Cgins User Guide: An Overture Solver for the Incompressible Navier–Stokes Equations on Composite Overlapping Grids,

William D. Henshaw[1]
Centre for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA, 94551.

June 9, 2014

LLNL-SM-455851

Abstract: This is the user guide for Cgins. Cgins is a program that can be used to solve incompressible fluid flow problems in complex geometry in two and three dimensions using composite overlapping grids. It is built upon the **Overture** object-oriented framework. Cgins solves the incompressible Navier-Stokes equations using a pressure-Poisson formulation. Second-order accurate and fourth-order accurate approximations are available. Cgins can be used to

- solve problems in two and three dimensional complex domains,
- solve problems on moving grids (specified motion and rigid body motion),
- solve temperature dependent flows with buoyancy using the Boussinesq approximation,
- solve axisymmetric flows.

This user guide describes how to get started and how to run Cgins. Various examples are given. The options for running the code along with specification of initial conditions and boundary conditions are described.

[1]This work was performed under the auspices of the U.S. Department of Energy (DOE) by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 and by DOE contracts from the ASCR Applied Math Program.

Contents

1 Introduction

Cgins[2] is an incompressible fluid flow solver for overlapping grids built upon the **Overture** framework [**?**],[**?**],[**?**]. More information about **Overture** can be found on the **Overture** home page, `http://www.llnl.gov/casc/Overture`.

Cgins solves the incompressible Navier-Stokes equations using a pressure-Poisson formulation [**?**]. Second-order accurate and fourth-order accurate approximations are available. Cgins can be used to

- solve problems in two and three dimensional complex domains,
- solve problems on moving grids (specified motion and rigid body motion),
- solve temperature dependent flows with buoyancy using the Boussinesq approximation,
- solve axisymmetric flows.

This user guide describes how to get started and how to run Cgins. Various examples are given. The options for running the code along with specification of initial conditions and boundary conditions are described.

The Cgins solver is found in the `ins` directory in the **cg** distribution and has sub-directories

`bin` : contains the executable, cgins. You may want to put this directory in your path.

`check` : contains regression tests.

`cmd` : sample command files for running cgins, see section (2).

`lib` : contains the Cgins library, `libCgins.so`.

`src` : source files

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

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

1.1 Basic steps

Here are the basic steps to solve a problem with Cgins.

1. Generate an overlapping grid with ogen. Make the grid with 2 ghost lines (this is the default).
2. Run cgins (note lowercase 'c', found in the `bin/cgins` 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.

[2]Thanks to Kyle Chand for all his contributions to Cgins including the development of the AFS algorithm.

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 cgins can be saved in a command file (by default they are saved in the file 'cgins.cmd'). This command file can be used to re-run the same problem by typing 'cgins 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 cgins, these are explained next in section (2). Papers that describe some of the algorithms used in Cgins include

1. *A Fourth-Order Accurate Method for the Incompressible Navier-Stokes Equations on Overlapping Grids* [**?**],

2. *A Split-Step Scheme for the Incompressible Navier-Stokes Equations* [**?**],

3. *Composite Overlapping Meshes for the Solution of Partial Differential Equations* [**?**],

4. *Analysis of a Difference Approximation for the Incompressible Navier-Stokes Equations* [**?**].

