranged to the following form composed of shear stress and heat flux terms:

$$\vec{F}'_V \text{ or } \vec{G}'_V = \frac{1}{J}\begin{bmatrix} 0 \\ m_x\tau_{xx} + m_y\tau_{xy} \\ m_x\tau_{xy} + m_y\tau_{yy} \\ m_x\left(u\tau_{xx} + v\tau_{xy} - q_x\right) + m_y\left(u\tau_{xy} + v\tau_{yy} - q_y\right) \end{bmatrix}$$

where,

$$m_x = \xi_x \quad \text{and} \quad m_y = \xi_y \quad \text{for} \quad \vec{F}'_V$$
$$m_x = \eta_x \quad \text{and} \quad m_y = \eta_y \quad \text{for} \quad \vec{G}'_V$$

The nondimensional form of shear stress and heat flux terms are given by:

$$\tau_{xx} = \frac{2\mu}{3\mathrm{Re}_L}\left(2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right)$$
$$\tau_{yy} = \frac{2\mu}{3\mathrm{Re}_L}\left(2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x}\right)$$
$$\tau_{xy} = \frac{\mu}{\mathrm{Re}_L}\left(2\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)$$
$$q_x = -\frac{\mu}{(\gamma-1)M_\infty^2 \mathrm{Re}_L \mathrm{Pr}}\frac{\partial T}{\partial x}$$
$$q_y = -\frac{\mu}{(\gamma-1)M_\infty^2 \mathrm{Re}_L \mathrm{Pr}}\frac{\partial T}{\partial y}$$

where M_∞ is the free stream Mach number,

$$M_\infty = \frac{V_\infty}{\sqrt{\gamma R T_\infty}}$$

and the perfect gas equations of state become

$$p = (\gamma - 1)\rho e$$
$$T = \frac{\rho M_\infty^2 p}{\rho}$$

Note that the above equations should be evaluated by non-dimensional forms.

The coefficients of viscosity and thermal conductivity can be related to the thermodynamic variables according to the gas kinetic theory. Based on this theory, Sutherland's formulas can be applied to evaluate the mass and thermal diffusivity by solving:

$$\mu = C_1 \frac{T^{3/2}}{T + C_2} \quad , \quad k = C_3 \frac{T^{3/2}}{T + C_4}$$

where C_1 to C_4 are constants for a given gas. For air at moderate temperatures, $C_1 = 1.458 \times 10^{-6}$, $C_2 = 110.4$, $C_3 = 2.495 \times 10^{-3}$ and $C_4 = 194$ in SI units. Note that the temperature T here must be here assumed to be dimenional variable in SI unit.

In this project, the Prandtl number is assumed to variable based on the following definition:

$$\mathrm{Pr} = \frac{c_p \mu}{k}$$

3.2 Initial Conditions

At the beginning of simulation, the 2DNS code sets the initial condition. After then the code set the boundary conditions at every time step. The initial conditions at all grid points is set on the basis of following pre-specified flow quantities in nondimensional forms:

$$M = 2.0, \quad \rho = 1.0, \quad u = 1.0, \quad v = 1.0, \quad \gamma = 1.4, \quad p = \frac{1}{\gamma M^2}, \quad T = 1.0$$

The reference free stream conditions used to nondimensionalize the flow variables are given by:

$$\rho_\infty = 0.01[\mathrm{kg/m^3}], \quad T_\infty = 300[\mathrm{K}], \quad V_\infty = 694.44[\mathrm{m/sec}], \quad L_\infty = 1.0[\mathrm{m}]$$

3.3 Boundary Conditions

The flow is assumed to be coming in and blowing out at both inlet and outlet under a supersonic condition. Thus the following boundary conditions can be suitable:

Inflow: $\vec{u}^n_{1,j}$ = fixed at initial conditions at every time step

Outflow: $\vec{u}^n_{imax,j} = \vec{u}^n_{imax-1,j}$ (1st order extrapolation for all n)

Inviscid wall (top): No velocity in the η direction. The 2DNS code uses a 2nd order extrapolation that is described in the 3rd computer project assignment.

Bottom wall:

In this project, two different wall boundary conditions are employed based on the treatment of wall temperature: adiabatic wall and isothermal wall boundaries. For the adiabatic wall boundary condition, the heat flux normal to the surface is enforced to be zero by:

$$T_{i,1} = T_{i,2}$$

For the isothermal wall boundary, the pre-specified wall temperature is applied to every $j = 1$ node points. In this project, 300 K is applied to the wall temperature as isothermal boundary condition.

