OverBlown : A Fluid Flow Solver For Overlapping Grids, User Guide, Version 1.0

William D. Henshaw
Centre for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA, 94551.
henshaw@llnl.gov
http://www.llnl.gov/casc/people/henshaw
http://www.llnl.gov/casc/Overture

November 6, 2003

UCRL-MA-134288

Abstract:

OverBlown is a program that can be used to solve fluid flow problems on overlapping grids. It is built upon the **Overture** object-oriented framework. **OverBlown** has a number of different algorithms that can be used to solve problems for a range of Mach numbers. The Mach number, M, is the ratio of the flow speed to the speed of sound. In particular there are algorithms suited for

- incompressible flow, $M = 0$, (method INS)
- low Mach number flows, $M < .5$, (method ASF)
- moderate Mach numbers $.25 < M < 1.0$, (method CNS) and high Mach number flows $.25 < M$, (method CNSCAD).
- reactive Euler equations in 2D (method CNSGOD).

OverBlown can be used to solve problems on moving grids. **OverBlown** can also be used to solve simple chemically reacting flows.

Contents

1 Introduction

OverBlown is a fluid flow solver for overlapping grids built upon the **Overture** framework [2],[6],[3]. **OverBlown** can be used to solve the incompressible Navier-Stokes equations (INS), and the compressible Navier-Stokes equations using either an all-speed flow algorithm (ASF), a moderate Mach number algorithm (CNS) or a high Mach number algorithm (CNSCAD or CNSGOD). The ASF algorithm would be appropriate from low to moderate Mach number, say $M < .5$, the CNS algorithm best for $.25 < M < 1$. while the CNSCAD algorithm is best for $M > .25$ (approximately). CNSGOD is for the 2D Euler equations (inviscid Navier-Stokes) with some optional chemical reactions.

More information about **Overture** can be found on the **Overture** home page, `http://www.llnl.gov/casc/-Overture`. For installation procedures see section (7).

The **OverBlown** distribution consists of a directory, `OverBlown`, plus subdirectories:

`OverBlown/bin` : contains the executable, **overBlown** . You may want to put this directory in your path.

`OverBlown/ins` : sample command files for running computations of the incompressible Navier-Stokes equations, see section (2).

`OverBlown/cns` : sample command files for running computations of the compressible Navier-Stokes equations.

`OverBlown/asf` : sample command files for running computations with the all-speed-flow solver for the compressible Navier-Stokes equations (this option needs more work).

`OverBlown/lib` : contains the **OverBlown** library, `libOverBlown.a`.

`OverBlown/src` : source files (.C files) for OverBlown.

`OverBlown/check` : contains testing routines for comparing the answers on test problems to previously run cases.

Other documents of interest that are available through the **Overture** home page are

- The **OverBlown** Reference Guide [11] for detailed descriptions of the equations, algorithms and discretizations.
- The overlapping grid generator, `Ogen`, [8]. Use this program to make grids for **OverBlown** .
- Mapping class documentation : `mapping.tex`, [7]. Many of the mappings that are used to create an overlapping grid are documented here.
- Interactive plotting : `PlotStuff.tex`, [10].
- `Oges` overlapping grid equation solver, used by **OverBlown** to solve implicit time stepping equations and the Poisson equation for the pressure, [9].

1.1 Basic steps

Here are the basic steps to solve a problem with **OverBlown** .

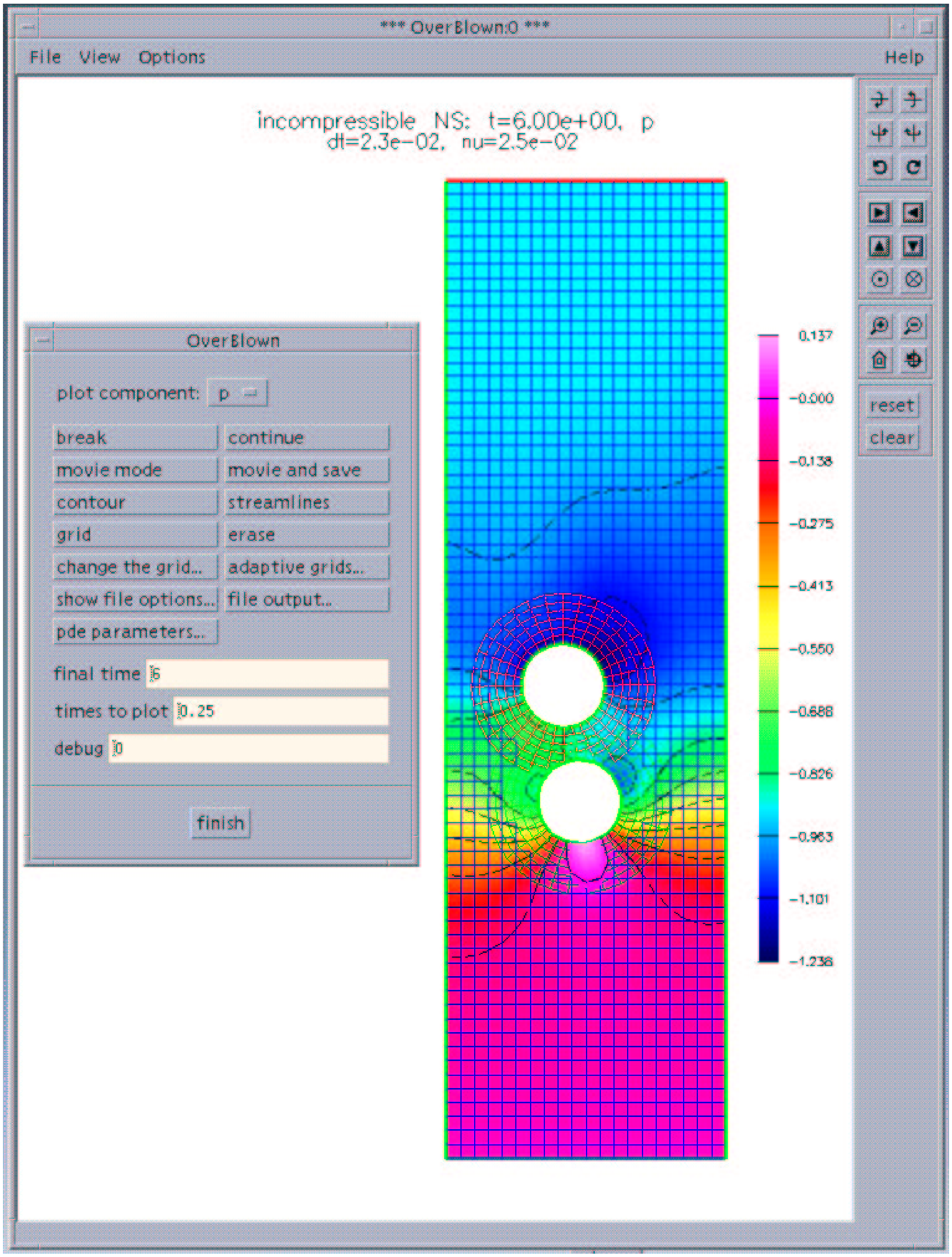
1. Generate an overlapping grid with ogen. Make the grid with 2 ghost lines.
2. Run **overBlown** (note lowercase 'o', found in the `OverBlown/bin` directory) and choose the PDE you want to solve.
3. Assign the boundary conditions and initial conditions.
4. Choose the parameters for the PDE (Reynold's number, Mach number etc.)
5. Choose run time parameters, time to integrate to, time stepping method etc.
6. Compute the solution (optionally plotting the results as the code runs).
7. When the code is finished you can look at the results (provided you saved a 'show file') using `plotStuff`.

The commands that you enter to run **OverBlown** can be saved in a command file (by default they are saved in the file 'overBlown.cmd'). This command file can be used to re-run the same problem by typing 'overBlown file.cmd'. The command file can be edited to change parameters.

To get started you can run one of the demo's that come with **OverBlown** , these are explained next in section (2).

Papers that describe some of the algorithms used in OverBlown include

1. *Composite Overlapping Meshes for the Solution of Partial Differential Equations* [4].
2. *A Fourth-Order Accurate Method for the Incompressible Navier-Stokes Equations on Overlapping Grids* [5].
3. *Analysis of a Difference Approximation for the Incompressible Navier-Stokes Equations* [12].



plot component: p

final time 6

times to plot 0.25

debug 0

Figure 1: Snapshot of OverBlown showing the run time dialog menu. The figure shows two falling bodies in an incompressible flow, computed with the command file twoDrop.cmd.

2 Sample command files for running OverBlown

Command files are supported throughout the Overture. They are files that contain lists of commands. These commands can initially be saved when the user is interactively choosing options. The command files can then be used to re-run the job. Command files can be edited and changed.

In this section we present a number of command files that can be used to run **OverBlown** .

2.1 Running a command file

Given a command file for **OverBlown** such as `cylinder.cmd`, found in `OverBlown/ins/cylinder.cmd`, one can type '`overBlown cylinder.cmd`' to run this command file . You can also just type '`overBlown cylinder`, leaving off the `.cmd` suffix. Typing '`overBlown noplot cylinder`' will run without interactive graphics (unless the command file turns on graphics). Note that here I assume that the `OverBlown/bin` directory is in your path so that the `overBlown` command is found when you type it's name. The **OverBlown** sample command files will automatically look for an overlapping grid in the `Overture/sampleGrids` directory, unless the grid is first found in the location specified in the command file.

When you run a command file a graphics screen will appear and after some processing the run-time dialog should appear and the initial conditions will be plotted. The program will also print out some information about the problem being solved. At this point choose `continue` or `movie mode`. Section (3.3) describes the options avialable in the run time dialog.

2.2 Incompressible flow past a cylinder in a long channel

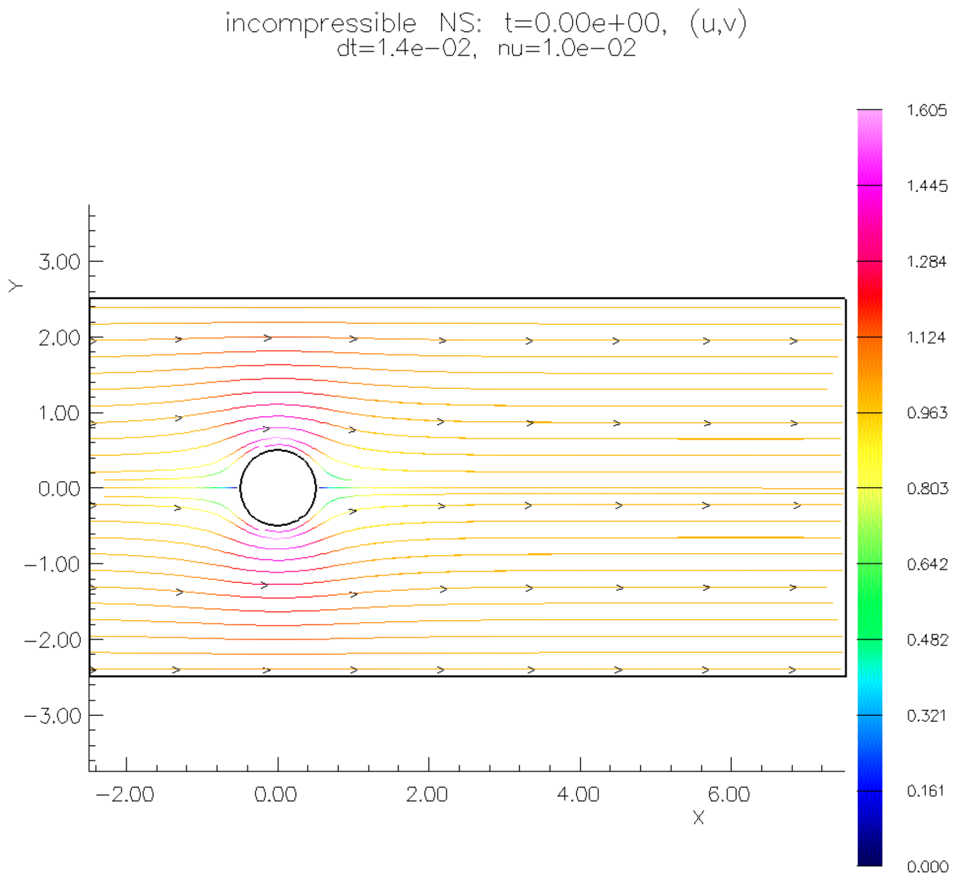
The command file `cylinder.cmd` in `OverBlown/ins` can be used to compute the incompressible flow past a cylinder in a channel, figure (2.2). This example uses the grid `Overture/sampleGrids/cilc.hdf`.