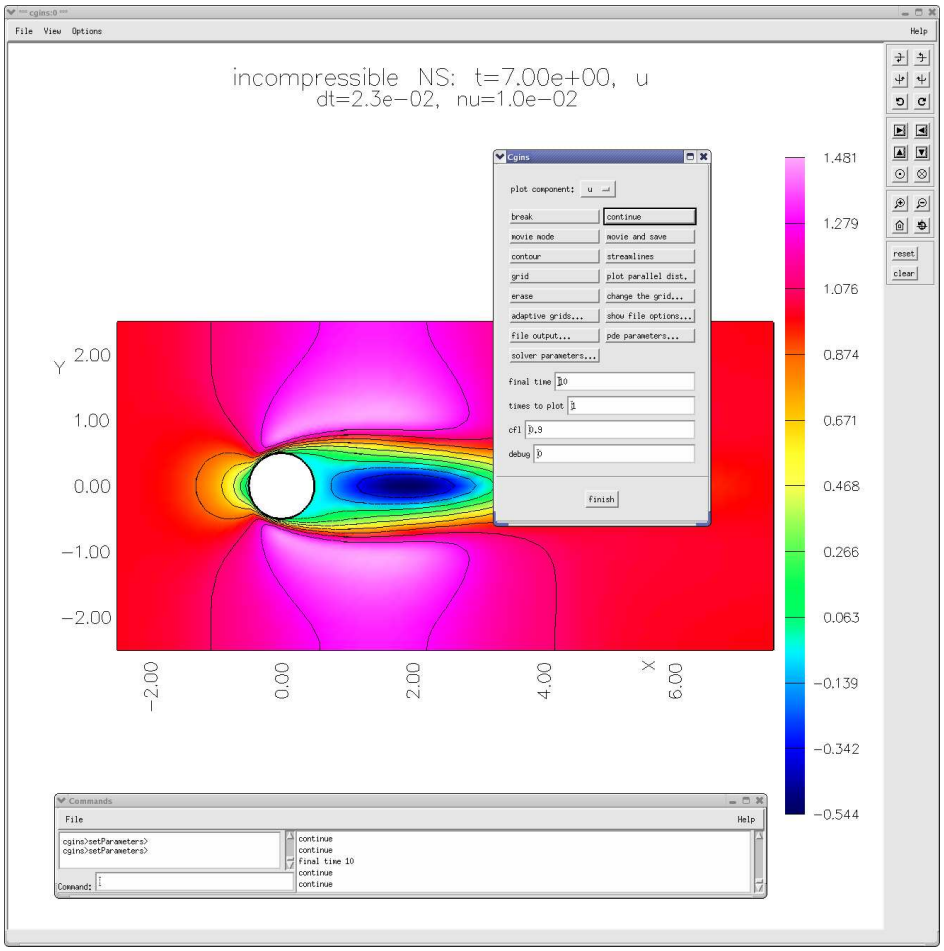

Figure 1: Snapshot of cgins showing the run time dialog menu.

2 Sample command files for running cgins

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 cgins. See the file `cg/ins/cmd/Readme` for a list and brief description of the command files found in `cg/ins/cmd`.

2.1 Running a command file

Given a command file for cgins such as `cylinder.cmd`, found in `cmd/cylinder.cmd`, one can type '`cgins cylinder.cmd`' to run this command file . You can also just type '`cgins cylinder`, leaving off the `.cmd` suffix. Typing '`cgins noplot cylinder`' will run without interactive graphics (unless the command file turns on graphics). Note that here it is assumed that the `bin` directory is in your path so that the `cgins` command is found when you type it's name. The Cgins 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, see Figure 1.1. The program will also print out some information about the problem being solved. Many of the command files such as the stirring-stick problem, `stir.cmd`, can take command line arguments. For example, here are two command lines that run a problem with different grids and parameters:

```
cgins stir -g=stir.hdf -nu=.05 -tf=1. -tp=.025
cgins stir -g=stir2.hdf -nu=.01 -tf=1. -tp=.002 -rate=8.
```

See the comments at the top of `stir.cmd` for further explanation and examples.

2.2 Incompressible flow past a cylinder in a long channel

The command file `cg/ins/cmd/cylinder.cmd` can be used to compute the incompressible flow past a cylinder in a channel, see Figure 2. This example uses the grid `Overture/sampleGrids/cilc.hdf`.