In addition to the wall temperature BC, viscous wall boundary should be taken account. This can be made by assuming no-slip wall boundary for the moderate gas pressure. In this approach, the velocity right at the wall is set to zero. Then the surface pressure is calculated from the assumption that the normal component of the momentum equation is zero. This can be implemented by resolving the following relation for a non-orthogonal grid with no-slip condition:

$$\left(x_\xi^2 + y_\xi^2\right) \frac{\partial p}{\partial \eta} = \left(x_\xi x_\eta + y_\xi y_\eta\right) \frac{\partial p}{\partial \xi}$$

From the pressure and temperature resolved above, the density is then enforced by solving the gas equations of state.

3.4 Convergence Log (RMS error)

In order to see the convergence history, the 2DNS code calculates the following RMS error at every time step:

$$\text{RMS}^n = \sqrt{\frac{1}{N} \sum_{m=1}^{4} \sum_{i=1}^{imax} \sum_{j=1}^{jmax} \left[\left(\vec{U}^{n+1}_{i,j} - \vec{U}^n_{i,j} \right)^2 \right]}$$

This error log was used for checking convergence history only not for the termination of program. Every cases in this project was run by 40,000 iterations.

CHAPTER 4

How to run the code

4.1 Machine platform for development

This 2DNS code has been developed on personal computer operating on linux system (Ubuntu Linux 3.2.0-38-generic x86_64). Machine specification is summarized as shown below:

vendor_id : GenuineIntel

cpu family : 6

model name : Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz

cpu cores : 4

Memory : 16418112 kB

4.2 Code setup

The 2DNS source code has been developed with version management tool, GIT. The git repository was built on 'github.com'. Thus, the source code as well as related document files can be cloned into user's local machine by following command:

```
$ git clone http://github.com/sayop/2DNS.git
```

If you open the git-cloned folder **CouetteFlow**, you will see two different folders and README file. The **CODEdev** folder contains again **bin** folder, **Python** folder, and **src** folder. In order to run the code, use should run **setup.sh** script in the **bin** folder. **Python** folder contains python scripts that are used to postprocess data. It may contain **build** folder, which might have been created in the different platform. Thus it is recommended that user should remove **build** folder before setting up the code. Note that the **setup.sh** script will run **cmake** command. Thus, make sure to have cmake installed on your system:

```
$ rm -rf build
$ ./setup.sh
-- The C compiler identification is GNU 4.8.1
-- The CXX compiler identification is GNU 4.8.1
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
```

```
-- Detecting CXX compiler ABI info - done
-- The Fortran compiler identification is GNU
-- Check for working Fortran compiler: /usr/bin/gfortran
-- Check for working Fortran compiler: /usr/bin/gfortran  -- works
-- Detecting Fortran compiler ABI info
-- Detecting Fortran compiler ABI info - done
-- Checking whether /usr/bin/gfortran supports Fortran 90
-- Checking whether /usr/bin/gfortran supports Fortran 90 -- yes
-- Configuring done
-- Generating done
-- Build files have been written to: /home/sayop/data/Devel/GitHub.Clones/2DNS/CODEdev/bin/build
Scanning dependencies of target cfd.x
[  7%] Building Fortran object CMakeFiles/cfd.x.dir/main/Parameters.F90.o
[ 15%] Building Fortran object CMakeFiles/cfd.x.dir/main/SimulationVars.F90.o
[ 23%] Building Fortran object CMakeFiles/cfd.x.dir/main/GridJacobian.F90.o
[ 30%] Building Fortran object CMakeFiles/cfd.x.dir/main/AUSMPWplus/AUSMPWplus.F90.o
[ 38%] Building Fortran object CMakeFiles/cfd.x.dir/main/BCvisc/BCvisc.F90.o
[ 46%] Building Fortran object CMakeFiles/cfd.x.dir/main/VISCflux/VISCflux.F90.o
[ 53%] Building Fortran object CMakeFiles/cfd.x.dir/main/TimeIntegration.F90.o
[ 61%] Building Fortran object CMakeFiles/cfd.x.dir/io/io.F90.o
[ 69%] Building Fortran object CMakeFiles/cfd.x.dir/main/SimulationSetup.F90.o
[ 76%] Building Fortran object CMakeFiles/cfd.x.dir/io/RestartDataOut.F90.o
[ 84%] Building Fortran object CMakeFiles/cfd.x.dir/main/MainLoop.F90.o
[ 92%] Building Fortran object CMakeFiles/cfd.x.dir/io/ReadGrid.F90.o
[100%] Building Fortran object CMakeFiles/cfd.x.dir/main/main.F90.o
Linking Fortran executable cfd.x
[100%] Built target cfd.x
```

If you run this, you will get executable named **cfd.x** and **input.dat** files. The input file is made by default. You can quickly change the required input options.