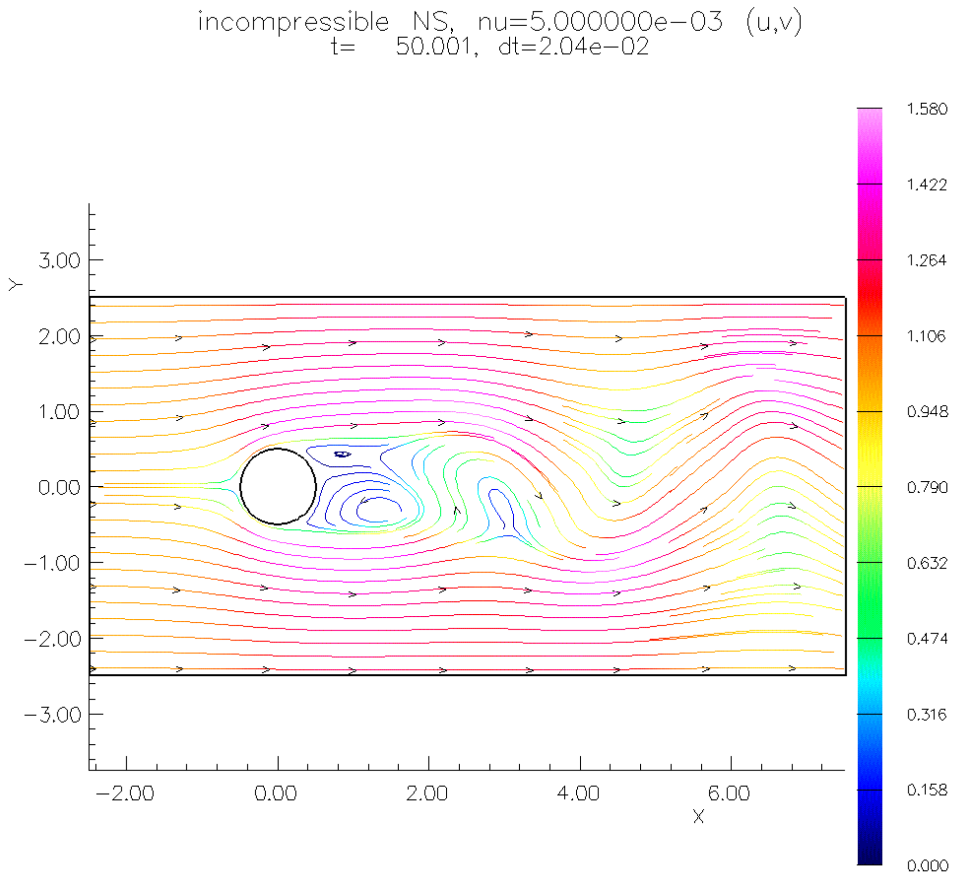
```
*
* OverBlown command file for flow past a cylinder
*
* specify the overlapping grid to use:
cilc.hdf
* Specify the equations we solve:
  incompressible Navier Stokes
  exit
*
* Next specify the file to save the results in.
* This file can be viewed with Overture/bin/plotStuff.
  show file options
    * compressed
    * open
    *  cylinder.show
    frequency to flush
      5
    exit
*   display parameters
  turn off twilight zone
* choose implicit time stepping:
  implicit
* but integrate the square explicitly:
  choose grids for implicit
    all=implicit
    square=explicit
    done
  final time (tf=)
    5.
  times to plot (tp=)
    1.
  plot and always wait
**  no plotting
  pde parameters
    nu
      .01
     turn off second order artificial diffusion
     turn off fourth order artificial diffusion
*     turn on second order artificial diffusion
*     OBPDE:second-order artificial diffusion 1
*     OBPDE:ad21,ad22 2,2
*       OBPDE:fourth-order artificial diffusion 1
*       OBPDE:ad41,ad42 1,1
    done
* cfl
*   .25
  boundary conditions
    all=noSlipWall
    square(0,0)=inflowWithVelocityGiven, uniform(p=1.,u=1.)
    square(1,0)=outflow
    square(0,1)=slipWall
    square(1,1)=slipWall
    done
  initial conditions
*      read from a show file
*      cylinder.show
*       9
  uniform flow
    p=1., u=1.
  exit
  project initial conditions
  continue


  movie mode
  finish

```

To run this command file from the `OverBlown/ins` directory type `‘overBlown cylinder’` (or `../bin/overBlown cylinder’` if you have not set your path).

Notes:

- You may have to change the name of the overlapping grid file, `cilc.hdf` (specified near the top of the command file) to be the correct location of the file, such as `henshaw/myGrids/cilc.hdf`. The suffix `‘.hdf’` is optional when specifying grids as Overture will tack on `‘.hdf’` if necessary. If **overBlown** does not find the file specified it will also by default look for the file in the `Overture/sampleGrids` directory.
- The initial conditions are assigned to be a uniform flow, $(u, v) = (1, 0)$. These initial conditions are projected to nearly satisfy $\nabla \cdot \mathbf{u} = 0$ by using the `‘project initial conditions’` option.
- The time-stepping method is chosen so that the grid around the cylinder uses implicit time-stepping while the back-ground grid uses explicit time-stepping. This was done for efficiency. The grids around the cylinder have small grid spacings so that implicit time stepping is especially useful. The back-ground grid does not have small grid spacings so there is not much of an advantage in using implicit time stepping. By treating the back-ground grid explicitly the implicit time stepping equations require less storage and cpu time to solve.
- By default the elliptic pressure equation is solved with a direct sparse solver. This usually is the best approach for 2D problems, unless the grids get large, since the matrix is factored only once. In later 3D examples it is shown how to specify an iterative method.

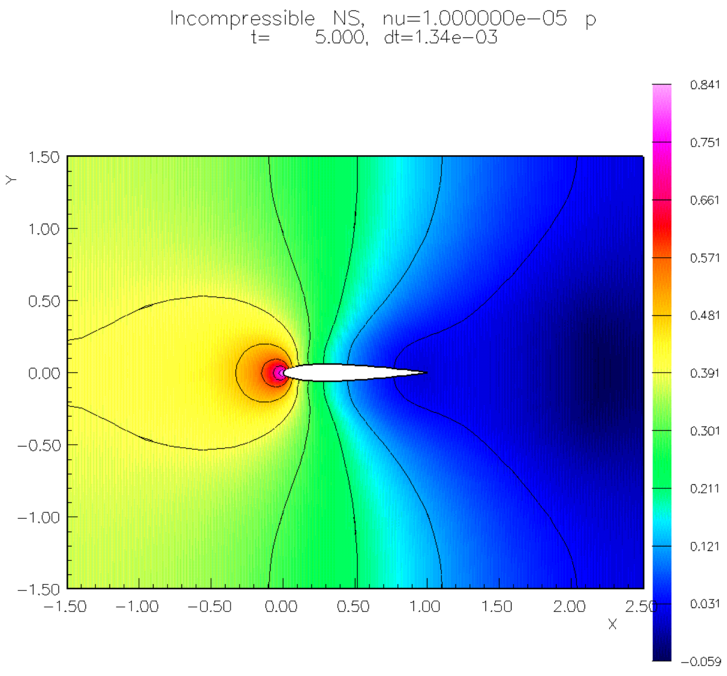

incompressible NS: t=0.00e+00, (u,v)
dt=1.4e-02, nu=1.0e-02
Y
3.00
2.00
1.00
0.00
−1.00
−2.00
−3.00
−2.00
0.00
2.00
4.00
6.00
X
1.605
1.445
1.284
1.124
0.963
0.803
0.642
0.482
0.321
0.161
0.000


incompressible NS, nu=5.000000e−03 (u,v)
t= 50.001, dt=2.04e−02
Y
3.00
2.00
1.00
0.00
−1.00
−2.00
−3.00
−2.00
0.00
2.00
4.00
6.00
X
1.580
1.422
1.264
1.106
0.948
0.790
0.632
0.474
0.316
0.158
0.000

2.3 Incompressible flow around a naca airfoil

The following command file can be used with **OverBlown** to compute the flow around a naca airfoil. This example uses the overlapping grid `Overture/sampleGrids/naca0012.hdf` generated using the command file `Overture/-sampleGrids/naca0012.cmd` (or the grid `OverBlown/grids/naca.hype.hdf` generated with `Overture/-sampleGrids/naca.hype.cmd`)

```
*
* OverBlown command file for flow past a naca0012 airfoil
*  use either naca0012.hdf or naca.hype.hdf
*
* naca.hype
naca0012
*
  incompressible Navier Stokes
  exit
*
  show file options
    open
     naca.show
  exit
  turn off twilight zone
  implicit
  choose grids for implicit
    all=implicit
    backGround=explicit
    done
  implicit factor
    0.75
  final time (tf=)
    1.
  times to plot (tp=)
    .1
  plot and always wait
  * no plotting
  pde parameters
    * the next value for nu is too small
    nu
      1.e-8
   turn on second order artificial diffus
  done
*  cfl
*    .75
  boundary conditions
    all=noSlipWall
    backGround(0,0)=inflowWithVelocityGiv
    backGround(1,0)=outflow
    backGround(0,1)=slipWall
    backGround(1,1)=slipWall
    done
  initial conditions
  uniform flow
    p=1., u=1.
  done
  project initial conditions
  continue
```



Incompressible flow around a NACA 0012 airfoil.

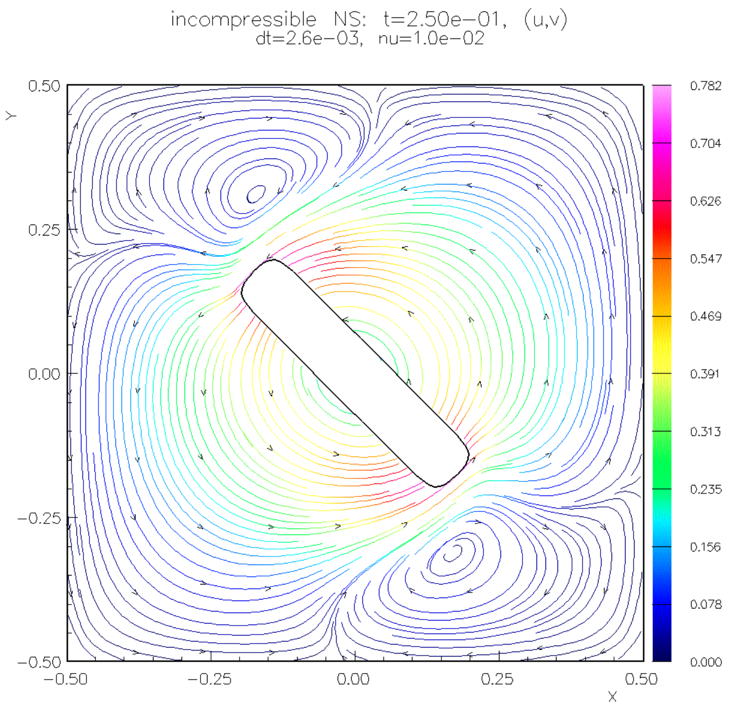
If you run this example you will notice messages printed to the effect that the divergence is large on some parts of the grid. By looking at the show file with `plotStuff` and plotting the divergence it can be seen that the divergence is large near the leading edge where there are large gradients in the solution. This is not unexpected when using artificial diffusion since only a minimal amount of smoothing is added. However, it could also indicate that either I need a more refined grid there or perhaps a better discretization method such as a finite volume method might work better.

This example demonstrates the use of the second-order artificial diffusion as described in section (3.8). The value of the artificial diffusion is determined in a local way that depends on the velocity gradients so as to keep the solution nicely behaved but with a minimum of dissipation. There is still some fiddling to do to get the coefficients of the artificial diffusion correct. The values are always around 1. Here I used a value of 2 where often a value of .5 will work fine.

2.4 Incompressible flow around a moving stirring stick

The following command file can be used with **OverBlown** to compute the flow around a rotating tongue depressor. This example uses the overlapping grid `Overture/sampleGrids/stir.hdf` generated using the command file `Overture/-sampleGrids/stir.cmd`.

```
*
* stirring stick
*
  stir.hdf
*  stir1.hdf
  incompressible Navier Stokes
  exit
  show file options
    * compressed
    open
     stir.show
    frequency to flush
      5
    exit
  turn off twilight zone
  project initial conditions
*
  turn on moving grids
  specify grids to move
    rotate
      0. 0. 0.
*    specify rate and rampInterval (rampI
      .5 .0
    stir
    done
  done
  * use implicit time stepping
  * implicit
  * choose grids for implicit
  *    all=explicit
  *    stir=implicit
  *   done
  pde parameters
    nu
       .05
    done
  boundary conditions
    all=noSlipWall
    done
  initial conditions
    uniform flow
     p=1.
  exit
  final time (tf=)
    .5
  times to plot (tp=)
    .025
  plot and always wait
  * no plotting
  continue
```



Incompressible flow around a rotating stirring stick.

This example demonstrates how to make some grids move. The options for moving grids is a bit primitive so far. There is currently no way to have the flow accelerate the body, only predetermined motion is supported so far. **Warning:** moving grids have not been tested in 3D very much so avoid doing this for now.

2.5 Axisymetric incompressible flow past a sphere

The command file `OverBlown/ins/halfCylinder.cmd` can be used to compute the axisymmetric flow past a sphere. The (two-dimensional) grid can be created with `Overture/sampleGrids/halfCylinder.hdf`. **OverBlown** assumes that the axis of symmetry is the x-axis ($y = 0$).