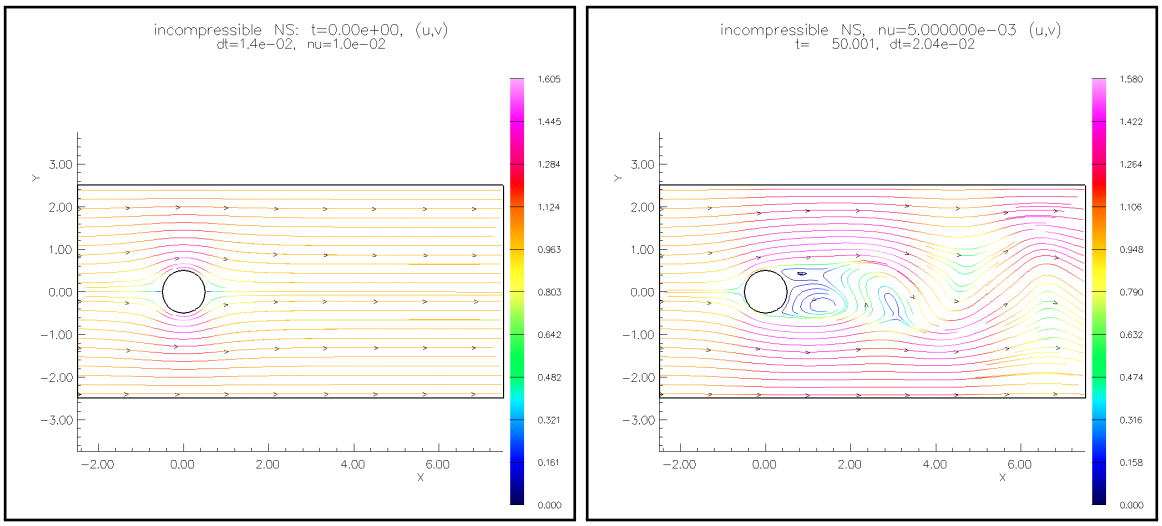

Figure 2: Incompressible flow around a cylinder. Left: the initial conditions are obtained by projecting a uniform flow, $(u, v) = (1, 0)$. Right: the solution at time $t = 40$.

To run cgins with the `cylinder.cmd` script, type

```
cgins $ins/cmd/cylinder.cmd
```

Then choose one of:

`continue` : to advance to the next output time.
`plot component` : to plot different solution components.
`streamlines` : to plot streamlines (choose `erase first`).
`grid` : to plot the grid (choose `erase first`).
`movie mode` : to run and plot.
`break` : to break from movie mode.
`final time 50.` : to increase the final time.

Further description of the options available in the run time dialog can be found in Section **??**.

Note: If cgins cannot find the cilc.hdf grid file, you can generate it with:

```
ogen -noplot $sampleGrids/cilc.cmd
```

Here are some more examples using the command line arguments to choose different options:

`cgins cylinder -g=cilc.hdf -tf=50. -tp=1.` : specify the grid, final time and time to plot.
`cgins cylinder -g=cilce2.order4 -tf=50. -tp=.1 -nu=.01` : solve a fourth-order accurate problem [3].

Notes:

- If the overlapping grid file, `cilc.hdf` is not found then you may need to specify the full path to its location, `-g=/home/henshaw/myGrids/cilc.hdf`. The suffix `'.hdf'` is optional when specifying grids as Overture will tack on `'.hdf'` if necessary. If Cgins 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 (see [**?**]).
- 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 implicit equations and elliptic pressure equation are 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 examples it will be shown how to specify an iterative method.

[3] By default, Cgins determines the order of accuracy from the grid: the order of accuracy of the grid is specified when the grid is constructed using ogen. A fourth-order accurate grid will have two layers of interpolation points (to support a 5 point stencil) and use high-order accurate interpolation with a interpolation stencil width of 5 in each direction.

2.3 Flow past two cylinders using different time-stepping methods

In this section we compute the flow past two cylinders in order to demonstrate the use of some different time-stepping methods for Cgins. These time-stepping methods include the explicit predictor-corrector (PC), the implicit predictor-corrector (IM) the approximate factored scheme (AFS), and the (pseudo) steady-state line solver (SS). Figure 3 shows two grids at different resolutions(-factor=2,4 with -order=2 as described below) and the solution (on a grid -factor=8, -order=4).