4.3 Input file setup

The 2DNS code allows user to set multiple options to solve the unsteady 2-dimensional Navier-Stokes problem by reading **input.dat** file at the beginning of the computation. Followings are default setup values you can find in the input file when you run **setup.sh** script:

```
#Input file for tecplot print
2-D Euler solver
imax            65
jmax            57
ngl             3
gridFile        NSgrid.dat
#Initial conditions (SI)
density         1.0
u               1.0
v               0.0
pressure        0.1785714
temp            1.0
gamma           1.4
#Free stream conditions
dens_ref        0.01
temp_ref        300.0
Uvel_ref        694.44
Leng_ref        1.0
Cp              1003.5
```

```
#Simulation parameters
restart         0
nmax            1000
CFL             0.5
errorLimit      1e-06
#AUSMPW+ parameters
alpha           0.0
epsil           0
limiter         0
kappa           0.0
#Viscous solver parameters
visc            0
wallT           0.0
#Data Post-Processing (BL profile)
xLoc            0.0
```

- **First line** ('2-D Navier-Stokes solver' by default): Project Name
- **imax**: number of grid points in i-direction
- **jmax**: number of grid points in j-direction
- **ngl**: number of ghost layers (not available in this project)
- **gridFile**: grid file name to be read
- **density**: incoming flow density
- **u**: incoming flow velocity in x-direction [Non-dimension]
- **v**: incoming flow velocity in y-direction [Non-dimension]
- **pressure**: incoming flow pressure [Non-dimension]
- **temp**: incoming flow temperature [Non-dimension]
- **gamma**: incoming flow heat specific ratio
- **dens_ref**: Reference flow density [Dimensional]
- **temp_ref**: Reference flow temperature [Dimensional]
- **Uvel_ref**: Reference flow axial velocity [Dimensional]
- **Leng_ref**: Reference length [Dimensional]
- **Cp**: Constant pressure specific heat capacity
- **restart**: If 1, the solver will restart from the specified iteration number. If you run the code from scratch, it must be set to zero.
- **nmax**: maximum number of iteration to be allowed and terminate case running
- **CFL**: CFL number
- **errorLimit**: normalized RMS error limit for convergence
- **alpha**: coefficient in AUSMPW+ (not used in this project)
- **epsil**: switch of second order accurate MUSCL differencing
- **limiter**: switch of MUSCL minmod limiter
- **kappa**: control parameter for 1st/2nd order accurate upwind differencing of MUSCL
- **visc**: If 0, the code runs as 2D Euler solver. If 1, the code runs as 2D Navier-Stokes solver.

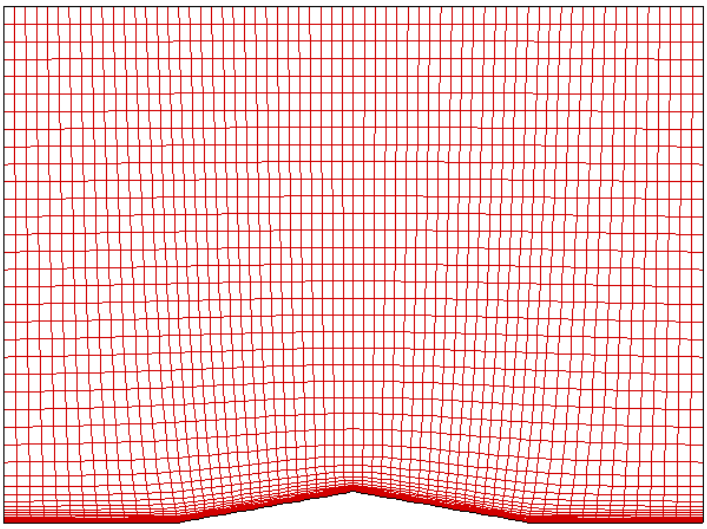
- **wallT**: Bottom wall temperature. If it is set to a positive value, the code set the wall to isothermal wall boundary. If it is set to negative value, the code assumes the adiabatic wall boundary condition.
- **xLoc**: x-position to collect the boundary layer profile. The code collects the non-dimensional axial velocity and dimensional temperature and stores them to separate file called 'BoundaryLayer.dat'.

