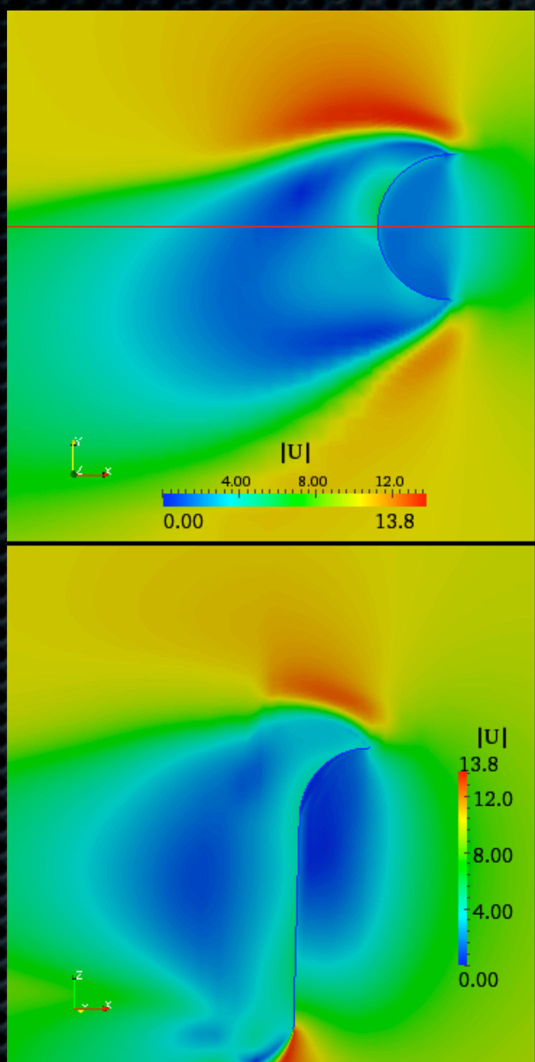
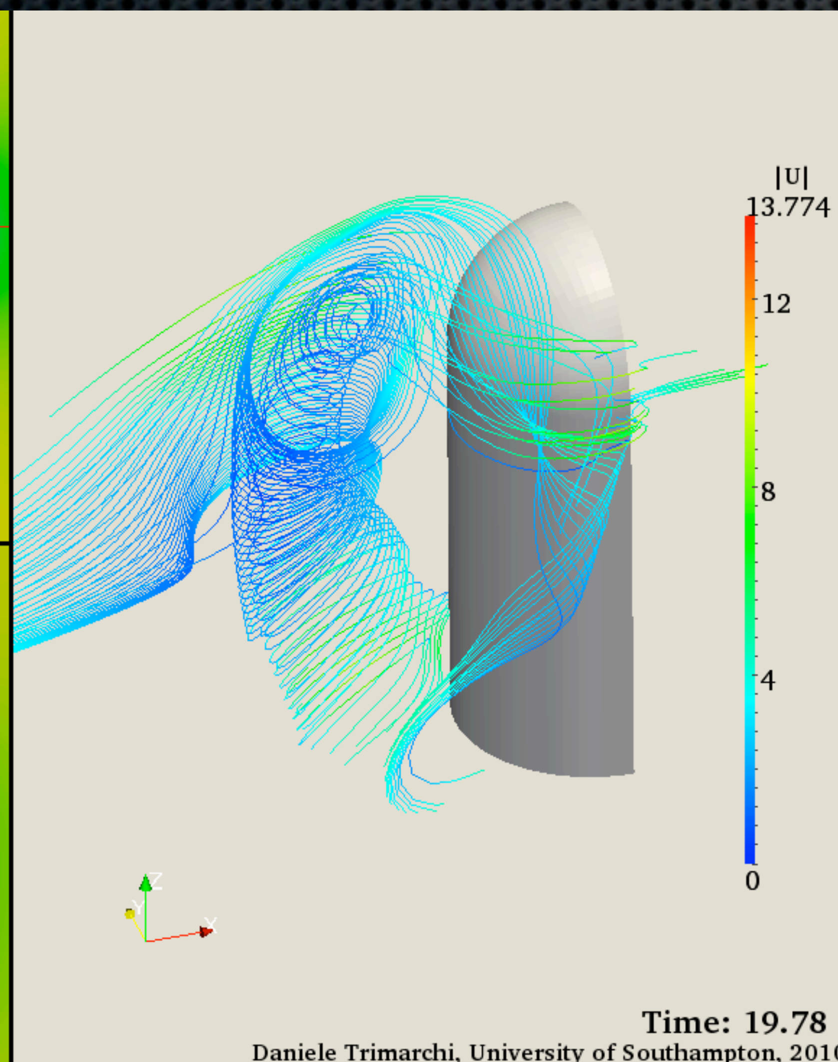
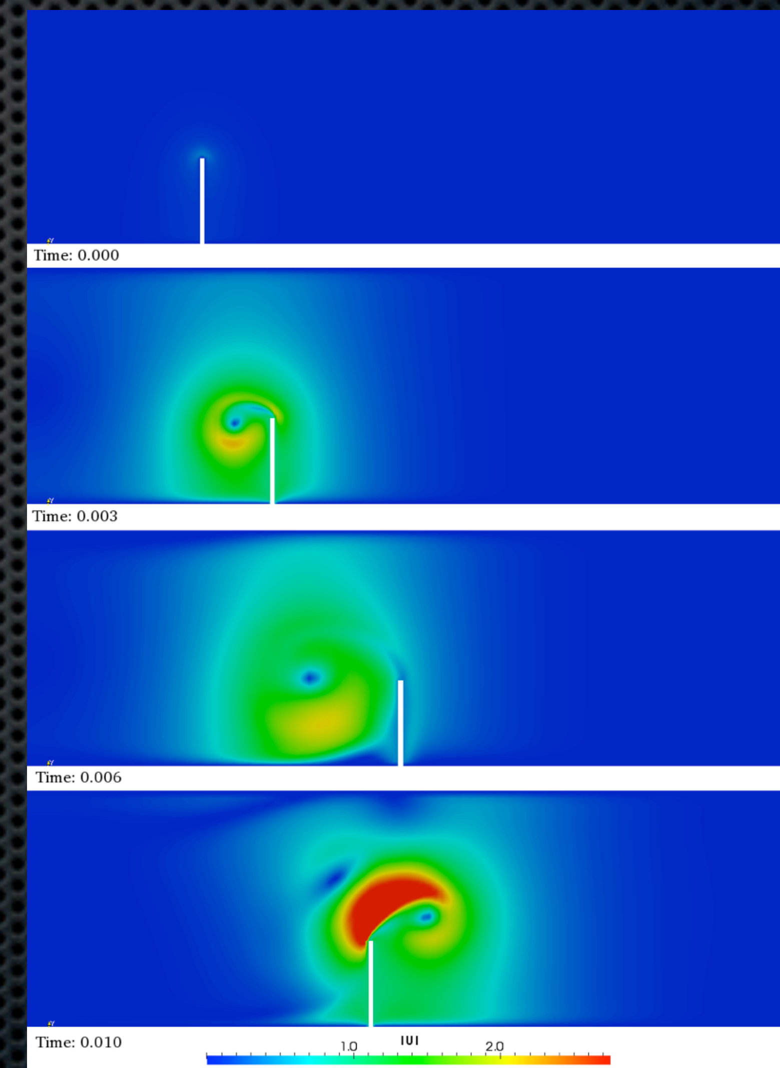
Open∇FOAM Workshop







Daniele Trimarchi
daniele.trimarchi@soton.ac.uk
University of Southampton, 01-11-2010

Table of contents:

Table of contents:

Table of contents:

Table of contents:

- OpenFOAM, UNIX AND THE MAC
- MESH GENERATION
  - presentation of the routine airfoil.exe

Table of contents:

- OpenFOAM, UNIX AND THE MAC
- MESH GENERATION
  - presentation of the routine airfoil.exe
  - GMSH: tutorial on the meshing of a simple geometry

Table of contents:

Table of contents:

Table of contents:

Table of contents:

Table of contents:

- OpenFOAM, UNIX AND THE MAC
- MESH GENERATION
  - presentation of the routine airfoil.exe
  - GMSH: tutorial on the meshing of a simple geometry
  - GMSH conversion: the utility gmshToFOAM and CreateBaffles
- UNSTEADY RANSE
  - Imposing unsteady BC using ramp files
  - forces, coefficients and samples extraction
  - Airfoil case study: the pisoFOAM solver, SST turb. model & unsteady BC

Table of contents:

Table of contents:

- OpenFOAM, UNIX AND THE MAC
- MESH GENERATION
  - presentation of the routine airfoil.exe
  - GMSH: tutorial on the meshing of a simple geometry
  - GMSH conversion: the utility gmshToFOAM and CreateBaffles
- UNSTEADY RANSE
  - Imposing unsteady BC using ramp files
  - forces, coefficients and samples extraction
  - Airfoil case study: the pisoFOAM solver, SST turb. model & unsteady BC
- DYNAMIC MESH HANDLING
  - Principles of mesh motion: ALE => the PimpleDyMFOAM solver, and the velocityLaplacianMotionSolver

Table of contents:

Table of contents:

Table of contents:

Table of contents:

OpenFOAM and the MAC

OpenFOAM and the MAC

- MAC core is UNIX, therefore it is possible to build OF

OpenFOAM and the MAC

- MAC core is UNIX, therefore it is possible to build OF
- It is however less easy than in Ubuntu....

OpenFOAM and the MAC

- MAC core is UNIX, therefore it is possible to build OF
- It is however less easy than in Ubuntu....
- Very good guidance in the forum:
  - **http://www.cfd-online.com/Forums/openfoam-installation/77570-patches-openfoam-1-7-macos-x.html**

OpenFOAM and the MAC

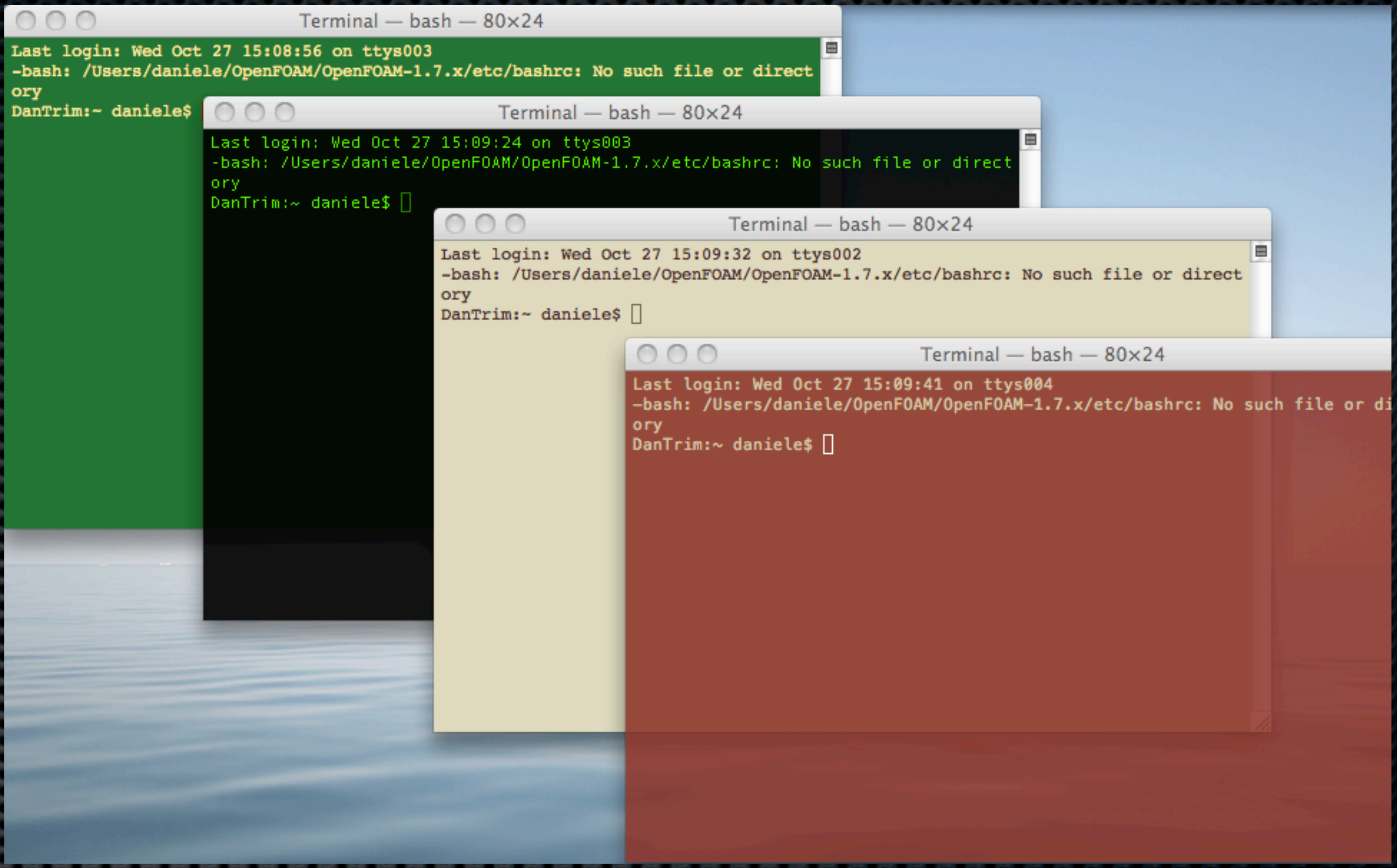
- MAC core is UNIX, therefore it is possible to build OF
- It is however less easy than in Ubuntu....
- Very good guidance in the forum:
  - **http://www.cfd-online.com/Forums/openfoam-installation/77570-patches-openfoam-1-7-macos-x.html**
- You need Xode
  - **http://developer.apple.com/technologies/tools**

OpenFOAM and the MAC

- MAC core is UNIX, therefore it is possible to build OF
- It is however less easy than in Ubuntu....
- Very good guidance in the forum:
  - **http://www.cfd-online.com/Forums/openfoam-installation/77570-patches-openfoam-1-7-macos-x.html**
- You need Xode
  - **http://developer.apple.com/technologies/tools**
- And macPorts is a very confortable tool..!
  - **www.macports.org**