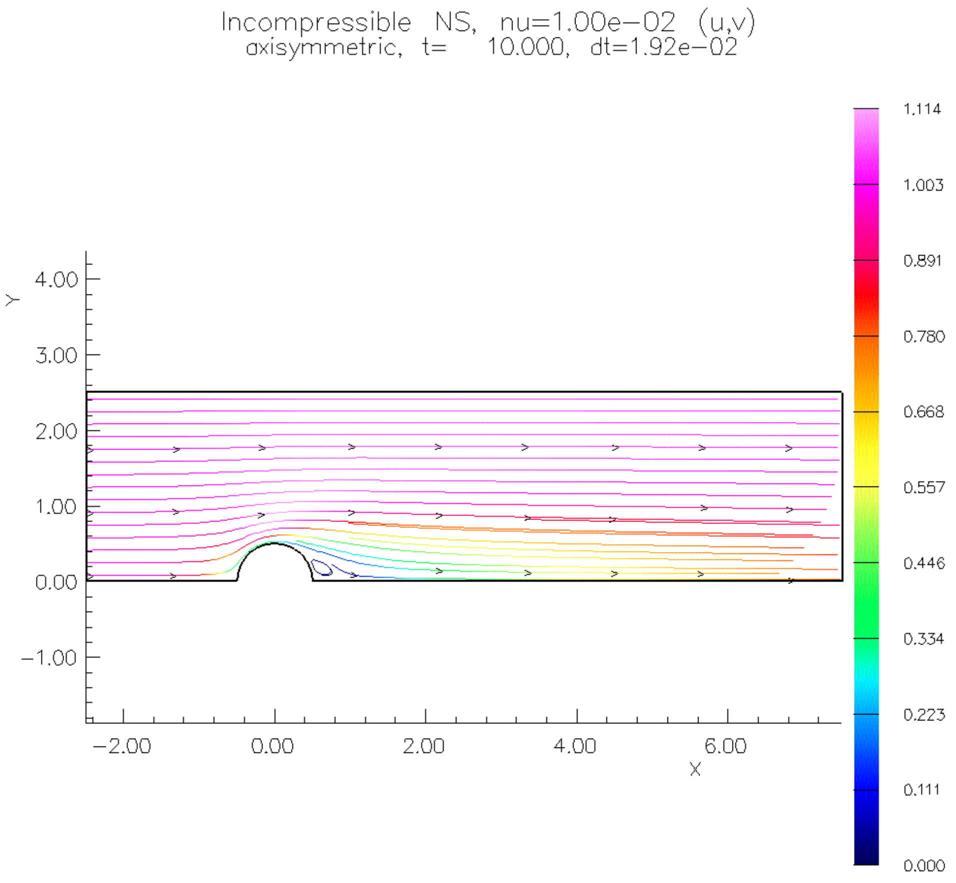


Figure 3: Incompressible axisymmetric flow past a sphere.

2.6 Incompressible flow past a backward facing step, a C-Grid and an H-grid

The command files `OverBlown/ins/backStep.cmd`, `OverBlown/ins/cgrid.cmd` and `OverBlown/ins/hgrid.cmd` can be used to solve the problems illustrated in this section. All of these grids use the *mixed boundary* condition feature, where portions of a physical boundary interpolate from another grid. The grids can be generated with the command files `Overture/sampleGrids/ins/backStep.cmd`, `Overture/sampleGrids/ins/cgrid.cmd` and `Overture/sampleGrids/ins/hgrid.cmd`.

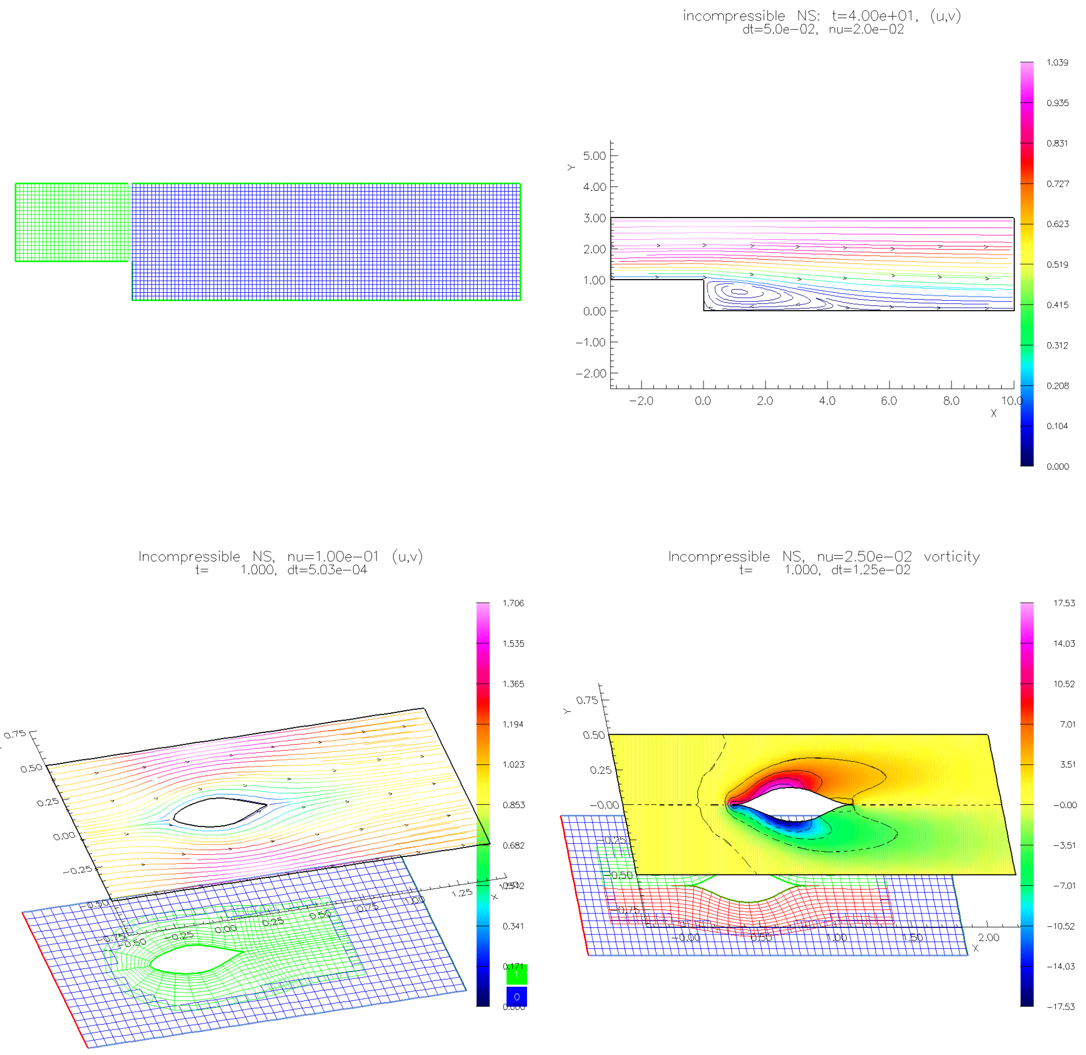
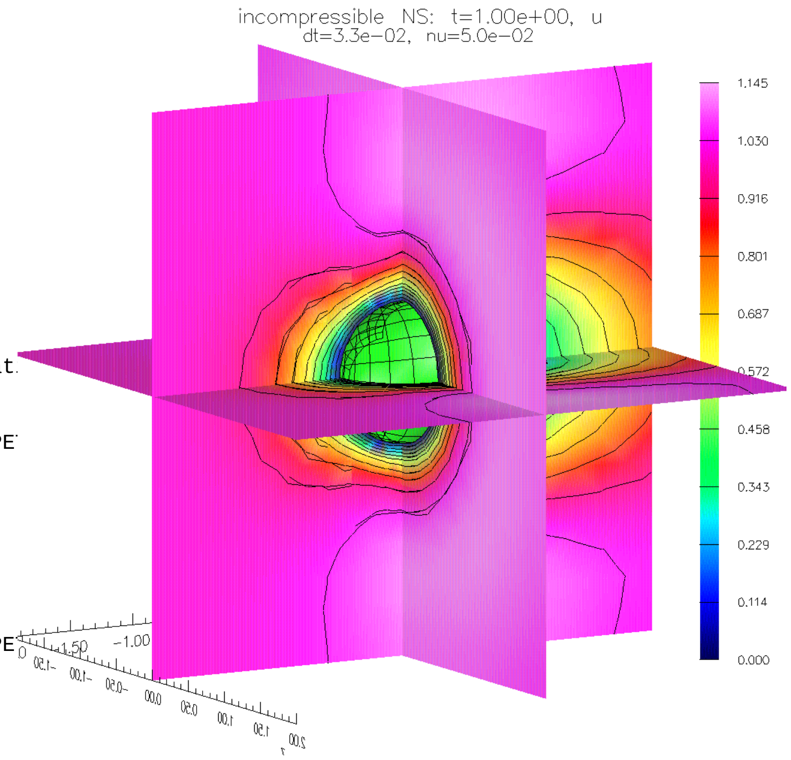


Figure 4: Incompressible flow past a backward facing step, a C-grid and an H-grid.

2.7 Incompressible flow past a sphere

The following command file can be used with **OverBlown** to compute the flow past a sphere in a box. This example uses the overlapping grid `Overture/sampleGrids/sib.hdf`.
File `OverBlown/ins/sib.cmd`.

```
*
* OverBlown command file for a sphere in a box
*
* grid name:
sib.hdf
  incompressible Navier Stokes
  exit
  turn off twilight zone
* implicit time stepping on viscous terms
  implicit
  show file options
    * compressed
    * open
    *  sib.show
    exit
*  but outer box is done explicitly
  choose grids for implicit
    all=implicit
    box=explicit
    done
  final time (tf=)
  .2
  times to plot (tp=)
  .1
   plot and always wait
  no plotting
  pde parameters
    nu
    .05
    done
  * use GMRES to solve the pressure equat
  pressure solver options
     choose best iterative solver
     * these tolerances are chosen for PE
     relative tolerance
       1.e-4
     absolute tolerance
       1.e-6
    exit
  implicit time step solver options
     choose best iterative solver
     * these tolerances are chosen for PE
     relative tolerance
       1.e-5
     absolute tolerance
       1.e-7
    exit
*
  project initial conditions
  initial conditions
    uniform flow
     u=1., p=1.
  exit
  boundary conditions
    all=slipWall
    box(0,0)=inflowWithVelocityGiven, uniform(p=1.,u=1.)
    box(1,0)=outflow
    north-pole=noSlipWall
    south-pole=noSlipWall
    done
  exit
  y+r:0 25
  x+r:0 25
```



Incompressible flow past a sphere.

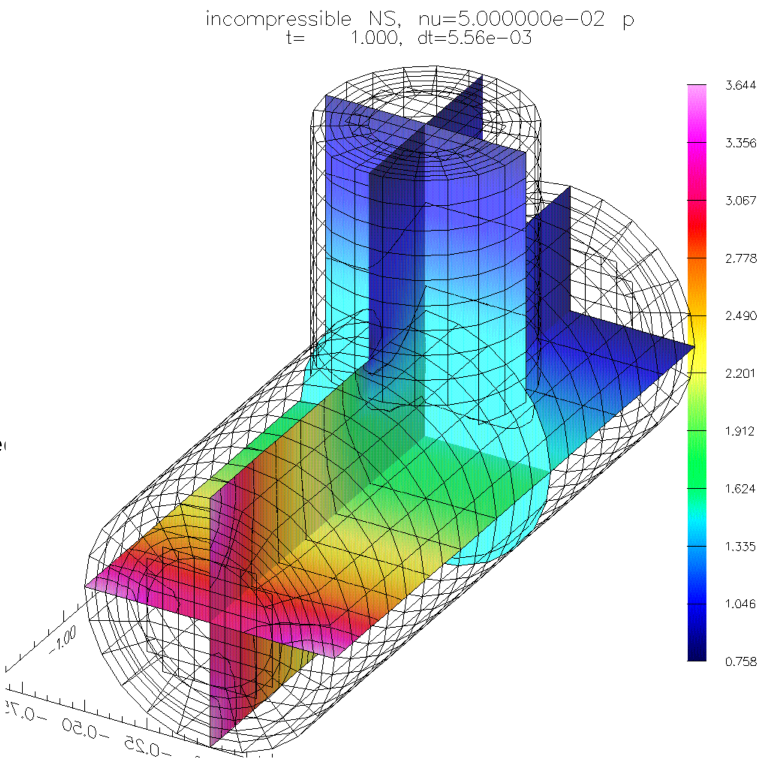
For 3D problems it is almost always necessary to use an iterative solver for the pressure equation and the implicit time stepping equations. In the above case we use the GMRES solver with a convergence tolerance of 10^{-3}. You may have to play around with this tolerance since I don’t have a good automatic way to do this yet. See the Oges documentation[9] for more information.

2.8 Incompressible flow through some intersecting pipes

The following command file can be used with **OverBlown** to compute the flow around a sphere in a box. This example uses the overlapping grid `Overture/sampleGrids/pipes.hdf` generated using the command file `Overture/-sampleGrids/pipes.cmd`.