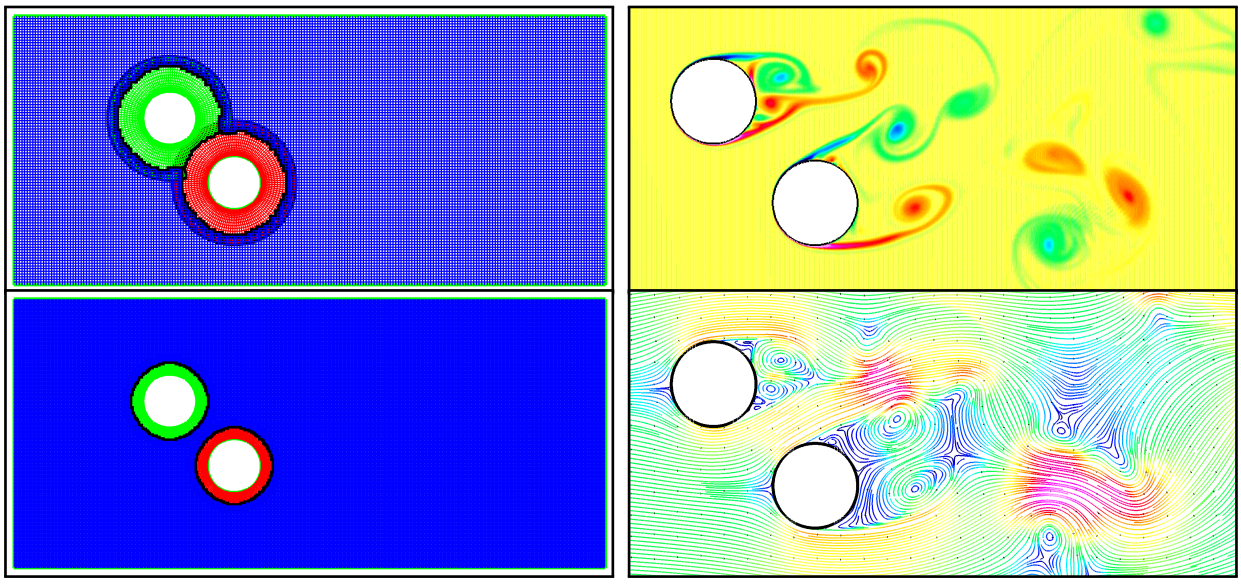

Figure 3: Incompressible flow past two cylinders. Left: overlapping grids at two resolutions. Right: vorticity and streamlines at time $t = 10$, (AFS24, tcilce8.order4.ml3, $\nu = 10^{-4}$).

The examples in this section use the Cgins command file `cg/ins/cmd/tcilc.cmd`. This script optionally takes numerous command line arguments that can be used to change parameters:

```
cgins [-noplot] tcilc -g=<grid-name> -nu=<> -tf=<> -tp=<> -show=<> -debug=<>
-project=[0|1] -ad2=[0|1] -ad4=[0|1] -ad41=<> -ad42=<> ...  -ts=[pc|im|afs|ss]
-solver=[best|mg|yale] -psolver=[best|mg|yale] -rtolp=<>

  -tf, -tp:  final time and times to plot.
  -show :  name of show file.
  -project :  1=project initial conditions to be approximate divergence free.
  -ad2 :  1=turn on 2nd-order artificial dissipation.
  -ad4 :  1=turn on 4th-order artificial dissipation.
  -ad41, -ad42 :  coefficients in fourth-order non-linear dissipation.
  -ts :  time-stepping method.
  -solver :  linear solver for implicit time-stepping equation.
  -psolver :  linear solver for the pressure equation.
  -rtolp :  relative convergence tolerance for the pressure equation solver.
```

The different grids used in this section can be computed with the Ogen command file `Overture/sampleGrids/tcilc.cmd`. Here are some examples

```
ogen -noplot tcilc -order=2 -interp=e -factor=2 : [tcilce2.order2.hdf]
ogen -noplot tcilc -order=2 -interp=e -factor=4 : [tcilce4.order2.hdf]
ogen -noplot tcilc -order=2 -interp=e -ml=3 -factor=8: [tcilce8.order2.ml3.hdf]
```

The `-factor` option specifies the grid resolution factor: the grid spacing is approximately 0.1 divided by this factor. The `-ml` option indicates that the grid should support at least this many multigrid levels (needed if the multigrid solver is used as a linear solver).

Here is a list of commands that can be used to simulate the flow using different time-stepping methods and parameters.

PC22 : second-order accurate predictor corrector, Yale direct sparse solver for pressure:
```
cgins tcilc -g=tcilce4.order2 -ts=pc -nu=1.e-4 -ad2=1 -tp=.05 -tf=5.-debug=3
-psolver=yale -show="tcilc.show" -go=halt
```

IM22 : second-order accurate implicit predictor corrector (viscous terms implicit), PETSc BiCG-Stab ILU(3) implicit solver and pressure solver:
```
cgins -noplot tcilc -g=tcilce32.order2 -ts=im -nu=.2e-4 -ad2=1 -tp=.1 -tf=10.
-solver=best -psolver=best -rtolp=1.e-4 -go=go
```

PC24 : second-order accurate in time, fourth-order accurate in space, predictor corrector, multigrid pressure solver:
```
cgins tcilc -g=tcilce32.order4.ml4 -ts=pc -nu=.2e-4 -ad4=1 -tp=.5 -tf=5.  -psolver=mg
-rtolp=1.e-4 -go=halt
```

IM24 : second-order accurate in time, fourth-order accurate in space, implicit predictor corrector, multigrid implicit solver, multigrid pressure solver, (note: decrease ad42 coefficient to avoid time-step restriction):
```
cgins tcilc -g=tcilce16.order4.ml3 -ts=im -nu=1e-4 -ad4=1 -ad42=.1 -tp=.1 -tf=5.
-solver=mg -psolver=mg -rtolp=1.e-4 -debug=3 -go=halt
```