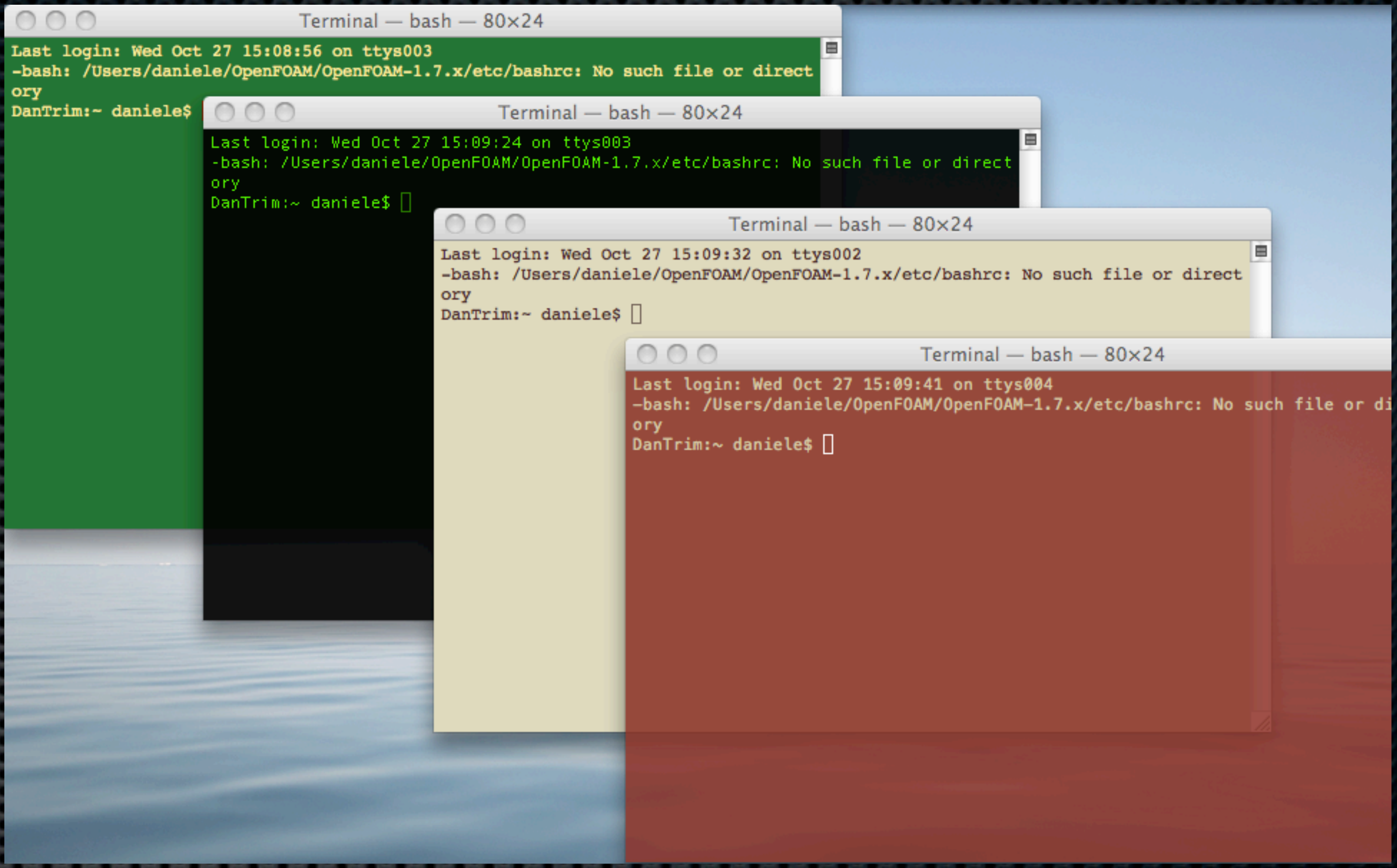
Unix: the use of the terminal

Unix: the use of the terminal



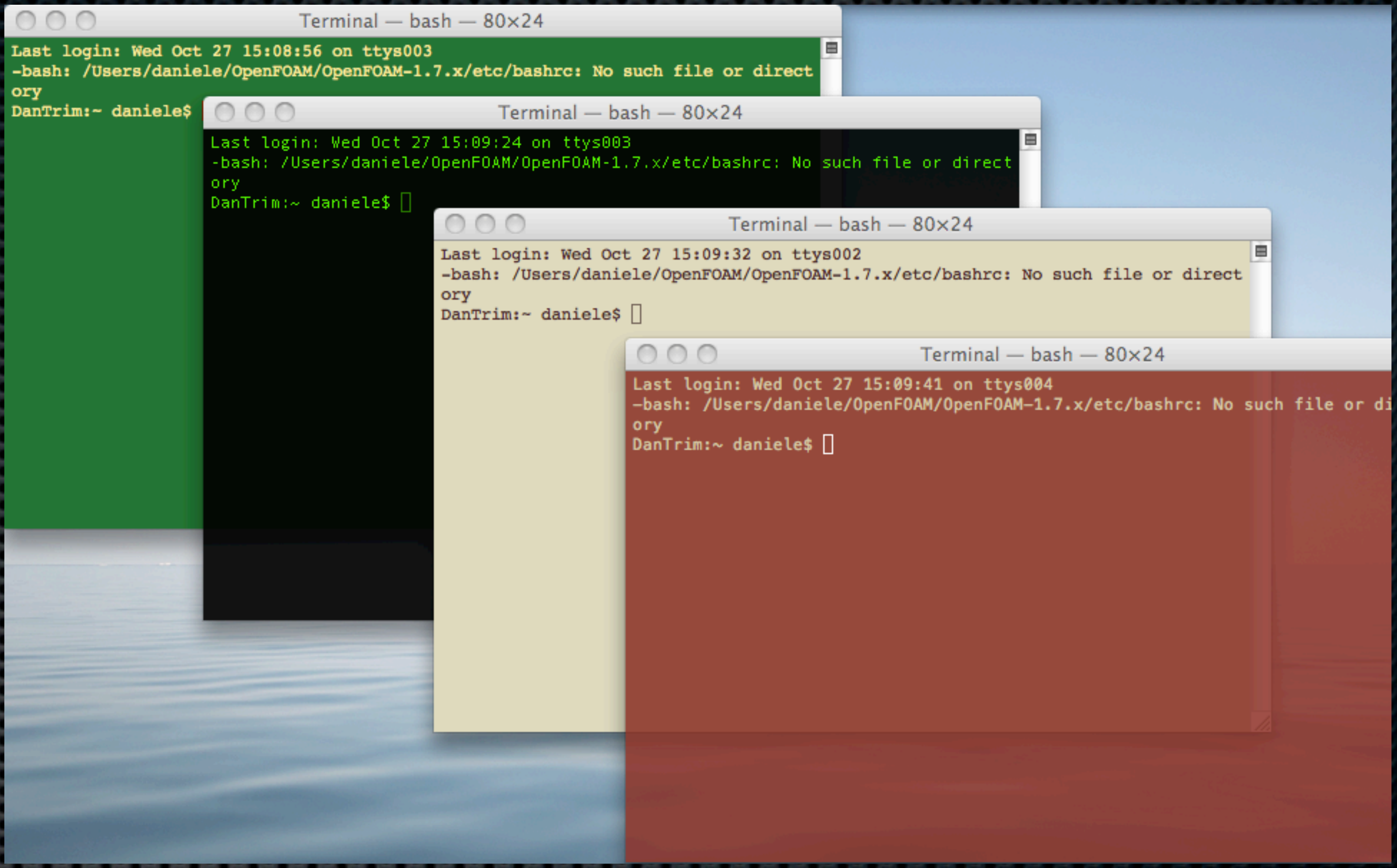
Unix: the use of the terminal

- ls



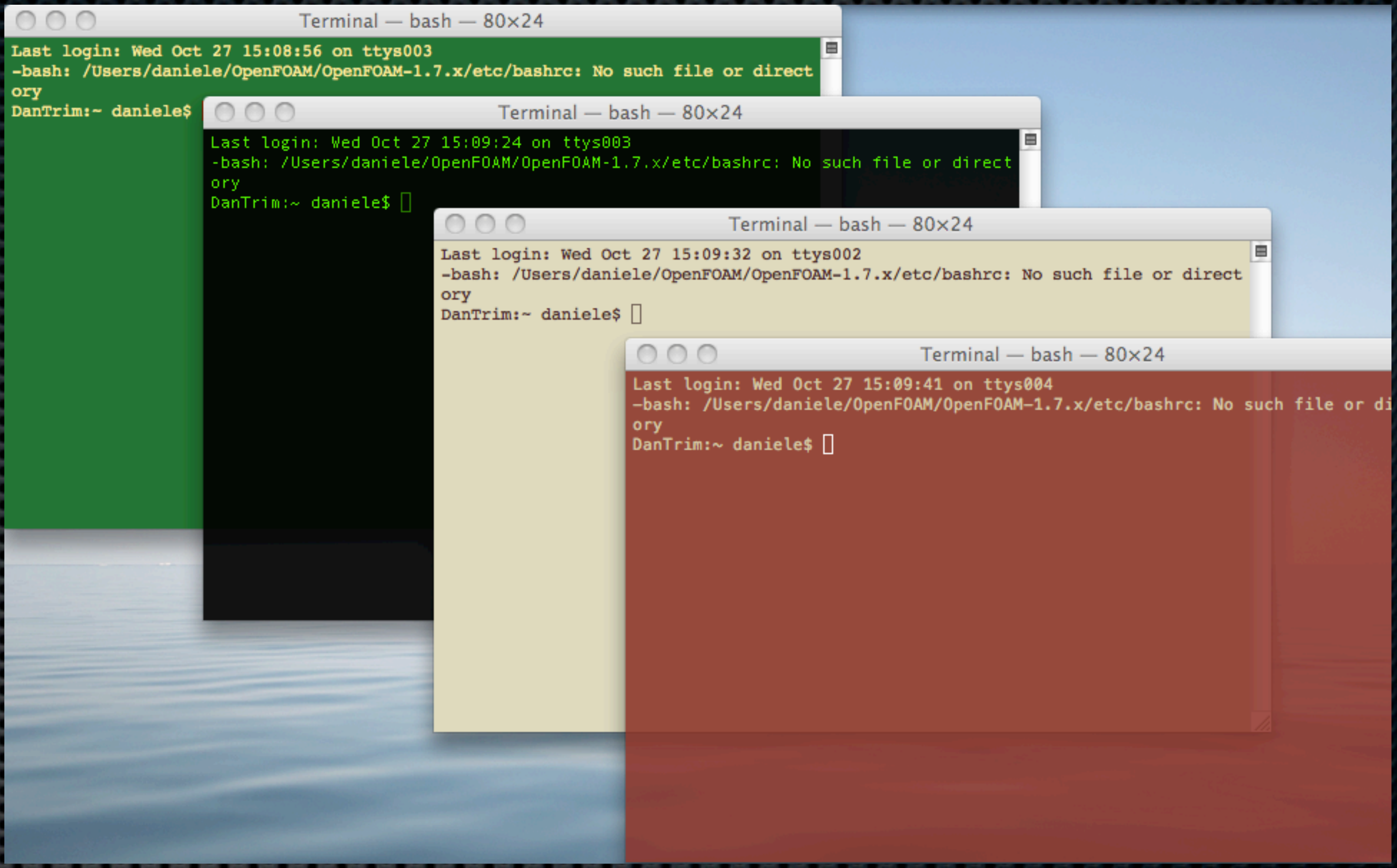
Unix: the use of the terminal

- ls
- cd



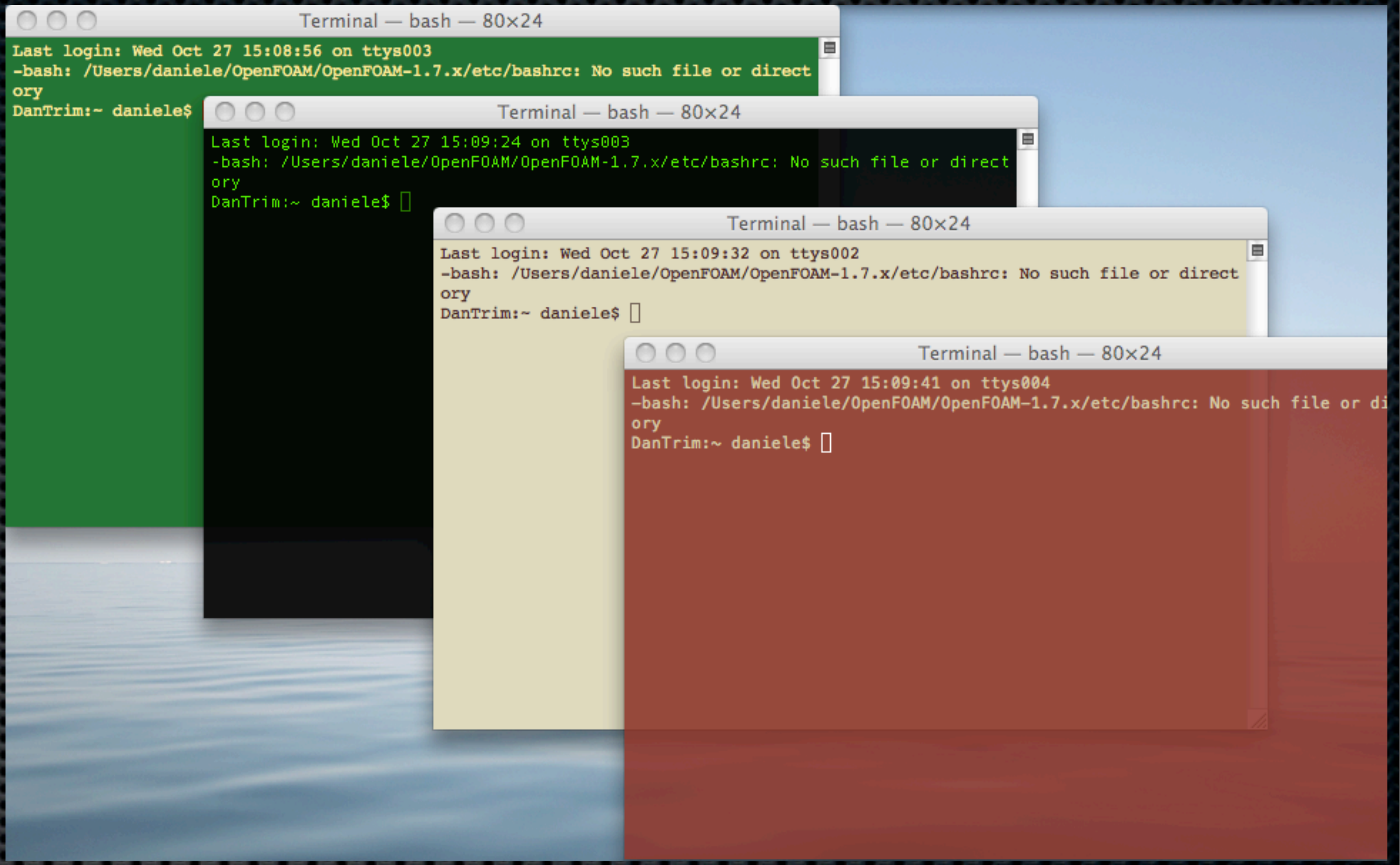
Unix: the use of the terminal

- ls
- cd
- pwd



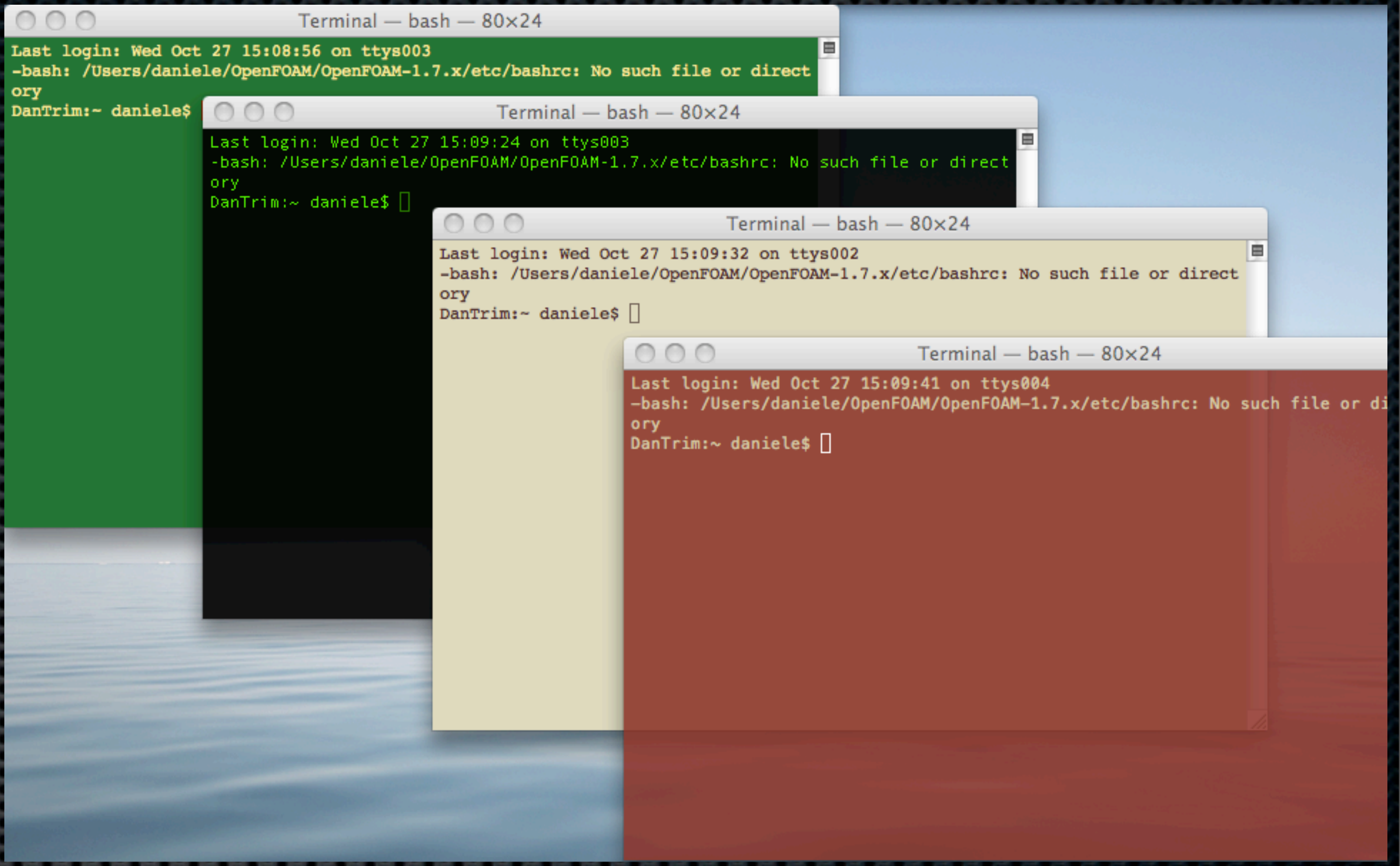
Unix: the use of the terminal

- ls
- cd
- pwd
- mkdir



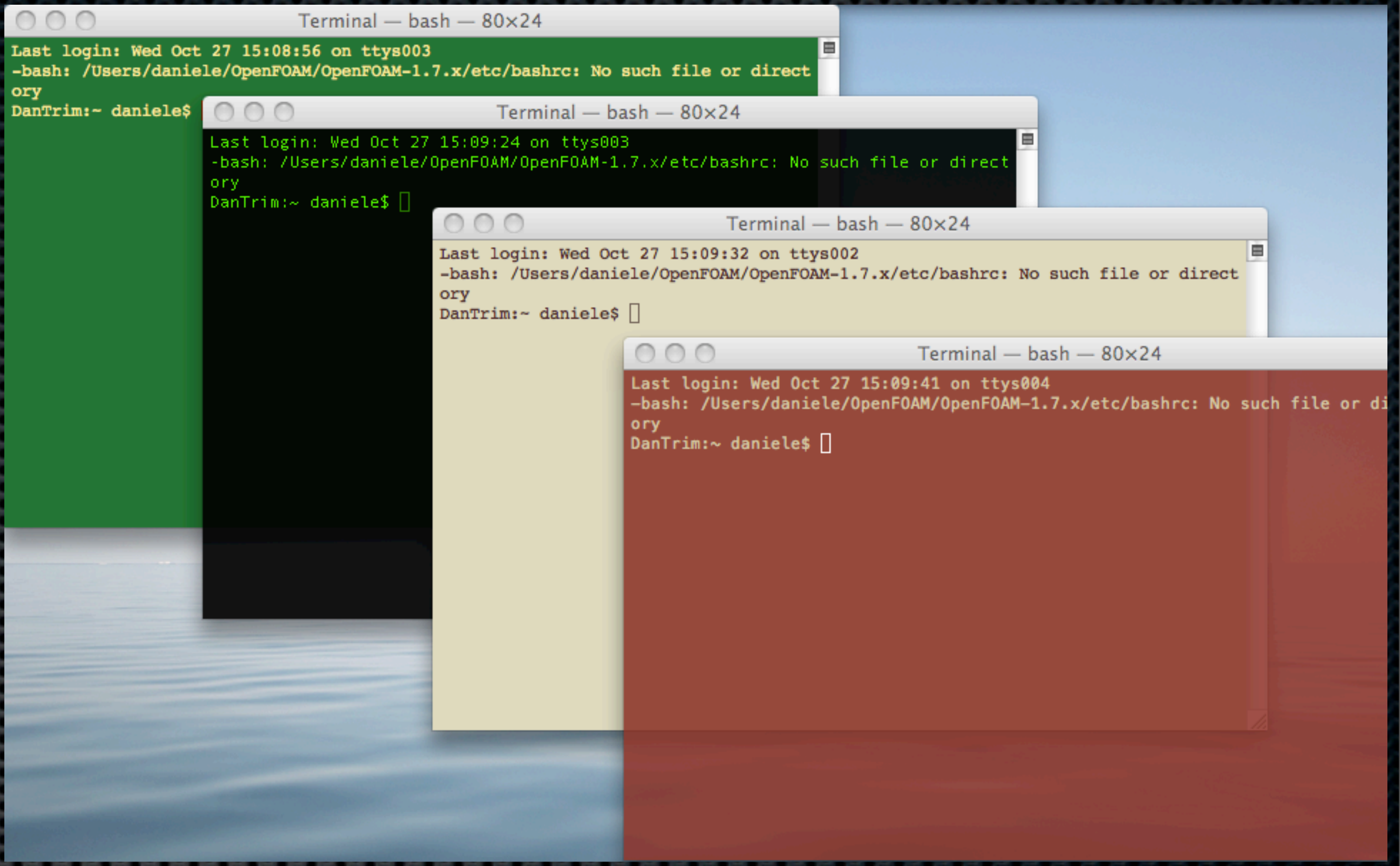
Unix: the use of the terminal

- ls
- cd
- pwd
- mkdir
- >



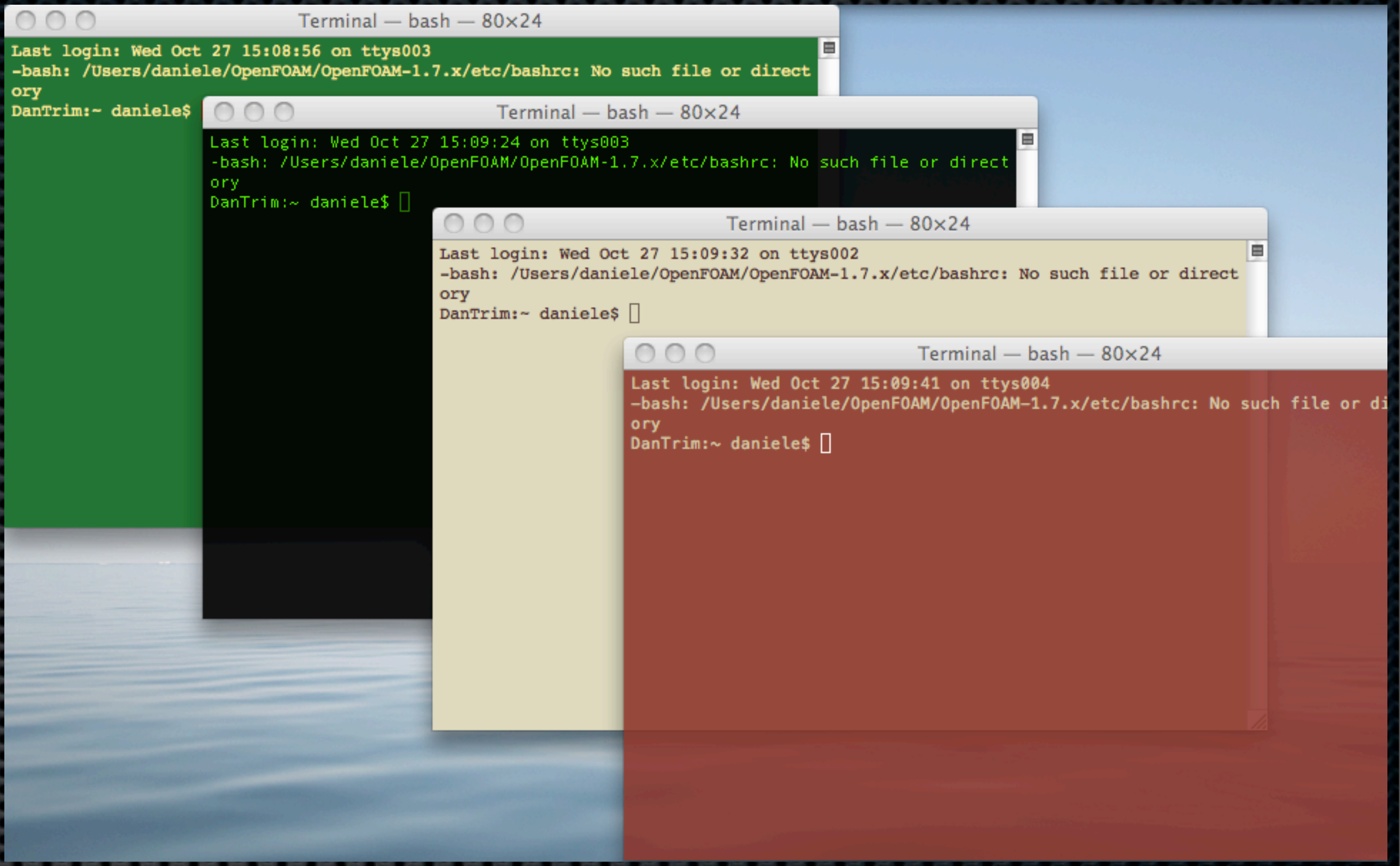
Unix: the use of the terminal

- ls
- cd
- pwd
- mkdir
- >
- man



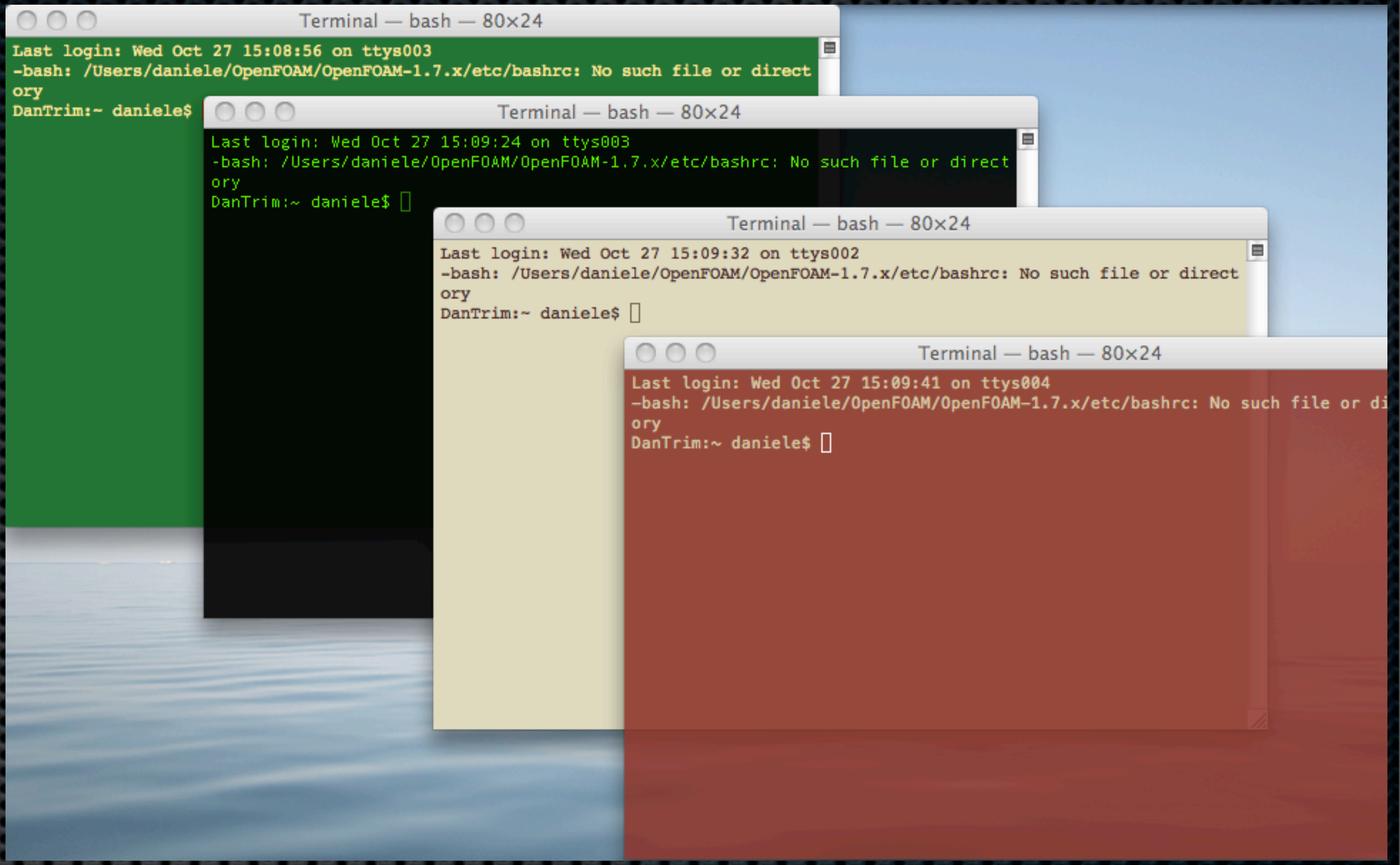
Unix: the use of the terminal

- ls
- cd
- pwd
- mkdir
- >
- man
- rm / rm -r [!!!]



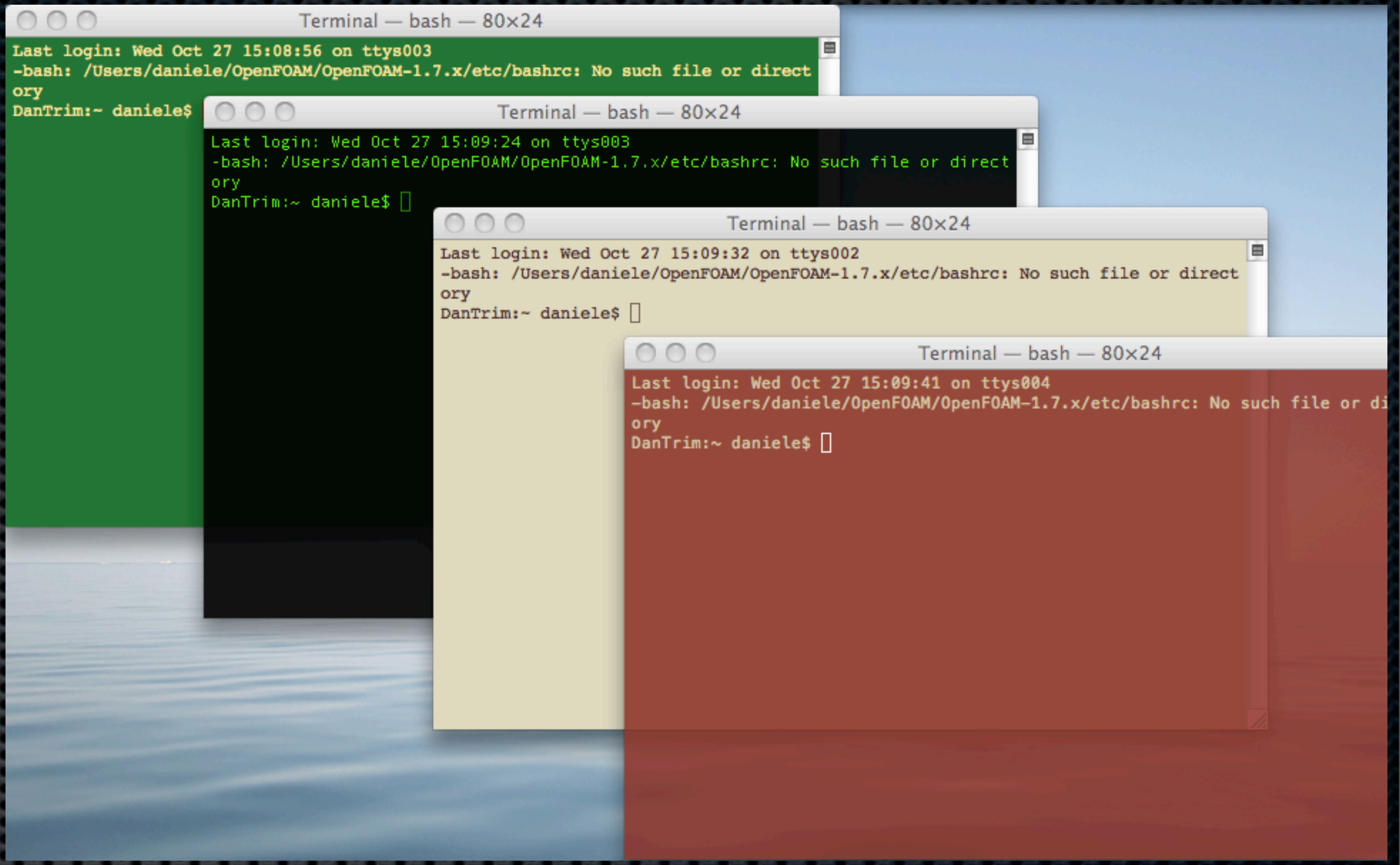
Unix: the use of the terminal

- ls
- cd
- pwd
- mkdir
- >
- man
- rm / rm -r [!!!]
- the .bash files



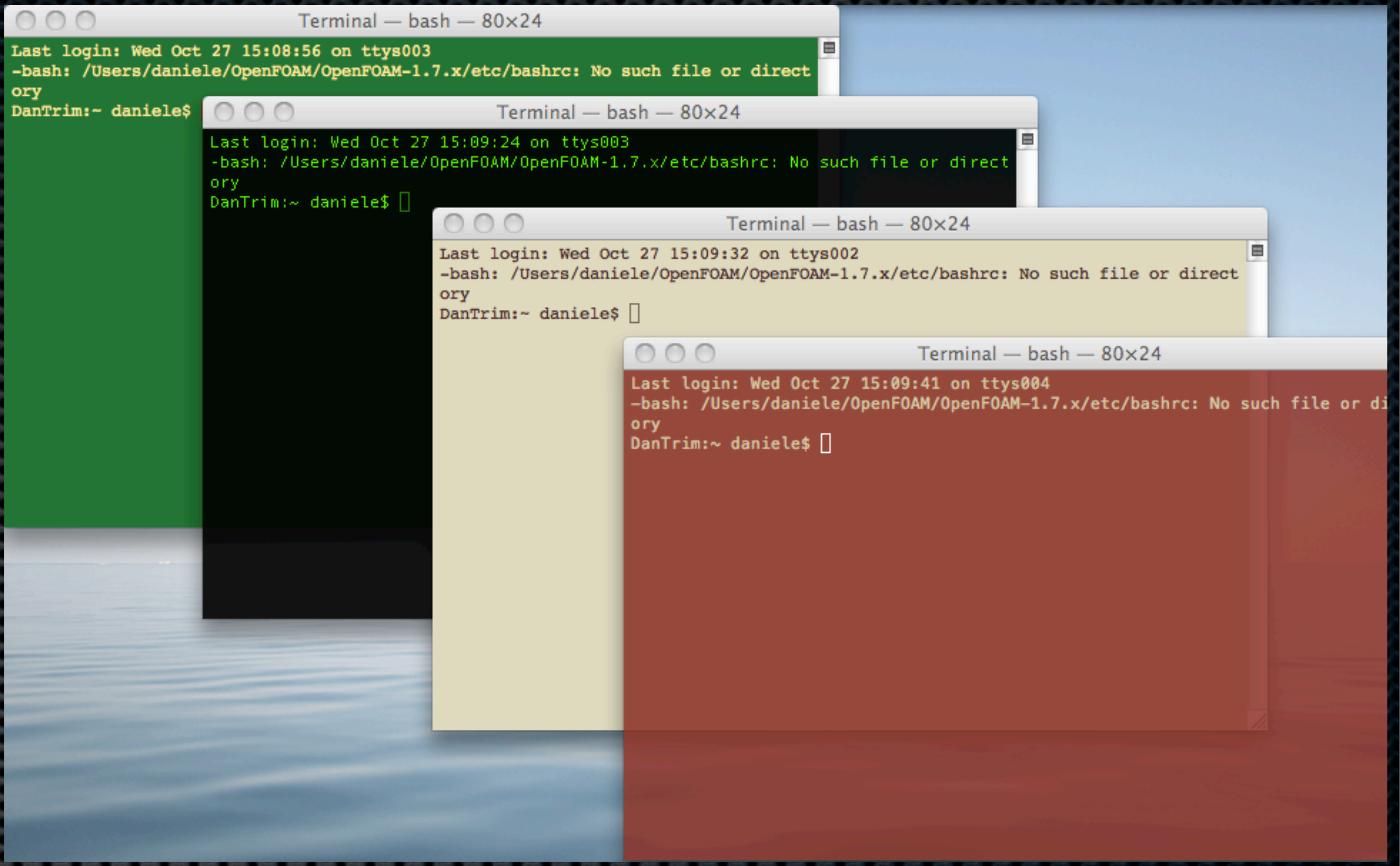
Unix: the use of the terminal

- ls
- cd
- pwd
- mkdir
- >
- man
- rm / rm -r [!!!]
- the .bash files
- make ➙ wmake



Unix: the use of the terminal

- ls
- cd
- pwd
- mkdir
- >
- man
- rm / rm -r [!!!]
- the .bash files
- make ➙ wmake



http://info.ee.surrey.ac.uk/Teaching/Unix/

Airfoil.exe

Airfoil.exe

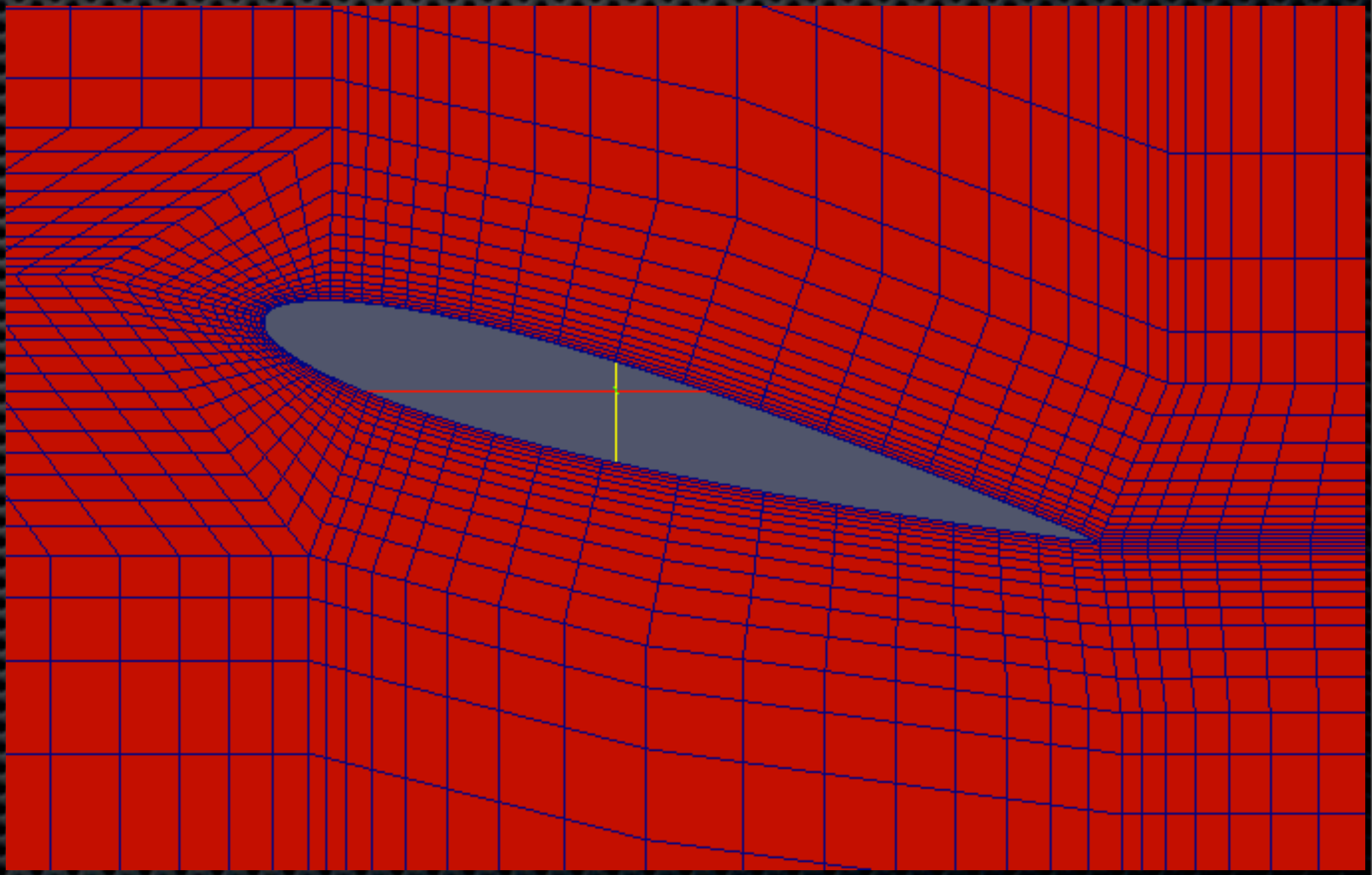
- Fortran program for creating a BlockMesh input file

Airfoil.exe

- Fortran program for creating a BlockMesh input file
- Geometry: 2D airfoil types (http://www.ae.illinois.edu/m-selig/ads.html)

Airfoil.exe

- Fortran program for creating a BlockMesh input file
- Geometry: 2D airfoil types (http://www.ae.illinois.edu/m-selig/ads.html)



Airfoil.exe

- Fortran program for creating a BlockMesh input file
- Geometry: 2D airfoil types (http://www.ae.illinois.edu/m-selig/ads.html)

Airfoil.exe

- Fortran program for creating a BlockMesh input file
- Geometry: 2D airfoil types (http://www.ae.illinois.edu/m-selig/ads.html)
- 2 Input files: airfoil.data ; input.data

Airfoil.exe

- Fortran program for creating a BlockMesh input file
- Geometry: 2D airfoil types (http://www.ae.illinois.edu/m-selig/ads.html)
- 2 Input files: airfoil.data ; input.data

```
63
0       0
0.00931490000000000     0.00255950000000000
0.0145291000000000      0.00328040000000000
0.0208771000000000      0.00415190000000000
0.0283441000000000      0.00516850000000000
0.0369127000000000      0.00632380000000000
0.0465628000000000      0.00761080000000000
0.0572720000000000      0.00902170000000000
0.0690152000000000      0.0105485000000000
0.0817649000000000      0.0121823000000000
0.0954915000000000      0.0139143000000000
0.110162800000000       0.0157351000000000
0.125744600000000       0.0176353000000000
0.142200500000000       0.0196051000000000
0.159492100000000       0.0216347000000000
0.177578900000000       0.0237142000000000
0.196418700000000       0.0258337000000000
0.215967600000000       0.0279828000000000
0.236179900000000       0.0301515000000000
0.257008300000000       0.0323294000000000
0.278404200000000       0.0345058000000000
0.300317700000000       0.0366700000000000
0.322697600000000       0.0388109000000000
0.345491500000000       0.0409174000000000
0.368646300000000       0.0429778000000000
0.392107900000000       0.0449802000000000
0.415821500000000       0.0469124000000000
0.439731700000000       0.0487619000000000
0.463782600000000       0.0505161000000000
0.487918100000000       0.0521620000000000
```