CHAPTER 5

Results and discussions

5.1 Computational Grid

The grid used in this project has a resolution of 65 X 57 in i- and j-directions as shown below. In this project, unsteady 2-dimensional Navier-Stokes solution is being resolved by performing the explicit time-integration.



<Computational grid>

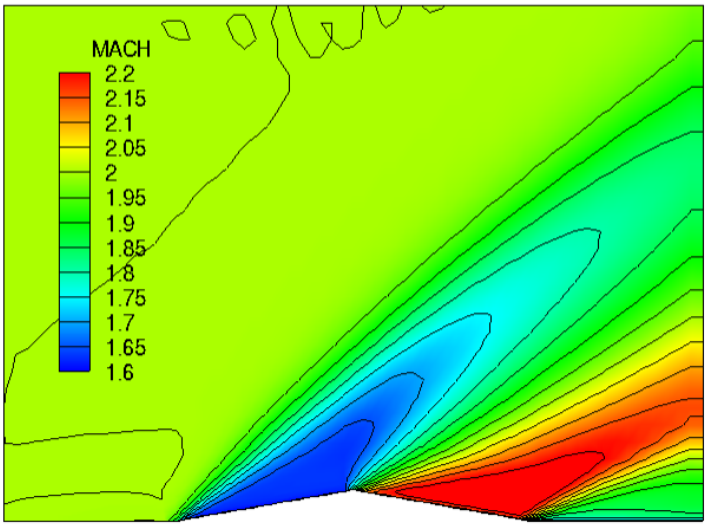
5.2 Inviscid solution (CASE 1 & CASE 2)

Following cases were achieved by running the 2DNS code with inviscid option off. The solution was obtained by running 40,000 iterations.

CASE 1

- Inviscid solution
- 1st order accuracy (CFL = 0.5)

For the first order accurate solution, central differencing was applied to solve the inviscid flux terms with AUSMPW+ scheme and to resolve the first derivatives in shear stress and heat flux terms. The following figure represents the fully developed shock waves across the airfoil in terms of local Mach number.

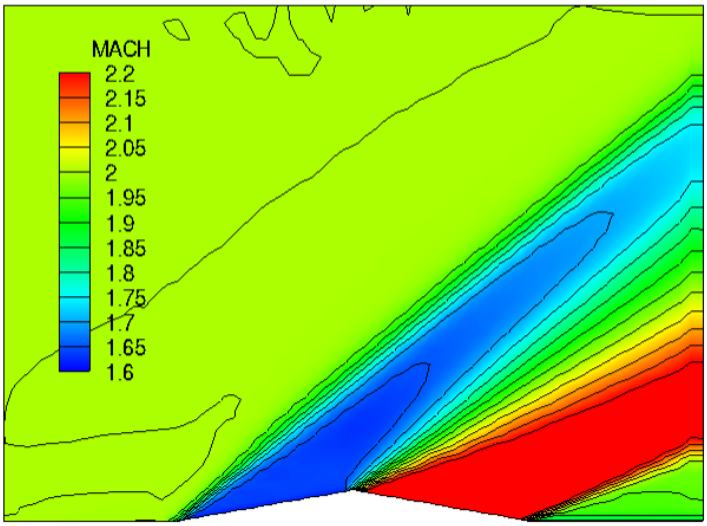


<Contour plot of Mach number: CASE 1>

CASE 2

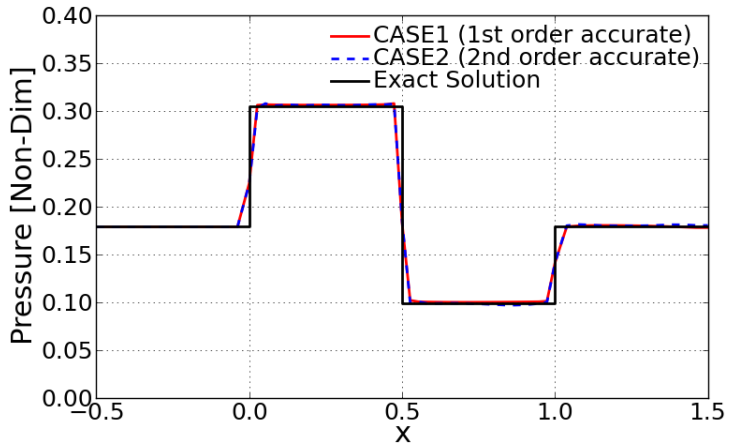
- 2nd order accurate (CFL = 0.5)
- with basic minmod limiter

For the second approach to resolve the inviscid solution, the fully upwind scheme for extrapolation was employed based on the characteristics wave direction. Then the basic minmod limiter was turned on to get TVD(Total Variation Diminishing) solution.