File `OverBlown/ins/pipes.cmd`.

```
*
* OverBlown command file for flow in some pipes
*
* grid name:
pipes
* equations to solve:
  incompressible Navier Stokes
  exit
*
turn off twilight zone
project initial conditions
final time (tf=)
  1.
times to plot (tp=)
  .05
plot and always wait
* save the speed in the show file:
show file variables
  speed
done
pde parameters
  nu
  .05
  done
* use iterative solver for the pressure e
  pressure solver options
     choose best iterative solver
     relative tolerance
       1.e-6
     absolute tolerance
       1.e-7
  exit
initial conditions
  uniform flow
   u=0., p=1.
done
boundary conditions
  all=noSlipWall
  mainPipe(0,1)=inflowWithVelocityGiven, parabolic(d=.2,p=1.,u=1.)
  mainCore(0,0)=inflowWithVelocityGiven, parabolic(d=.2,p=1.,u=1.)
  mainPipe(1,1)=outflow
  mainCore(1,0)=outflow
  branchCore(1,1)=outflow
  branchPipe(1,1)=outflow
done
continue
* plot grids with wire frame
  grid
    plot shaded surfaces (3D) toggle 0
    exit this menu
  continue
```



Incompressible flow through some pipes.

This example demonstrates the use of the parabolic profile for the inflow boundary condition as described in section (3.5.1).

2.9 Two falling drops in an incompressible flow

The command file `OverBlown/ins/twoDrop.cmd` can be used with **OverBlown** to compute two drops falling in an incompressible flow. The grid can be created with `Overture/sampleGrids/twoDrop.cmd`. The initial conditions for the drops include their initial position, velocity, and angular velocity. The mass and moments of inertia must be specified for each drop. There can be problems for the grid generator if the drops get too close together since there will not be enough grid points in the gap between the drops. To avoid this problem there is an option "detect collisions" that has been turned on that will detect when the drops get close and perform an elastic collision. This collision detection currently only works for circular drops.

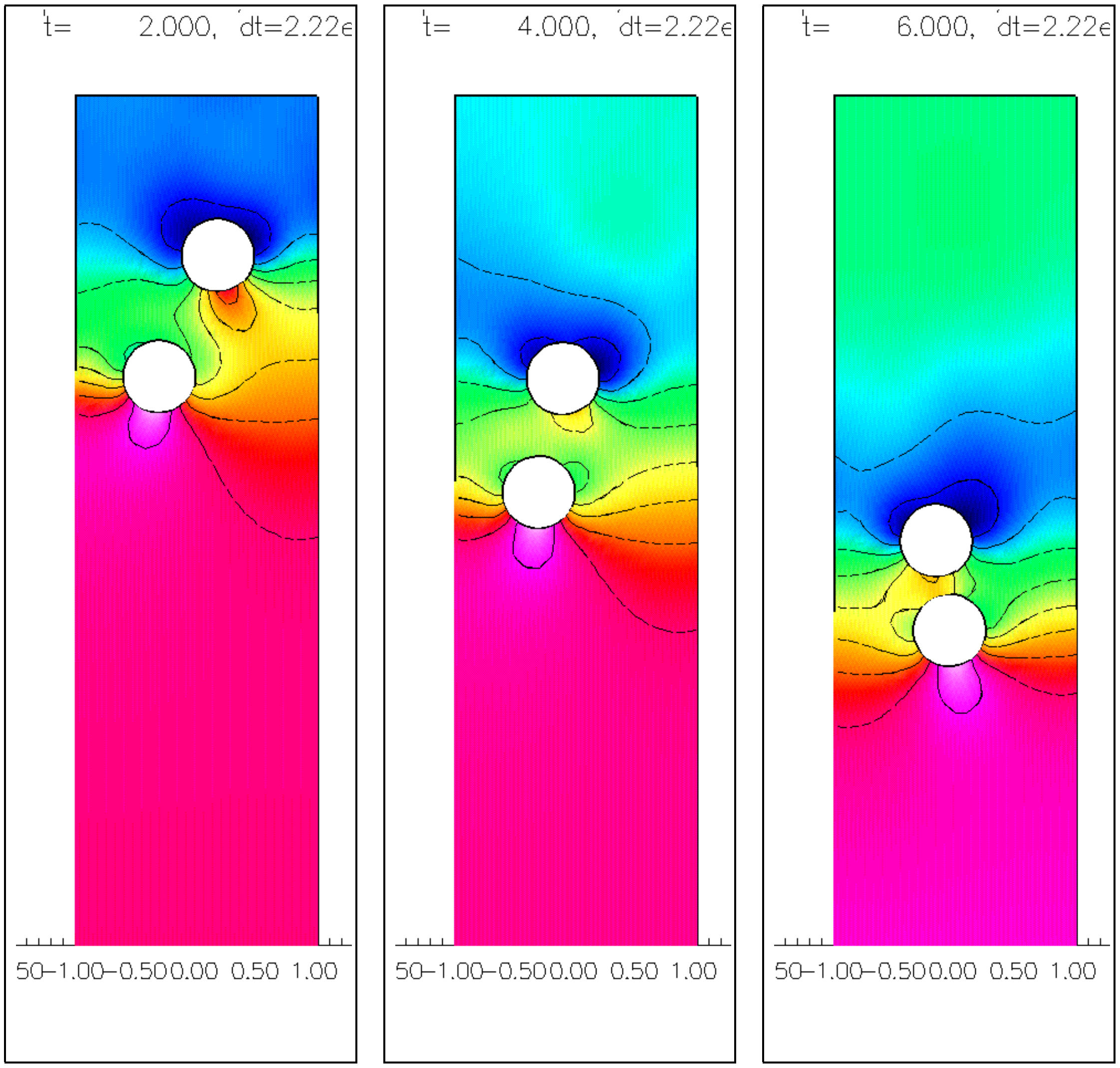


Figure 5: Two drops falling in an incompressible flow. The upper drop wants to "draft" in behind the lower drop where the pressure is lower.

2.10 Compressible flow past two offset cylinders

This example demonstrates the method CNSCAD, solution of the compressible Navier-Stokes equations using a conservative discretization with artificial diffusion.

The command file `OverBlown/cns/twoBump.cmd` can be used with **OverBlown** to compute the two-dimensional flow of a shock traveling past two offset cylinders. This example uses (a finer version) of the overlapping grid `Overture/-sampleGrids/twoBump.hdf` generated using the command file `Overture/sampleGrids/twoBump.cmd`. The coefficients of viscosity and heat conduction have been set to zero so that we are solving the inviscid Euler equations.

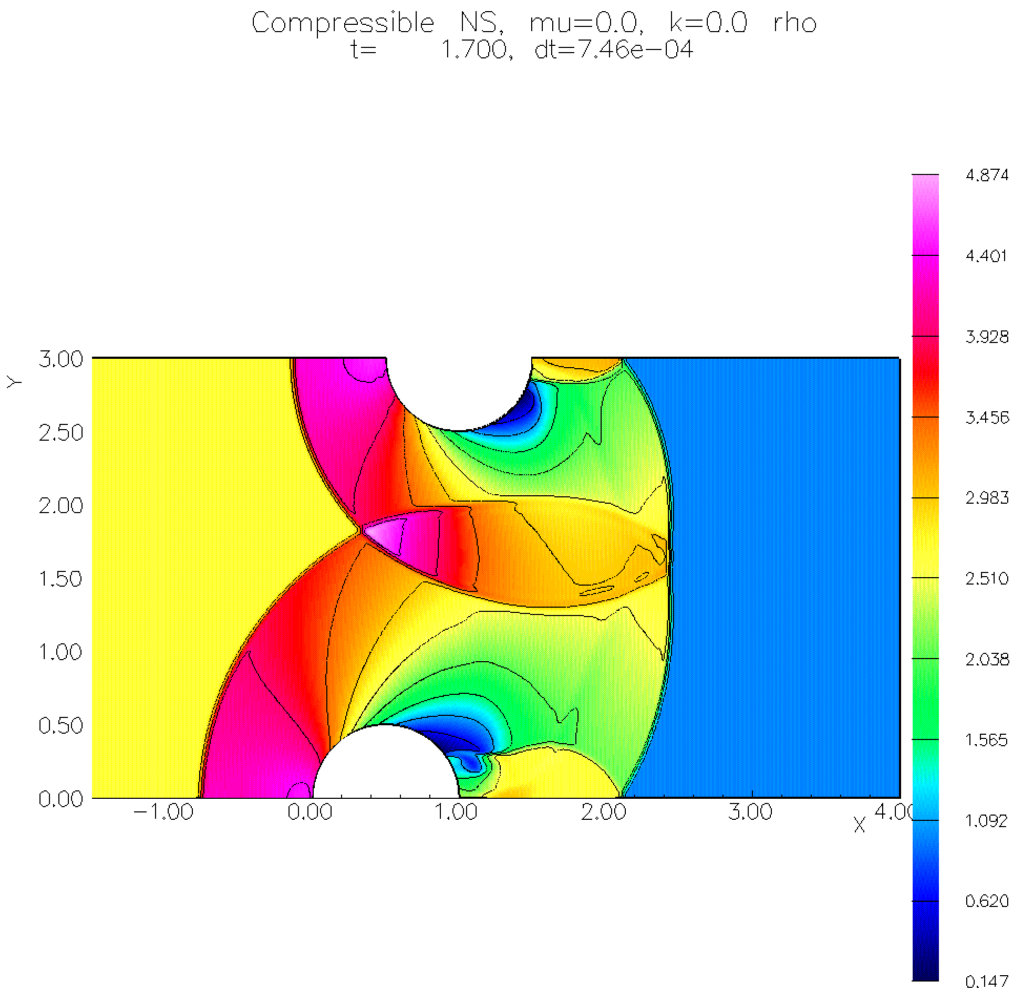


Figure 6: Solution of the compressible Euler equations: an initially plane shock, traveling from left to right, hits two offset cylinders.

2.11 Solving the Euler Equations with AMR

This example demonstrates the use adaptive mesh refinement with **OverBlown** using `OverBlown/cns/cicShockg.cmd`. We solve the compressible Euler equations with a conservative Godunov method (written by Don Schwendeman).

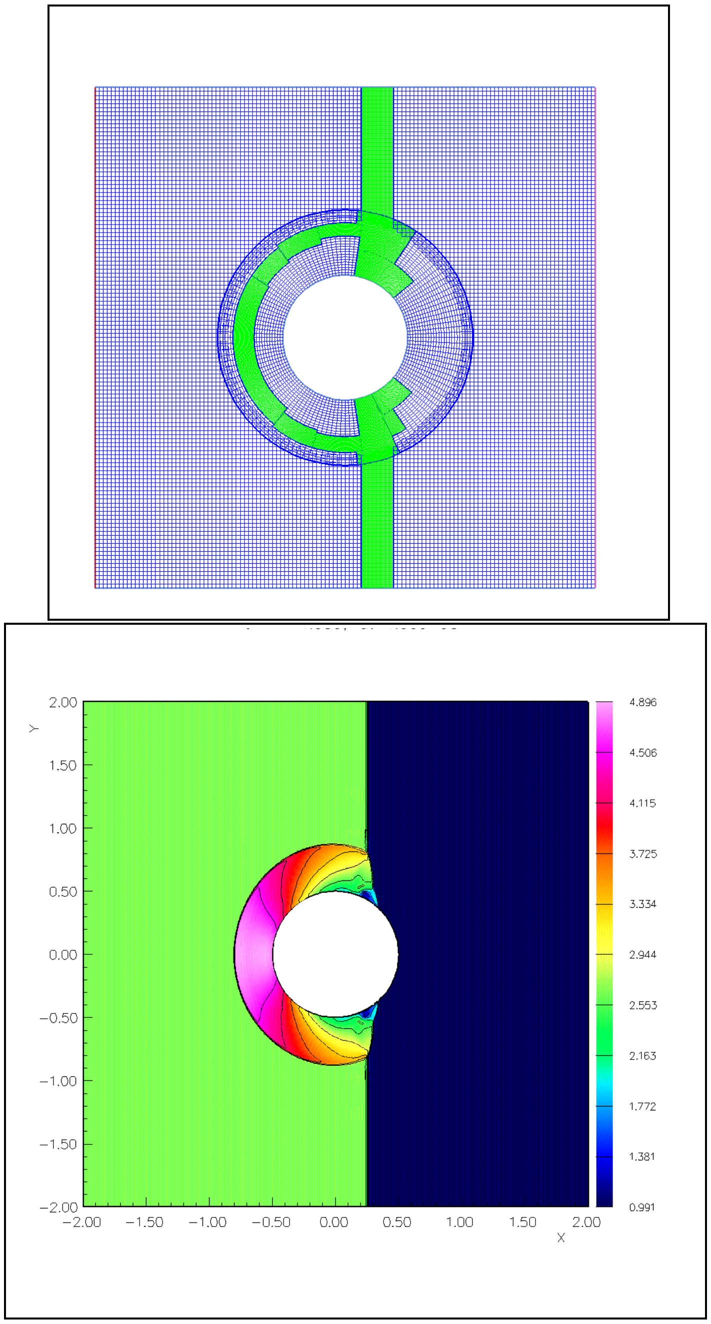
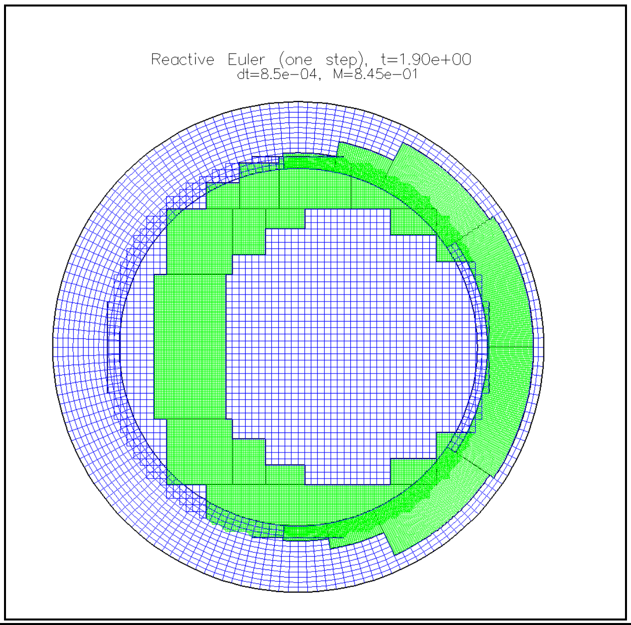


Figure 7: A shock hitting a cylinder. Adaptive mesh refinement is used to resolve the shock.