Airfoil.exe

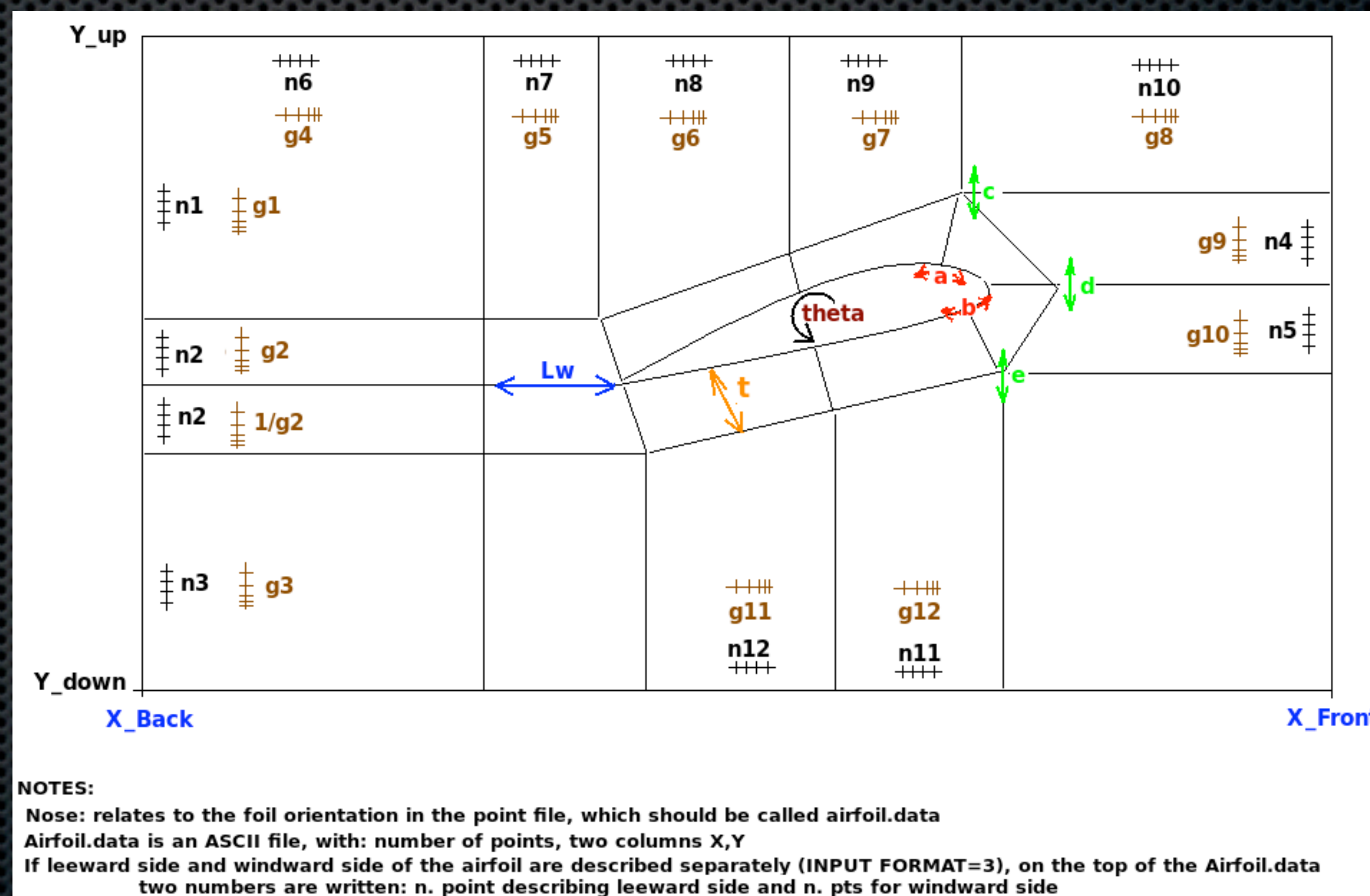
- Fortran program for creating a BlockMesh input file
- Geometry: 2D airfoil types (http://www.ae.illinois.edu/m-selig/ads.html)
- 2 Input files: airfoil.data ; input.data

```
63
0       0
0.00931490000000000     0.00255950000000000
0.0145291000000000      0.00328040000000000
0.0208771000000000      0.00415190000000000
0.0283441000000000      0.00516850000000000
0.0369127000000000      0.00632380000000000
0.0465628000000000      0.00761080000000000
0.0572720000000000      0.00902170000000000
0.0690152000000000      0.0105485000000000
0.0817649000000000      0.0121823000000000
0.0954915000000000      0.0139143000000000
0.110162800000000       0.0157351000000000
0.125744600000000       0.0176353000000000
0.142200500000000       0.0196051000000000
0.159492100000000       0.0216347000000000
0.177578900000000       0.0237142000000000
0.196418700000000       0.0258337000000000
0.215967600000000       0.0279828000000000
0.236179900000000       0.0301515000000000
0.257008300000000       0.0323294000000000
0.278404200000000       0.0345058000000000
0.300317700000000       0.0366700000000000
0.322697600000000       0.0388109000000000
0.345491500000000       0.0409174000000000
0.368646300000000       0.0429778000000000
0.392107900000000       0.0449802000000000
0.415821500000000       0.0469124000000000
0.439731700000000       0.0487619000000000
0.463782600000000       0.0505161000000000
0.487918100000000       0.0521620000000000
```

Airfoil.exe

- Fortran program for creating a BlockMesh input file
- Geometry: 2D airfoil types (http://www.ae.illinois.edu/m-selig/ads.html)
- 2 Input files: airfoil.data ; input.data

63	
0	0
0.00931490000000000	0.00255950000000000
0.0145291000000000	0.00328040000000000
0.0208771000000000	0.00415190000000000
0.0283441000000000	0.00516850000000000
0.0369127000000000	0.00632380000000000
0.0465628000000000	0.00761080000000000
0.0572720000000000	0.00902170000000000
0.0690152000000000	0.0105485000000000
0.0817649000000000	0.0121823000000000
0.0954915000000000	0.0139143000000000
0.110162800000000	0.0157351000000000
0.125744600000000	0.0176353000000000
0.142200500000000	0.0196051000000000
0.159492100000000	0.0216347000000000
0.177578900000000	0.0237142000000000
0.196418700000000	0.0258337000000000
0.215967600000000	0.0279828000000000
0.236179900000000	0.0301515000000000
0.257008300000000	0.0323294000000000
0.278404200000000	0.0345058000000000
0.300317700000000	0.0366700000000000
0.322697600000000	0.0388109000000000
0.345491500000000	0.0409174000000000
0.368646300000000	0.0429778000000000
0.392107900000000	0.0449802000000000
0.415821500000000	0.0469124000000000
0.439731700000000	0.0487619000000000
0.463782600000000	0.0505161000000000
0.487918100000000	0.0521620000000000



Airfoil.exe

- Fortran program for creating a BlockMesh input file
- Geometry: 2D airfoil types (http://www.ae.illinois.edu/m-selig/ads.html)
- 2 Input files: airfoil.data ; input.data

Airfoil.exe

- Fortran program for creating a BlockMesh input file
- Geometry: 2D airfoil types (http://www.ae.illinois.edu/m-selig/ads.html)
- 2 Input files: airfoil.data ; input.data
- complile the program: Makefile

Airfoil.exe

- Fortran program for creating a BlockMesh input file
- Geometry: 2D airfoil types (http://www.ae.illinois.edu/m-selig/ads.html)
- 2 Input files: airfoil.data ; input.data
- complile the program: Makefile

Airfoil.data 1 | Makefile 2

```
FFLAGS= -O
.DEFAULT: .f
.SUFFIXES:.f

.f.o: $*.f
	gfortran -c $(FFLAGS) $*.f

#-----------------------
#source file(s) name
OBJS= Airfoil.o max.o mid_A.o quA.o mid_B.o quB.o cGrid.o ruot.o \
	inv.o inv2.o simm.o mirror.o
#-----------------------

airfoil: $(OBJS)
	gfortran $(FFLAGS) $(OBJS) -o airfoil.exe
```

Airfoil.exe

- Fortran program for creating a BlockMesh input file
- Geometry: 2D airfoil types (http://www.ae.illinois.edu/m-selig/ads.html)
- 2 Input files: airfoil.data ; input.data
- complile the program: Makefile

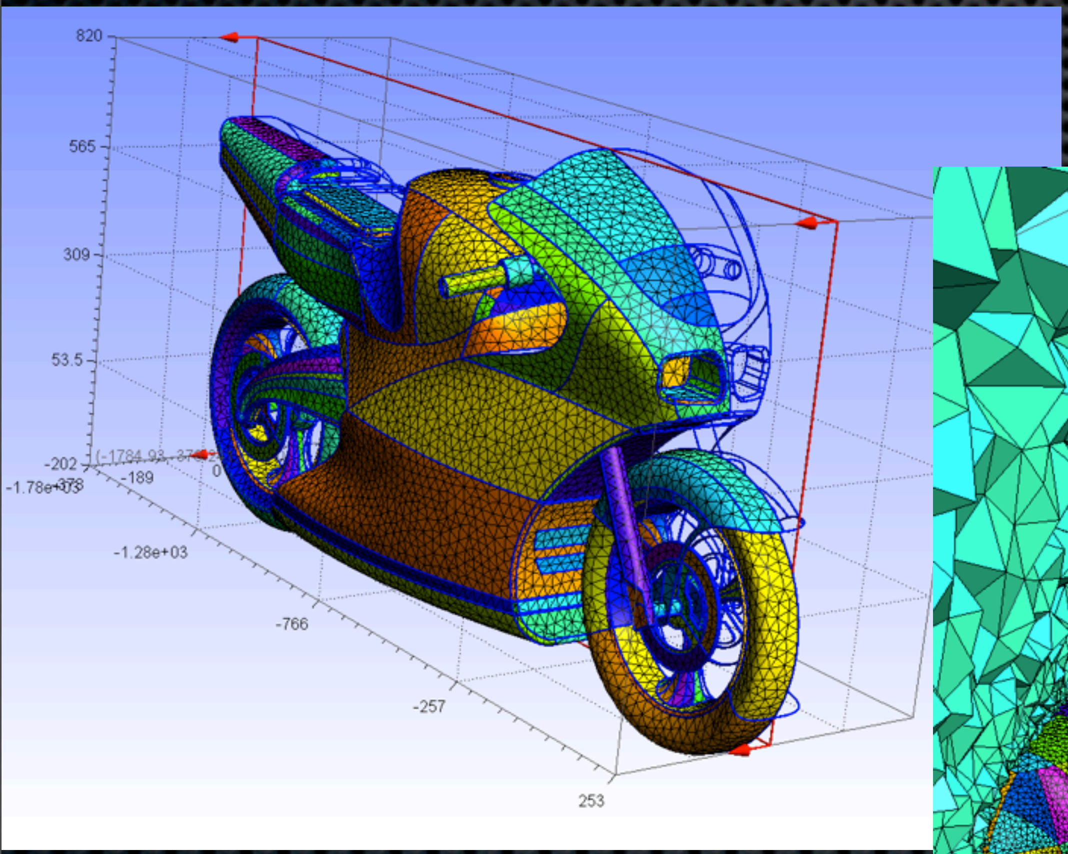
Airfoil.exe

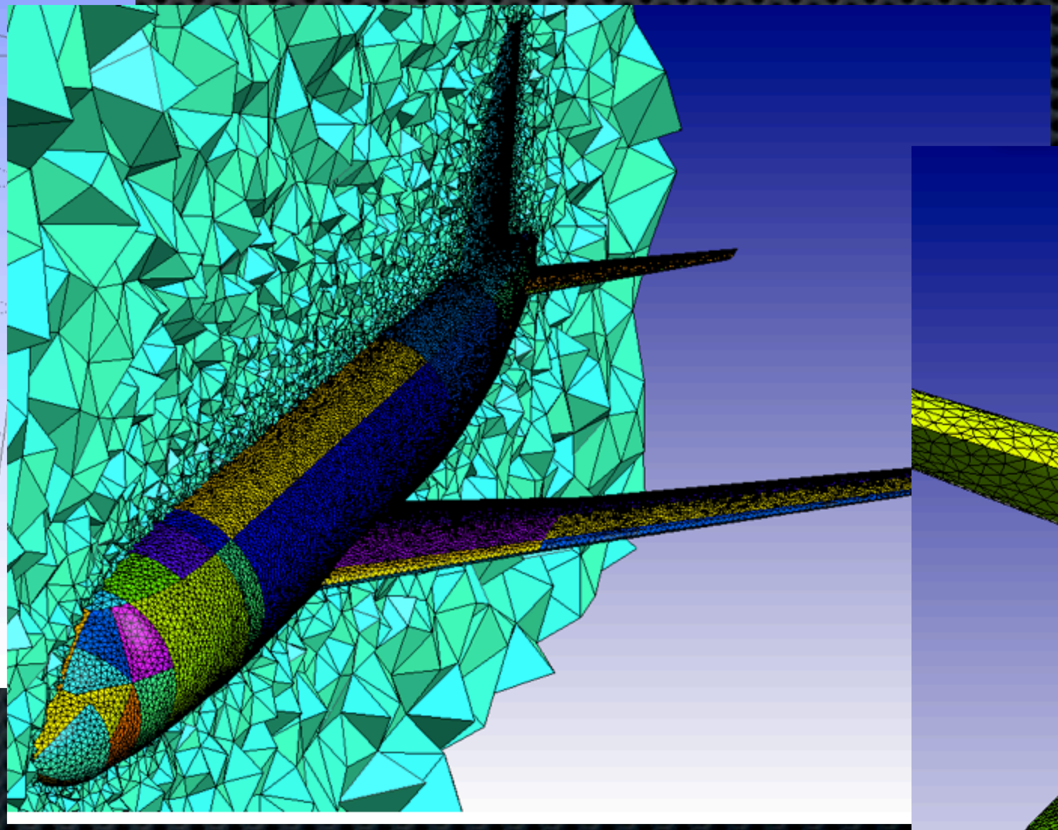
- Fortran program for creating a BlockMesh input file
- Geometry: 2D airfoil types (http://www.ae.illinois.edu/m-selig/ads.html)
- 2 Input files: airfoil.data ; input.data
- complile the program: Makefile

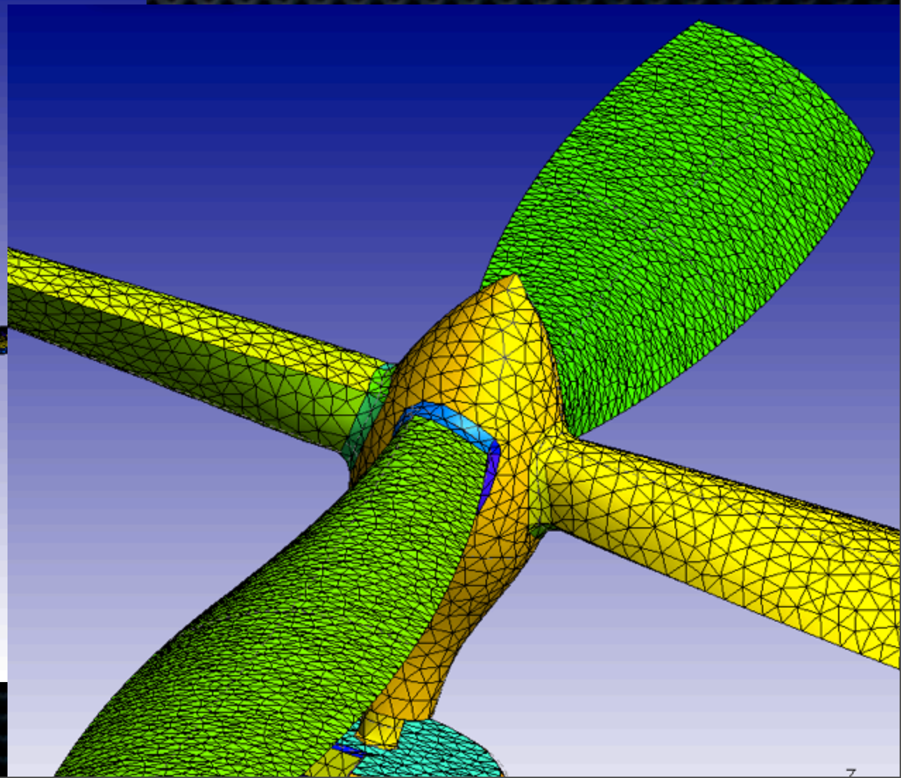
- http://www-roc.inria.fr/MACS/spip.php?rubrique69

Gmsh

- OpenSource meshing program with GUI and scripting language (for Mac, Unix, Win)
- Pro: Easy to use, cross plattform
- Cons: difficult for complex geometries, and no hybrid meshes (..?)
- Online: http://geuz.org/gmsh/ => **gmsh2.4**







Gmsh: basic scripting commands

Gmsh: basic scripting commands

- Point(0) = {x, y, z};

Gmsh: basic scripting commands

- Point(0) = {x, y, z};
- Line(0) = {0, 1};

Gmsh: basic scripting commands

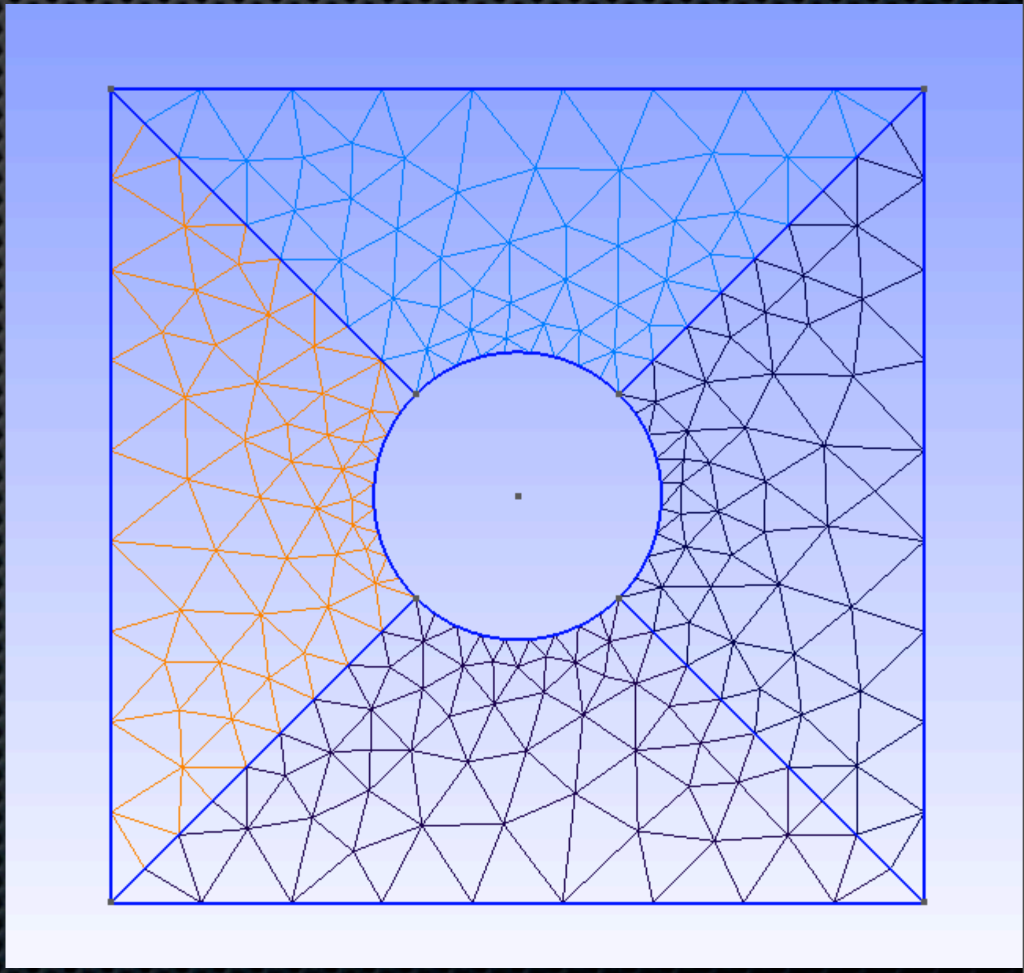
- Point(0) = {x, y, z};
- Line(0) = {0, 1};
- Transfinite Line {0} = N Using Progression P;

Gmsh: basic scripting commands

- Point(0) = {x, y, z};
- Line(0) = {0, 1};
- Transfinite Line {0} = N Using Progression P;
- Line Loop(2) = {0, 1, -2, -3};
  Plane Surface(1) = {2};

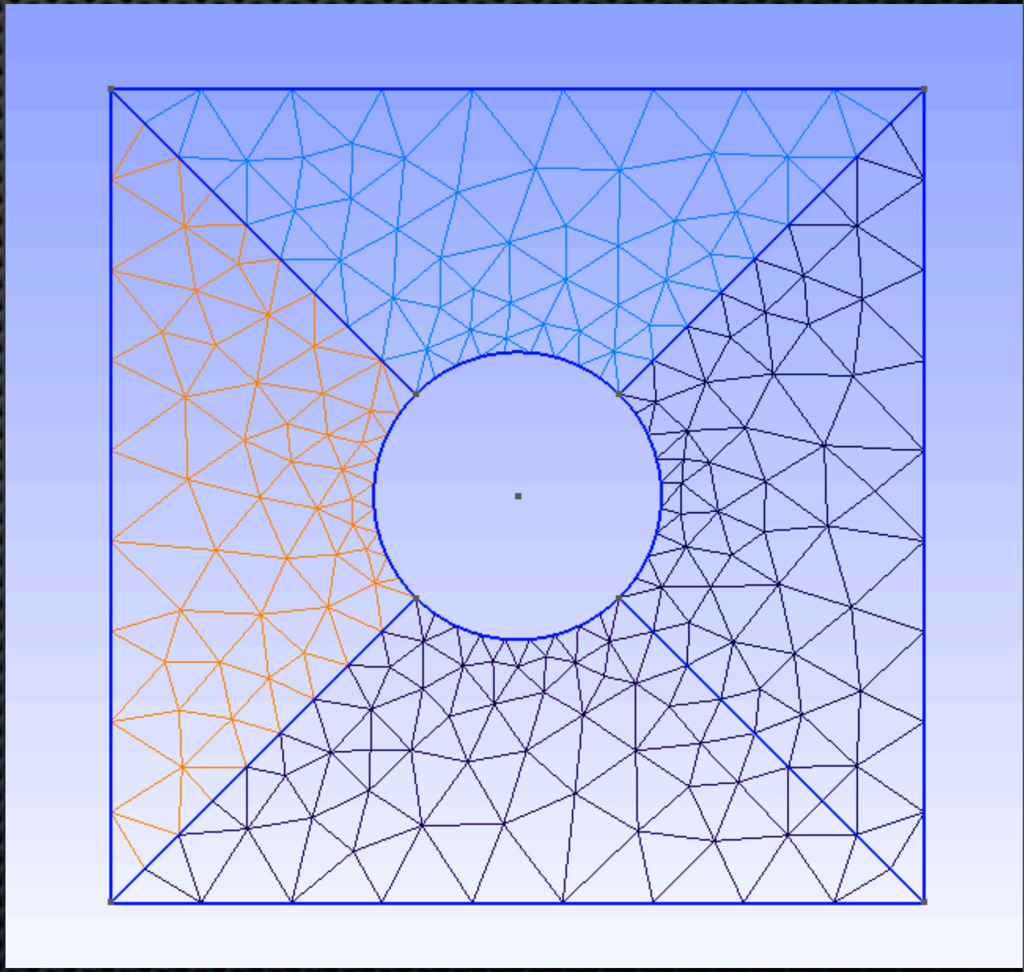
Gmsh: basic scripting commands

- Point(0) = {x, y, z};
- Line(0) = {0, 1};
- Transfinite Line {0} = N Using Progression P;
- Line Loop(2) = {0, 1, -2, -3};
  Plane Surface(1) = {2};



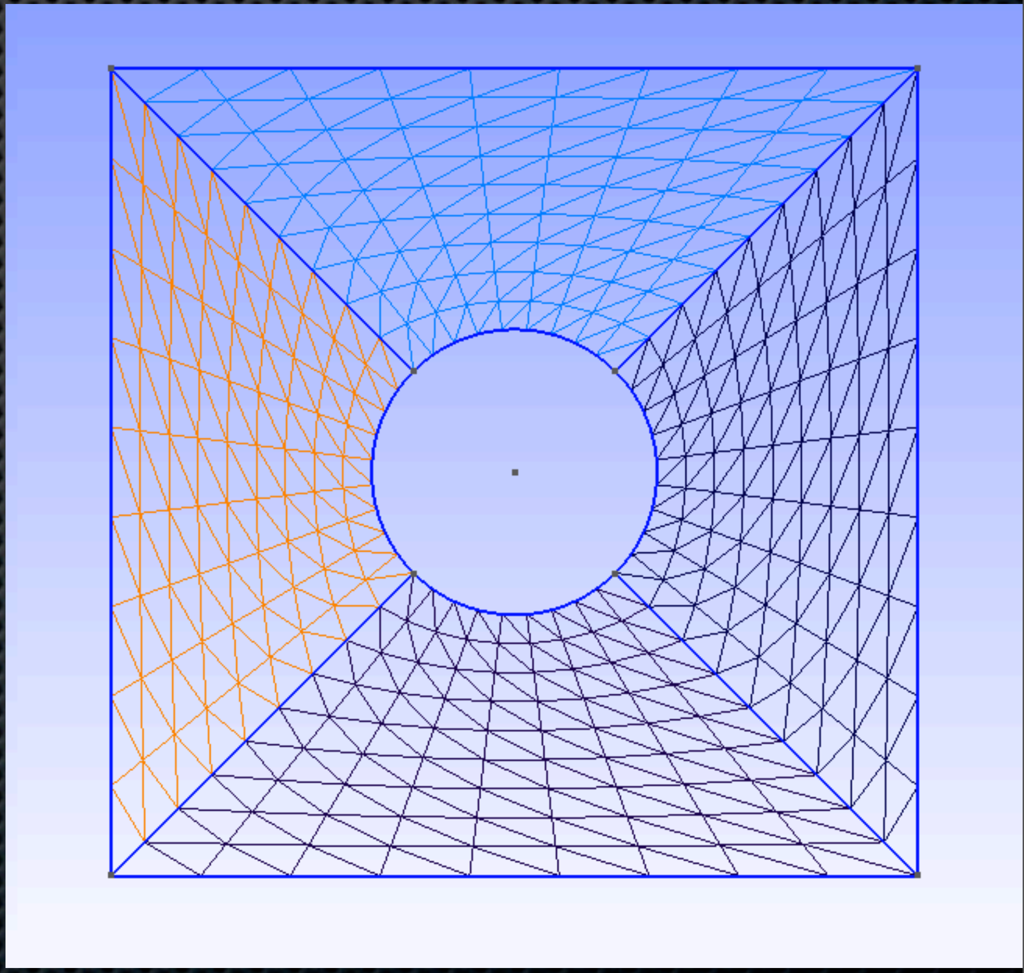
Gmsh: basic scripting commands

- Point(0) = {x, y, z};
- Line(0) = {0, 1};
- Transfinite Line {0} = N Using Progression P;
- Line Loop(2) = {0, 1, -2, -3};
  Plane Surface(1) = {2};
- Transfinite Surface {1};



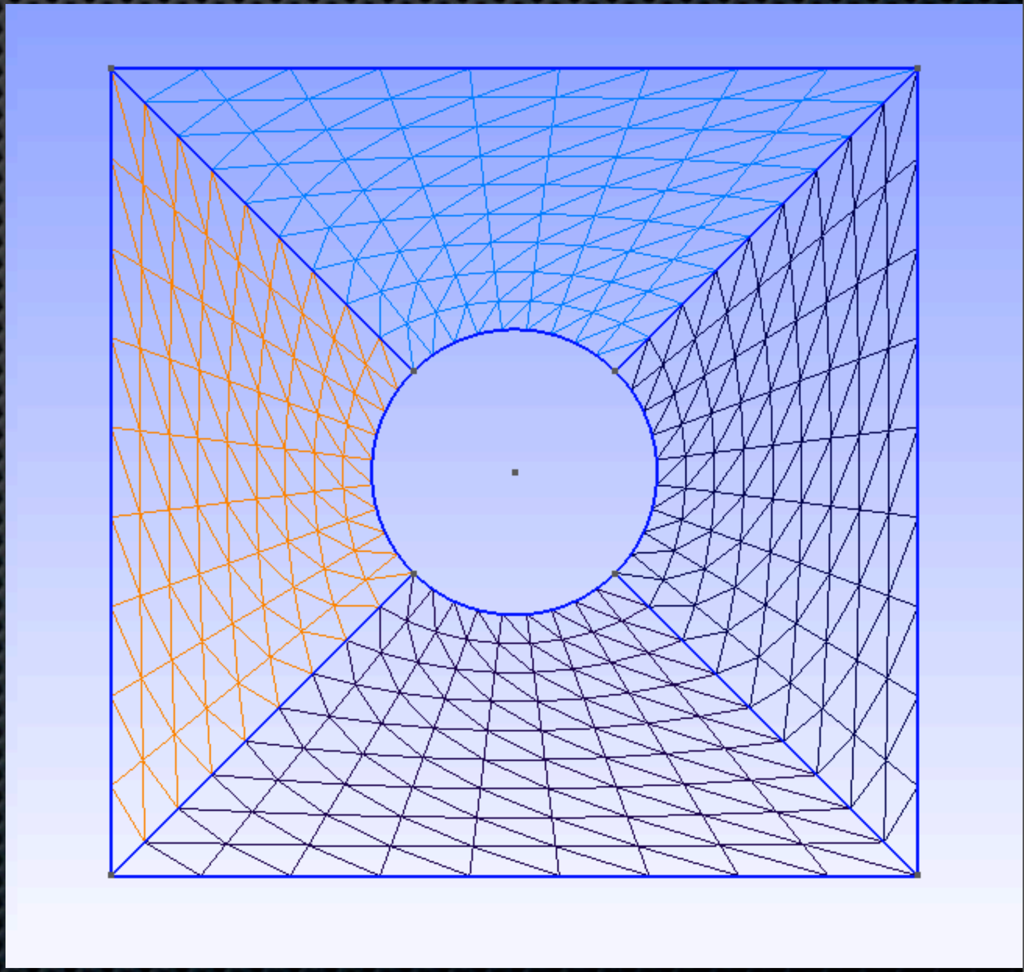
Gmsh: basic scripting commands

- Point(0) = {x, y, z};
- Line(0) = {0, 1};
- Transfinite Line {0} = N Using Progression P;
- Line Loop(2) = {0, 1, -2, -3};
  Plane Surface(1) = {2};
- Transfinite Surface {1};