AFS24 : second-order accurate in time, fourth-order accurate in space predictor corrector, multigrid pressure solver:
```
cgins tcilc -g=tcilce32.order4.ml4 -nu=.2e-4 -ad4=1 -ts=afs -cfl=4 -tp=.1 -tf=.2
-psolver=mg -rtolp=1.e-4 -go=halt
```

SS2 : second-order accurate pseudo-steady-state line solver (local time stepping), MG pressure solver solver:
```
cgins -noplot tcilc -g=tcilce4.order2.ml2 -nu=.1 -ts=ss -plotIterations=100
-maxIterations=5000 -psolver=mg -show="tcilc.show" -go=go
```

IM22-FI : second-order accurate full-implicit solver:
```
cgins tcilc -g=tcilce2.order2 -nu=.1 -ts=im -implicitVariation=full -implicitFactor=1.
-refactorFrequency=20 -tp=.5 -tf=100.  -dtMax=.1 -solver=yale -psolver=yale
-plotResiduals=1
```

Note: The implicit methods (IM22, IM24 and AFS24) are generally faster when there is a fine boundary layer grid since the viscous terms generally require explicit schemes to have a small time-step.

Note: The Yale direct sparse solver is usually fastest linear solver for small to moderate size 2D problems. The MG solver is otherwise generally the fastest.

Note: The AFS solver can run at CFL numbers bigger than one. Currently this requires some trial and error to determine how to choose the `-cfl` parameter and other AFS parameters, see Section **??** for more details.

Note: The generally fastest overall solver for bigger problems, or moving grid problems is AFS24 with the MG pressure solver.

Note: The SS2 solver is very memory efficient and is useful for problems where time accuracy is not important or for low Reynolds number flows where the solution reaches a steady state.

Note: The full implicit solver IM22-FI treates the momentum equations with all terms implicit (expect the pressure gradient). This method requires more memory.

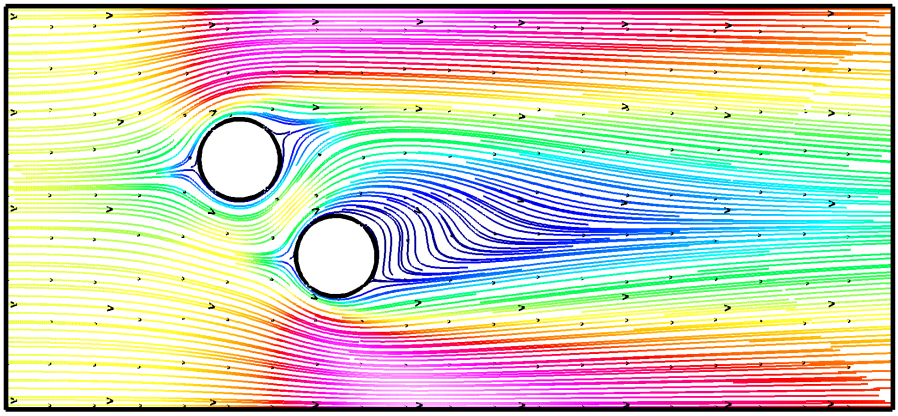

Figure 4: Incompressible flow past two cylinders. Steady state solution computed with the steady-state line solver (or full implicit solver), $\nu = 0.1$.

2.4 Incompressible flow around a naca airfoil

The command file `cg/ins/cmd/naca.cmd` can be used with cgins 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` generated with `Overture/sampleGrids/naca.hype.cmd`)

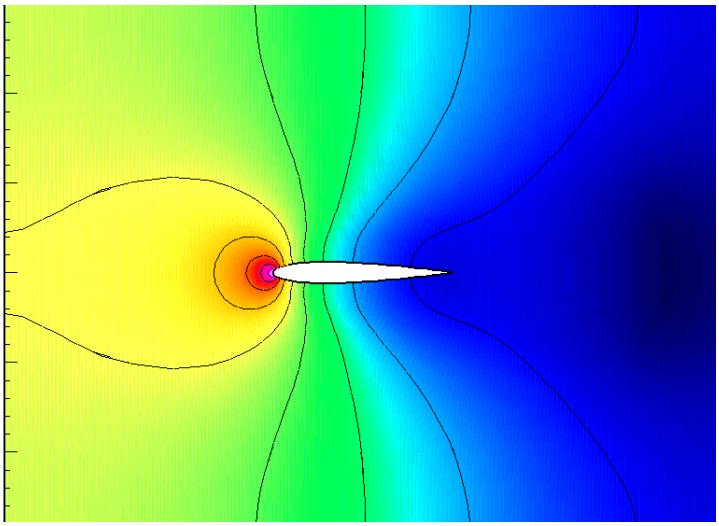

Figure 5: Incompressible flow past a NACA 0012 airfoil, pressure.