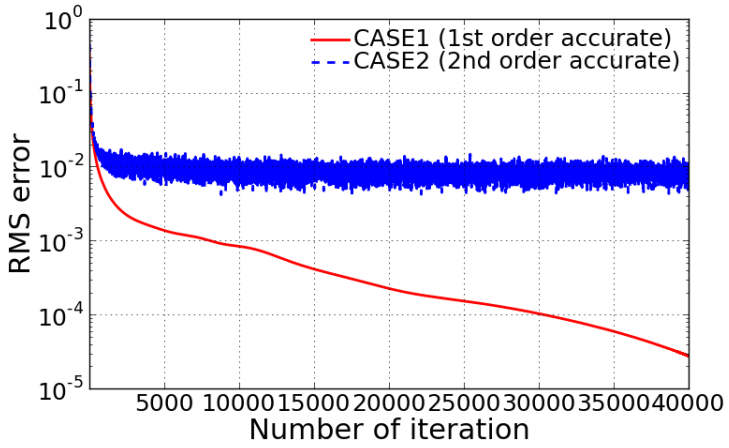
<Contour plot of Mach number: CASE 2>

The figure shown below illustrates the resolved pressure profile along the bottom wall boundary for the inviscid fluid. The resulted pressures seem to be overlapped each other. However, very slightly dispersive nature can be found for the 2nd order accurate solution at the shock wave anchored location around x = 0.0 and x = 1.0. On the other hand, dissipative nature is dominated in the first order accurate solution. This can be observed in the contour lines shown above. The CASE 2 contour lines along the shock surface shows wave-looking lines.



<Comparison of static pressure along the bottom wall boundary>

The following figure present the RMS residual history at every iterations. It is observed that the first order accurate solution converges very smoothly with no oscillation nature as discussed above. However, the second order accurate solution clearly represents the TV (Total Varation) dominant phenomenon. This is also called Limit Cycle Oscillations (LCO).



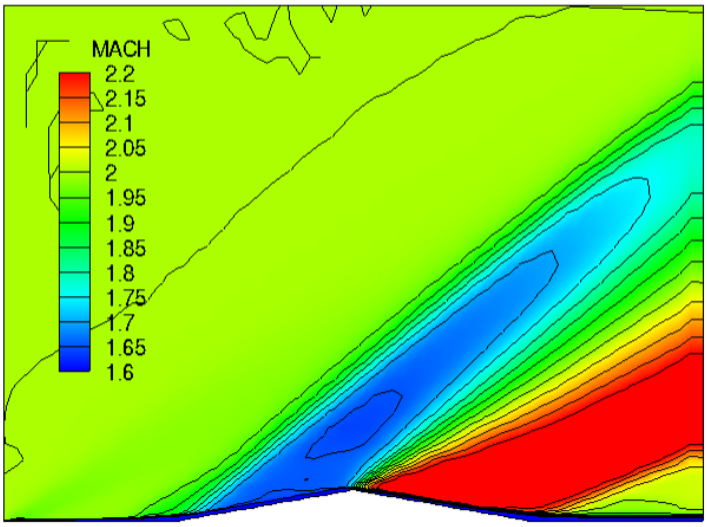
<Comparison of RMS error log>

5.3 Viscous flow solution (CASE 3 & CASE 4)

Following cases were achieved by running the 2DNS code with inviscid option on. The solution was obtained by running 40,000 iterations.