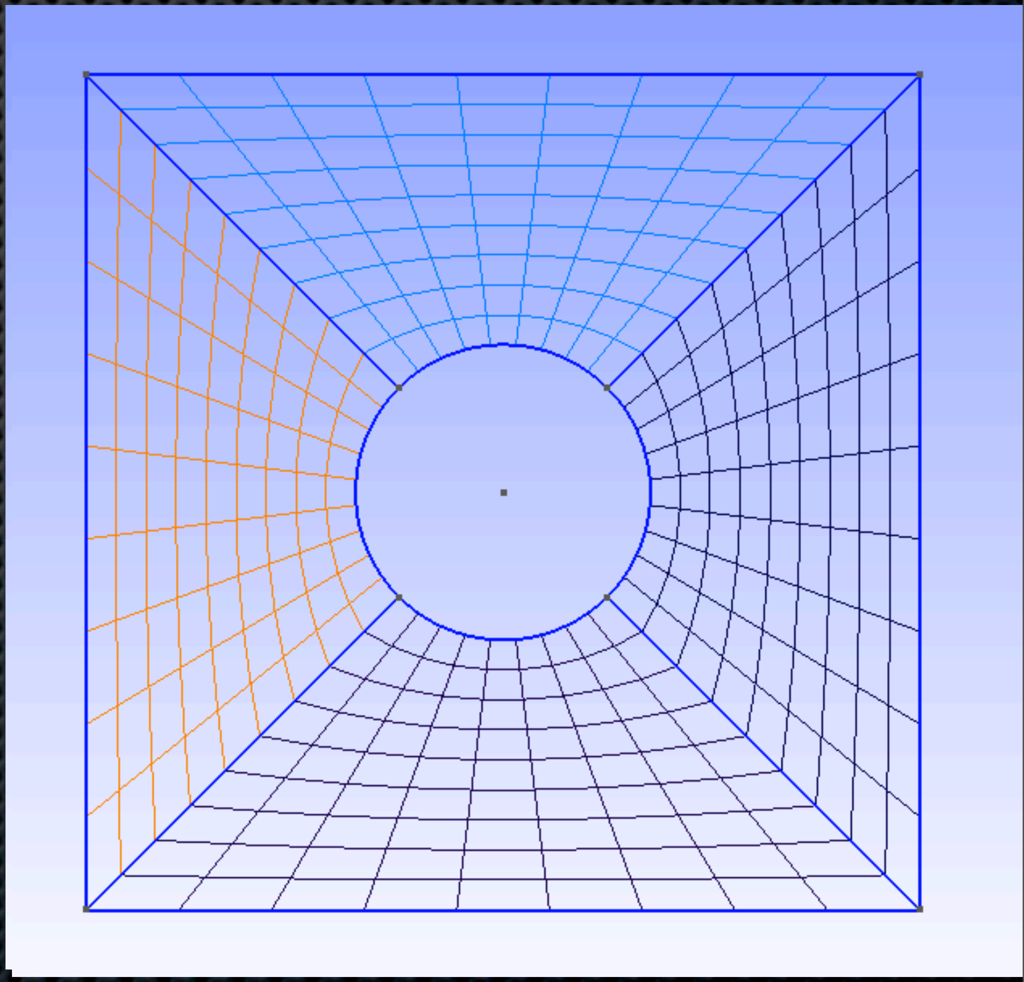
Gmsh: basic scripting commands

- Point(0) = {x, y, z};
- Line(0) = {0, 1};
- Transfinite Line {0} = N Using Progression P;
- Line Loop(2) = {0, 1, -2, -3};
  Plane Surface(1) = {2};
- Transfinite Surface {1};
- Recombine Surface {1};



Gmsh: basic scripting commands

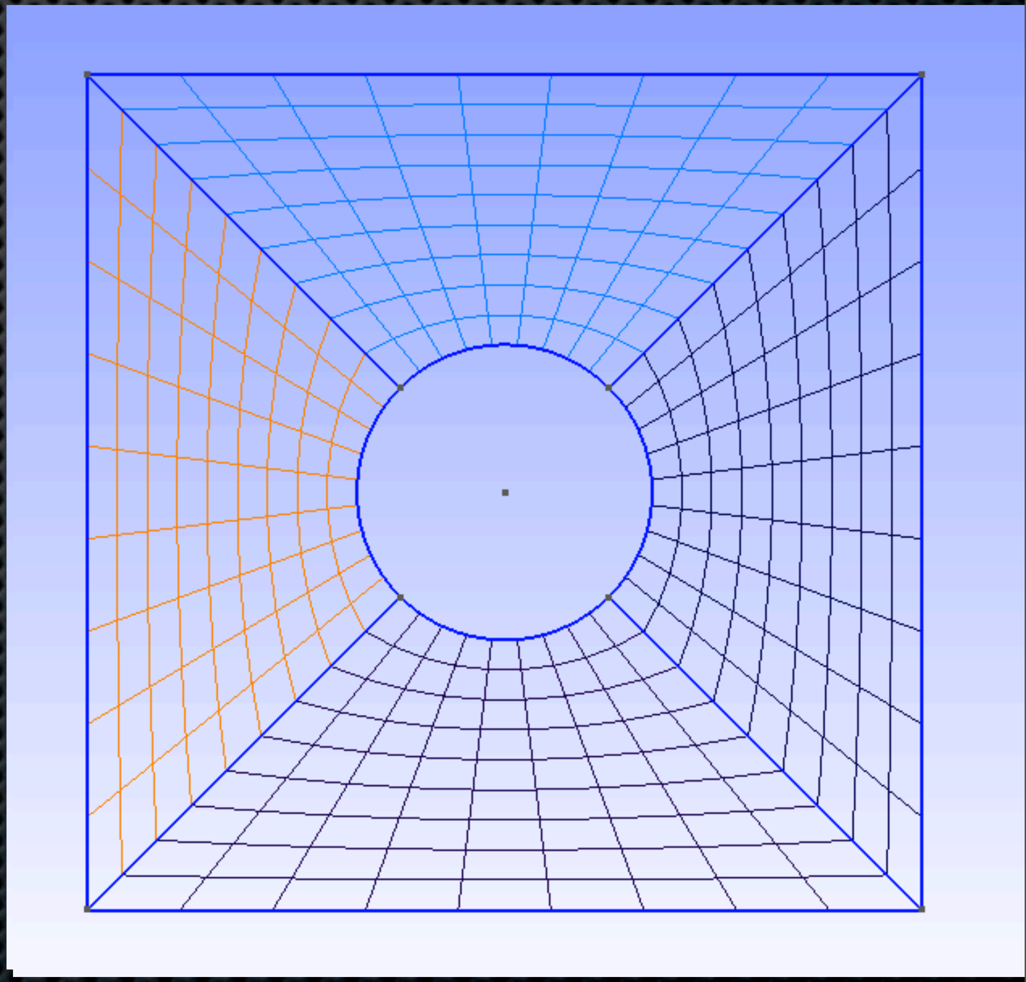
- Point(0) = {x, y, z};
- Line(0) = {0, 1};
- Transfinite Line {0} = N Using Progression P;
- Line Loop(2) = {0, 1, -2, -3};
  Plane Surface(1) = {2};
- Transfinite Surface {1};
- Recombine Surface {1};



Gmsh: basic scripting commands

- Point(0) = {x, y, z};
- Line(0) = {0, 1};
- Transfinite Line {0} = N Using Progression P;
- Line Loop(2) = {0, 1, -2, -3};
  Plane Surface(1) = {2};
- Transfinite Surface {1};
- Recombine Surface {1};
- Extrude {0, 0, H}

```
{
 Surface{1};
 Layers{{n1,n2,n3,n4},{.25,.5,.75,1}};
 Recombine;
}
```

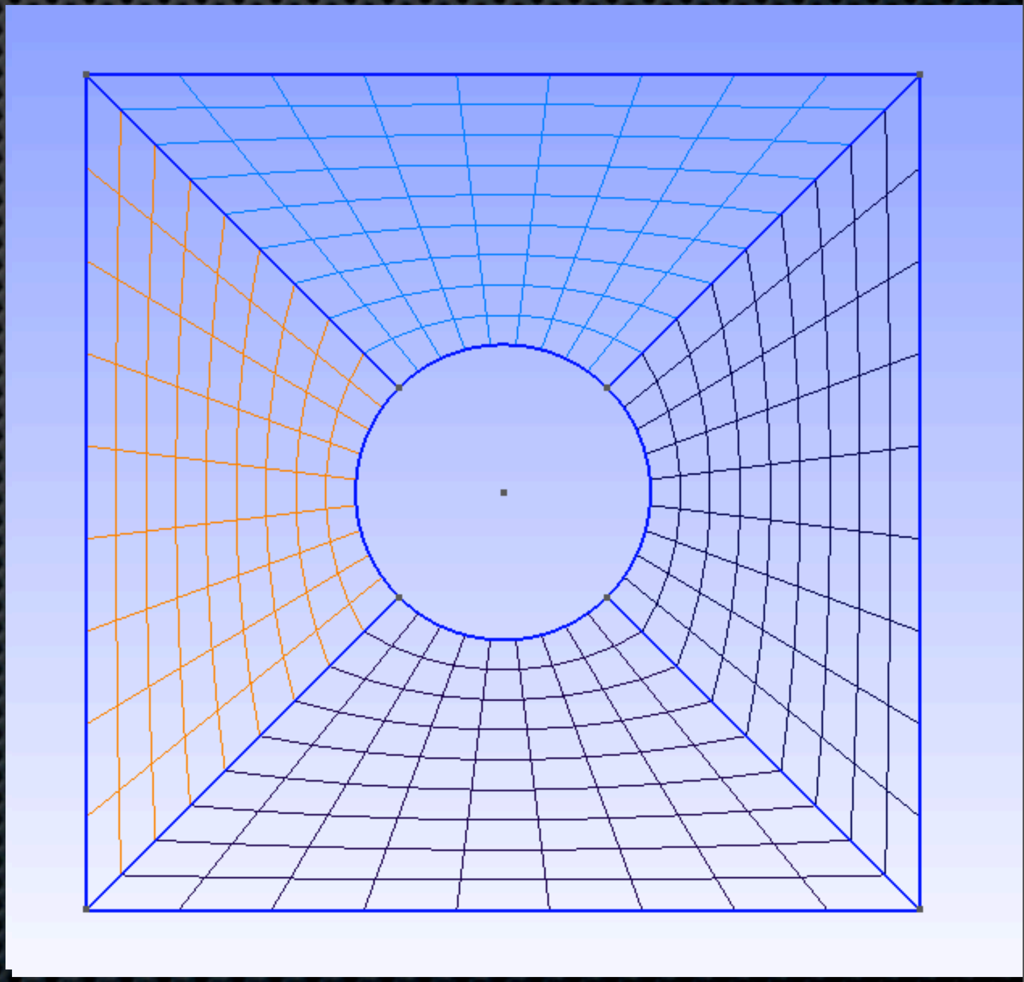


Gmsh: basic scripting commands

- Point(0) = {x, y, z};
- Line(0) = {0, 1};
- Transfinite Line {0} = N Using Progression P;
- Line Loop(2) = {0, 1, -2, -3};
  Plane Surface(1) = {2};
- Transfinite Surface {1};
- Recombine Surface {1};
- Extrude {0, 0, H}

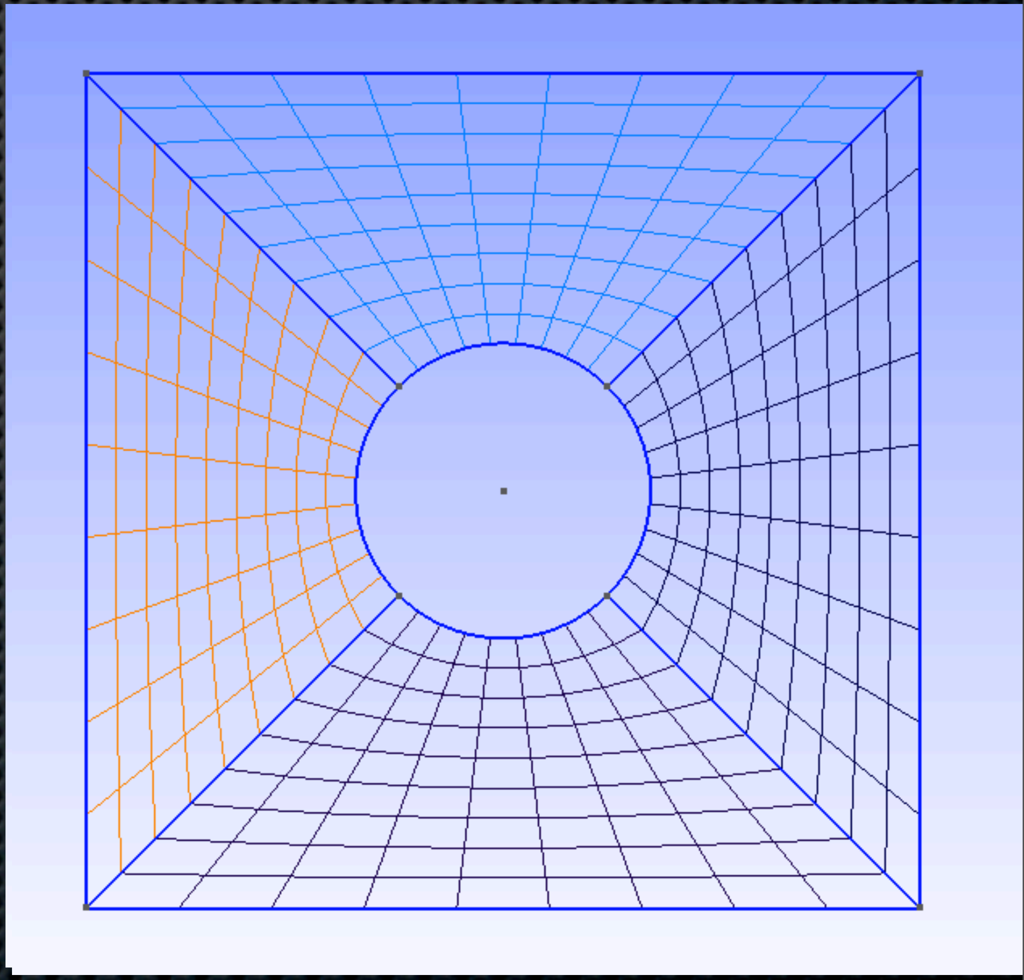
```
{
 Surface{1};
 Layers{{n1,n2,n3,n4},{.25,.5,.75,1}};
 Recombine;
}
```

- Physical Surface("Inlet") = {1};



Gmsh: basic scripting commands

- Point(0) = {x, y, z};
- Line(0) = {0, 1};
- Transfinite Line {0} = N Using Progression P;
- Line Loop(2) = {0, 1, -2, -3};
  Plane Surface(1) = {2};
- Transfinite Surface {1};
- Recombine Surface {1};
- Extrude {0, 0, H}
  {
   Surface{1};
   Layers{{n1,n2,n3,n4},{.25,.5,.75,1}};
   Recombine;
  }
- Physical Surface("Inlet") = {1};
- Physical Volume("AIR") = {1,2, 3, 4};



gmshToFoam

gmshToFoam

- Once created in .msh format, the mesh has to be converted for OpenFOAM

gmshToFoam

- Once created in .msh format, the mesh has to be converted for OpenFOAM
- Create directory System, which containes:

gmshToFoam

- Once created in .msh format, the mesh has to be converted for OpenFOAM
- Create directory System, which containes:

  ‘ControlDict’, ‘FvSchemes’ and ‘Fvsolution’

gmshToFoam

- Once created in .msh format, the mesh has to be converted for OpenFOAM
- Create directory System, which containes:

    ‘ControlDict’, ‘FvSchemes’ and ‘Fvsolution’

- type in terminal: gmshToFoam MyMesh.msh

CreateBaffles

CreateBaffles

- Example of the 2D spinnaker section: download the file (**http://www-roc.inria.fr/MACS/spip.php?rubrique69**)

CreateBaffles

- Example of the 2D spinnaker section: download the file (**http://www-roc.inria.fr/MACS/spip.php?rubrique69**)
- Create dirs and the files; run gmshToFoam

CreateBaffles

- Example of the 2D spinnaker section: download the file (**http://www-roc.inria.fr/MACS/spip.php?rubrique69**)
- Create dirs and the files; run gmshToFoam
- look at the output in terminal:

CreateBaffles

- Example of the 2D spinnaker section: download the file (**http://www-roc.inria.fr/MACS/spip.php?rubrique69**)
- Create dirs and the files; run gmshToFoam
- look at the output in terminal:

```
.....
Patch 0 gets name walls
Patch 1 gets name outlet
Patch 2 gets name wing
Patch 3 gets name inlet
.....
Writing zone 0 to cellZone Air and cellSet
.....
Writing zone 2 to faceZone faceZone_2 and faceSet
```

CreateBaffles

- Example of the 2D spinnaker section: download the file
  (**http://www-roc.inria.fr/MACS/spip.php?rubrique69**)
- Create dirs and the files; run gmshToFoam
- look at the output in terminal:

```
.....
Patch 0 gets name walls
Patch 1 gets name outlet
Patch 2 gets name wing
Patch 3 gets name inlet
.....
Writing zone 0 to cellZone Air and cellSet
.....
Writing zone 2 to faceZone faceZone_2 and faceSet
```

- The sail surface, called (in gmsh) ‘wing’ has the facezone label ‘2’ assigned.

CreateBaffles

- Example of the 2D spinnaker section: download the file (**http://www-roc.inria.fr/MACS/spip.php?rubrique69**)
- Create dirs and the files; run gmshToFoam
- look at the output in terminal:

```
.....
Patch 0 gets name walls
Patch 1 gets name outlet
Patch 2 gets name wing
Patch 3 gets name inlet
.....
Writing zone 0 to cellZone Air and cellSet
.....
Writing zone 2 to faceZone faceZone_2 and faceSet
```

- The sail surface, called (in gmsh) ‘wing’ has the facezone label ‘2’ assigned.
- type in terminal: createBaffles faceZone_2 wing

...and finally the end...

...and finally the end...

- The new mesh is written in a new directory, called as the time-step in the ControlDict (es: 0.02)

...and finally the end...

- The new mesh is written in a new directory, called as the time-step in the ControlDict (es: 0.02)
- Still a modification has to be done in constant/polymesh

...and finally the end...

- The new mesh is written in a new directory, called as the time-step in the ControlDict (es: 0.02)
- Still a modification has to be done in constant/polymesh
- All surfaces are now patches, therefore walls and empty entries are to be substituted:

...and finally the end...

- The new mesh is written in a new directory, called as the time-step in the ControlDict (es: 0.02)
- Still a modification has to be done in constant/polymesh
- All surfaces are now patches, therefore walls and empty entries are to be substituted:

```
walls
{
  type          patch;
  nFaces          244;
  startFace      38241;
}
```

```
walls
{
  type    wall;
  nFaces   244;
  startFace   38241;
}
```

Unsteady BC with ramp files

Unsteady BC with ramp files

- Very easy strategy for imposing variable boundary conditions
  Ex: variation in inlet velocity or body motion

Unsteady BC with ramp files

- Very easy strategy for imposing variable boundary conditions
  Ex: variation in inlet velocity or body motion
- Use a ‘ramp’ file for describing the variation in time

Unsteady BC with ramp files

- Very easy strategy for imposing variable boundary conditions
  Ex: variation in inlet velocity or body motion
- Use a 'ramp' file for describing the variation in time

```
Ramp
(
 ( 0.0  ( 0 0 0 ) )
 ( 0.015  ( .5 0 0 ) )
 ( 0.025  ( 0 0 0 ) )
 ( 0.05  ( -1 0 0 ) )
)
```

Unsteady BC with ramp files

- Very easy strategy for imposing variable boundary conditions
  Ex: variation in inlet velocity or body motion
- Use a ‘ramp’ file for describing the variation in time

Ramp

```
(
 ( 0.0  ( 0 0 0 ) )
 ( 0.015  ( .5 0 0 ) )
 ( 0.025  ( 0 0 0 ) )
 ( 0.05  ( -1 0 0 ) )
)
```

U (cellMotionU, PointMotionU.....)

```
boundaryField
{
   inlet
    {
      type
      timeVaryingUniformFixedValue;
      fileName "ramp"
      outOfBounds repeat;
    }
    ...
    ...
}
```

Force extraction

Force extraction

- Add in the system/controlDict file the entry for force extraction

Force extraction

- Add in the system/controlDict file the entry for force extraction
- Forces can be extracted with ‘libforce’, lift and drag coeffs with ‘libforceCoeffs’

Force extraction

- Add in the system/controlDict file the entry for force extraction
- Forces can be extracted with ‘libforce’, lift and drag coeffs with ‘libforceCoeffs’

controlDict

```
....
forces
{
  type forces;
  functionObjectLibs ("libforces.dylib"); // .dylib on Mac and .so on Linux
  outputControl outputTime;
  patches (wing); //Name of patche to integrate forces
  pName p;
  Uname U;
  rhoName rhoInf;
  rhoInf 1.2; //Reference density for fluid
  pRef 0;
  CofR (0 0 0); //Origin for moment calculations
}
```

Sampling

www.openfoam.com/docs/user

Sampling

www.openfoam.com/docs/user

- Add in workdir/system a file called sampleDict

Sampling

www.openfoam.com/docs/user

- Add in workdir/system a file called sampleDict
- Samples can be extracted during the execution, or post-execution, typing; ‘sample’

Sampling

www.openfoam.com/docs/user

- Add in workdir/system a file called sampleDict
- Samples can be extracted during the execution, or post-execution, typing; 'sample'

sampleDict

```
interpolationScheme cellPoint;
setFormat       raw;
sets
(
   MySample   //Filename
   {
      type    uniform;
      axis    xyz;
      start   ( 3.8 1.5 0.005 );
      end     ( 4 1.5 0.005 );
      nPoints 3;
   }
);
surfaces        ();
fields          ( U );
```

Sampling

www.openfoam.com/docs/user

- Add in workdir/system a file called sampleDict
- Samples can be extracted during the execution, or post-execution, typing; 'sample'

sampleDict

```
interpolationScheme cellPoint;
setFormat       raw;
sets
(
   MySample   //Filename
   {
      type    uniform;
      axis    xyz;
      start   ( 3.8 1.5 0.005 );
      end     ( 4 1.5 0.005 );
      nPoints 3;
   }
);
surfaces        ();
fields          ( U );
```

cell
cellPoint
CellPointFace

Sampling

www.openfoam.com/docs/user

- Add in workdir/system a file called sampleDict
- Samples can be extracted during the execution, or post-execution, typing; 'sample'

```
sampleDict
interpolationScheme cellPoint;
setFormat        raw;
sets
(
   MySample   //Filename
   {
      type    uniform;
      axis    xyz;
      start   ( 3.8 1.5 0.005 );
      end     ( 4 1.5 0.005 );
      nPoints 3;
   }
);
surfaces          ();
fields         ( U );
```

cell
cellPoint
CellPointFace

raw
GnuPlot
...

Sampling

www.openfoam.com/docs/user

- Add in workdir/system a file called sampleDict
- Samples can be extracted during the execution, or post-execution, typing; ‘sample’

sampleDict

```
interpolationScheme cellPoint;
setFormat       raw;
sets
(
    MySample    //Filename
    {
        type    uniform;
        axis    xyz;
        start   ( 3.8 1.5 0.005 );
        end     ( 4 1.5 0.005 );
        nPoints 3;
    }
);
surfaces        ();
fields          ( U );
```

cell
cellPoint
CellPointFace

raw
GnuPlot
...

uniform
face
midPoint
midPointAndFace
curve
cloud

Sampling

www.openfoam.com/docs/user

- Add in workdir/system a file called sampleDict
- Samples can be extracted during the execution, or post-execution, typing; 'sample'

```
sampleDict
interpolationScheme cellPoint;
setFormat       raw;
sets
(
   MySample   //Filename
   {
      type    uniform;
      axis    xyz;
      start   ( 3.8 1.5 0.005 );
      end     ( 4 1.5 0.005 );
      nPoints 3;
   }
);
surfaces        ();
fields          ( U );
```

cell
cellPoint
CellPointFace

raw
GnuPlot
...

uniform
face
midPoint
midPointAndFace
curve
cloud

x
y
z
xyz
distance

Sampling

www.openfoam.com/docs/user

- Add in workdir/system a file called sampleDict
- Samples can be extracted during the execution, or post-execution, typing; 'sample'

sampleDict

```
interpolationScheme cellPoint;
setFormat       raw;
sets
(
   MySample   //Filename
   {
      type    uniform;
      axis    xyz;
      start   ( 3.8 1.5 0.005 );
      end     ( 4 1.5 0.005 );
      nPoints 3;
   }
);
surfaces        ();
fields          ( U );
```

cell
cellPoint
CellPointFace

raw
GnuPlot
...

uniform
face
midPoint
midPointAndFace
curve
cloud

x
y
z
xyz
distance

patcName
interpolate
triangulate

Sampling

www.openfoam.com/docs/user

- Add in workdir/system a file called sampleDict
- Samples can be extracted during the execution, or post-execution, typing; 'sample'

sampleDict

```
interpolationScheme cellPoint;
setFormat       raw;
sets
(
   MySample   //Filename
   {
      type    uniform;
      axis    xyz;
      start   ( 3.8 1.5 0.005 );
      end     ( 4 1.5 0.005 );
      nPoints 3;
   }
);
surfaces        ();
fields          ( U );
```

cell
cellPoint
CellPointFace

raw
GnuPlot
...

uniform
face
midPoint
midPointAndFace
curve
cloud

x
y
z
xyz
distance

patcName
interpolate
triangulate

Airfoil case study:

pisoFOAM, SST turbulence model & unsteady BC

Airfoil case study:

pisoFOAM, SST turbulence model & unsteady BC

- pisoFOAM: unsteady solver of the SIMPLE family (Semi-Implicit Method for Pressure Linked Equations) => iterative method:

Airfoil case study:

pisoFOAM, SST turbulence model & unsteady BC

- pisoFOAM: unsteady solver of the SIMPLE family (Semi-Implicit Method for Pressure Linked Equations) => iterative method:

guess p* and solve u*

Airfoil case study:

pisoFOAM, SST turbulence model & unsteady BC

- pisoFOAM: unsteady solver of the SIMPLE family (Semi-Implicit Method for Pressure Linked Equations) => iterative method:

  guess p* and solve u*

  find correction for p* (by mean of ∇·u*=0)

Airfoil case study:

pisoFOAM, SST turbulence model & unsteady BC

- pisoFOAM: unsteady solver of the SIMPLE family (Semi-Implicit Method for Pressure Linked Equations) => iterative method:

guess p* and solve u*

find correction for p* (by mean of ∇·u*=0)

Airfoil case study:

pisoFOAM, SST turbulence model & unsteady BC

- pisoFOAM: unsteady solver of the SIMPLE family (Semi-Implicit Method for Pressure Linked Equations) => iterative method:

  guess p* and solve u*

  find correction for p* (by mean of $\nabla \cdot u^*=0$)

- SST: blend between k-ε (far from the wall) and k-ω (close to the wall). Good model for engineering type applications

Airfoil case study:

pisoFOAM, SST turbulence model & unsteady BC

- pisoFOAM: unsteady solver of the SIMPLE family (Semi-Implicit Method for Pressure Linked Equations) => iterative method:

  guess p^* and solve u^*

  find correction for p^* (by mean of $\nabla \cdot u^*=0$)

- SST: blend between k-ε (far from the wall) and k-ω (close to the wall). Good model for engineering type applications
- Set variable inlet velocity with a ramp file

Airfoil case study:

pisoFOAM, SST turbulence model & unsteady BC

- pisoFOAM: unsteady solver of the SIMPLE family (Semi-Implicit Method for Pressure Linked Equations) => iterative method:

  guess p* and solve u*

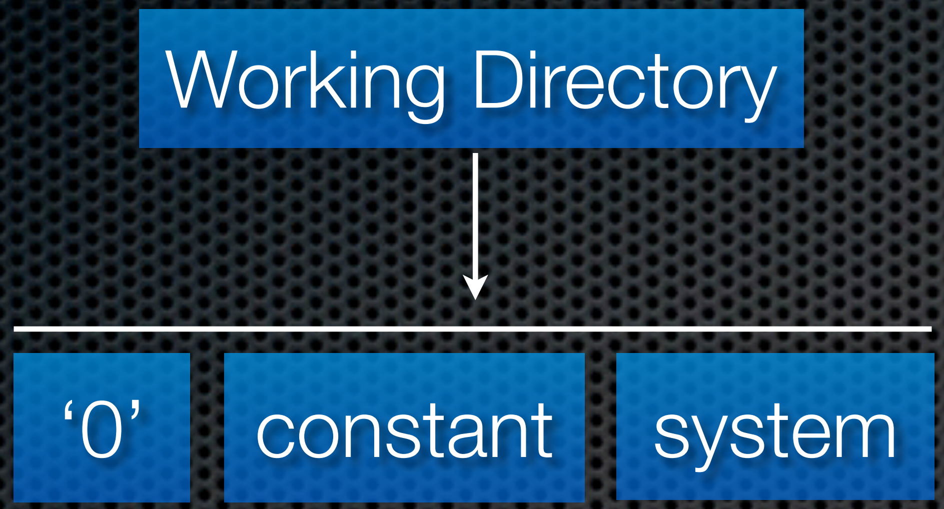
  find correction for p* (by mean of $\nabla \cdot u^* = 0$)

- SST: blend between k-ε (far from the wall) and k-ω (close to the wall). Good model for engineering type applications
- Set variable inlet velocity with a ramp file
- Extract forces on the wing