This example demonstrates the use of the second-order non-linear artificial diffusion (described in the Cgins reference guide [**?**]). The default value for $\nu = 10^{-8}$ is much too small to have any affect on the solution for the grid being used. 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 sometimes some fiddling required to get the coefficients of the artificial diffusion correct. The values are usually always around be .1 and 5..

2.5 Incompressible flow around a moving stirring stick

The command file `cg/ins/cmd/stir.cmd` can be used with cgins 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`.

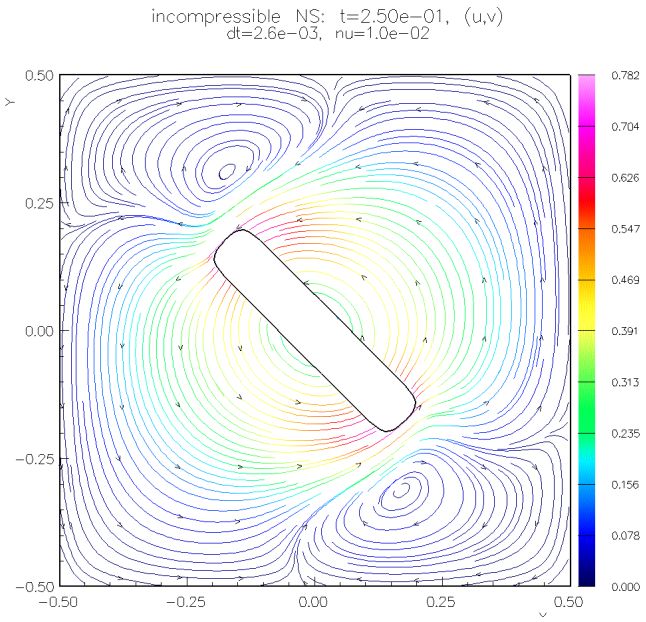

Figure 6: Incompressible flow around a rotating stirring stick demonstrating the use of moving grids.

2.6 Axisymetric incompressible flow past a sphere

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

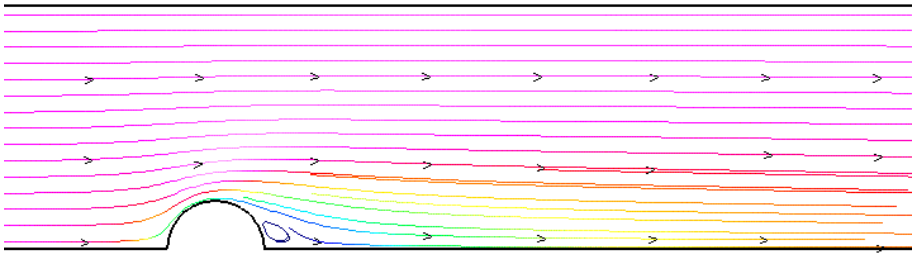

Figure 7: Incompressible axisymmetric flow past a sphere.

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

The command files `cg/ins/cmd/backStep.cmd`, `cg/ins/cmd/cgrid.cmd` and `cg/ins/cmd/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/backStep.cmd`, `Overture/sampleGrids/cgrid.cmd` and `Overture/sampleGrids/hgrid.cmd`.

In general it is preferable NOT to use grids that have been generated with a *mixed boundary*. For example, the flow around the convex corner in the backward facing step is not well resolved. It is recommended to round off convex corners. See the command file `cg/ins/cmd/backStepSmooth.cmd` for flow past a back step with a smoothed corner.

2.8 Incompressible flow past a sphere

The command file `cg/ins/cmd/sib.cmd` can be used with cgins to compute the flow past a sphere in a box.