2.12 Solving the Reactive Euler Equations with AMR

In this example we solve the reactive Euler equations with AMR using `OverBlown/cns/circleDetonation.check.cmd`. The chemistry is defined by a simple one-step reaction. An initial temperature profile is generated using an option from the user defined initial conditions, file `UserDefinedInitialConditions4.C`. A detonation forms at the hot spot, expands and reflects off the boundaries.



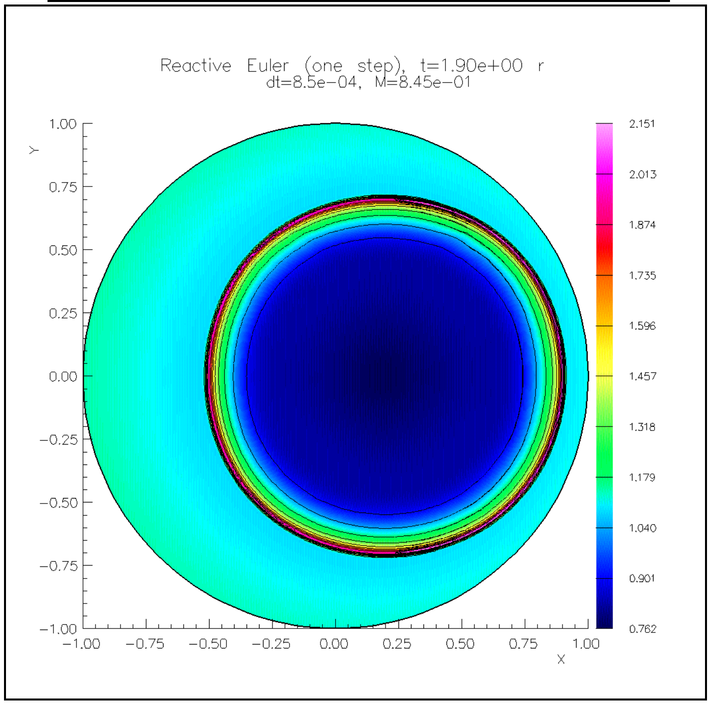


Figure 8: Solving the reactive Euler equations. Adaptive mesh refinement is used to resolve the detonation.

2.13 Low Mach number flow past an ellipse

This example demonstrates the method ASF, solution of the slightly compressible Navier-Stokes equations using an all-speed-flow algorithm.

This example uses the overlapping grid `Overture/sampleGrids/ellipse.hdf` generated using the command file `Overture/sampleGrids/ellipse.cmd`.
File `OverBlown/asf/ellipse.cmd`.

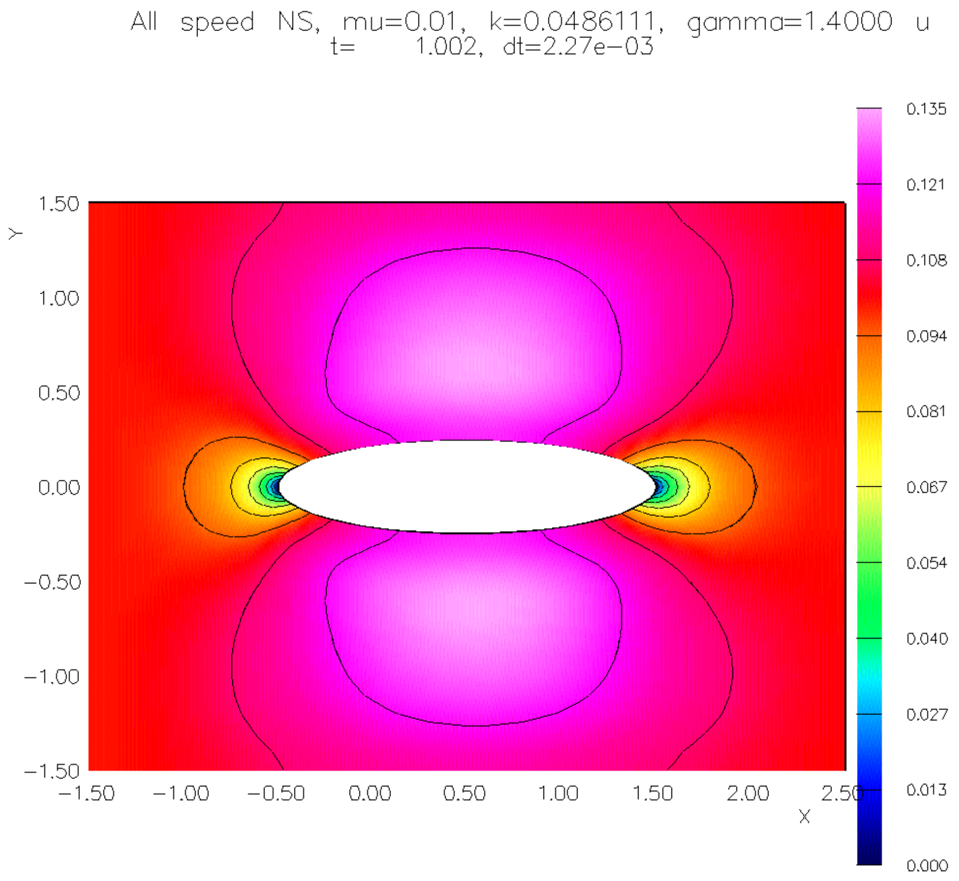


Figure 9: Solution of the slightly compressible Navier-Stokes equations: the Mach number at inflow is .1

```
*
* OverBlown command file
*
ellipse.hdf
*
  all speed Navier Stokes
  exit
*
  turn off twilight zone
  linearized all speed implicit
  final time (tf=)
    4.
  times to plot (tp=)
    .1
* Next specify the file to save the results in.
* This file can be viewed with Overture/bin/plotStuff.
  show file options
    * compressed
```

```
     open
      ellipse.show
     frequency to flush
       5
     exit
 *
   plot and always wait
   * no plotting
   pde parameters
     Mach number
      1.
     Reynolds number
      100.
     done
   cfl
     .5
   boundary conditions
     all=slipWall  uniform(T=1.)
     backGround(0,0)=subSonicInflow uniform(r=1.,u=.1,v=0.,T=1.)
     backGround(1,0)=subSonicOutflow uniform(T=1.)
     backGround(0,1)=slipWall
     backGround(1,1)=slipWall
     done
   initial conditions
     uniform flow
       r=1., u=.1, T=1.
   exit
   project initial conditions
   continue
```

Note that the parameters for this run were specified in terms in the (global) Reynolds number and the Mach number. With the global Mach number being M=1, then an inflow velocity of $(u, v) = (.1, 0)$ corresponds to a local inflow Mach number of $u/M = .1$.

2.14 Specifying the boundary conditions correctly

It can be confusing to get all the boundary conditions correct. To help you do this you should plot the grid and display the boundaries coloured by the boundary condition number (this is the default) as shown in figure (10). In 3D you will need to `'plot shaded surfaces'` to see the boundary colours. This will help you see if all the faces are correct. **OverBlown** prints out the number that corresponds to each boundary.

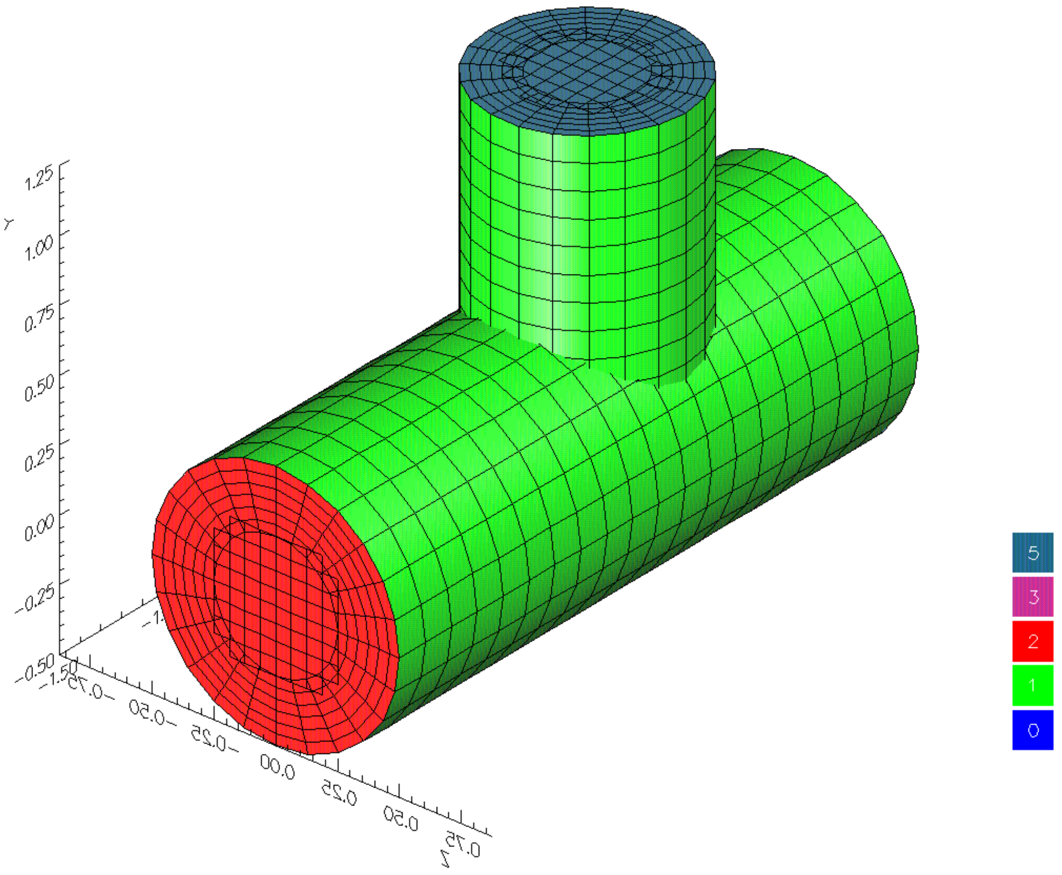


Figure 10: After specifying boundary conditions it is helpful to plot the grid with boundaries coloured by the boundary condition number. Here we see that the inflow boundary for the main pipe is number 2 (`inflowWithVelocityGiven`) the outflow boundary for the branch pipe is number 5 (`outflow`) and the walls are number 1 (`noSlipWall`). This figure is best seen in **colour**.

3 Options and Parameters

There are many options and parameters for **OverBlown** . Be warned that not all combinations of options will work. It is best to start from an existing command file and make make minor changes.

3.1 Setup menu

The setup popup menu appears after the `overBlown` is run and a grid is chosen. At this point one specifies which PDE to solve.

The choices are

incompressible Navier Stokes : solve the incompressible Navier-Stokes in 2D or 3D or 2D-axisymmetric.

all Speed Navier Stokes :

compressible Navier Stokes (Jameson) : use a Jameson style scheme for 2D/3D Euler/Navier-Stokes.

compressible Navier Stokes (Godunov I) : use a Godunov scheme for the Euler equations or reactive Euler equations.

Note that there will also be illusions to options for a **turbulence model** and or **reaction mechanism** but most of these are either un-implemented or unavailable for general use at the present time.

3.2 Parameters Dialog and Popup menu

After choosing the PDE to solve the user will be given the opportunity to change the parameters that define the problem.

At the current time there is both a dialog menu (new) and a popup menu (old). There are some options that appear in both menus. Eventually most of the popup menu should disappear. From the main *OverBlown Parameters* dialog window one can open other dialog windows such as the *Time Stepping Parameters* dialog. Some dialogs, such as the *Boundary Conditions* dialog are entitled *Under Construction*. In this case you should use the popup menu instead.

Here is a desciption of the menu options available for changing parameters. This main parameter menu appears when **OverBlown** is run and is found in the `OverBlown::setParametersInteractively()` function.

continue choose this item to exit this menu and continue on to the run-time dialog.

time stepping parameters... : open the time stepping parameters dialog

- **time stepping method** : Not all schemes work for all PDEs.
  - **forwardEuler** : For CNS Godunov
  - **adamsBashforth2** : For INS.
  - **adamsPredictorCorrector2** : For INS.
  - **variableTimeStepAdamsPredictorCorrector** for CNS.
  - **midpoint** : For INS.
  - **implicit** : For INS. Treat the viscous terms implicitly. One may optinally specify that some grids are integrated explicitly and some implicitly (see `choose grids for implicit`).
  - **all speed implicit** : For All-speed-flow.
  - **linearized all speed implicit** : For All-speed-flow. Linearize the implicit equations so that the implicit matrix is only formed and factored every ?? steps.
- **final time** : Integrate to this time.
- **cfl** : Set the `cfl` parameter. The maximum time step based on stability is scaled by this factor. By default `cfl=.9`.
- **dtMax** : Restrict the time step to be no larger than this value.
- **implicit factor** :This value in $[0., 1.]$ is used with the implicit time-stepping. A value of $.5$ will correspond to a 2nd-order Crank-Nicolson approach for the viscous terms, a value of $1.$ will be backward-Euler and a value of $0.$ will be forward-Euler. See the the reference manual for more details.
- **recompute dt every** : The time step, dt, is recomputed every time the solution is plotted/saved. In addition you may specify the maximum number of steps that will be taken before dt is recomputed. Use this if the solution is not plotted very often.