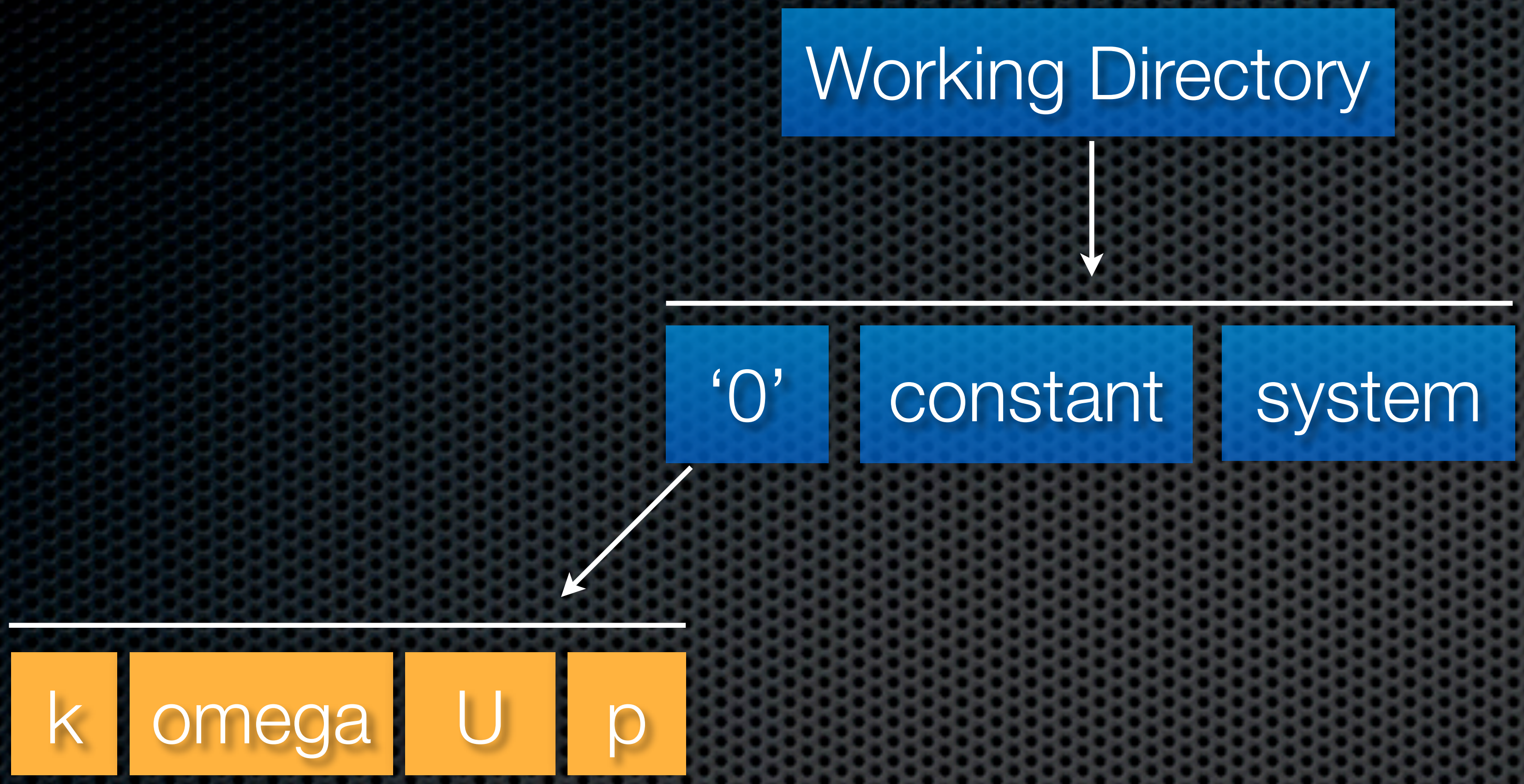
Airfoil case study: directory listing

Working Directory

Airfoil case study: directory listing



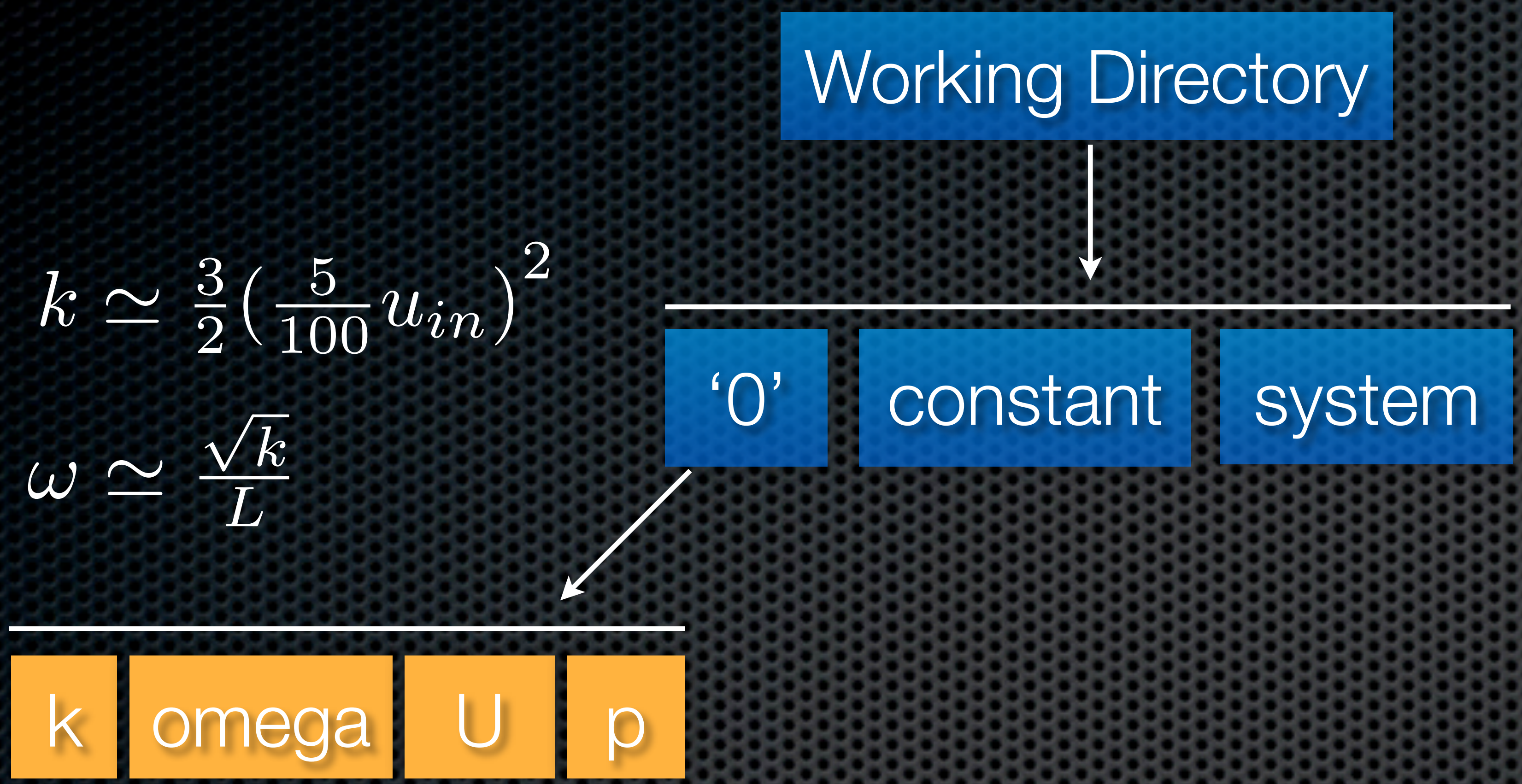
Airfoil case study: directory listing



Airfoil case study: directory listing

$$k \simeq \frac{3}{2}\left(\frac{5}{100}u_{in}\right)^2$$

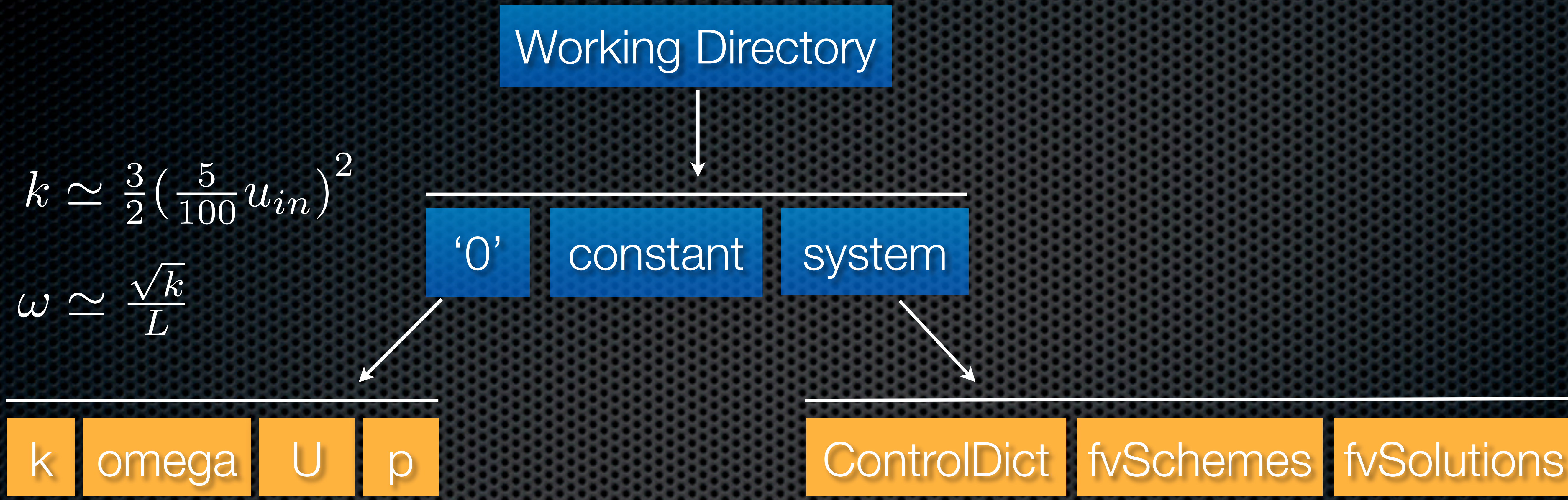
$$\omega \simeq \frac{\sqrt{k}}{L}$$



Airfoil case study: directory listing

$$k \simeq \frac{3}{2}\left(\frac{5}{100}u_{in}\right)^2$$

$$\omega \simeq \frac{\sqrt{k}}{L}$$



Airfoil case study: directory listing

$$k \simeq \frac{3}{2}\left(\frac{5}{100}u_{in}\right)^2$$

$$\omega \simeq \frac{\sqrt{k}}{L}$$

- Working Directory
  - '0'
    - k
    - omega
    - U
    - p
  - constant
    - polyMesh
    - RASProperties
    - transportProperties
    - turbulenceProperties
  - system
    - ControlDict
    - fvSchemes
    - fvSolutions

Airfoil case study: directory listing

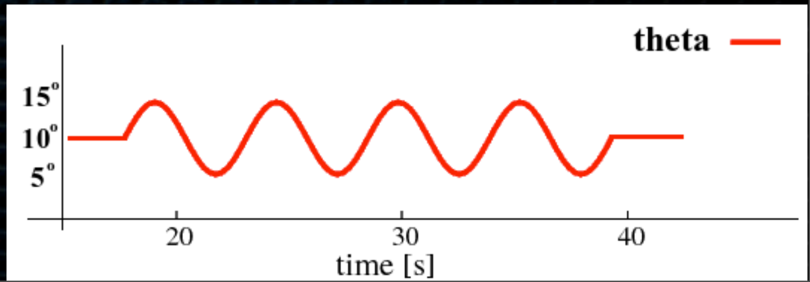
$$k \simeq \frac{3}{2}\left(\frac{5}{100}u_{in}\right)^2$$

$$\omega \simeq \frac{\sqrt{k}}{L}$$

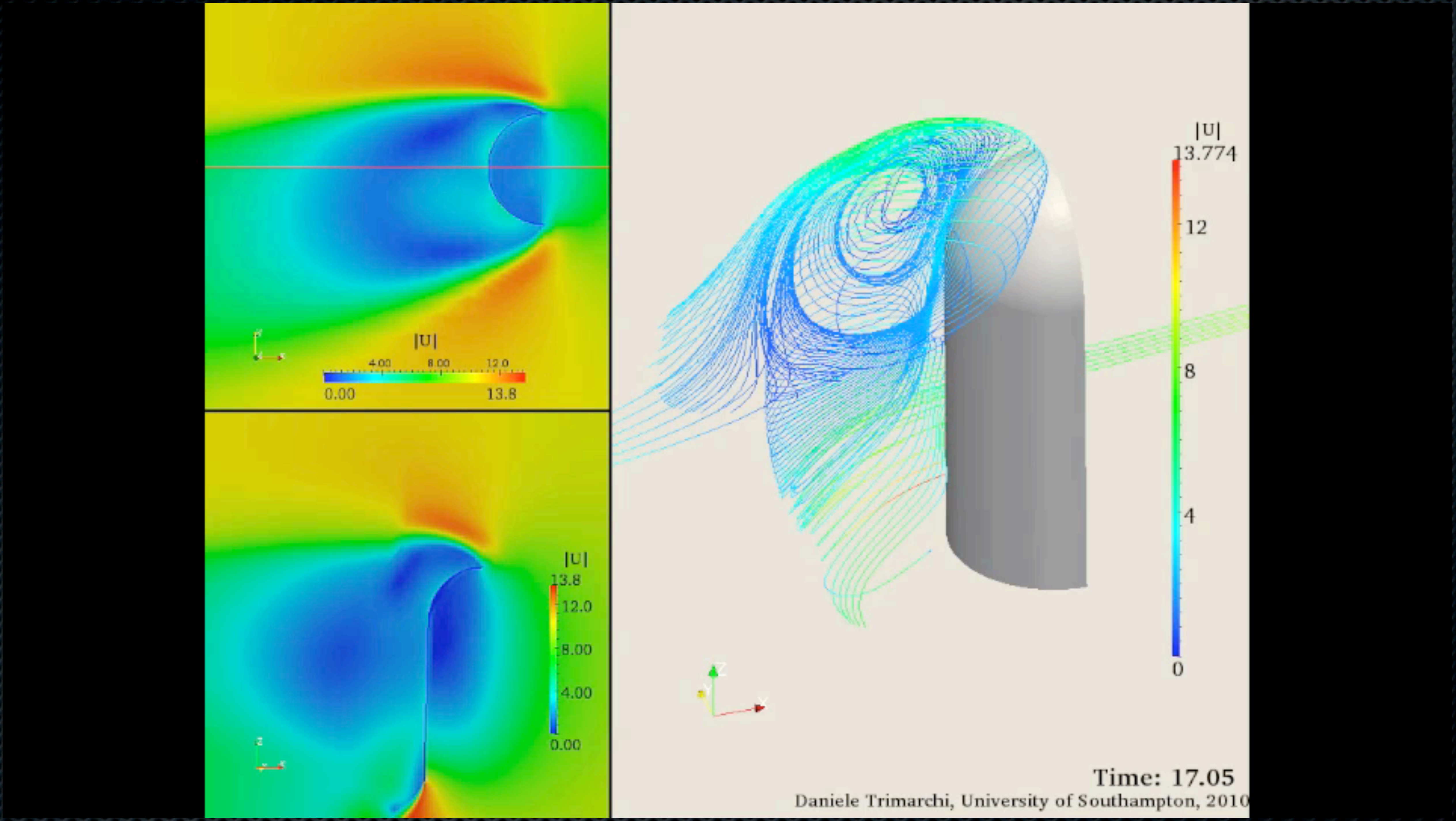
- Working Directory
  - '0'
    - k
    - omega
    - U
    - p
  - constant
    - polyMesh
      - boundary
      - cellZone
      - faces
      - faceZones
      - ...
    - RASProperties
    - transportProperties
    - turbulenceProperties
  - system
    - ControlDict
    - fvSchemes
    - fvSolutions

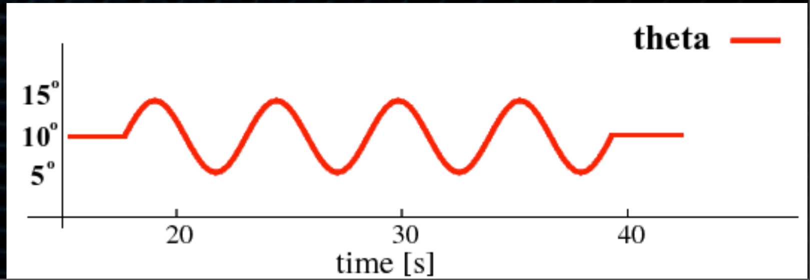
Case:

undeformable Spinnaker geometry in gusts



Case:
undeformable Spinnaker geometry in gusts





Principles of mesh motion:
ALE and the PimpleDyMFOAM solver

Principles of mesh motion:
ALE and the PimpleDyMFOAM solver

- Arbitrary Lagrangian Eulerian methods:

Principles of mesh motion: ALE and the PimpleDyMFOAM solver

- Arbitrary Lagrangian Eulerian methods:
  - Deform the Eulerian fluid mesh in lagrangian way on the moving interface

Principles of mesh motion:
ALE and the PimpleDyMFOAM solver

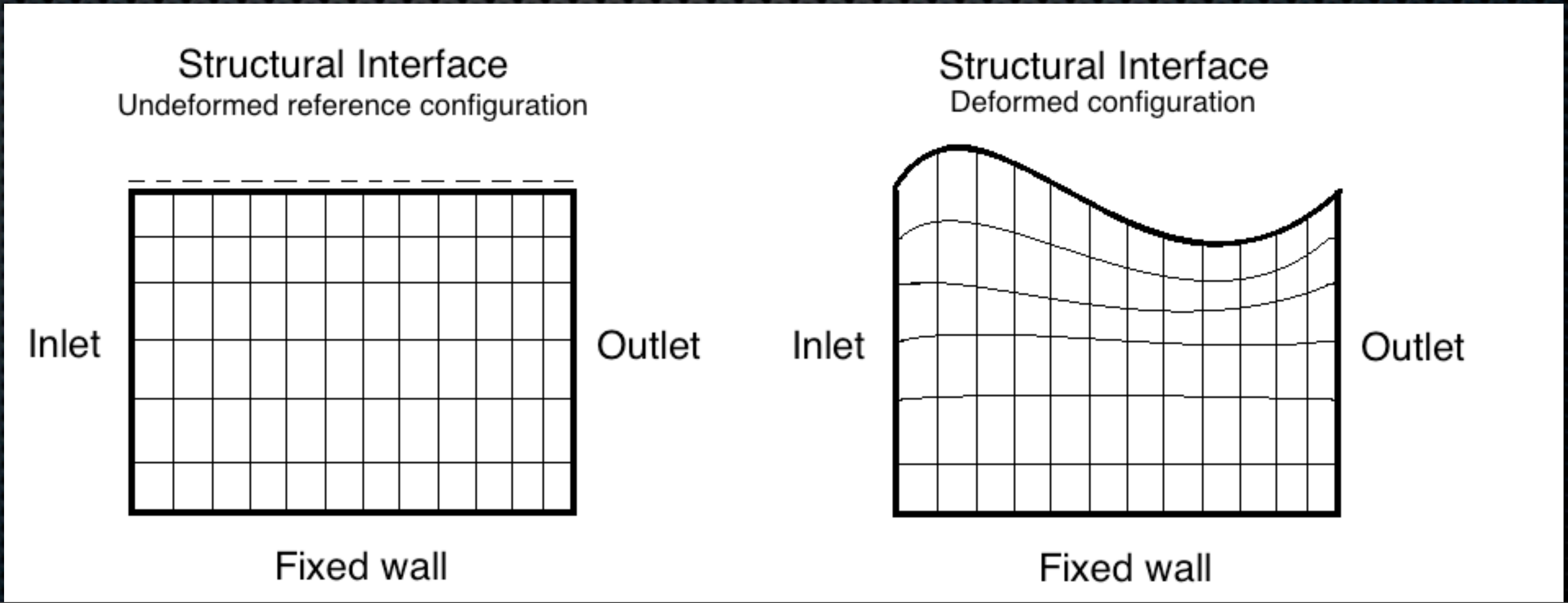
- Arbitrary Lagrangian Eulerian methods:
  - Deform the Eulerian fluid mesh in lagrangian way on the moving interface
  - Mesh boundaries remain unchanged

Principles of mesh motion: ALE and the PimpleDyMFOAM solver

- Arbitrary Lagrangian Eulerian methods:
  - Deform the Eulerian fluid mesh in lagrangian way on the moving interface
  - Mesh boundaries remain unchanged
  - In the domain all is allowed, provided that some regularity is respected

Principles of mesh motion: ALE and the PimpleDyMFOAM solver

- Arbitrary Lagrangian Eulerian methods:
  - Deform the Eulerian fluid mesh in lagrangian way on the moving interface
  - Mesh boundaries remain unchanged
  - In the domain all is allowed, provided that some regularity is respected



Principles of mesh motion:

Principles of mesh motion:

The Grid has now its own velocity => this should be considered in order to be conservative

Principles of mesh motion:

The Grid has now its own velocity => this should be considered in order to be conservative

Navier-Stokes equation are reformulated in the ALE framework, by taking into account the mesh velocity w

Principles of mesh motion:

The Grid has now its own velocity => this should be considered in order to be conservative

Navier-Stokes equation are reformulated in the ALE framework, by taking into account the mesh velocity w

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} - \nu\Delta\boldsymbol{u} + \nabla p = \boldsymbol{f} \\ \nabla\cdot\boldsymbol{u} = 0 \end{cases} \quad \text{in } \Omega \subset \mathbb{R}^d$$

Principles of mesh motion:

The Grid has now its own velocity => this should be considered in order to be conservative

Navier-Stokes equation are reformulated in the ALE framework, by taking into account the mesh velocity w

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} - \nu\Delta\boldsymbol{u} + \nabla p = \boldsymbol{f} \\ \nabla\cdot\boldsymbol{u} = 0 \end{cases} \quad \text{in} \quad \Omega\subset\mathbb{R}^d$$

$$\begin{cases} \left.\dfrac{\partial \boldsymbol{u}}{\partial t}\right|_{\boldsymbol{\xi}} + [(\boldsymbol{u}-\boldsymbol{w})\cdot\nabla_{\boldsymbol{x}}]\,\boldsymbol{u} - \nu\Delta_{\boldsymbol{x}}\boldsymbol{u} + \nabla_{\boldsymbol{x}} p = \boldsymbol{f} \\ \nabla_{\boldsymbol{x}}\cdot\boldsymbol{u} = 0 \end{cases} \quad \text{in} \quad \Omega\subset\mathbb{R}^d.$$

Principles of mesh motion:

The Grid has now its own velocity => this should be considered in order to be conservative

Navier-Stokes equation are reformulated in the ALE framework, by taking into account the mesh velocity w

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} - \nu \Delta \boldsymbol{u} + \nabla p = \boldsymbol{f} \\ & \text{in} \quad \Omega \subset \mathbb{R}^d \\ \nabla \cdot \boldsymbol{u} = 0 \end{cases}$$

$$\begin{cases} \left.\dfrac{\partial \boldsymbol{u}}{\partial t}\right|_{\boldsymbol{\xi}} + [(\boldsymbol{u} - \boldsymbol{w}) \cdot \nabla_{\boldsymbol{x}}]\, \boldsymbol{u} - \nu \Delta_{\boldsymbol{x}} \boldsymbol{u} + \nabla_{\boldsymbol{x}} p = \boldsymbol{f} \\ & \text{in} \quad \Omega \subset \mathbb{R}^d. \\ \nabla_{\boldsymbol{x}} \cdot \boldsymbol{u} = 0 \end{cases}$$

Formally, the convective term only changes

Principles of mesh motion:

The Grid has now its own velocity => this should be considered in order to be conservative

Navier-Stokes equation are reformulated in the ALE framework, by taking into account the mesh velocity w

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} - \nu \Delta \boldsymbol{u} + \nabla p = \boldsymbol{f} & \\ & \text{in} \quad \Omega \subset \mathbb{R}^d \\ \nabla \cdot \boldsymbol{u} = 0 & \end{cases}$$

$$\begin{cases} \left.\dfrac{\partial \boldsymbol{u}}{\partial t}\right|_{\boldsymbol{\xi}} + [(\boldsymbol{u} - \boldsymbol{w}) \cdot \nabla_{\boldsymbol{x}}]\, \boldsymbol{u} - \nu \Delta_{\boldsymbol{x}} \boldsymbol{u} + \nabla_{\boldsymbol{x}} p = \boldsymbol{f} & \\ & \text{in} \quad \Omega \subset \mathbb{R}^d. \\ \nabla_{\boldsymbol{x}} \cdot \boldsymbol{u} = 0 & \end{cases}$$

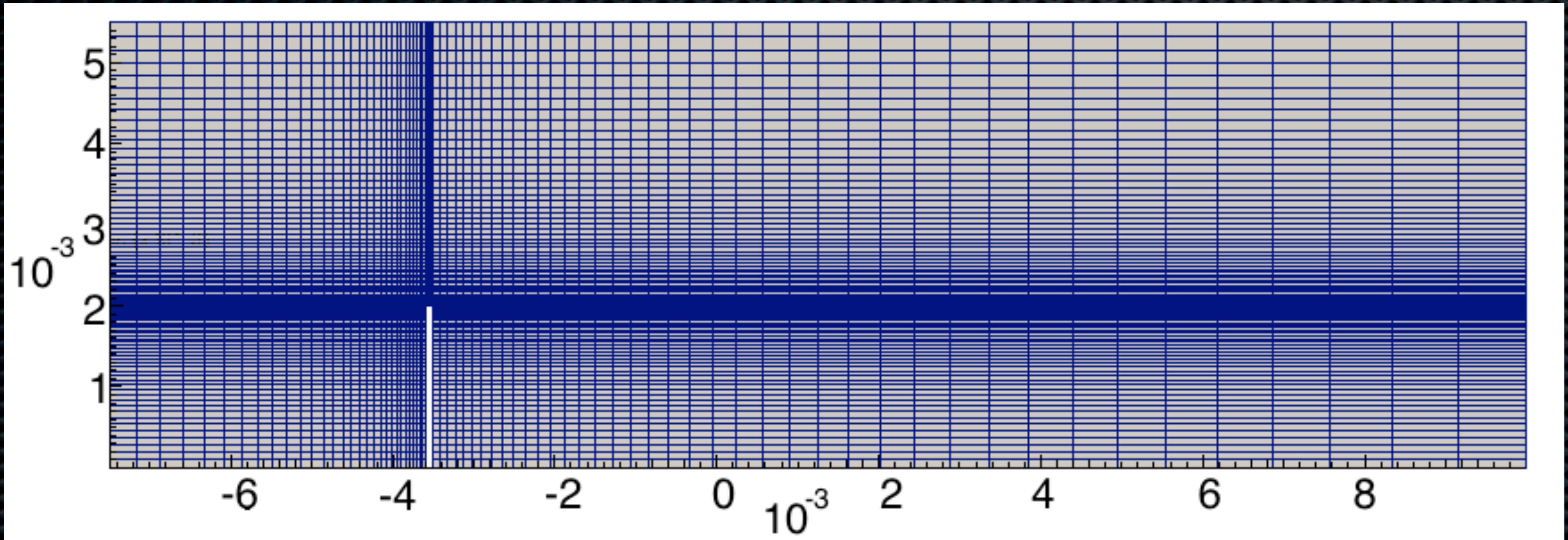
Formally, the convective term only changes

This is equivalent to make velocities relative to the mesh motion. In pimpleDyMFOAM:

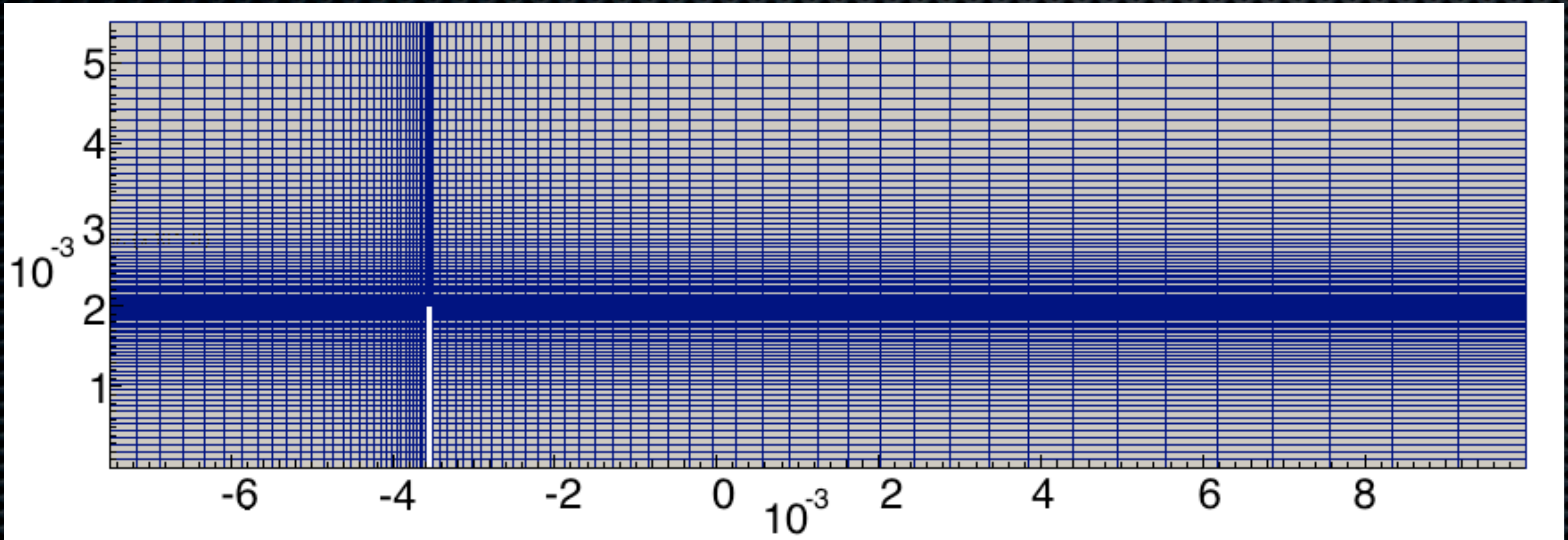
```
// Make the fluxes relative to the mesh motion
fvc::makeRelative(phi, U);
```

Set up the case for the moving beam:

Set up the case for the moving beam:

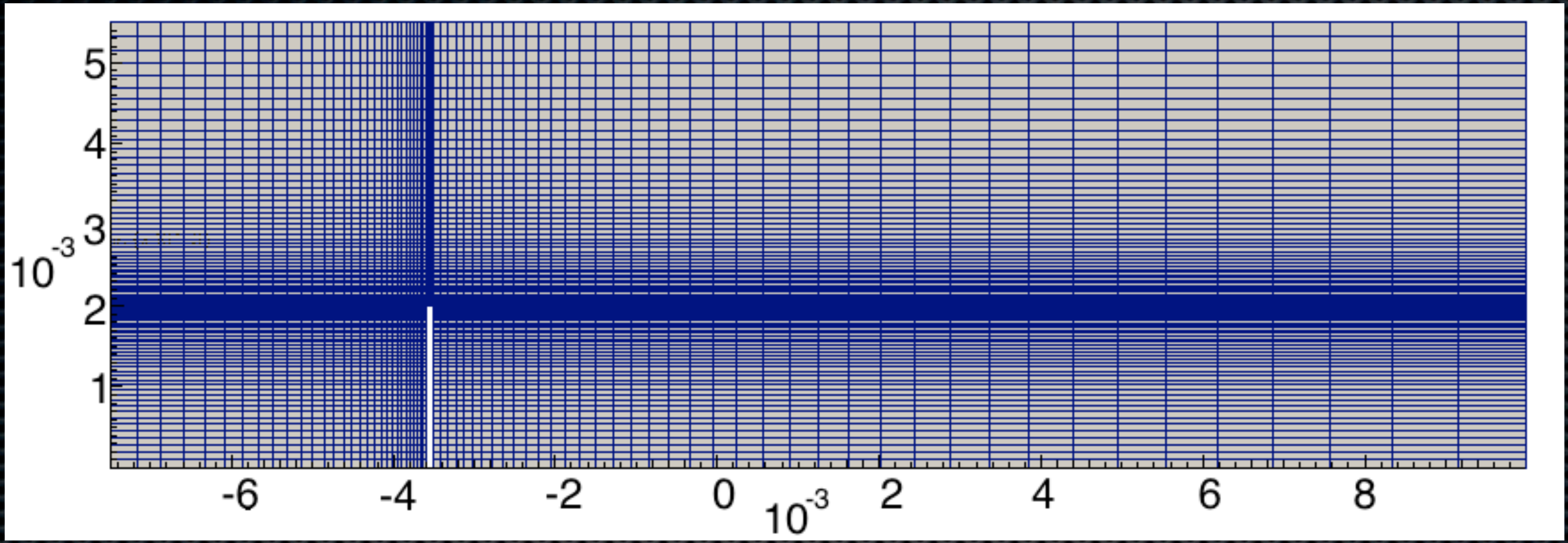


Set up the case for the moving beam:



	Beam	Top	Bottom	Inlet	Outlet
U	ramp	0	0	pressureInletOutletVelocity	0
p	ZeroG.	ZeroG.	ZeroG.	totalPressure - 0	ZeroG.