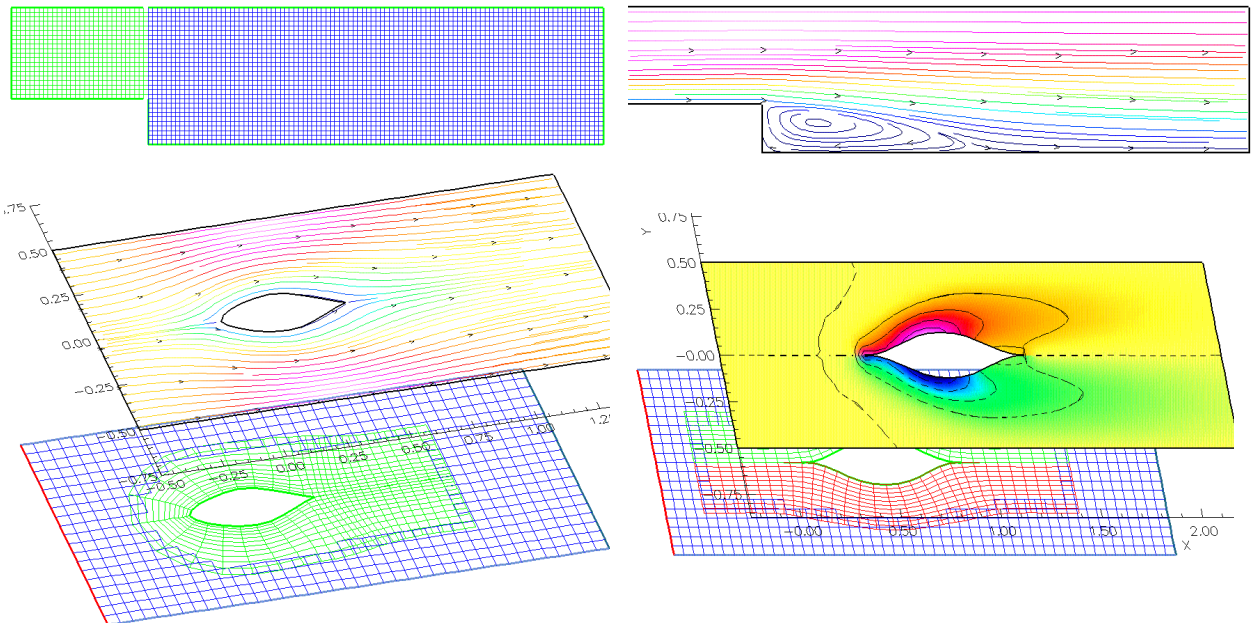

Figure 8: Incompressible flow past a backward facing step, a C-grid and an H-grid.

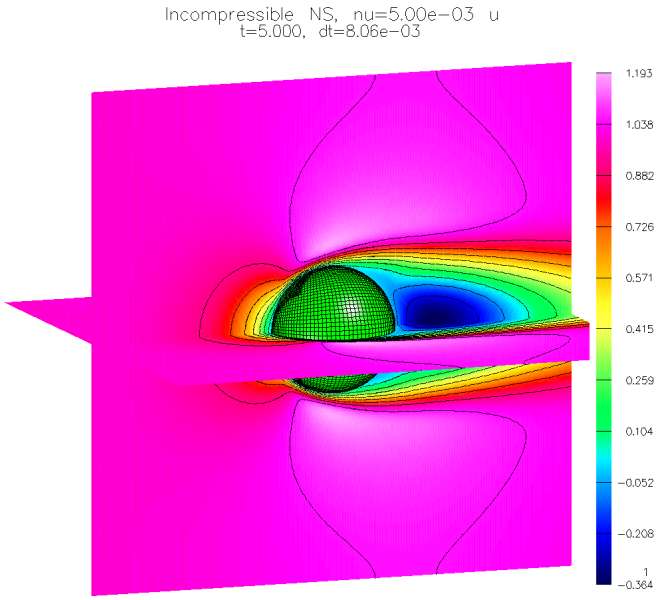

Figure 9: Incompressible flow past a sphere.

For 3D problems it is almost always necessary to use an iterative solver for the pressure equation and the implicit time stepping equations. The PETSc linear solvers are recommended. Iterative solvers require a convergence tolerance and you may have to play around with this tolerance (if the tolerance for the pressure equation is too large, then the divergence may grow large). See the Oges documentation[**?**] for more information on linear solvers.

2.9 Incompressible flow through some intersecting pipes

The command file can be `cg/ins/cmd/pipes.cmd` used with cgins to compute the flow through two intersecting pipes. The pipes intersect using the *poor man's* intersection option. This example uses the overlap-

ping grid `Overture/sampleGrids/pipes.hdf` generated using the command file `Overture/sampleGrids/-pipes.cmd`. See also the command file `cg/ins/cmd/twoPipes.cmd` where the pipes are joined with a smooth fillet.