- **slow start time** : Ramp the time step Δt from a small value (determined by slow start cfl) to it maximum value (as determined by the `cfl` parameter over this time interval.
- **slow start cfl** : The initial time step for the slow start option is determined by this cfl value, default$=.25$..

pde options... open the pde options menu. The dialog which opens depends on which PDE was chosen and is described below.

initial conditions options... open the initial conditions dialog. This dialog is under construction.

- **read from a show file** : read initial conditions from a show file. This can either be a show file generated from OverBlown or one that you have built yourself.
- **read from a restart file** : read initial conditions from a restart file.
- **uniform state** : specify a uniform state.

show file options... open the showfile dialog.

- **show variables** : toggle on/off variables that should be saved in the show file.
- **mode** : specify the mode as compressed or uncompressed. A compressed file will be smaller (especially for AMR runs that create many grids) but a compressed file will not be readable by future versions of OverBlown.
- **open** : open a show file. You will be prompted for the name.
- **close** : close the show file.
- **frequency to save** : By default the solution is saved in the show file as often as it is plotted according to `'times to plot'`. To save the solution less often set this integer value to be greater than 1. A value of 2 for example will save solutions every 2nd time the solution is plotted.
- **frequency to flush** : Save this many solutions in each show file so that multiple show files will be created (these are automatically handled by plotStuff). See section (3.6.1) for why you might do this.

display parameters : print current values for parameters.

output options... open the output options dialog.

output options : Here are the output options.

- **plot option-menu** :
  - **plot and wait first time** :
  - **plot with no waiting** :
  - **plot and always wait** :
  - **no plotting** : do not plot. If you want to turn off all graphics you must choose this option and also run overBlown with the noplot option.
- **output periodically to a file** : output data to a file at each time step
- **times to plot** : Specify the time interval between plotting (and saving in a show file).
- **show file options...** open the show file options dialog.
- **save a restart file** : save or do not save a restart file.
- **allow user defined output** : call the userDefinedOutput routine at every step.
- **times to plot** : change the time interval between plotting (and output).
- **check file cutoffs** : used internally for regression tests.

boundary conditions... open the boundary condition options dialog. This dialog is under construction

twilight zone options... : open the twilight zone (method of analytic solutions) dialog.

- **type** : specify the type of analytic solution
  - **polynomial** :
    - **turn on polynomial** : Make the twilight-zone function be a polynomial.
    - **degree in space** : 0,1, or 2
    - **degree in time** : 0,1, or 2

trigonometric turn on trigonometric : Make the twilight-zone function be a trigonometric polymoinal.

frequencies : arguments to the trig functions are Π ? times the frequency specified here.

twilight zone flow : toggle on or off. When this option is on the equations are forced so that the true solution is equal to some analytically defined function. This is used to test the accuracy of the code.

use 2D function in 3D : use a 2D analytic function in 3D .

compare 3D run to 2D : make adjustments so that an extruded 3D geoemtry can be compared to a 3D computation.

degree in space : degree of the spatial polynomial

degree in time : degree of the temporal polynomial

frequencies (x,y,z,t) : frequencies to use with the trigonometric analytic solution.

plot the grid : plot the grid.

project initial conditions : (popup menu)

project initial conditions : Project initial conditions to nearly satisfy $\nabla \cdot \mathbf{u} = 0$. This option applies to INS and ASF.

do not project initial conditions : (popup menu)

time stepping options : Here are options that affect the time step.

choose grids for implicit : For use with the `implicit` time stepping option. Choose which grids to integrate implicitly and which to integrate explicitly. Normally one should choose thoses grids with fine grid spacing (such as in boundary layers) to be implicit while a back-ground grid could be explicit. See section (3.2.2).

boundary conditions : Brings up a new menu described in section (3.4).

data for boundary conditions : Brings up the sub-menu described in section (3.5).

initial conditions : Brings up a new menu described in section (3.2.3).

pde parameters : This brings up a new menu described below in section (3.2.4).

axisymmetric flow : solve an axisymmetric problem with cylindrical symmetry.

turn on axisymmetric flow : The solution is assumed to have cylindrical symmetry about the axis $y = 0$ with the grid defined only in the region $y \geq 0$.

turn off axisymmetric flow :

adaptive grids : use adaptive mesh refinement.

turn on adaptive grids

turn off adaptive grids

error threshold

truncation error coefficient :

order of AMR interpolation :

regrid frequency :

change adaptive grid parameters : change AMR regridding parameters (class Regrid).

change error estimator parameters : change parameters in the error estimator (class ErrorEstimator).

show amr error function : add the error function used for AMR regridding to the items that can be plotted.

Debugging :

debug file options : turn out various output to the debug file, ob.debug.

print solution/errors : print solution (or errors if known) at each time.

check error on ghost : also check errors of ghost points.

print classify array : print the classify array for sparse coefficient matrixes.

print sparse matrix : print the sparse matrix generated by Oges (big).

- **debug (debug=)** : This is a bit flag that turns on various messages. The more bits turned on, the more detailed the messages that appear. Thus a value of debug=3 (1+2) would have the first 2 bits turned on and would display few messages. A value of debug=63 (1+2+4+8+16+32) would have 6 bits turned on and would results in a lot of information.
- **Oges::debug (od=)** : bit flag debug variable for Oges.
- **Reactions::debug (rd=)** :
- **compare 3D run to 2D** : this option will adjust the equations and forcing so that a 3D run on an extruded 2D grid can be compared to the 2D computation. This includes setting the twilight-zone function to be 2D and changing the divergence damping (INS) to be two-dimensional (otherwise it is scaled in the wrong way).

reduce interpolation width : specify a new interpolation width. For example, when solving the inviscid Navier-Stokes equations one may want to use linear interpolation (width=2) instead of of quadratic interpolation (width=3) since this may reduce wiggles. If the grid was built with width=3 interpolation you can reduce the order of interpolation with this option.

sparse solver options :

- **pressure solver options** : Choosing this item will allow you to change any `Oges` related parameters as they apply to the elliptic equation for the pressure. See the `Oges` documentation for a description of these parameters [9].
- **implicit time step solver options** :Choosing this item will allow you to change any `Oges` related parameters as they apply to the mplicit time stepping equations. See the `Oges` documentation for a description of these parameters [9].

moving grids : Options related to moving grids.

- **turn on moving grids** : Allow grids to move.
- **turn off moving grids** : do not allow grids to move.
- **specify grids to move** : indicate which grids move and how. You must also choose `'turn on moving grids'` if you really want these grids to move. See section (**??**).
- **detect collisions** : detect collisions for some types of rigid bodies (wip)
- **do not detect collisions** : turn off collision detection.
- **minimum separation for collisions** : minimum allowed distance between colliding bodies. This distance is in grid lines and should be chosen large enough so that a valid grid can still be generated. Usually value will be from 2 to 3 but may need to be more for some grids.

plot the grid : plot the grid. Useful to see if boundary conditions have been plotted correctly.

erase : erase the graphics screen.

exit : exit this menu and continue on (same as 'continue').

3.2.1 Show file options

Here are the options related to show files, these options are from the `updateShowFile` function in the `OB_Parameters` class.

open : open a new show file.

close : close any open show file.

show file variables : specify extra derived quantities, such as the divergence or vorticity, that should be saved in the show file in addition to the standard variables.

frequency to save : By default the solution is saved in the show file as often as it is plotted according to `'times to plot'`. To save the solution less often set this integer value to be greater than 1. A value of 2 for example will save solutions every 2nd time the solution is plot.

frequency to flush : Save this many solutions in each show file so that multiple show files will be created (these are automatically handled by plotStuff). See section (3.6.1) for why you might do this.

properties :

- **uncompressed** : save the show file uncompressed. This is a more portable format that can be read by newer versions of Overture.
- **compressed** : save the show file compressed. This is a less portable format.

3.2.2 Choosing grids for implicit time stepping.

When the option `'choose grids for implicit'` is chosen from the main parameter menu one can specify which grids should be treated implicitly or explicitly with the **implicit** time stepping option. Type a line of the form

```
<grid name>=[explicit][implicit]
```

where `<grid name>` is the name of a grid or `'all'`. Type `'help'` to see the names. Examples:

```
square=explicit
all=implicit
cylinder=implicit
```

Type `'done'` when finished.

3.2.3 Initial condition options

Here are the options for specifying initial conditions. This menu appears when `'initial conditions'` is chosen from main parameter menu.

uniform flow : specify a uniform flow. Enter values in the form `'p=1., u=2., ...'`. Variables not specified will get default values (usually zero).

step function : specify two uniform conditions separted by a step

read from a show file : read the initial conditions from a solution in a show file.

read from a restart file : read the initial conditions from a solution in a restart file.

3.2.4 PDE parameters for INS

Here are the pde parameters that can be changed when solving the incompressible Navier-Stokes equations. This menu appears when `'pde parameters'` is chosen from main menu.

nu : kinematic viscosity (constant).

divergence damping

artificial diffusion : see section (3.8) for a description of the artificial diffusion terms.

- **second order artifical diffusion turn on second order artificial diffusion**
  - **turn off second order artificial diffusion**
  - **ad21 : coefficient of linear term**
  - **ad22 : coefficient of non-linear term**
- **fourth order artificial diffusion turn on fourth order artificial diffusion**
  - **turn off fourth order artificial diffusion**
  - **ad41 : coefficient of linear term**
  - **ad42 : coefficient of non-linear term**

3.2.5 PDE parameters for CNS

Here are the pde parameters that can be changed when solving the compressible Navier-Stokes equations. This menu appears when `'pde parameters'` is chosen from main menu and you are solving the compressible Navier-Stokes equations. Normally one would specify either the `Mach number` and `Reynolds number` or alternatively one could specify values for `mu`, and ...

Mach number : global Mach number.

Reynolds number : global Reynolds number.

mu : viscosity (currently constant)

Prandtl number :

kThermal : thermal conductivity (currently constant).

Rg : gas constant

gamma : ratio of specific heats.

gravity : a vector specifying the acceleration per unit mass due to gravity.

algorithms :

- **conservative with artificial dissipation** : Use conservative differencing with a Jameson style artificial dissipation that mixes a second-order and fourth order dissipation.
- **non-conservative** : use a centered non-conservative scheme, not recommended if you have un-resolved shocks.
- **conservative Godunov** : Use a conservative Godunov Scheme by Don Schwendeman

3.2.6 PDE parameters for ASF

Here are the pde parameters that can be changed when solving the all-speed flow version of the compressible Navier-Stokes equations. This menu appears when `'pde parameters'` is chosen from main menu and you are solving the `allSpeedNavierStokes`. Normally one would specify either the `Mach number` and `Reynolds number` or alternatively one could specify values for `mu`, and ...

Mach number : global Mach number.

Reynolds number : global Reynolds number.

mu : viscosity (currently constant)

Prandtl number :

kThermal : thermal conductivity (currently constant).

Rg : gas constant

gamma : ratio of specific heats.

gravity : a vector specifying the acceleration per unit mass due to gravity.

nuRho :

pressure level : the constant background level of the pressure, normally determined automatically from the Mach number.

remove fast pressure waves (toggle) : remove the p_{tt} term from the pressure equation to eliminate sound waves with a fast time scale.

3.3 Run time dialog

After the equations have been specified, parameters set and initial conditions chosen, the run time dialog window will appear, see figure(1.1). Note that **OverBlown** is in the process of converting from popup menus (left mouse button) to dialog windows so sometimes you will need to look for the command in the popup menu if it is not in the dialog.

plot component: choose the solution component to plot.

break : If running in movie mode this command will cause the program to halt at the next time to plot.

continue : compute the solution to the next time to plot.

movie mode : compute the solution to the final time without waiting. The solution will be plotted at each output time interval.

movie and save : movie mode plus save each frame as a ppm file.

contour : enter the contour plotting function in `PlotStuff`. Here you will more options to change the plot.

streamlines : enter the streamlines plotting function from `PlotStuff`.

grid : enter the grid plotting function from `PlotStuff`. If you don't first erase the contour plot then both the contours and the grid will be shown.

erase : erase the screen.

change the grid : add, remove or change existing grids. (poor man's adaptive mesh refinement).

adaptive grids... : open up a new dialog to show parameters adaptive grids.

> **use adaptive grids** : turn adaptive grids on or off.
>
> **error threshold** : specify the error threshold.

show file options... : choose show file options; e.g. open or close a show file.

file output... : specify options for saving solutions to an ascii file (for plotting with matlab for example). There are a number of options available as to what data should be saved. See also the userDefinedOutput routine where you can customize output.