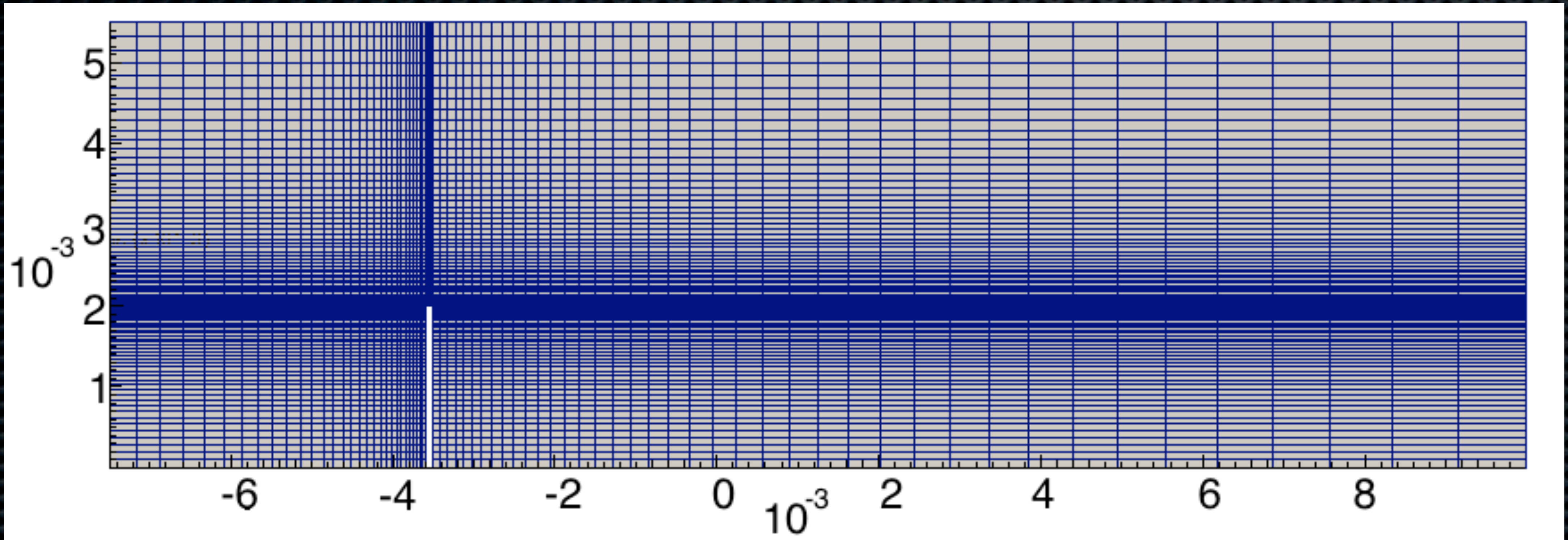
Set up the case for the moving beam:



	Beam	Top	Bottom	Inlet	Outlet
U	ramp	0	0	pressureInletOutletVelocity	0
p	ZeroG.	ZeroG.	ZeroG.	totalPressure - 0	ZeroG.

A dynamic mesh type and a mesh motion solver have to be chosen

Set up the case for the moving beam:

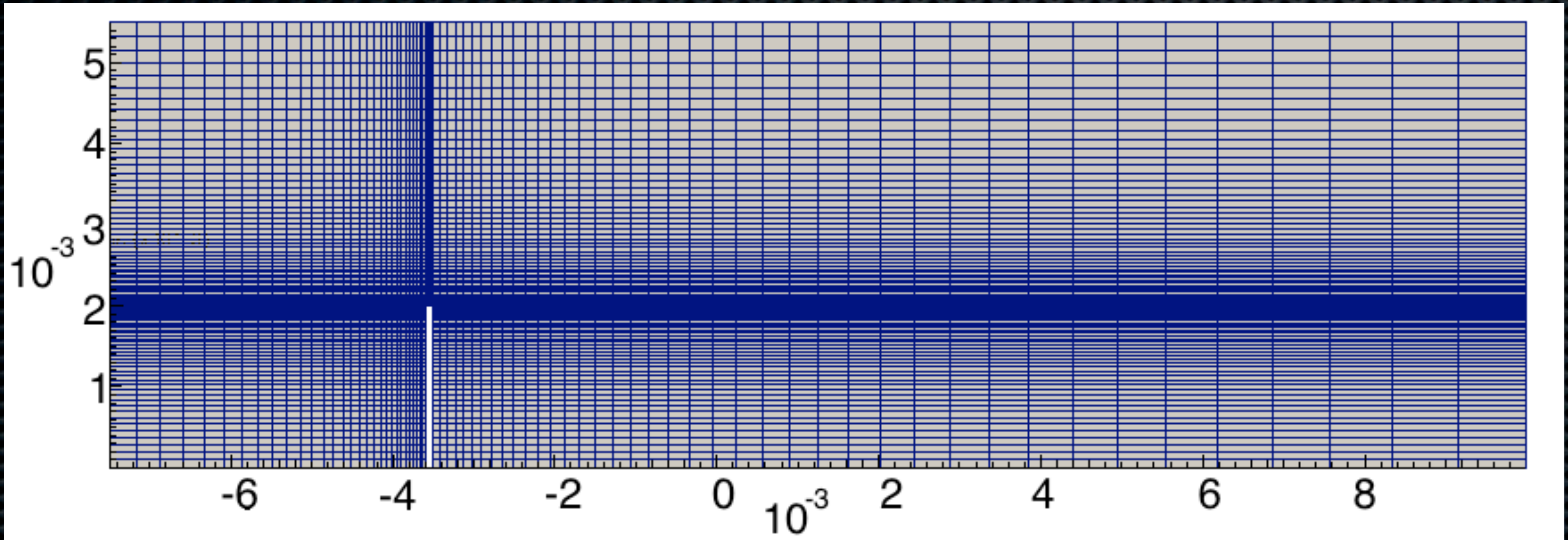


	Beam	Top	Bottom	Inlet	Outlet
U	ramp	0	0	pressureInletOutletVelocity	0
p	ZeroG.	ZeroG.	ZeroG.	totalPressure - 0	ZeroG.

A dynamic mesh type and a mesh motion solver have to be chosen

This is done in Constant/dynamicMeshDict

Set up the case for the moving beam:



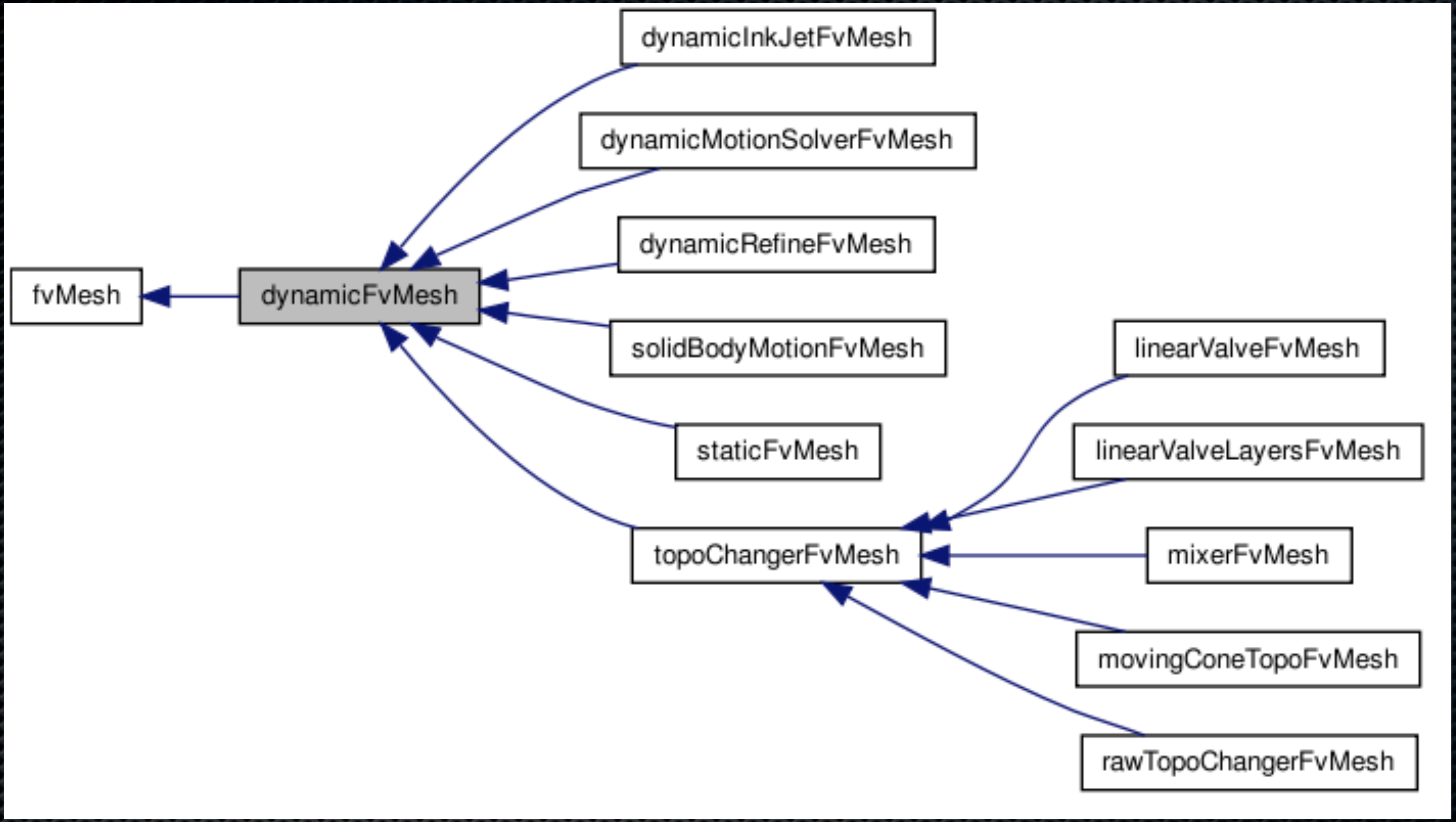
	Beam	Top	Bottom	Inlet	Outlet
U	ramp	0	0	pressureInletOutletVelocity	0
p	ZeroG.	ZeroG.	ZeroG.	totalPressure - 0	ZeroG.

A dynamic mesh type and a mesh motion solver have to be chosen

This is done in Constant/dynamicMeshDict

This choice determines the additional input files to be added

Choice of the dynamic mesh class:

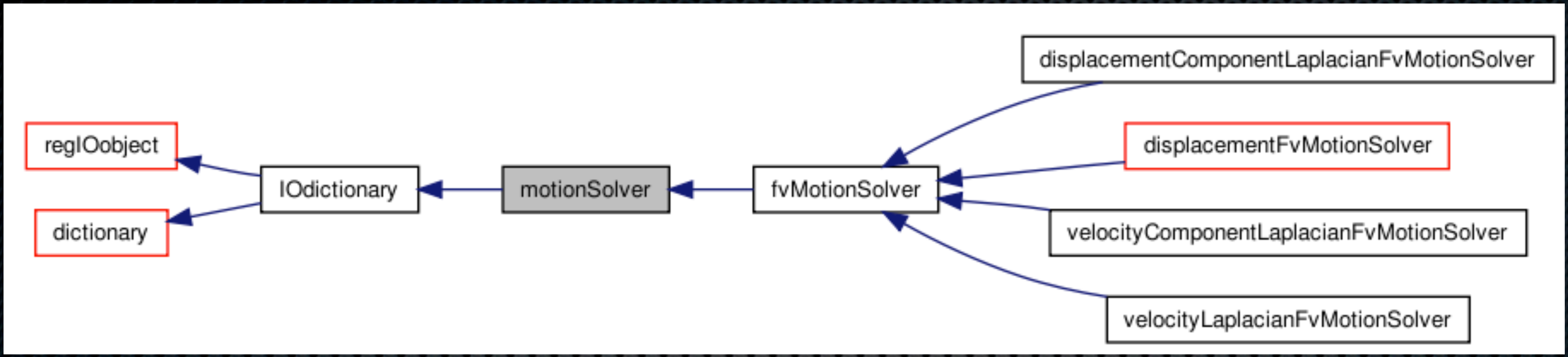


dynamicInkJetFvMesh: moves the mesh using analytical expression (See tutorial from Gonzales, Chalmers university)

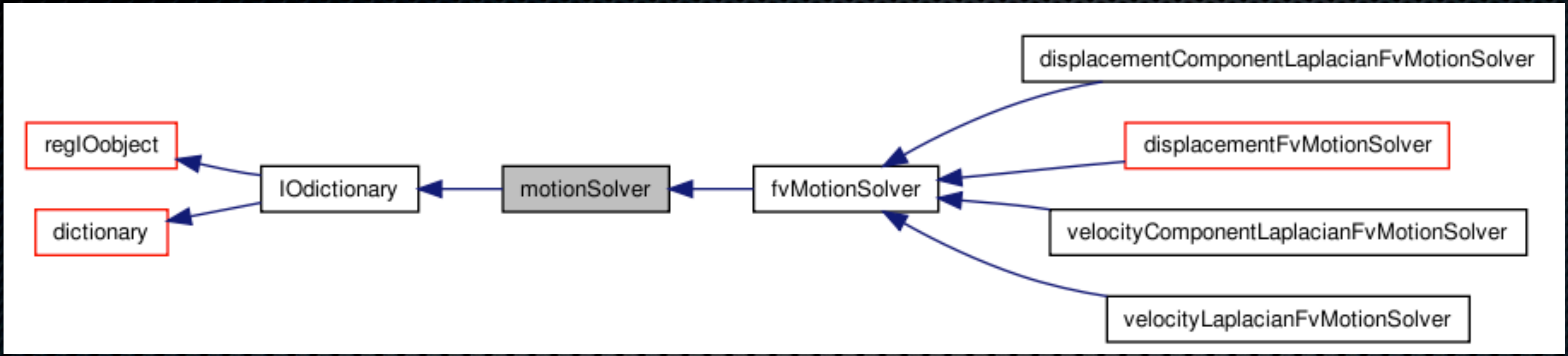
dynamicMotionSolverFvMesh: prescribes mesh motions, for example on boundaries, and it allows the direct specification of mesh points motions (velocities or displacements).

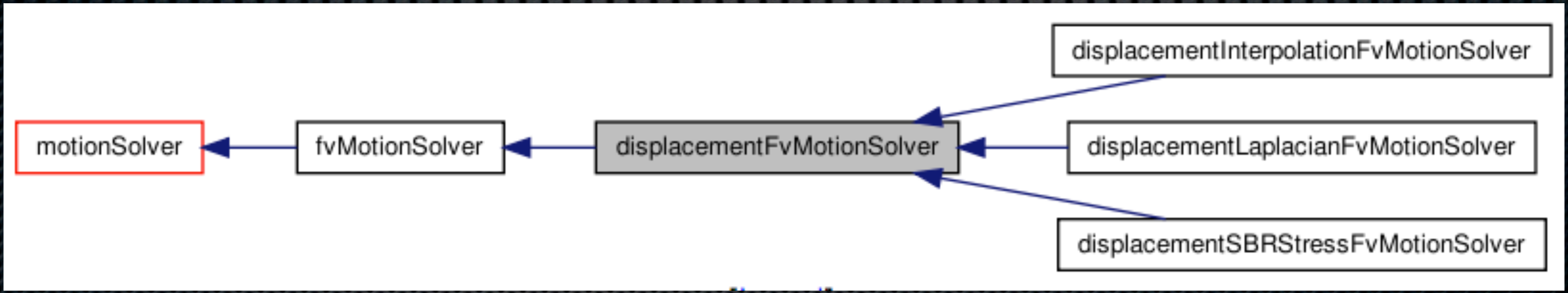
Choice of the motion solver:

Choice of the motion solver:

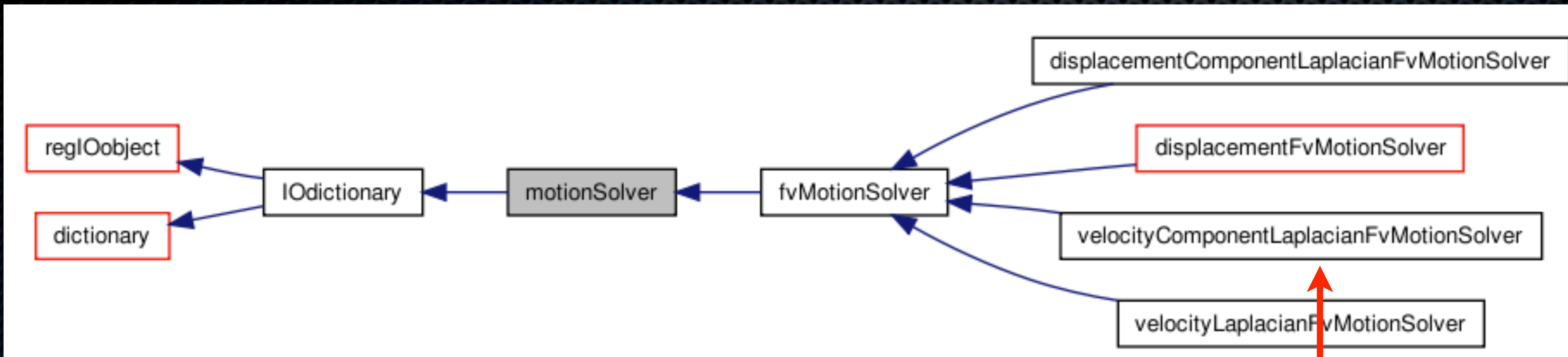


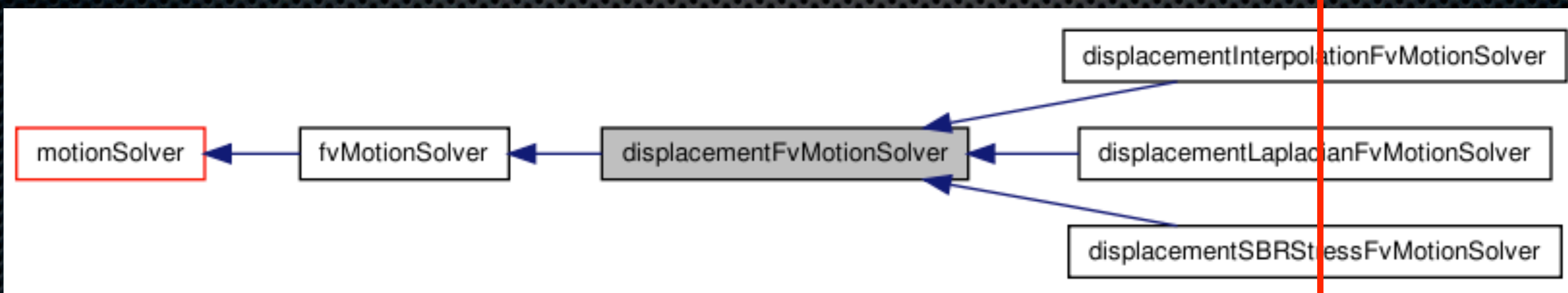
Choice of the motion solver:





Choice of the motion solver:





Constant/dinamicMeshDict

```
------------------------------------------------------------
dynamicFvMesh  dynamicMotionSolverFvMesh;
motionSolverLibs ( "libfvMotionSolvers.so" );
solver          velocityLaplacian;
diffusivity     directional ( 1 200 0 );
------------------------------------------------------------
```

Using the velocityLaplacianMotionSolver:

Using the velocityLaplacianMotionSolver:

- Basic Idea: define the motion of the boundaries in terms of velocity, and solve a laplacian (therefore apply a diffusion) in the rest of the domain

Using the velocityLaplacianMotionSolver:

- Basic Idea: define the motion of the boundaries in terms of velocity, and solve a laplacian (therefore apply a diffusion) in the rest of the domain
- The motion can be defined as a constant velocity or via a ‘ramp’ file

Using the velocityLaplacianMotionSolver:

- Basic Idea: define t … in terms of velocity, and solve a laplacia … n) in the rest of the domain

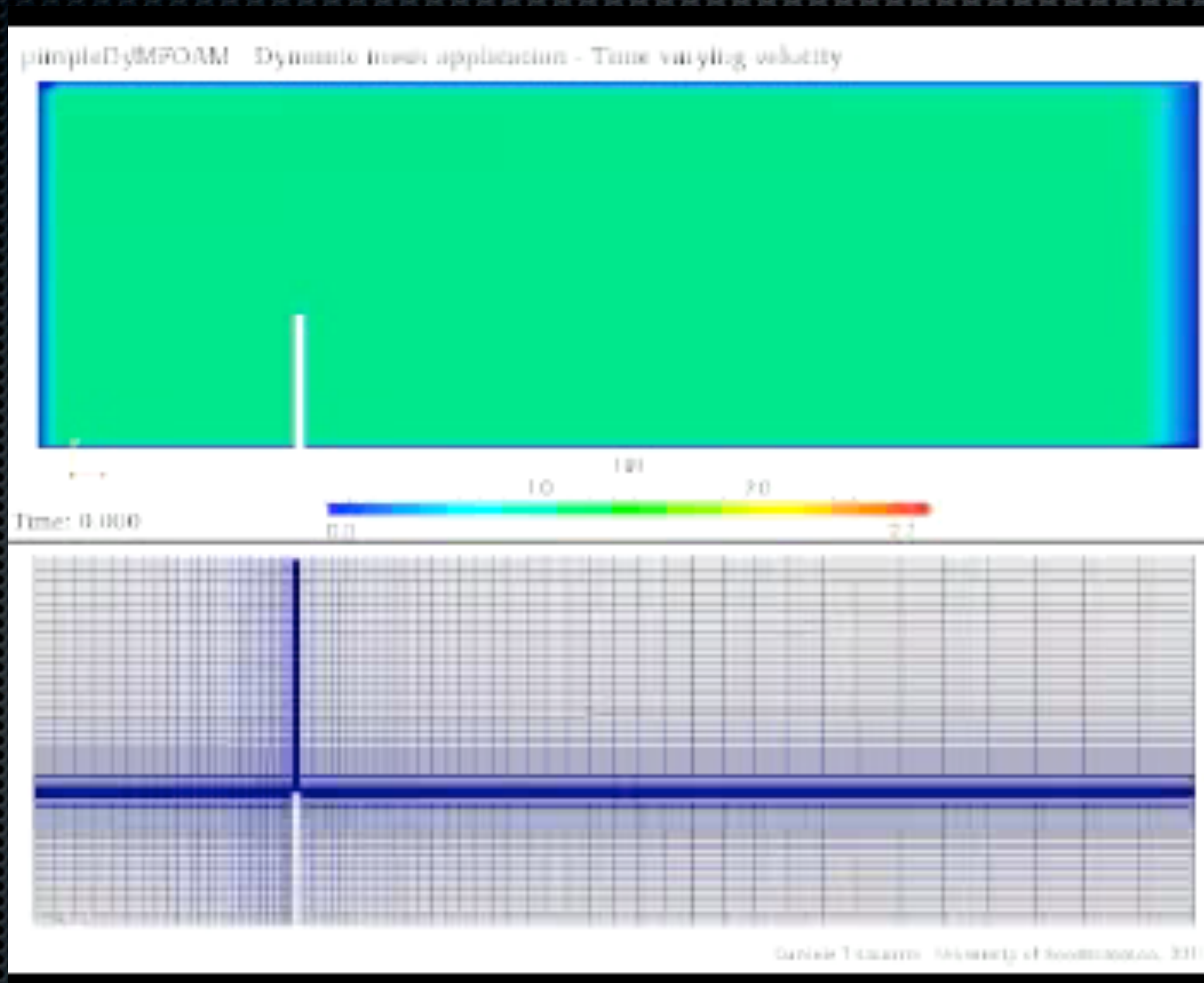
- The motion can be … y or via a 'ramp' file

```
dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    beam
    {
        type  timeVaryingUniformFixedValue;
        fileName "ramp";
        outOfBounds repeat;
    }
    topmoving
    {
        type            slip;
    }
    outlet
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
```

- Two files have to be added in the '0' directory: cellMotionU and pointMotionU. Their definition is as usual for FOAM files, and in this case they are equal:

And finally some results:

And finally some results:



WRITING AND COMPILING USER APPLICATIONS AND LIBRARIES

Tutorial from:
http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2009/

Report, files and presentation by: Andreu Oliver Gonzalez

Before to get into the code...

Before to get into the code...

C++ is an object oriented programming language, OpenFOAM is build up to objects and classes

www.cplusplus.com/doc

Deitel & Deitel: C++ How to program

Before to get into the code...

C++ is an object oriented programming language, OpenFOAM is build up to objects and classes

www.cplusplus.com/doc

Deitel & Deitel: C++ How to program

A reference: & indicates an address in the computer memory, and it can be seen as a bookmark. Treat variables as references is good, since the object is not copied or transferred in the memory

Before to get into the code...

C++ is an object oriented programming language, OpenFOAM is build up to objects and classes

www.cplusplus.com/doc

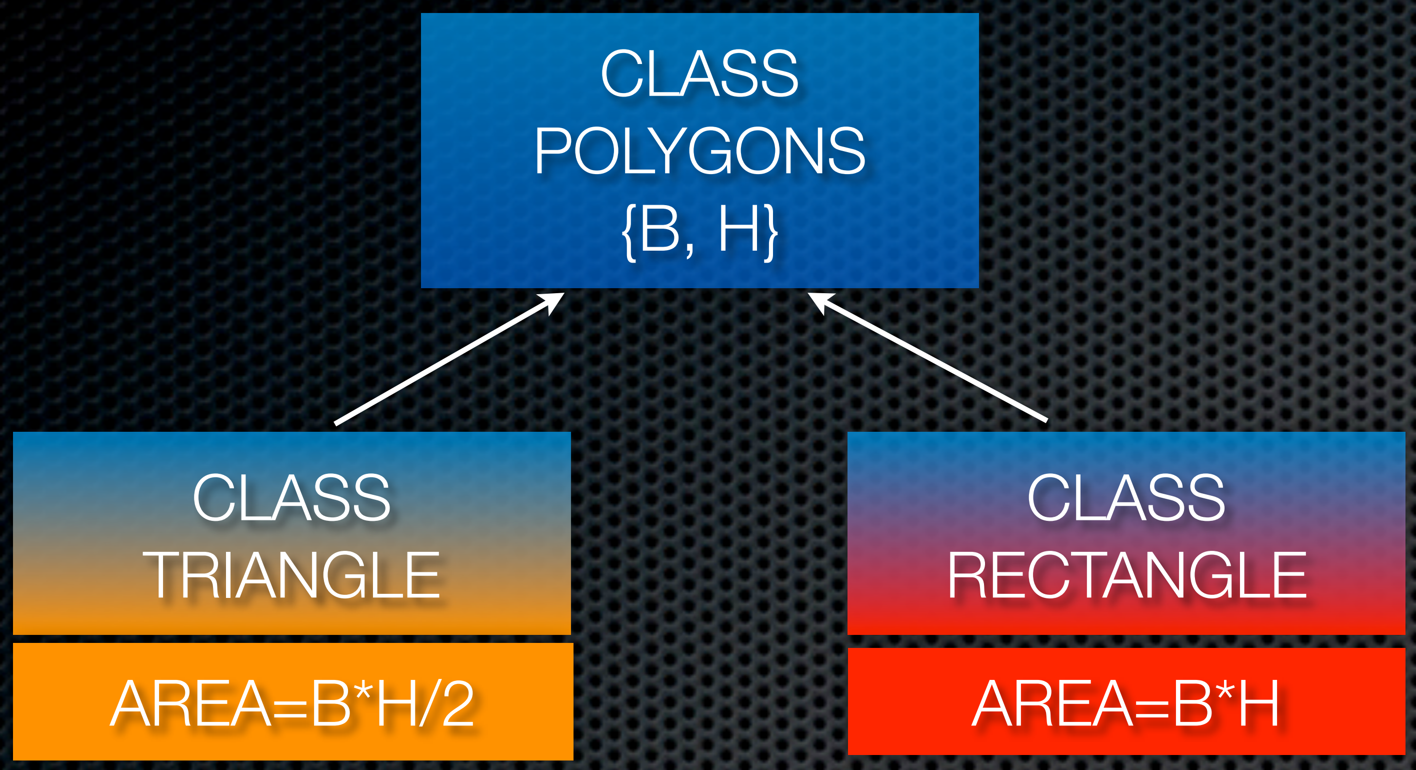
Deitel & Deitel: C++ How to program

A reference: & indicates an address in the computer memory, and it can be seen as a bookmark. Treat variables as references is good, since the object is not copied or transferred in the memory