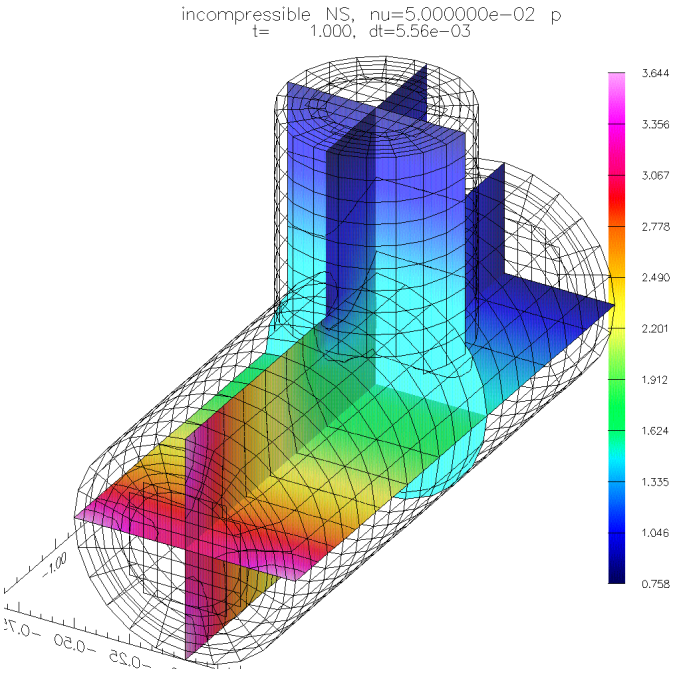

Figure 10: Incompressible flow through some pipes.

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

2.10 Two falling drops in an incompressible flow

Figure (11) shows two rigid bodies ("drops") failing under the influence of gravity in an incompressible flow.

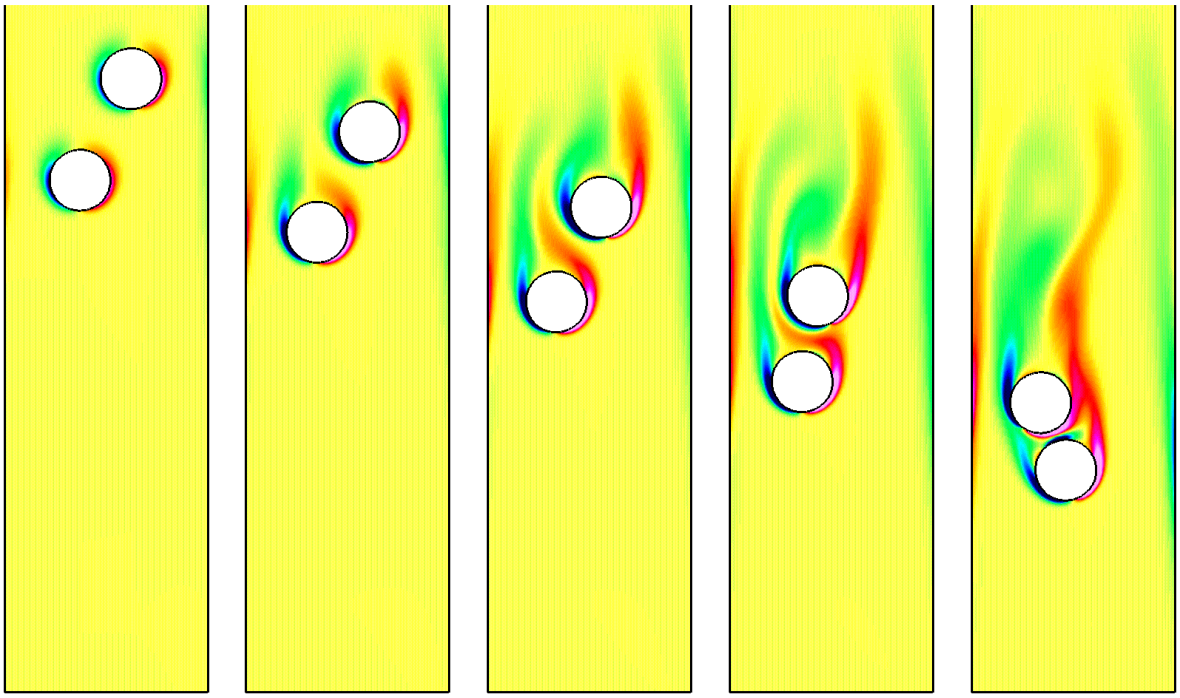

Figure 11: Two drops falling in an incompressible flow, contour plots of the vorticity. The upper drop wants to "draft" in behind the lower drop where the pressure is lower.

This computation used the command file `cg/ins/cmd/twoDrop.cmd`. The grid can be created with `Overture/sampleGrids/twoDrop.cmd` (finer grids with `Overture/sampleGrids/twoDropArg.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.