> **output periodically to a file** : Open a file for output; specify how often to save data in the file (every step, every second step...); specify what data to save in the file (only grid 1, only values on some boundaries etc). Each time this menu item is selected a new file is opened, allowing one, for example, to save certain information every step and other information every tenth step.
>
> **close an output file** : Close a file opened by the command 'output periodically to a file'.
>
> **save a restart file** : save the current solution as a restart file; usually I just use the show file for restarts.

pde parameters... change PDE parameters at run time.

final time : change the value for the final time to integrate to.

times to plot : change the time interval between plotting (and output).

debug : enter an integer to turn on debugging info. This is a bit flag with debug=1 turning on just a bit of info, debug=3 (1+2) showing more, debug=7 (1+2+4) even more etc.

finish : do not compute any further, exit and save the show files etc.

Thus, for example, you can choose `'continue'` and the solution will be computed and plotted at the next time interval.

3.4 Boundary Conditions

In order to compute the correct flow the user must choose the correct boundary conditions. Each physical boundary of each grid must be given a boundary condition.

The names of the available boundary conditions are given in the `OB_Parameters::BoundaryCondition` enumerator:

```
enum BoundaryCondition
{
  interpolation=0,
  noSlipWall,
  inflowWithVelocityGiven,
  inflowWithPressureAndTangentialVelocityGiven,
  slipWall,
  outflow,
  superSonicInflow,
  superSonicOutflow,
  subSonicInflow,
  subSonicInflow2,
  subSonicOutflow,
  symmetry,
  dirichletBoundaryCondition,
  axisymmetric,
  convectiveOutflow,
  tractionFree,
  numberOfBCNames     // counts number of entries
};
```

Not all boundary conditions can be used with all PDEs. Boundary conditions are specified interactively (or in a command file) by choosing the `'boundary condition'` option from the main parameters menu and then typing a string that takes one of the following forms

<grid name>(side,axis)=< boundary condition name> [,option] [,option] ...

to change the boundary condition on a given side of a given grid, or

<grid name>=<boundary condition name> [,option] [,option] ...

to change all boundaries on a given grid, or

bcNumber<num>=<boundary condition name> [,option] [,option] ...

to change all boundaries that currently have a boundary condition value equal to the integer 'num'. Here <**grid name**> is the name of the grid, side=0,1 and axis=0,1,2. <**grid name**> can also be 'all'. The optional arguments specify data for the boundary conditions:

option = 'uniform(p=1.,u=1.,...)' : to specify a uniform inflow profile

option = 'parabolic(d=2,p=1.,...)' : to specify a parabolic inflow profile

option = 'jet(r=1.,x=0.,y=0,z=0.,d=.1,p=1.,u=$U_{\max}$,v=$V_{\max}$,...)' : specify a jet inflow profile.

option = 'pressure(.1*p+1.*p.n=0.)' : pressure boundary condition at outflow

option = 'oscillate(t0=.5,omega=1.,a0=.5,a1=.5,u0=0.,v0=0.,w0=0.)' : oscillating inflow parameters

option = 'ramp(ta=0.,tb=1.,ua=0.,ub=1.,...)' : ramped inflow parameters

option = 'userDefinedBoundaryData' : use a user defined boundary value option.

Examples:

square(0,0)=inflowWithVelocityGiven , uniform(p=1.,u=1.)
square(1,0)=outflow
annulus=noSlipWall
all=slipWall
bcNumber1=noSlipWall
square(0,1)=outflow , pressure(.1*p+1.*p.n=0.)
square(0,0)=inflowWithVelocityGiven , parabolic(d=.25,p=1.,u=1.) , oscillate(t0=0.,omega=1.,a0=.5,a1=.5)
square(0,0)=inflowWithVelocityGiven , userDefinedBoundaryData square(0,0)=inflowWithVelocityGiven ,
parabolic(d=.25,p=1.,u=1.) , userDefinedBoundaryData

The first example, `square(0,0)=inflowWithVelocityGiven`, will set the left edge of the square to be an inflow BC, while `square(1,0)=outflow` will set the right edge to be an outflow boundary. The line, `annulus=noSlipWall`, will set all physical boundaries of the annulus to be no-slip walls. Note that an annulus will normally have a branch cut and possibly an interpolation boundary. The boundary conditions on these non-physical boundaries are never changed. The command, `all=slipWall`, will make all physical boundaries slip-walls (and thus over-ride any previous changes to boundary conditions).

3.5 Data for Boundary Conditions

Some boundary conditions require 'data', such as an inflow boundary that requires values for certain quantities such as the velocity. These data values are optionally specified when the boundary condition is given. Here are some examples:

```
square(0,0)=inflowWithVelocityGiven , uniform(p=1.,u=1.)
square(0,1)=outflow , pressure(.1*p+1.*p.n=0.)
square(0,0)=inflowWithVelocityGiven , parabolic(d=.2,p=1.,u=1.), oscillate(t0=.3,omega=2.5)
```

The available options are

uniform(component=value [,component=value]...) Specify a uniform inflow profile and supply values for some of the components (components not specified will have a value of zero). Here `component0` is the name of a component such as ‘p’ or ‘u’.

parabolic([d=boundary layer width][,component=value]...) Specify a parabolic inflow profile with a given width. See section (3.5.1) for more details.

pressure(a*p+b*p.n=c) Specify the parameters a,b,c for a pressure outflow boundary condition. Here p=pressure and p.n=normal derivative of p. **Note that a and b should have the same sign or else the condition is unstable.**

oscillate([t0=value][,omega=value]) Specify parameters for an oscillating inflow boundary condition. See section (3.5.3) for more details.

ramp([ta=value][,tb=value][,...]) : specify values for a *ramped* inflow. See section (3.5.4).

userDefinedBoundaryData : choose from the currently available user defined options. See section (4) for how to define your own boundary conditions.

Note that not all options can be used with all boundary conditions.

3.5.1 Parabolic velocity profile

A ‘parabolic’ profile can be specified as a Dirichlet type boundary condition. The parabolic profile is zero at the boundary and increases to a specified value $U_{\max}$ at a distance d from the boundary:

$$u(\mathbf{x}) = \begin{cases} U_{\max}(2 - s/d)s/d & \text{if } s \le d \\ U_{\max} & \text{if } s > d \end{cases}$$

Here s is the shortest distance between the point $\mathbf{x}$ on the inflow face to the next nearest adjacent boundary. and d is the user specified *boundary layer width*. **OverBlown** is quite smart at correctly determining the distance s even if the inflow boundary is covered by one or more overlapping grids (such as the pipe flow example or inlet-outlet grid).

The parabolic profile can be useful, for example, in specifying the velocity profile at an inflow boundary that is adjacent to a no-slip wall. A uniform profile would have a discontinuity at the wall.

3.5.2 Jet velocity profile

The jet option is `'jet(r=1.,x=0.,y=0,z=0.,d=.1,p=1.,u=`$U_{\max}$`,v=`$V_{\max}$`,w=`$W_{\max}$`,...)'`.

A ‘jet’ profile can be used to define inflow over a portion of a boundary. The jet has a a center, (x_0, y_0, z_0), a radius r, and a maximum value of $U_{\max}$ for u (or $V_{\max}$ for v or $W_{\max}$ for w) at $r = 0$:

$$u(\mathbf{x}) = \begin{cases} U_{\max} & \text{if } |\mathbf{x} - \mathbf{x}_0| \le r \\ 0 & \text{if } |\mathbf{x} - \mathbf{x}_0| > r \end{cases}$$

In 3D the jet will have a cylindrical cross section. The jet can also be defined to go to zero at it’s boundary using the parameter d which defines the width of the transition layer,

$$u(\mathbf{x}) = \begin{cases} U_{\max} & \text{if } |\mathbf{x} - \mathbf{x}_0| \le r - d \\ U_{\max}[1 - (\xi/d)^2] & \text{if } r - d \le |\mathbf{x} - \mathbf{x}_0| < r \\ 0 & \text{if } |\mathbf{x} - \mathbf{x}_0| > r \end{cases}$$

Here $\xi = |\mathbf{x} - \mathbf{x}_0| - (r - d)$.

3.5.3 Oscillating values

An inflow boundary condition, `uniformInflow` or `parabolicInflow`, can be given an oscillating time dependence of the form

$$\{a_0 + a_1 \cos[2\pi\omega(t - t_0)]\} \times \{\text{uniform/parabolic profile}\} + \mathbf{u}_0$$

The parameters `omega,t0,a0,a1,u0,v0,w0` are specified with the `oscillate` option. Here $\mathbf{u}_0 = (u0, v0, w0)$.

3.5.4 Ramped Inflow

An inflow boundary condition can be ramped from one value (usually zero) to another value. The ramp function is a cubic polynomial on the interval (t_a, t_b). The polynomial is montone increasing on this interval with slope zero at the ends. The variables (u, v, w) vary from (u_a, v_a, w_a) to (u_b, v_b, w_b). Thus the u boundary condition ramp function would be:

$$u(t) = \begin{cases} u_a & \text{for } t \le t_a \\ (t-t_a)^2(-(t-t-a)/3 + (t_b-t_a)/2)6\frac{(u_b-u_a)}{(t_b-t_a)^3} + ua & \text{for } t_a < t < t_b \\ u_b & \text{for } t \ge t_b \end{cases}$$

The ramped inflow can also be combined with the parabolic profile as in

```
square(0,0)=inflowWithVelocityGiven , parabolic(d=.25,p=1.,u=1.) , ramp(ta=0.,tb=1.,ua=0.,ub=2
```

to give a ramped parabolic profile.

3.6 The show file

The 'show file' is a data base file of a particular format that contains the solutions from **OverBlown** . The post-processing routine `plotStuff` [10] knows how to read this file and find all the solutions and the different grids if the grids are moving or adaptive. The show can be looked at by typing `'plotStuff file.show'` or just `'plotStuff file'`, where `file.show` is the name that you gave to the show file when running **OverBlown** . The program `plotStuff` is found in `Overture/bin`.

3.6.1 Flushing the show file

It is not possible to look at results in a show file while the program is running and the show file is open and being written to. As a result, if the program crashes for some reason you will not be able to look at the results. To overcome this problem it is possible to automatically save multiple show files, with each show file containing one or more solutions. The number of solutions saved in each show file is determined by the frequency the show file is flushed. Use the `'frequency to flush the show file'` option to specify how many solutions should be saved in each show file. The files are named 'file.show', 'file.show1', 'file.show2' etc. where 'file.show' was the name given to the show file. The plotStuff program will automatically read all these different files if you just type 'plotStuff file.show'.

It is thus possible to look at the solutions when **OverBlown** crashes or while **OverBlown** is still running. Only the most recent solutions that belong to the most recent (open) show file will be unavailable.

3.7 Restarts

The easiest way to restart is to choose your initial conditions to come from the show file that you saved in a previous run, see section (3.2.3). The program will let one choose any solution in the show file as an initial condition. Remember to rename the show file from the previous run so that it doesn't get over-written before you have a chance to read from it.

You can also restart using an explicit **restart file**. To do this you need to turn on the saving of a restart file, see section (3). In this case **overBlown** will write a restart file every time the solution is output. Actually, to be safe, two files are created named 'ob1.restart' and 'ob2.restart'. This is in case the program crashes while the restart file is being written. Usually both files will be valid as use for restart files.

To **read the restart file** you simply specify this option and the file to use when assigning initial conditions, see section (3.2.3).

3.8 Artificial Diffusion

OverBlown implements artificial diffusions based either on a second-order undivided difference or a fourth-order undivided difference.

The artificial diffusions are

$$\mathbf{d}_{2,i} = (\texttt{ad21} + \texttt{ad22}|\nabla_h \mathbf{V}_i|_1) \sum_{m=1}^{n_d} \Delta_{m+}\Delta_{m-}\mathbf{V}_i \tag{1}$$

in the second-order case and

$$\mathbf{d}_{4,i} = -(\texttt{ad41} + \texttt{ad42}|\nabla_h \mathbf{V}_i|_1) \sum_{m=1}^{n_d} \Delta^2_{m+}\Delta^2_{m-}\mathbf{V}_i \tag{2}$$

in the fourth-order case. Here $|\nabla_h \mathbf{V}_i|_1$ is the magnitude of the gradient of the velocity and $\Delta_{m\pm}$ are the forward and backward undivided difference operators in direction m

$$\begin{array}{rcl}
|\nabla_h \mathbf{V}_i|_1 & = & n_d^{-2} \sum_{m=1}^{n_d} \sum_{n=1}^{n_d} |D_{m,h} V_{ni}| \\
\Delta_{1+} \mathbf{V}_i & = & \mathbf{V}_{i_1+1} - \mathbf{V}_i \\
\Delta_{1-} \mathbf{V}_i & = & \mathbf{V}_i - \mathbf{V}_{i_1+1} \\
\Delta_{2+} \mathbf{V}_i & = & \mathbf{V}_{i_2+1} - \mathbf{V}_i \\
\Delta_{2-} \mathbf{V}_i & = & \mathbf{V}_i - \mathbf{V}_{i_2+1} \quad \text{etc.}
\end{array}$$

The artificial diffusion is added to the momentum equations

$$\frac{d}{dt}\mathbf{V}_i + (\mathbf{V}_i \cdot \nabla_h)\mathbf{V}_i + \nabla_h P_i - \nu \Delta_h \mathbf{V}_i - \mathbf{f}(\mathbf{x}_i, t) - \mathbf{d}_{m,i} = 0$$

but does not change the pressure equation. Typical choices for the constants $\texttt{ad21} = \texttt{ad41} = 1$ and $\texttt{ad22} = \texttt{ad42} = 1.$. These artificial diffusions should not affect the order of accuracy of the method. With the artificial diffusion turned on to a sufficient degree, the real viscosity can be set at low as zero, $\texttt{nu} = 0$.