CASE 3

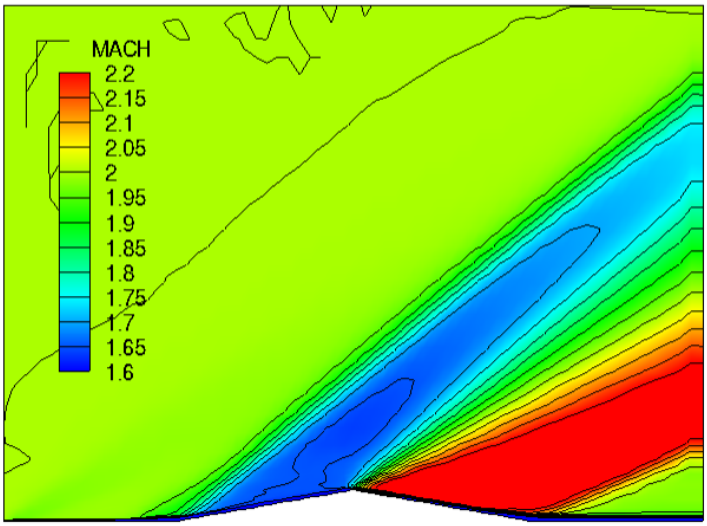
- Navier-Storkes solution
- 2nd order accurate (CFL = 0.5)
- with basic minmod limiter
- Adiabatic wall BC



<Contour plot of Mach number: CASE 3>

CASE 4

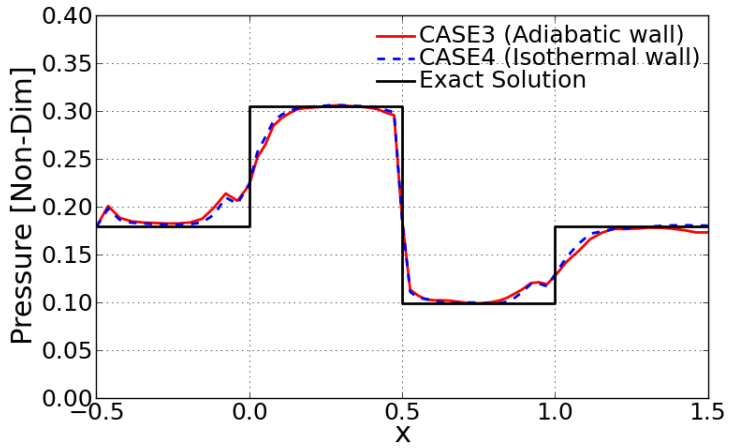
- Navier-Storkes solution
- 2nd order accurate (CFL = 0.5)
- with basic minmod limiter
- Isothermal wall BC (T_{wall} = 300 k)



<Contour plot of Mach number: CASE 4>

The figure shown below illustrates the resolved pressure profile along the bottom wall boundary. Here, remarkable difference from the inviscid solution can be observed. The Navier-Stokes solution tends to smear the solution out across the shock forming location. This is a clue of viscous fluid. We observed that the first order accurate solution has a dissipative nature due to the artificial viscosity. Likewise, physical viscous fluids can play a important role in dissipative solution in the high gradient region.

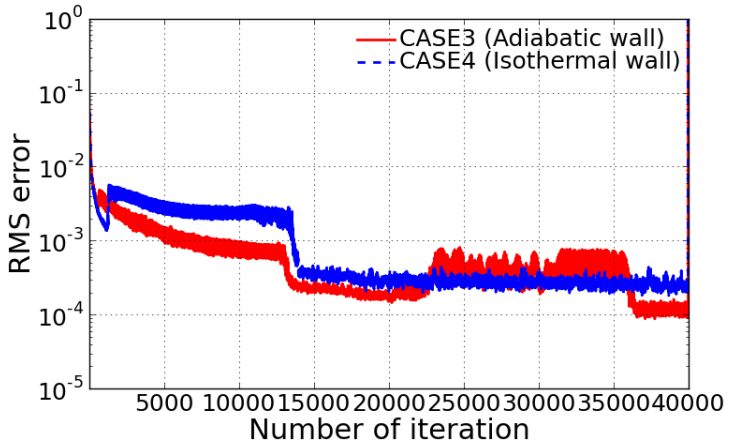
In addition to the dissipative profile, we observe the small bump in the leading edge. This is because a very weak shock forms as the bounday layer grows and leads to the slight change of flow angle.



<Comparison of static pressure along the bottom wall boundary>

Contrary to the inviscid solution, viscous flow solution tends to show some unsteadiness in the RMS residual history. We alreay observe that the RMS log goes slowly down and smoothly converges. However, both CASE 3 and CASE 4 show a slight bump-up and bump-down a while later. This represents the developed separation flow in the boundary layer. This is a distinct nature of viscous flow. Thus the unsteadiness of RMS log is simply due to the physical unsteadiness of the viscous flow.