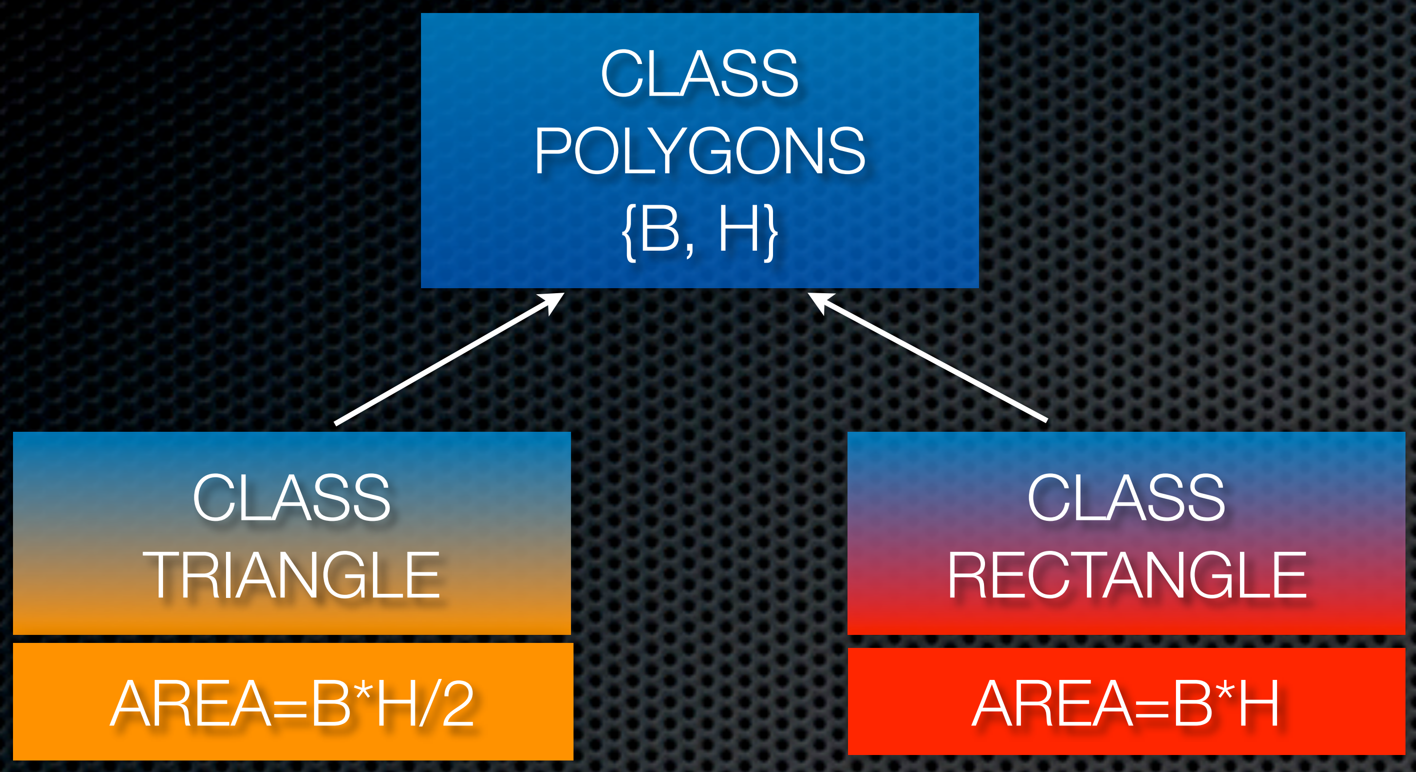
Classes have member functions, the same named function applied to different objects produces different calls. Functions should be (generally) searched in the object's class, or in the parent classes

Before to get into the code... Inheritance

Before to get into the code... Inheritance

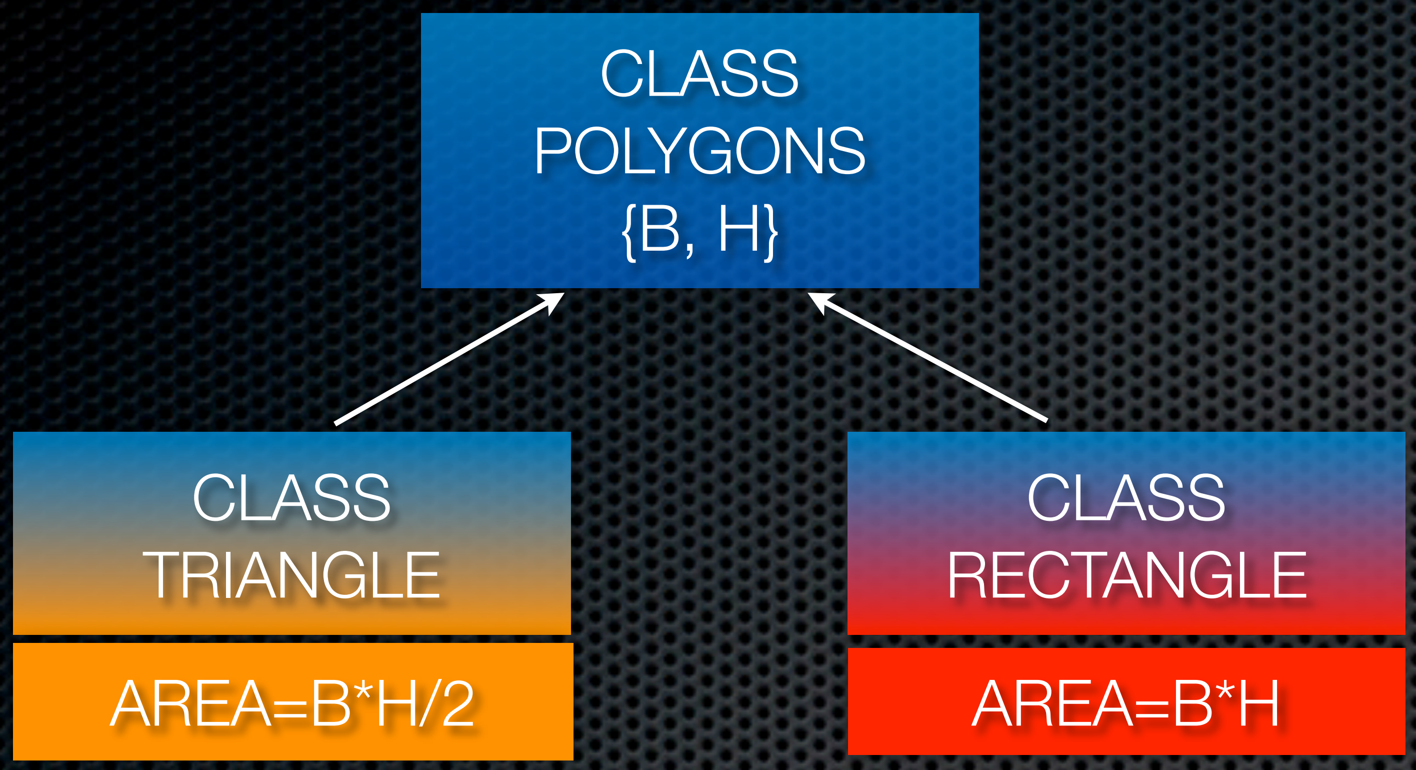


Before to get into the code... Inheritance




```
//Define objects:
  Triangle Tria{2,3}
  Rectangle Rec{3,4}
//Calculate area:
  Tria.area();
  Rec.area();
```

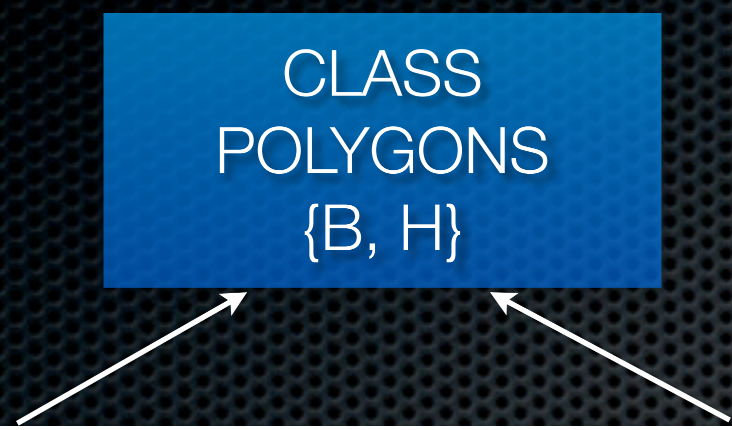
Before to get into the code... Inheritance



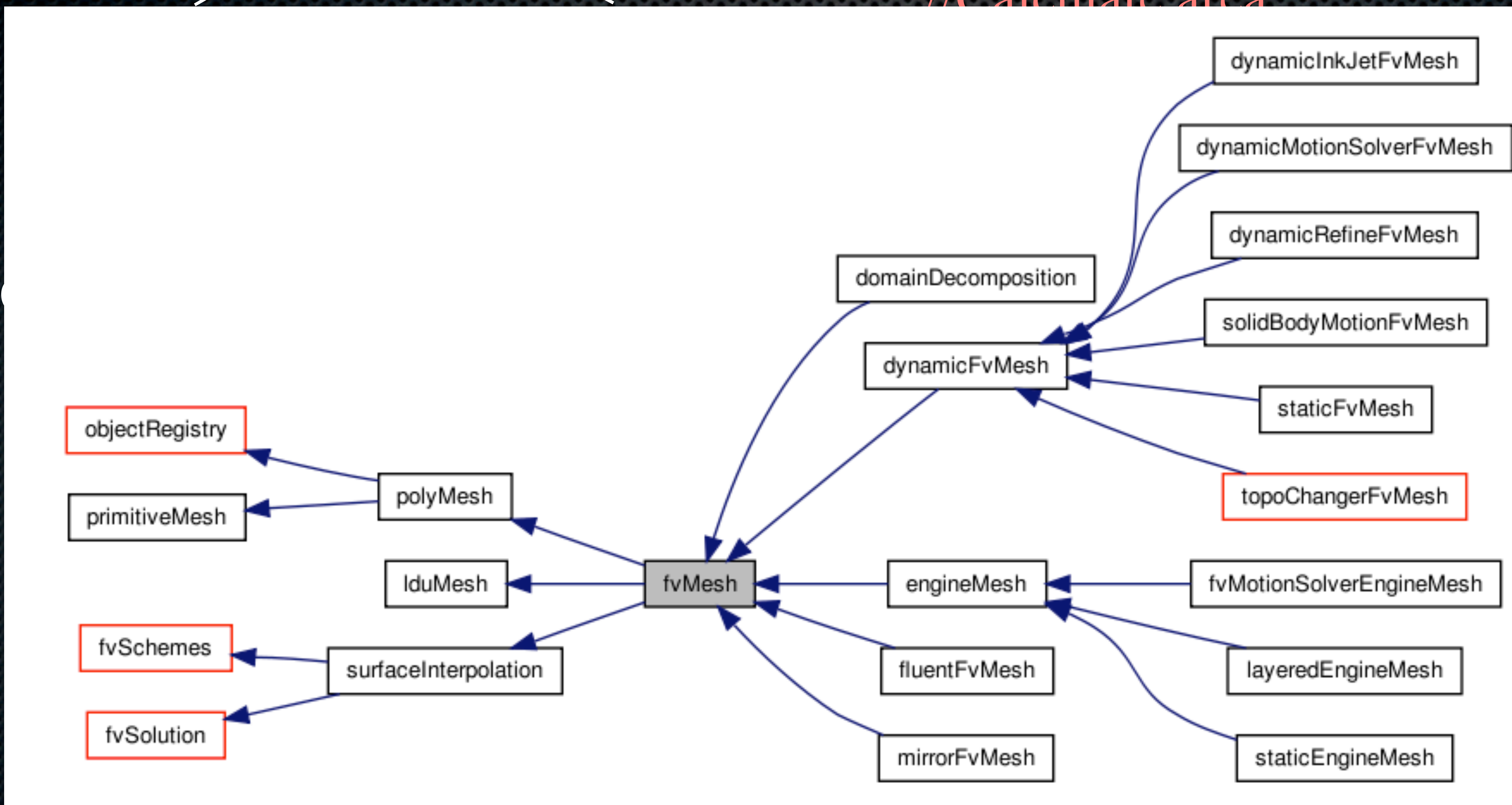

```
//Define objects:
    Triangle Tria{2,3}
    Rectangle Rec{3,4}
//Calculate area:
    Tria.area();
    Rec.area();
```

OpenFOAM makes a large use of ineritance, therefore it is necessary to use the doxygen (http://foam.sourceforge.net/doc/Doxygen/html/)

Before to get into the code... Inheritance




```
//Define objects:
  Triangle Tria{2,3}
  Rectangle Rec{3,4}
//Calculate area:
```

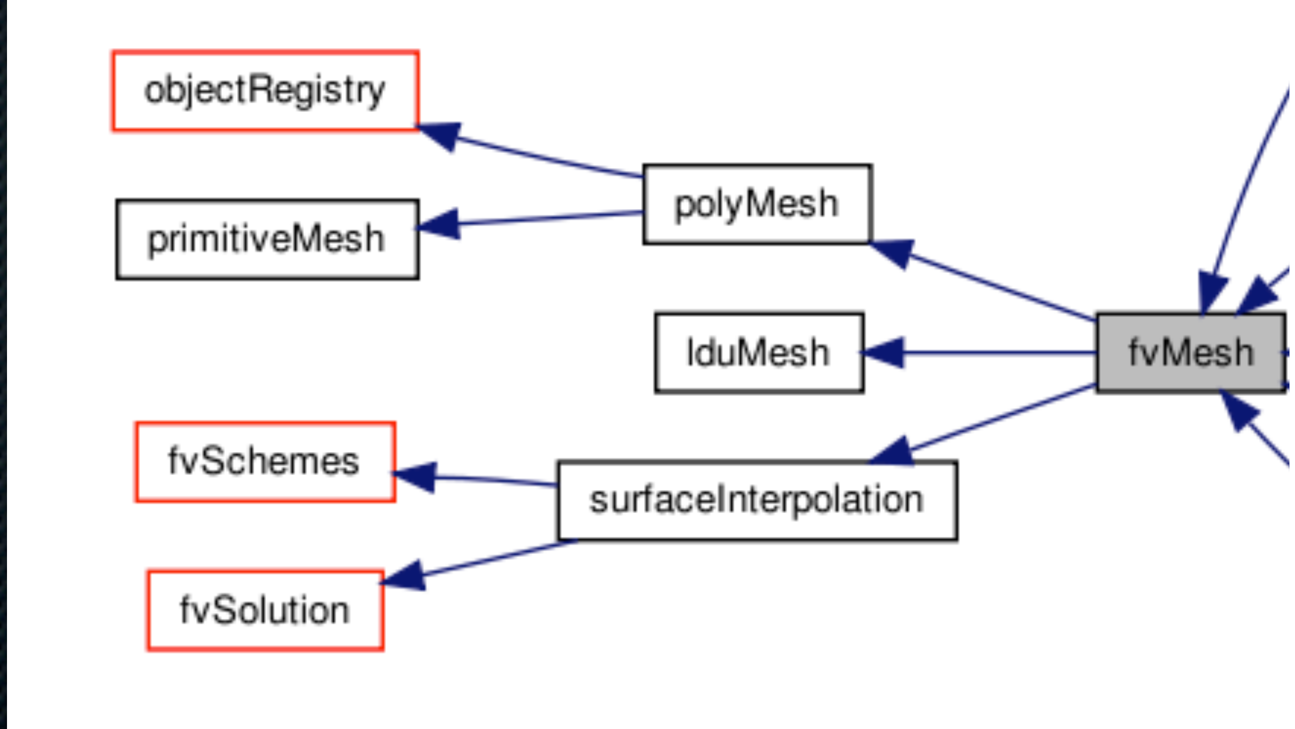


to

Before to get into the code... Inheritance

CLASS
POLYGONS
{B, H}

//Define objects:
Triangle Tria{2,3}



Public Member Functions

	ClassName ("fvMesh")
	fvMesh (const IOobject &io)
	Construct from IOobject.
	fvMesh (const IOobject &io, const Xfer<
	Construct from components without bou
	fvMesh (const IOobject &io, const Xfer<
	Construct without boundary from cells r
virtual	~fvMesh ()
void	addFvPatches (const List< polyPatch * >
	Add boundary patches. Constructor help
virtual readUpdateState	readUpdate ()
	Update the mesh based on the mesh file
const Time &	time () const
	Return the top-level database.
virtual const objectRegistry &	thisDb () const
	Return the object registry - resolve conf
const word &	name () const
	Return reference to name.
const fvBoundaryMesh &	boundary () const
	Return reference to boundary mesh.
virtual const lduAddressing &	lduAddr () const
	Return ldu addressing.
virtual lduInterfacePtrsList	interfaces () const
	Return a list of pointers for each patch.
const unallocLabelList &	owner () const
	Internal face owner.
const unallocLabelList &	neighbour () const
	Internal face neighbour.
const DimensionedField< scalar, volMesh > &	V () const
	Return cell volumes.

How to find entries:

}

How to find entries:

Looking into the code:
OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 225 to 299

```
// * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * //

void Foam::forces::read(const dictionary& dict)
{
    ...
    const fvMesh& mesh = refCast<const fvMesh>(obr_);
    patchSet_ =
            mesh.boundaryMesh().patchSet(wordList(dict.lookup("patches")));
    ...
    pName_ = dict.lookupOrDefault<word>("pName", "p");
    UName_ = dict.lookupOrDefault<word>("UName", "U");
    rhoName_ = dict.lookupOrDefault<word>("rhoName", "rho");
    ...
    rhoRef_ = readScalar(dict.lookup("rhoInf"));
    pRef_ = dict.lookupOrDefault<scalar>("pRef", 0.0);
    CofR_ = dict.lookup("CofR");

}
```

How to find entries:

Looking into the code:
OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 225 to 299

```
// * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * //

void Foam::forces::read(const dictionary& dict)
{
    ...
    const fvMesh& mesh = refCast<const fvMesh>(obr_);
    patchSet_ =
            mesh.boundaryMesh().patchSet(wordList(dict.lookup("patches")));
    ...
    pName_ = dict.lookupOrDefault<word>("pNar
    UName_ = dict.lookupOrDefault<word>("UNa
    rhoName_ = dict.lookupOrDefault<word>("rhc
    ...
    rhoRef_ = readScalar(dict.lookup("rhoInf"));
    pRef_ = dict.lookupOrDefault<scalar>("pRef",
    CofR_ = dict.lookup("CofR");

}
```

obr: Pointer to the objectRegistry [1.174] :

```
Foam::forces::forces
(
 const word& name,
 const objectRegistry& obr,
 const dictionary& dict,
 const bool loadFromFiles
)
:
   name_(name),
   obr_(obr),
   active_(true),
   ...
```

How to find entries:

Looking into the code:

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 225 to 299

```
// * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * //

void Foam::forces::read(const dictionary& dict)
{
    ...
    const fvMesh& mesh = refCast<const fvMesh>(obr_);
    patchSet_ =
            mesh.boundaryMesh().patchSet(wordList(dict.lookup("patches")));
    ...
    pName_ = dict.lookupOrDefault<word>("pName", "p");
    UName_ = dict.lookupOrDefault<word>("UName", "U");
    rhoName_ = dict.lookupOrDefault<word>("rhoName", "rho");
    ...
    rhoRef_ = readScalar(dict.lookup("rhoInf"));
    pRef_ = dict.lookupOrDefault<scalar>("pRef", 0.0);
    CofR_ = dict.lookup("CofR");

}
```

How to find entries:

Looking into the code:
OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 225 to 299

```
// * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * * //

void Foam::forces::read(const dictionary& dict)
{
    ...
    const fvMesh& mesh = refCast<const fvMesh>(obr_);
    patchSet_ =
            mesh.boundaryMesh().patchSet(wordList(dict.lookup("patches")));
    ...

    pName_ = dict.lookupOrDefault<word>("pName", "p");
    U
    rh
    ..

    rhoRef_ = readScalar(dict.lookup("rhoInf"));
    pRef_ = dict.lookupOrDefault<scalar>("pRef", 0.0);
    CofR_ = dict.lookup("CofR");

}
```

patchSet is then the list of patches specified in the controlDict

How to find entries:

Looking into the code:
OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 225 to 299

```
// * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * * //

void Foam::forces::read(const dictionary& dict)
{
    ...
    const fvMesh& mesh = refCast<const fvMesh>(obr_);
    patchSet_ =
            mesh.boundaryMesh().patchSet(wordList(dict.lookup("patches")));
    ...
    pName_ = dict.lookupOrDefault<word>("pName", "p");
    UName_ = dict.lookupOrDefault<word>("UName", "U");
    rhoName_ = dict.lookupOrDefault<word>("rhoName", "rho");
    ...
    rhoRef_ = readScalar(dict.lookup("rhoInf"));
    pRef_ = dict.lookupOrDefault<scalar>("pRef", 0.0);
    CofR_ = dict.lookup("CofR");

}
```

How to find entries:

Looking into the code:
OpenFOAM/src/postProcessing/functionObjects/forces/f

```
// * * * * * * * * * * * * * * * Member Functions  * * * *

void Foam::forces::read(const dictionary& dict)
{
    ...
    const fvMesh& mesh = refCast<const fvMesh>(obr_);
    patchSet_ =
            mesh.boundaryMesh().patchSet(wordList(dict.lookup("patches")));
    ...
    pName_ = dict.lookupOrDefault<word>("pName", "p");
    UName_ = dict.lookupOrDefault<word>("UName", "U");
    rhoName_ = dict.lookupOrDefault<word>("rhoName", "rho");
    ...
    rhoRef_ = readScalar(dict.lookup("rhoInf"));
    pRef_ = dict.lookupOrDefault<scalar>("pRef", 0.0);
    CofR_ = dict.lookup("CofR");

}
```

controlDict

```
forces
{
  type forces;
  functionObjectLibs ("libforces.dylib");
  outputControl outputTime;
  patches (wing);
  pName p;
  Uname U;
  rhoName rhoInf;
  rhoInf 1.2; //Reference density for fluid
  pRef 0;
  CofR (0 0 0); //Origin for moment calculations
}
```

And how the calculation is actually done (1):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 393 to 447

```
Foam::forces::forcesMoments Foam::forces::calcForcesMoment() const
{
    forcesMoments fm
      ...
  {
      const volVectorField& U = obr_.lookupObject<volVectorField>(UName_);
      const volScalarField& p = obr_.lookupObject<volScalarField>(pName_);
      const fvMesh& mesh = U.mesh();
      const surfaceVectorField::GeometricBoundaryField& Sfb =
          mesh.Sf().boundaryField();
      ...
```

And how the calculation is actually done (1):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 393 to 447

```
Foam::forces::forcesMoments Foam::forces::calcForcesMoment() const
{
   forcesMoments fm
     ...
  {
      const volVectorField& U = obr_.lookupObject<volVectorField>(UName_);
      const volScalarField& p = obr_.lookupObject<volScalarField>(pName_);
      const fvMesh& mesh = U.mesh();
      const surfaceVectorField::GeometricBoundaryField& Sfb =
          mesh.Sf().boundaryField();
      ...
```

was read from the controlDict

And how the calculation is actually done (1):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 393 to 447

```
Foam::forces::forcesMoments Foam::forces::calcForcesMoment() const
{
   forcesMoments fm
     ...
  {
      const volVectorField& U = obr_.lookupObject<volVectorField>(UName_);
      const volScalarField& p = obr_.lookupObject<volScalarField>(pName_);
      const fvMesh& mesh = U.mesh();
      const surfaceVectorField::GeometricBoundaryField& Sfb =
          mesh.Sf().boundaryField();
      ...
```

And how the calculation is actually done (1):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 393 to 447

```
Foam::forces::forcesMoments Foam::forces::calcForcesMoment() const
{
   forcesMoments fm
     ...
  {
      const volVectorField& U = obr_.lookupObject<volVectorField>(UName_);
      const volScalarField& p = obr_.lookupObject<volScalarField>(pName_);
      const fvMesh& mesh = U.mesh();
      const surfaceVectorField::GeometricBoundaryField& Sfb =
          mesh.Sf().boundaryField();
      ...
```

And how the calculation is actually done (1):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 393 to 447

```
Foam::forces::forcesMoments Foam::forces::calcForcesMoment() const
{
  forcesMoments fm
    ...
 {
    const volVectorField& U = obr_.lookupObject<volVectorField>(UName_);
    const volScalarField& p = obr_.lookupObject<volScalarField>(pName_);
    const fvMesh& mesh = U.mesh();
    const surfaceVectorField::GeometricBoundaryField& Sfb =
      mesh.Sf().boundaryField();
    ...
```

And how the calculation is actually done (1):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 393 to 447

```
Foam::forces::forcesMoments Foam::forces::calcForcesMoment() const
{
  forcesMoments fm
    ...
 {
    const volVectorField& U = obr_.lookupObject<volVectorField>(UName_);
    const volScalarField& p = obr_.lookupObject<volScalarField>(pName_);
    const fvMesh& mesh = U.mesh();
    const surfaceVectorField::GeometricBoundaryField& Sfb =
       mesh.Sf().boundaryField();
    ...
```

mesh() has to be a function of the class volScalarFIeld, returning a reference to the mesh associated with the field U. But volScalarField is class template... Where is it?

And how the calculation is actually done (1):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 393 to 447

```
Foam::forces::forcesMoments Foam::forces::calcForcesMoment() const
{
  forcesMoments fm
    ...
 {
     const volVectorField& U = obr_.lookupObject<volVectorField>(UName_);
     const volScalarField& p = obr_.lookupObject<volScalarField>(pName_);
     const fvMesh& mesh = U.mesh();
     const surfaceVectorField::GeometricBoundaryField& Sfb =
        mesh.Sf().boundaryField();
     ...
```

And how the calculation is actually done (1):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 393 to 447

```
Foam::forces::forcesMoments Foam::forces::calcForcesMoment() const
{
  forcesMoments fm
    ...
  {
      const volVectorField& U = obr_.lookupObject<volVectorField>(UName_);
      const volScalarField& p = obr_.lookupObject<volScalarField>(pName_);
      const fvMesh& mesh = U.mesh();
      const surfaceVectorField::GeometricBoundaryField& Sfb =
         mesh.Sf().boundaryField();
      ...
```

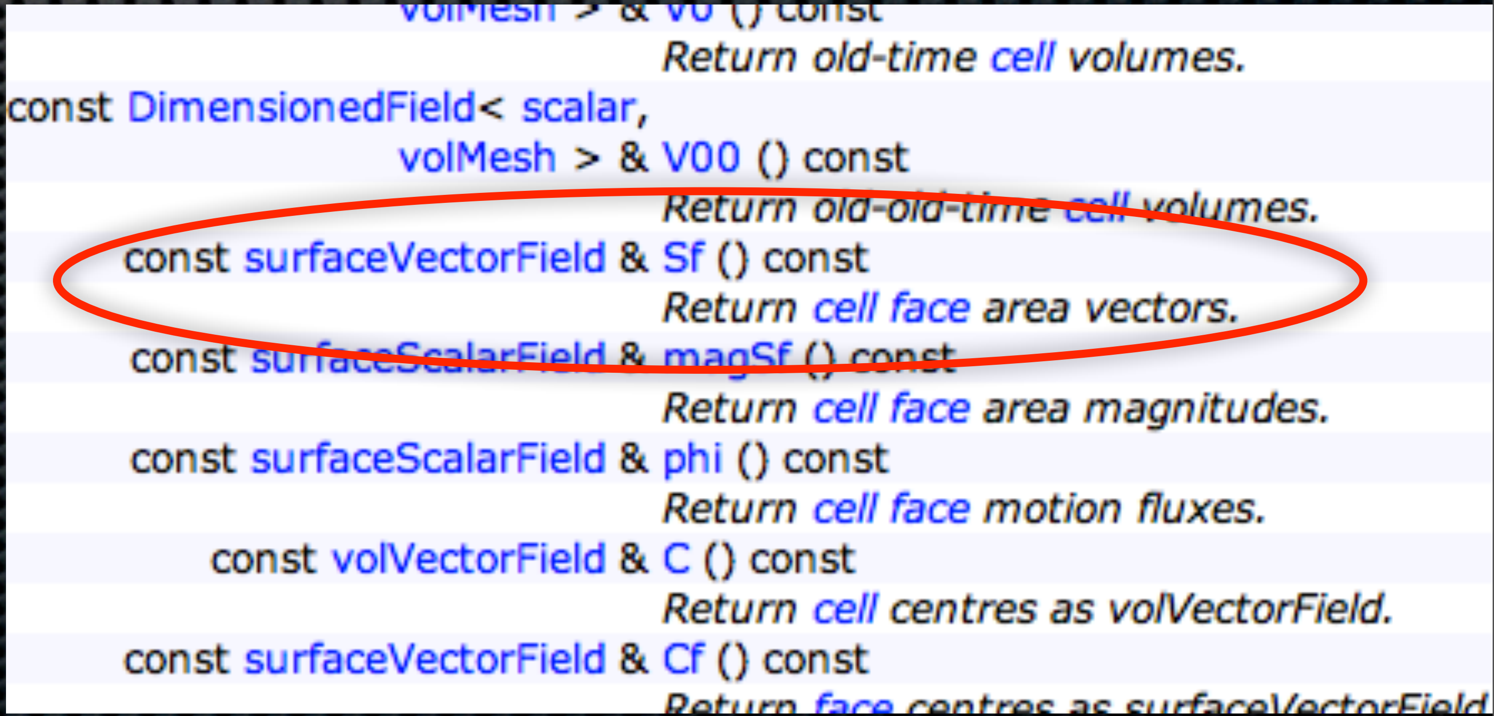
mesh is object of the class fvMesh.
searching in the Doxygen for this class:

And how the calculation is actually done (1):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 393 to 447

```
Foam::forces::forcesMoments Foam::forces::calcForcesMoment() const
{
  forcesMoments fm
    ...
  {
      const volVectorField& U = obr_.lookupObject<volVectorField>(UName_);
      const volScalarField& p = obr_.lookupObject<volScalarField>(pName_);
      const fvMesh& mesh = U.mesh();
      const surfaceVectorField::GeometricBoundaryField& Sfb =
         mesh.Sf().boundaryField();
      ...
```

mesh is object of the class fvMesh.
searching in the Doxygen for this class:



Return old-time cell volumes.

const DimensionedField< scalar, volMesh > & V00 () const
Return old-old-time cell volumes.

const surfaceVectorField & Sf () const
Return cell face area vectors.

const surfaceScalarField & magSf () const
Return cell face area magnitudes.

const surfaceScalarField & phi () const
Return cell face motion fluxes.

const volVectorField & C () const
Return cell centres as volVectorField.

const surfaceVectorField & Cf () const

And how the calculation is actually done (1):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 393 to 447

```
Foam::forces::forcesMoments Foam::forces::calcForcesMoment() const
{
    forcesMoments fm
      ...
  {
       const volVectorField& U = obr_.lookupObject<volVectorField>(UName_);
       const volScalarField& p = obr_.lookupObject<volScalarField>(pName_);
       const fvMesh& mesh = U.mesh();
       const surfaceVectorField::GeometricBoundaryField& Sfb =
           mesh.Sf().boundaryField();
       ...
```

And how the calculation is actually done (1):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 393 to 447

```
Foam::forces::forcesMoments Foam::forces::calcForcesMoment() const
{
  forcesMoments fm
    ...
  {
     const volVectorField& U = obr_.lookupObject<volVectorField>(UName_);
     const volScalarField& p = obr_.lookupObject<volScalarField>(pName_);
     const fvMesh& mesh = U.mesh();
     const surfaceVectorField::GeometricBoundaryField& Sfb =
        mesh.Sf().boundaryField();
     ...
```

Sfb is then a reference to the face area vector of the mesh boundaryField (inlet, outlet, walls, wing, ...) as it is defined in the mesh and in the boundary files (as p or U)

And how the calculation is actually done (2):

And how the calculation is actually done (2):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 461 to 490

```
        ...
        forAllConstIter(labelHashSet, patchSet_, iter)
        {
            label patchi = iter.key();

            vectorField Md = mesh.C().boundaryField()[patchi] - CofR_;

            vectorField pf = Sfb[patchi]*(p.boundaryField()[patchi] - pRef);

            fm.first().first() += rho(p)*sum(pf);

            fm.second().first() += rho(p)*sum(Md ^ pf);
           ...
        }
    }
    ...
    return fm;
}
```

And how the calculation is actually done (2):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 461 to 490

```
    ...
    forAllConstIter(labelHashSet, patchSet_, iter)
    {
        label patchi = iter.key();

        vectorField Md = mesh.C().boundaryField()[patchi] - CofR_;

        vectorField pf = Sfb[patchi]*(p.boundaryField()[patchi] - pRef);

        fm.first().first() += rho(p)*sum(pf);

        fm.second().first() += rho(p)*sum(Md ^ pf);
      ...
    }
  }
  ...
  return fm;
}
```

OpenFOAM iterator, as a ‘for’ cycle.