This form of the artificial diffusion is based on a theoretical result [13][14] that states that the minimum scale, $\lambda_{\min}$, of solutions to the incompressible Navier-Stokes equations is proportional to the square root of the kinematic viscosity divided by the square root of the maximum velocity gradient:

$$\lambda_{\min} \propto \sqrt{\frac{\nu}{|\nabla \mathbf{u}| + c}} .$$

This result is valid locally in space so that $|\nabla \mathbf{u}|$ measures the local value of the velocity gradient. The minimum scale measures the size of the smallest eddy or width of the sharpest shear layer as a function of the viscosity and the size of the gradients of $\mathbf{u}$. Scales smaller than the minimum scale are in the exponentially small part of the spectrum.

This result can be used to tell us the smallest value that we can choose for the (artificial) viscosity, ν_A, and still obtain a reasonable numerical solution. We require that the artificial viscosity be large enough so that the smallest (but still significant) features of the flow are resolved on the given mesh. If the local grid spacing is h, then we need

$$h \propto \sqrt{\frac{\nu_A}{|\nabla \mathbf{u}| + c}} .$$

This gives

$$\nu_A = (c_1 + c_2|\nabla \mathbf{u}|)h^2$$

and thus we can choose an artificial diffusion of

$$(c_1 + c_2|\nabla \mathbf{u}|)h^2 \Delta \mathbf{u}$$

which is just the form (1).

In the fourth-order accurate case we wish to add an artificial diffusion of the form

$$-\nu_A \Delta^2 \mathbf{u}$$

since, as we will see, this will lead to $\nu_A \propto h^4$. In this case, if we consider solutions to the incompressible Navier-Stokes equations with the diffusion term $\nu \Delta \mathbf{u}$ replaced by $-\nu_A \Delta^2 \mathbf{u}$ then the minimum scale would be

$$\lambda_{\min} \propto \left(\frac{\nu_A}{|\nabla \mathbf{u}|} \right)^{1/4}$$

Following the previous argument leads us to choose an artificial diffusion of the form

$$-(c_1 + c_2|\nabla \mathbf{u}|)h^4 \Delta^2 \mathbf{u}$$

which is just like (2).

4 User defined functions

Here is a list of functions that can be changed by a user. After rewriting any of these files, compile and link OverBlown with the new file.

4.1 User defined initial conditions

The function **userDefinedInitialConditions** defines initial conditions. Rewrite this function (userDefinedInitialConditions.C) to define arbitrary initial conditions. This function is accessed when interactively setting parameters in the `'initial conditions'` menu under `'user defined'`.

4.2 User defined boundary conditions

The functions **chooseUserDefinedBoundaryValues, userDefinedBoundaryValues** define values for boundary conditions. Change these functions in order to define new right-hand-side values for boundary conditions. For example, you may want to define the inflow velocity profile to have a certain shape and/or time dependence. The `chooseUserDefinedBoundaryValues` is accessed when you specify boundary conditions and choose the `userDefinedBoundaryData` option.

4.3 User defined grid motion

to appear...

5 Hints for running OverBlown

- Start out with a simple problem on a coarse grid so that the problem can be quickly run to determine if you have the boundary conditions correct etc.
- Start out by taking only a few time steps and looking at the solution to see if it looks correct.
- The rule of thumb for choosing the viscosity ν is that if the velocities are order 1 and the domain is order 1 then $\nu > h_{\max}^2$, where $h_{\max}$ is the maximum grid spacing as reported by **OverBlown** when it runs. This comes from the minimum scale result as discussed in section (3.8).
- If you want to use as small a viscosity as possible then set $\nu = 0$ and use artificial viscosity as discussed in sections (3.2.4,3.8).
- If OverBlown blows up it could be the time step is not computed correctly. Reduce the cfl parameter (default is .9) to a value like .5 or .25 to see if this is the problem. There are known problems with the time step determination for implicit time stepping and a large viscosity (relative to the grid spacing).

6 Post-processing: Reading a show file and computing some Aerodynamic Quantities

The program `OverBlown/bin/aero.C` shows how to read a show file that has been generated by OverBlown and access the solution values stored there. This program can then be used to plot the pressure coefficient on the surface of a body and to compute the lift and drag on a body.

The file `OverBlown/bin/aero.C` can be altered to compute other quantities that may be of particular interest to your application. All information about the grid, solutions and OverBlown parameters are accessible from the show file. You could, for example use this program to output the solution values to a data file format suitable for some other plotting or analysis program.

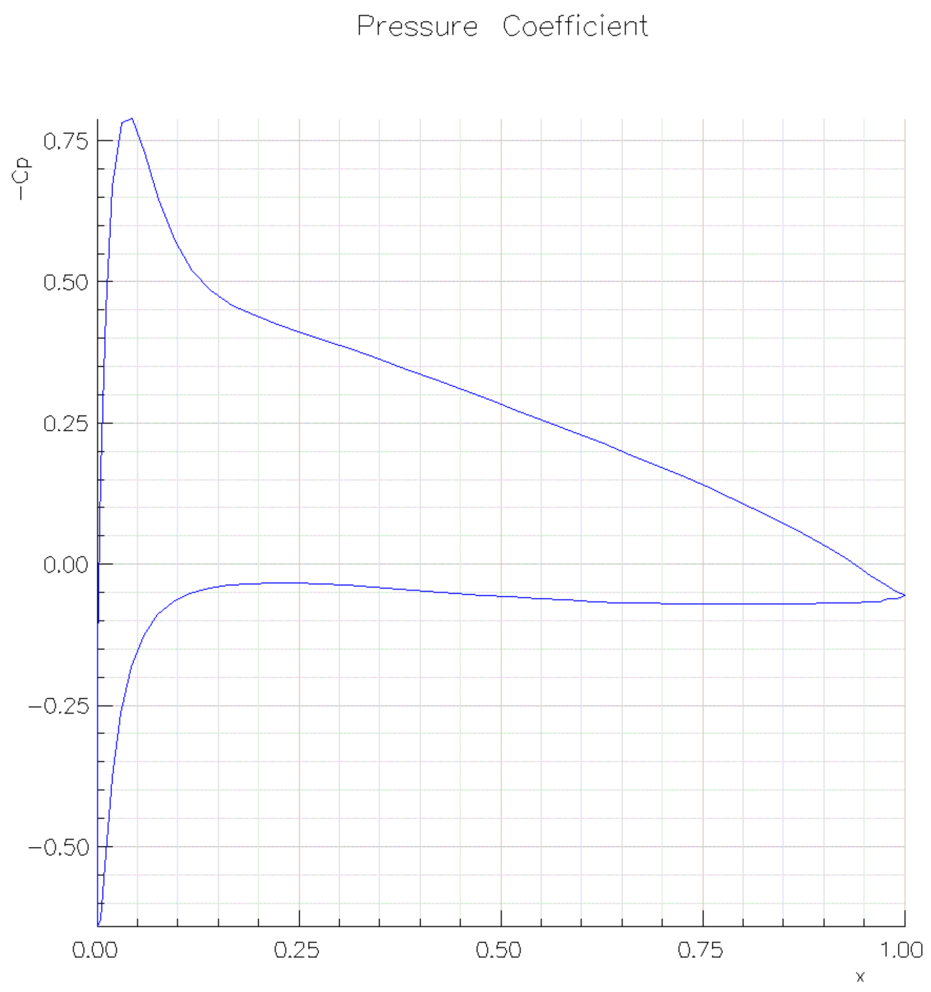


Figure 11: The areo.C program can be used to read a show file generated by OverBlown and compute the coefficient of pressure on the surface of a body.

7 Installing OverBlown

OverBlown is usually distributed as a gzipped tar file, `OverBlown.vXX.tar,gz`, where `XX` will be the version number. After uncompressing and untar'ing this file you will see that the **OverBlown** distribution consists of a directory, `OverBlown`, plus subdirectories:

`OverBlown/bin` : contains the executable, **overBlown** . You may want to put this directory in your path.

`OverBlown/ins` : sample command files for running computations of the incompressible Navier-Stokes equations, see section (2).

`OverBlown/cns` : sample command files for running computations of the compressible Navier-Stokes equations.

`OverBlown/asf` : sample command files for running computations with the all-speed-flow solver for the compressible Navier-Stokes equations.

`OverBlown/lib` : contains the **OverBlown** library, `libOverBlown.a`.

`OverBlown/src` : source files (.C files) for OverBlown.

`OverBlown/check` : contains testing routines for comparing the answers on test problems to previously run cases.

Here are the steps for installing **OverBlown** :

1. `cd OverBlown`
2. `configure` : run configure with the default options.
3. `make` : compile **OverBlown** .

To run the regression tests:

1. `cd OverBlown/check`
2. `make`
3. `check.p` : a perl script that will run **OverBlown** on a number of scripts.

To run an example:

1. `cd OverBlown/ins`
2. `../bin/overBlown cylinder.cmd` : compute flow past a cylinder.

Notes: To build **OverBlown** first run the **OverBlown** configure script (perl script) to create the Makefile's for the machine you are running on. For example type `'configure'` from the **OverBlown** directory to build Makefile's for OverBlown in double precision. Some options, such as 'precision=single', must match the corresponding options that were used to configure **Overture** . Type `'configure --help'` with no arguments to see all the options. The configure script will check to see that you have set up the appropriate environmental variables for **Overture** and **OverBlown** . See the `Overture/README` and `OverBlown/README` files for more info. Note that **OverBlown** uses the LAPACK libraries in addition to the libraries required by **Overture** . Once the Makefile's have been created, just type `'make'`. If the make is successful then an executable will be built, `OverBlown/bin/overBlown`.

7.1 Using PETSc and OverBlown

PETSc, the Portable Extensible Toolkit for Scientific computations[1], can be used in OverBlown to solve implicit systems.

To use PETSc you should

1. build or locate a version of PETSc. I have only built and linked OverBlown to the non-parallel version of PETSc using PETSc's internal replacement for mpi.
2. define the PETSC_LIB environmental variable (as required to use PETSc normally) and add it to your LD_LIBRARY_PATH.
3. configure OverBlown with the 'petsc' option as in 'configure petsc', and make OverBlown (only the bin directory needs to be rebuilt if you have already complied OverBlown without PETSc).

References

[1] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *PETSc 2.0 users manual*, Tech. Rep. ANL-95/11 - Revision 2.0.24, Argonee Naitional Laboratory, 1999.

[2] D. L. Brown, G. S. Chesshire, W. D. Henshaw, and D. J. Quinlan, *Overture: An object oriented software system for solving partial differential equations in serial and parallel environments*, in Proceedings of the Eight SIAM Conference on Parallel Processing for Scientific Computing, 1997.

[3] D. L. Brown, W. D. Henshaw, and D. J. Quinlan, *Overture: An object oriented framework for solving partial differential equations*, in Scientific Computing in Object-Oriented Parallel Environments, Springer Lecture Notes in Computer Science, **1343**, 1997.

[4] G. Chesshire and W. Henshaw, *Composite overlapping meshes for the solution of partial differential equations*, J. Comp. Phys., 90 (1990), pp. 1–64.

[5] W. Henshaw, *A fourth-order accurate method for the incompressible Navier-Stokes equations on overlapping grids*, J. Comp. Phys., 113 (1994), pp. 13–25.

[6] ——, *Overture: An object-oriented system for solving PDEs in moving geometries on overlapping grids*, in First AFOSR Conference on Dynamic Motion CFD, June 1996, L. Sakell and D. Knight, eds., 1996, pp. 281–290.

[7] ——, *Mappings for Overture, a description of the Mapping class and documentation for many useful Mappings*, Research Report UCRL-MA-132239, Lawrence Livermore National Laboratory, 1998.

[8] ——, *Ogen: An overlapping grid generator for Overture*, Research Report UCRL-MA-132237, Lawrence Livermore National Laboratory, 1998.

[9] ——, *Oges user guide, a solver for steady state boundary value problems on overlapping grids*, Research Report UCRL-MA-132234, Lawrence Livermore National Laboratory, 1998.

[10] ——, *Plotstuff: A class for plotting stuff from Overture*, Research Report UCRL-MA-132238, Lawrence Livermore National Laboratory, 1998.

[11] ——, *OverBlown: A fluid flow solver for overlapping grids, reference guide*, Research Report UCRL-MA-134289, Lawrence Livermore National Laboratory, 1999.

[12] W. Henshaw and H.-O. Kreiss, *Analysis of a difference approximation for the incompressible Navier-Stokes equations*, Research Report LA-UR-95-3536, Los Alamos National Laboratory, 1995.

[13] W. Henshaw, H.-O. Kreiss, and L. Reyna, *On the smallest scale for the incompressible Navier-Stokes equations*, Theoretical and Computational Fluid Dynamics, 1 (1989), pp. 1–32.

[14] ——, *Smallest scale estimates for the incompressible Navier-Stokes equations*, Arch. Rational Mech. Anal., 112 (1990), pp. 21–44.

Index