Furthermore, a slight reduction in the amplitude of the oscillations for the viscous flows is found compared to the previous inviscid flow. It is believed that the oscillatory nature of the viscous flow solution can be a little bit diminished. However, the nature of Limit Cycle Oscillations will never be lost because LCO is the distinct numerical phenomenon of the 2nd order accurate solution.



<Comparison of RMS error log>

5.4 Prediction of Boundary Layer Profile

Theoretical study of Boundary-Layer thickness

In the earlier study of van Driest (1952) proposed a semiempirical formula to approximate the boundary layer thickness in compressible fluids as a function of freestream Mach number. In this study, the dimensionless boundary-layer thickness was found to grow with the Mach number for both adiabatic and nonadiabatic walls. The predicted boundary

layer thickness can be determined by:

$$\frac{\delta}{x} Re_{xe}^{1/2} \approx C_w^{1/2} \left[5.0 + \left(0.2 + 0.9 \frac{T_w}{T_{aw}} \right) (\gamma - 1) Ma_e^2 \right]$$

Here, T_{aw} is a wall temperature that would have been achieved if it was set to the adiabatic wall for given freestream Mach number. The estimated adiabatic wall temperature can then be determined by:

$$\frac{T_{aw}}{T_w} = 1 + \mathrm{Pr}^{1/2} \left(\frac{\gamma - 1}{2} \right) Ma_e^2$$

where Pr is Prandtl number which is evaluated as 0.71432 for given freestream temperature, 300 K. In this project, C_w was set to unity. Finally, boundary layer thickness is then determined as a function of wall temperature T_w, distance from leading edge x, and freestream Mach number Ma_e.

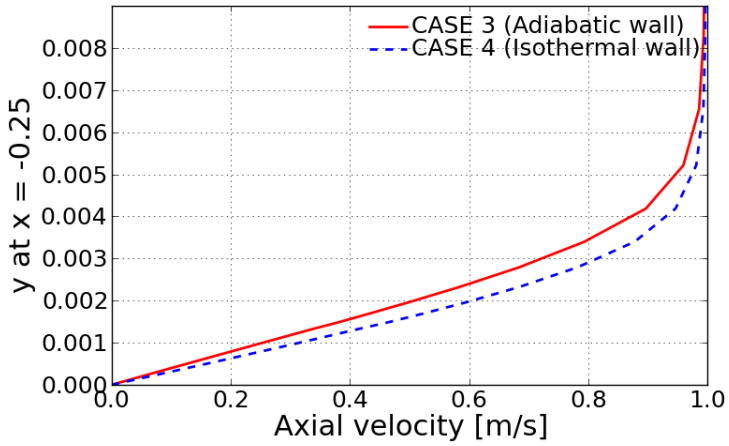
[Ref. Van Driest, E. R. (1951) "Turbulent Boundary Layer in Compressible Fluids", J. Aeronaut. Sci., vol. 18, pp. 145-160.]

x = -0.25

By solving the theoretical boundary layer thickness relation above, the following were obtained for both adiabatic wall and isothermal wall.

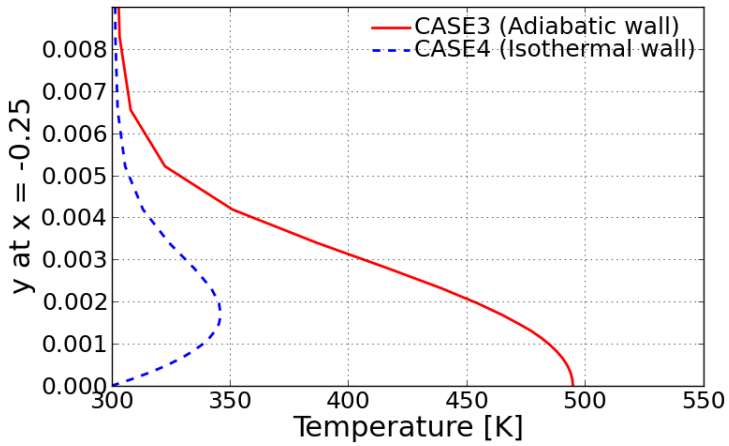
- Adiabatic wall: δ = 0.00664
- Isothermal wall: δ = 0.00493

The following figure presents the axial velocity profiles resolved by current CFD solution. As theoretically predicted, adiabatic wall condition gives higher boundary layer thickness. Comparing the CFD solution to the theoretical solution, it can be said that the current CFD solution well follows the theoretical approximation.



<Nondimensional axial velocity profile in boundary layer>

The figure shown below illustrates how the temperature profile looks like for the two different type of wall condition. As can be simply predicted, the adiabatic wall temperature is higher than the isothermal wall temperature which was originally identical to the freestream temperature. The higher temperature field in boundary layer for the adiabatic wall condition can become a primary reason of thicker boundary layer. In other words, the higher temperature the fluid gets, the higher molecular diffusivity becomes. Thus,the higher diffusivity in boundary layer may drag the upper layer of higher momentum.



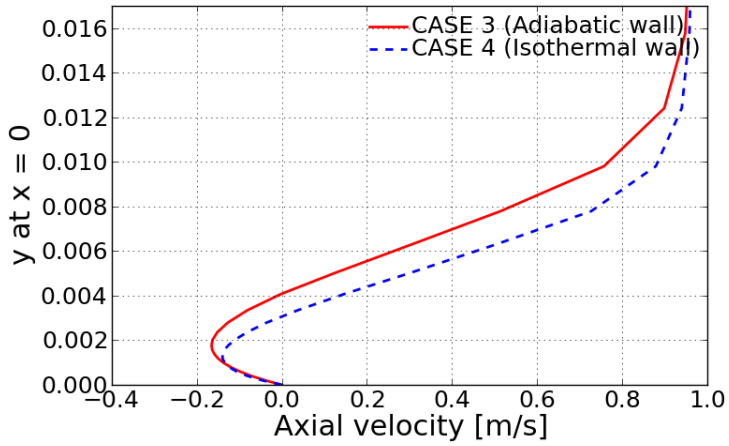
<Dimensional temperature profile in boundary layer>

x = 0

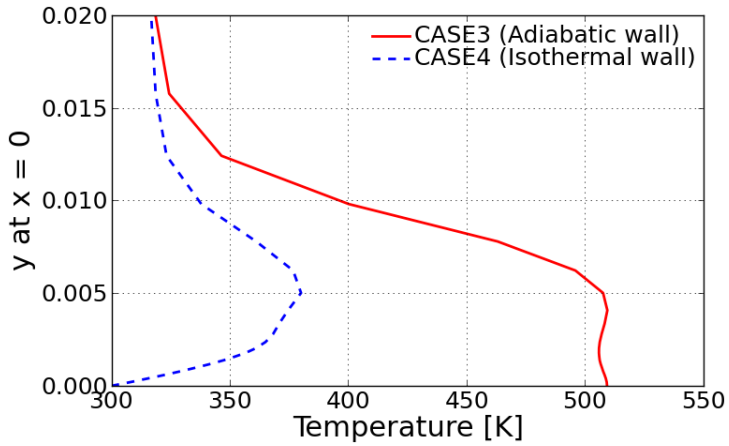
The same solution can be applied for the boundary layer thickness prediction at x = 0. In this solution, x should be evaluated as the distance from the leading edge so that x = 0.5 m. The theoretically obtained solution are:

- Adiabatic wall: $\delta = 0.00939$
- Isothermal wall: $\delta = 0.00698$

In this case, we observe big difference between CFD solution and theoretical solution. This can be explained by knowing that the theoretical solution does not take separation flow into account. As the negative axial velocity is observed from the figure below, the CFD solution predicts the separated flow phenomena at this location due to the adverse pressure gradient. As already discussed, the adiabatic wall boundary condition gives rise to the thicker boundary layer regardless of formation of separation flow. Thus it can be concluded that the boundary layer is an outcome of momentum transfer between two upper and lower layer that have different molecular diffusivities. Since we know that viscosity is a dominant function of temperature, the higher wall temperature condition is more likely to give more viscid flow and leads to thicker boundary layer.



<Nondimensional axial velocity profile in boundary layer>



<Dimensional temperature profile in boundary layer>

5.5 Computational performance

The following table compares the CPU time consumed for each case. We can clearly find that the Navier-Stokes solution consumes much more CPU resource in terms of computational speed because it has to calculate the additional flux terms and more derivatives associated with shear stress and heat conductivity.

CASE #	CFL	CPU Time [sec]
1	0.5	141.714
2	0.5	189.65
3	0.5	590.071
4	0.5	589.124

The CFL number to ensure the convergence varies for each case. The following tables listed the maximum CFL number that was obtained by experimenting the various CFL number at the interval of 0.05. The slight bigger CFL number than those number triggers the divergence of numerical solution.

CASE #	Maximum CFL
1	0.85
2	0.6
3	0.8
4	0.8