And how the calculation is actually done (2):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 461 to 490

```
        ...
        forAllConstIter(labelHashSet, patchSet_, iter)
        {
            label patchi = iter.key();

            vectorField Md = mesh.C().boundaryField()[patchi] - CofR_;

            vectorField pf = Sfb[patchi]*(p.boundaryField()[patchi] - pRef);

            fm.first().first() += rho(p)*sum(pf);

            fm.second().first() += rho(p)*sum(Md ^ pf);
          ...
        }
    }
    ...
    return fm;
}
```

OpenFOAM iterator, as a ‘for’ cycle.

It cycles (using the index ‘iter’) over the objects in the patchSet

And how the calculation is actually done (2):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 461 to 490

```
...
forAllConstIter(labelHashSet, patchSet_, iter)
{
    label patchi = iter.key();

    vectorField Md = mesh.C().boundaryField()[patchi] - CofR_;

    vectorField pf = Sfb[patchi]*(p.boundaryField()[patchi] - pRef);

    fm.first().first() += rho(p)*sum(pf);

    fm.second().first() += rho(p)*sum(Md ^ pf);
  ...
  }
}
...
return fm;
}
```

OpenFOAM iterator, as a 'for' cycle.

It cycles (using the index 'iter') over the objects in the patchSet

This has been previously defined as the list of the patches on which we want to integrate forces (coming from the controlDict)

And how the calculation is actually done (2):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 461 to 490

```
        ...
        forAllConstIter(labelHashSet, patchSet_, iter)
        {
            label patchi = iter.key();

            vectorField Md = mesh.C().boundaryField()[patchi] - CofR_;

            vectorField pf = Sfb[patchi]*(p.boundaryField()[patchi] - pRef);

            fm.first().first() += rho(p)*sum(pf);

            fm.second().first() += rho(p)*sum(Md ^ pf);
           ...
        }
    }
    ...
    return fm;
}
```

And how the calculation is actually done (2):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 461 to 490

```
    ...
    forAllConstIter(labelHashSet, patchSet_, iter)
    {
        label patchi = iter.key();

        vectorField Md = mesh.C().boundaryField()[patchi] - CofR_;

        vectorField pf = Sfb[patchi]*(p.boundaryField()[patchi] - pRef);

        fm.first().first() += rho(p)*sum(pf);

        fm.second().first() += rho(p)*sum(Md ^ pf);
      ...
    }
  }
  ...
  return fm;
}
```

mesh is object of the class fvMesh

And how the calculation is actually done (2):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 461 to 490

```
    ...
    forAllConstIter(labelHashSet, patchSet_, iter)
    {
        label patchi = iter.key();

        vectorField Md = mesh.C().boundaryField()[patchi] - CofR_;

        vectorField pf = Sfb[patchi]*(p.boundaryField()[patchi] - pRef);

        fm.first().first() += rho(p)*sum(pf);

        fm.second().first() += rho(p)*sum(Md ^ pf);
       ...
    }
  }
  ...
  return fm;
}
```

mesh is object of the class fvMesh

From the Doxygen then:

And how the calculation is actually done (2):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 461 to 490

```
    ...
    forAllConstIter(labelHashSet, patchSet_, iter)
    {
        label patchi = iter.key();

        vectorField Md = mesh.C().boundaryField()[patchi] - CofR_;

        vectorField pf = Sfb[patchi]*(p.boundaryField()[patchi] - pRef);

        fm.first().first() += rho(p)*sum(pf);

        fm.second().first() += rho(p)*sum(Md ^ pf);
        ...
    }
  }
  ...
  return fm;
}
```

mesh is object of the class fvMesh

From the Doxygen then:

```
                    volMesh > & V0 () const
                                 Return old-time cell volumes.
const DimensionedField< scalar,
                    volMesh > & V00 () const
                                 Return old-old-time cell volumes.
   const surfaceVectorField & Sf () const
                                 Return cell face area vectors.
   const surfaceScalarField & magSf () const
                                 Return cell face area magnitudes.
   const surfaceScalarField & phi () const
                                 Return cell face motion fluxes.
      const volVectorField & C () const
                                 Return cell centres as volVectorField.
   const surfaceVectorField & Cf () const
                                 Return face centres as surfaceVectorField.
```

And how the calculation is actually done (2):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 461 to 490

```
    ...
    forAllConstIter(labelHashSet, patchSet_, iter)
    {
        label patchi = iter.key();

        vectorField Md = mesh.C().boundaryField()[patchi] - CofR_;

        vectorField pf = Sfb[patchi]*(p.boundaryField()[patchi] - pRef);

        fm.first().first() += rho(p)*sum(pf);

        fm.second().first() += rho(p)*sum(Md ^ pf);
       ...
    }
  }
  ...
  return fm;
}
```

mesh is object of the class fvMesh

From the Doxygen then:

And how the calculation is actually done (2):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 461 to 490

```
    ...
    forAllConstIter(labelHashSet, patchSet_, iter)
    {
        label patchi = iter.key();

        vectorField Md = mesh.C().boundaryField()[patchi] - CofR_;

        vectorField pf = Sfb[patchi]*(p.boundaryField()[patchi] - pRef);

        fm.first().first() += rho(p)*sum(pf);

        fm.second().first() += rho(p)*sum(Md ^ pf);
      ...
    }
  }
  ...
  return fm;
}
```

mesh is object of the class fvMesh

From the Doxygen then:

The entire expression returns a vector with the cell centres of the patch that we chosed for integrating forces

And how the calculation is actually done (2):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 461 to 490

```
        ...
        forAllConstIter(labelHashSet, patchSet_, iter)
        {
            label patchi = iter.key();

            vectorField Md = mesh.C().boundaryField()[patchi] - CofR_;

            vectorField pf = Sfb[patchi]*(p.boundaryField()[patchi] - pRef);

            fm.first().first() += rho(p)*sum(pf);

            fm.second().first() += rho(p)*sum(Md ^ pf);
           ...
        }
    }
    ...
    return fm;
}
```

And how the calculation is actually done (2):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 461 to 490

```
        ...
        forAllConstIter(labelHashSet, patchSet_, iter)
        {
            label patchi = iter.key();

            vectorField Md = mesh.C().boundaryField()[patchi] - CofR_;

            vectorField pf = Sfb[patchi]*(p.boundaryField()[patchi] - pRef);

            fm.first().first() += rho(p)*sum(pf);

            fm.second().first() += rho(p)*sum(Md ^ pf);
          ...
        }
    }
    ...
    return fm;
}
```

Sfb is the (reference to) the face area vector

And how the calculation is actually done (2):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 461 to 490

```
    ...
    forAllConstIter(labelHashSet, patchSet_, iter)
    {
        label patchi = iter.key();

        vectorField Md = mesh.C().boundaryField()[patchi] - CofR_;

        vectorField pf = Sfb[patchi]*(p.boundaryField()[patchi] - pRef);

        fm.first().first() += rho(p)*sum(pf);

        fm.second().first() += rho(p)*sum(Md ^ pf);
      ...
    }
  }
  ...
  return fm;
}
```

Sfb is the (reference to) the face area vector

It is here multiplied for the pressure boundaryField => pf returns the vector of forces insisting on the chosen patch

And how the calculation is actually done (2):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 461 to 490

```
        ...
        forAllConstIter(labelHashSet, patchSet_, iter)
        {
            label patchi = iter.key();

            vectorField Md = mesh.C().boundaryField()[patchi] - CofR_;

            vectorField pf = Sfb[patchi]*(p.boundaryField()[patchi] - pRef);

            fm.first().first() += rho(p)*sum(pf);

            fm.second().first() += rho(p)*sum(Md ^ pf);
           ...
        }
    }
    ...
    return fm;
}
```

And how the calculation is actually done (2):

OpenFOAM/src/postProcessing/functionObjects/forces/forces/Force.C, lines 461 to 490

```
    ...
    forAllConstIter(labelHashSet, patchSet_, iter)
    {
        label patchi = iter.key();

        vectorField Md = mesh.C().boundaryField()[patchi] - CofR_;

        vectorField pf = Sfb[patchi]*(p.boundaryField()[patchi] - pRef);

        fm.first().first() += rho(p)*sum(pf);

        fm.second().first() += rho(p)*sum(Md ^ pf);
       ...
    }
  }
  ...
  return fm;
}
```

$$F = \rho \int p dA = \rho \sum p_i A_i$$

$$M = F \times r = \rho \sum f_i \times r_i$$

Modifying pimpleDyMFOAM for Fluid Structure interactions: imposing patch deformations:

Modifying pimpleDyMFOAM for Fluid Structure interactions: imposing patch deformations:

- Until now, it is easy to impose rigid body movements, in terms of velocities of displacements

Modifying pimpleDyMFOAM for Fluid Structure interactions: imposing patch deformations:

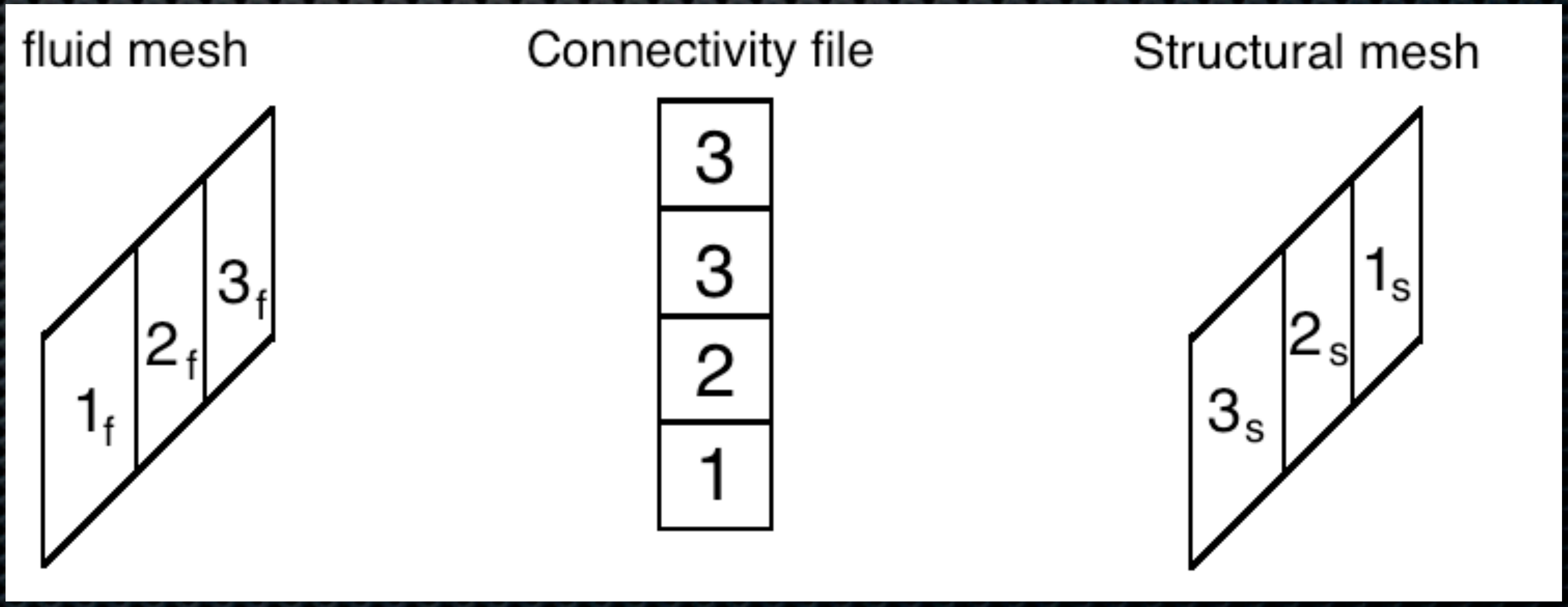
- Until now, it is easy to impose rigid body movements, in terms of velocities of displacements
- The solver has then be modified in order to read the cell displacements from a file, coming from a structural solver

Modifying pimpleDyMFOAM for Fluid Structure interactions: imposing patch deformations:

- Until now, it is easy to impose rigid body movements, in terms of velocities of displacements
- The solver has then be modified in order to read the cell displacements from a file, coming from a structural solver
- Three modifications:

Modifying pimpleDyMFOAM for Fluid Structure interactions: imposing patch deformations:

- Until now, it is easy to impose rigid body movements, in terms of velocities of displacements
- The solver has then be modified in order to read the cell displacements from a file, coming from a structural solver
- Three modifications:
  - 1_ In the initial phase an output file is created with the coordinates of the face-centres of the interface, namely the sail surface. This is used for calculating the connectivity file



Modifying pimpleDyMFOAM for Fluid Structure interactions: imposing patch deformations:

- Until now, it is easy to impose rigid body movements, in terms of velocities of displacements
- The solver has then be modified in order to read the cell displacements from a file, coming from a structural solver
- Three modifications:
  - 1_ In the initial phase an output file is created with the coordinates of the face-centres of the interface, namely the sail surface. This is used for calculating the connectivity file
  - 2_ Before the mesh update, displacements of the mesh (cells) are read from a file, which comes from the structural calculation.

Modifying pimpleDyMFOAM for Fluid Structure interactions: imposing patch deformations:

- Until now, it is easy to impose rigid body movements, in terms of velocities of displacements
- The solver has then be modified in order to read the cell displacements from a file, coming from a structural solver
- Three modifications:
  - 1_ In the initial phase an output file is created with the coordinates of the face-centres of the interface, namely the sail surface. This is used for calculating the connectivity file
  - 2_ Before the mesh update, displacements of the mesh (cells) are read from a file, which comes from the structural calculation.
  - 3_At the end of the routine, pressures at the interface are written to an output file

Output the patch face centres to file /1:

```
IOdictionary coupling
(
   IOobject
   (
   "CouplingDict",
        runTime.constant(),
   mesh,
   IOobject::MUST_READ,
   IOobject::NO_WRITE
   )
 );

 word sail=coupling.lookup("wing");

 label sailL = mesh.boundaryMesh().findPatchID(sail);

 const polyPatch &SailMesh=mesh.boundaryMesh()[sailL];

 int nFaces=SailMesh.size()/2;
```

Output the patch face centres to file /1:

```
IOdictionary coupling
(
   IOobject
   (
   "CouplingDict",
        runTime.constant(),
   mesh,
   IOobject::MUST_READ,
   IOobject::NO_WRITE
   )
);

word sail=coupling.lookup("wing");

label sailL = mesh.boundaryMesh().findPatchID(sail);

const polyPatch &SailMesh=mesh.boundaryMesh()[sailL];

int nFaces=SailMesh.size()/2;
```

Opens a Dictionary called CouplingDict and placed in Constant, and assigns this object to the variable “coupling”

Output the patch face centres to file /1:

```
IOdictionary coupling
(
   IOobject
   (
   "CouplingDict",
        runTime.constant(),
   mesh,
   IOobject::MUST_READ,
   IOobject::NO_WRITE
   )
 );

 word sail=coupling.lookup("wing");

 label sailL = mesh.boundaryMesh().findPatchID(sail);

 const polyPatch &SailMesh=mesh.boundaryMesh()[sailL];

 int nFaces=SailMesh.size()/2;
```

Output the patch face centres to file /1:

```
IOdictionary coupling
(
  IOobject
  (
  "CouplingDict",
      runTime.constant(),
  mesh,
  IOobject::MUST_READ,
  IOobject::NO_WRITE
  )
 );

 word sail=coupling.lookup("wing");

 label sailL = mesh.boundaryMesh().findPatchID(sail);

 const polyPatch &SailMesh=mesh.boundaryMesh()[sailL];

 int nFaces=SailMesh.size()/2;
```

Searches in the Dict the entry "wing", and assigns its (char) value to the variable "sail". This should correspond to a physical surface in the mesh.

Output the patch face centres to file /1:

```
IOdictionary coupling
(
   IOobject
   (
   "CouplingDict",
        runTime.constant(),
   mesh,
   IOobject::MUST_READ,
   IOobject::NO_WRITE
   )
);

word sail=coupling.lookup("wing");

label sailL = mesh.boundaryMesh().findPatchID(sail);

const polyPatch &SailMesh=mesh.boundaryMesh()[sailL];

int nFaces=SailMesh.size()/2;
```

CouplingDict

wing vela;

Searches in the Dict the entry "wing", and assigns its (char) value to the variable "sail". This should correspond to a physical surface in the mesh.

Output the patch face centres to file /1:

```
IOdictionary coupling
(
   IOobject
   (
   "CouplingDict",
        runTime.constant(),
   mesh,
   IOobject::MUST_READ,
   IOobject::NO_WRITE
   )
 );

 word sail=coupling.lookup("wing");

 label sailL = mesh.boundaryMesh().findPatchID(sail);

 const polyPatch &SailMesh=mesh.boundaryMesh()[sailL];

 int nFaces=SailMesh.size()/2;
```

Output the patch face centres to file /1:

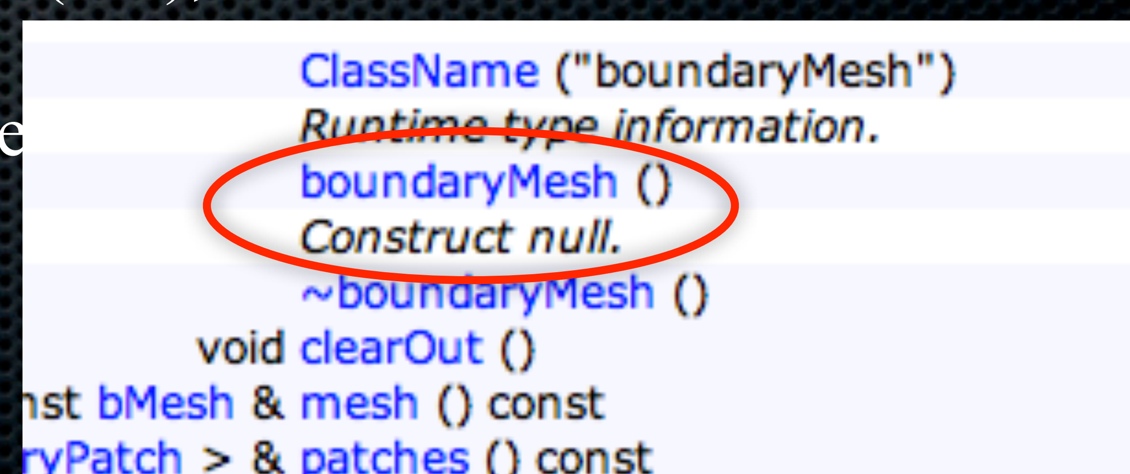
```
IOdictionary coupling
(
   IOobject
   (
   "CouplingDict",
        runTime.constant(),
   mesh,
   IOobject::MUST_READ,
   IOobject::NO_WRITE
   )
 );

 word sail=coupling.lookup("wing");

 label sailL = mesh.boundaryMesh().findPatchID(sail);

 const polyPatch &SailMesh=mesh.boundaryMesh()[sailL];

 int nFaces=SailMesh.size()/2;
```

- boundarymesh() is not a member function of fvMesh, BUT a class itself. (..?).

Output the patch face centres to file /1:

```
IOdictionary coupling
(
   IOobject
   (
   "CouplingDict",
        runTime.constant(),
   mesh,
   IOobject::MUST_READ,
   IOobject::NO_WRITE
   )
 );

 word sail=coupling.lookup("wing");

 label sailL = mesh.boundaryMesh().findPatchID(sail);

 const polyPatch &SailMesh=mesh.boundaryMe

 int nFaces=SailMesh.size()/2;
```

- boundarymesh() is not a member function of fvMesh, BUT a class itself. (..?).
- boundaryMesh() is its constructor



Output the patch face centres to file /1:

labelList getNearest (const primitiveMesh &pMesh, const vector &searchSpan

Get bMesh index of nearest face for every boundary face in.

void patchify (const labelList &nearest, const polyBoundaryMesh &oldPat

*Take over patches onto polyMesh from nearest face in *this.*

label whichPatch (const label faceI) const

Get index of patch face is in.

label findPatchID (const word &patchName) const

Get index of patch by name.

wordList patchNames () const

Get names of patches.

void addPatch (const word &patchName)

Add to back of patch list.

void deletePatch (const word &patchName)

```
    IOobject::MUST_READ,
    IOobject::NO_WRITE
  )
);

word sail=coupling.lookup("wing");

label sailL = mesh.boundaryMesh().findPatchID(sail);

const polyPatch &SailMesh=mesh.boundaryMe

int nFaces=SailMesh.size()/2;
```

- boundarymesh() is not a member function of fvMesh, BUT a class itself. (..?).
- boundaryMesh() is its constructor
- findPatchID is function member of this class

ClassName ("boundaryMesh")

Runtime type information.

boundaryMesh ()

Construct null.

~boundaryMesh ()

void clearOut ()

nst bMesh & mesh () const

ryPatch > & patches () const

Output the patch face centres to file /1:

```
IOdictionary coupling
(
   IOobject
   (
   "CouplingDict",
        runTime.constant(),
   mesh,
   IOobject::MUST_READ,
   IOobject::NO_WRITE
   )
 );

 word sail=coupling.lookup("wing");

 label sailL = mesh.boundaryMesh().findPatchID(sail);

 const polyPatch &SailMesh=mesh.boundaryMesh()[sailL];

 int nFaces=SailMesh.size()/2;
```

Output the patch face centres to file /1:

```
IOdictionary coupling
(
  IOobject
  (
  "CouplingDict",
      runTime.constant(),
  mesh,
  IOobject::MUST_READ,
  IOobject::NO_WRITE
  )
);

word sail=coupling.lookup("wing");

label sailL = mesh.boundaryMesh().findPatchID(sail);

const polyPatch &SailMesh=mesh.boundaryMesh()[sailL];

int nFaces=SailMesh.size()/2;
```

Globally, sailL is the label (the numbering) of the surface defined in the CouplingDict

Output the patch face centres to file /1:

```
IOdictionary coupling
(
   IOobject
   (
   "CouplingDict",
        runTime.constant(),
   mesh,
   IOobject::MUST_READ,
   IOobject::NO_WRITE
   )
 );

 word sail=coupling.lookup("wing");

 label sailL = mesh.boundaryMesh().findPatchID(sail);

 const polyPatch &SailMesh=mesh.boundaryMesh()[sailL];

 int nFaces=SailMesh.size()/2;
```

Output the patch face centres to file /1:

```
IOdictionary coupling
(
   IOobject
   (
   "CouplingDict",
        runTime.constant(),
   mesh,
   IOobject::MUST_READ,
   IOobject::NO_WRITE
   )
 );

 word sail=coupling.lookup("wing");

 label sailL = mesh.boundaryMesh().findPatchID(sail);

 const polyPatch &SailMesh=mesh.boundaryMesh()[sailL];

 int nFaces=SailMesh.size()/2;
```

and SailMesh a reference to the boundary-Mesh defining the surface in the mesh

Output the patch face centres to file /1:

```
IOdictionary coupling
(
   IOobject
   (
   "CouplingDict",
        runTime.constant(),
   mesh,
   IOobject::MUST_READ,
   IOobject::NO_WRITE
   )
 );

 word sail=coupling.lookup("wing");

 label sailL = mesh.boundaryMesh().findPatchID(sail);

 const polyPatch &SailMesh=mesh.boundaryMesh()[sailL];

 int nFaces=SailMesh.size()/2;
```

Output the patch face centres to file /1:

```
IOdictionary coupling
(
   IOobject
   (
   "CouplingDict",
        runTime.constant(),
   mesh,
   IOobject::MUST_READ,
   IOobject::NO_WRITE
   )
 );

 word sail=coupling.lookup("wing");

 label sailL = mesh.boundaryMesh().findPa

 const polyPatch &SailMesh=mesh.bounda

 int nFaces=SailMesh.size()/2;
```

Sail is function of a template class, therefore it is multi-purpose, applicable to different objects.

Output the patch face centres to file /1:

```
IOdictionary coupling
(
   IOobject
   (
   "CouplingDict",
        runTime.constant(),
   mesh,
   IOobject::MUST_READ,
   IOobject::NO_WRITE
   )
 );

 word sail=coupling.lookup("wing");

 label sailL = mesh.boundaryMesh().findPa

 const polyPatch &SailMesh=mesh.bounda

 int nFaces=SailMesh.size()/2;
```

Sail is function of a template class, therefore it is multi-purpose, applicable to different objects.
It is /2, since the zero thickness surface is described as a series of superposed faces

Output the patch face centres to file /1:

```
IOdictionary coupling
(
  IOobject
  (
  "CouplingDict",
      runTime.constant(),
  mesh,
  IOobject::MUST_READ,
  IOobject::NO_WRITE
  )
 );

 word sail=coupling.lookup("wing");

 label sailL = mesh.boundaryMesh().findPatchID(sail);

 const polyPatch &SailMesh=mesh.boundaryMesh()[sailL];

 int nFaces=SailMesh.size()/2;
```

Output the patch face centres to file /2:

```
std::ofstream myfile;
myfile.open ("FluidMesh_centres.txt", std::ios::app );

myfile<<nFaces<<"\n";
for (int n=0; n<nFaces; n++)
  {
    myfile<<SailMesh.faceCentres()[n][0]<<
                " " <<SailMesh.faceCentres()[n][1]<<
    " " << SailMesh.faceCentres()[n][2]<<"\n";
   }

 myfile.close();
```

Output the patch face centres to file /2:

```
std::ofstream myfile;
myfile.open ("FluidMesh_centres.txt", std::ios::app );

myfile<<nFaces<<"\n";
for (int n=0; n<nFaces; n++)
  {
    myfile<<SailMesh.faceCentres()[n][0]<<
                " " <<SailMesh.faceCentres()[n][1]<<
    " " << SailMesh.faceCentres()[n][2]<<"\n";
   }

 myfile.close();
```

faceCentres() is again a template function, applied here to an object of the class boundaryMesh, it returns the vector (x,y,z) with the coordinates of the centres of the mesh faces

Output the patch face centres to file /2:

```
std::ofstream myfile;
myfile.open ("FluidMesh_centres.txt", std::ios::app );

myfile<<nFaces<<"\n";
for (int n=0; n<nFaces; n++)
  {
    myfile<<SailMesh.faceCentres()[n][0]<<
                " " <<SailMesh.faceCentres()[n][1]<<
    " " << SailMesh.faceCentres()[n][2]<<"\n";
  }

 myfile.close();
```

faceCentres() is again a template function, applied here to an object of the class boundaryMesh, it returns the vector (x,y,z) with the coordinates of the centres of the mesh faces

The operator [] is used in C++ for accessing the vector entries

Output the patch face centres to file /2:

```
std::ofstream myfile;
myfile.open ("FluidMesh_centres.txt", std::ios::app );

myfile<<nFaces<<"\n";
for (int n=0; n<nFaces; n++)
  {
    myfile<<SailMesh.faceCentres()[n][0]<<
                " " <<SailMesh.faceCentres()[n][1]<<
    " " << SailMesh.faceCentres()[n][2]<<"\n";
   }

 myfile.close();
```

Read the cell displacements from file /1:

```
#include "readCellDisp.h"

volVectorField::GeometricBoundaryField &meshDisplacement =
        const_cast<volVectorField&>
  (mesh.objectRegistry::lookupObject<volVectorField>("cellDisplacement"))
                        .boundaryField();
```

Read the cell displacements from file /1:

```
#include "readCellDisp.h"

volVectorField::GeometricBoundaryField &meshDisplacement =
        const_cast<volVectorField&>
  (mesh.objectRegistry::lookupObject<volVectorField>("cellDisplacement"))
                        .boundaryField();
```

Searches in the objectRegistry of the mesh the entry “cellDisplacement”, which is an object of the class volVectorField
ObjectRegistry is function member of the class fvMesh

Read the cell displacements from file /1:

```
#include "readCellDisp.h"

volVectorField::GeometricBoundaryField &meshDisplacement =
        const_cast<volVectorField&>
    (mesh.objectRegistry::lookupObject<volVectorField>("cellDisplacement"))
                    .boundaryField();
```

Searches in the objectRegistry of the mesh the entry "cellDisplacement", which is an object of the class volVectorField
ObjectRegistry is function member of the class fvMesh

From the cellDisplacement vector, searches the entries corresponding to the boundaryField (inlet, outlet, walls, sail...)

Read the cell displacements from file /1:

```
#include "readCellDisp.h"

volVectorField::GeometricBoundaryField &meshDisplacement =
        const_cast<volVectorField&>
    (mesh.objectRegistry::lookupObject<volVectorField>("cellDisplacement"))
                        .boundaryField();
```

Whatever is returned, forces it to be a reference to a volVectorField.
We have then the address in the computer memory where displacements of the boundary field cells are stored

Searches in the objectRegistry of the mesh the entry “cellDisplacement”, which is an object of the class volVectorField
ObjectRegistry is function member of the class fvMesh

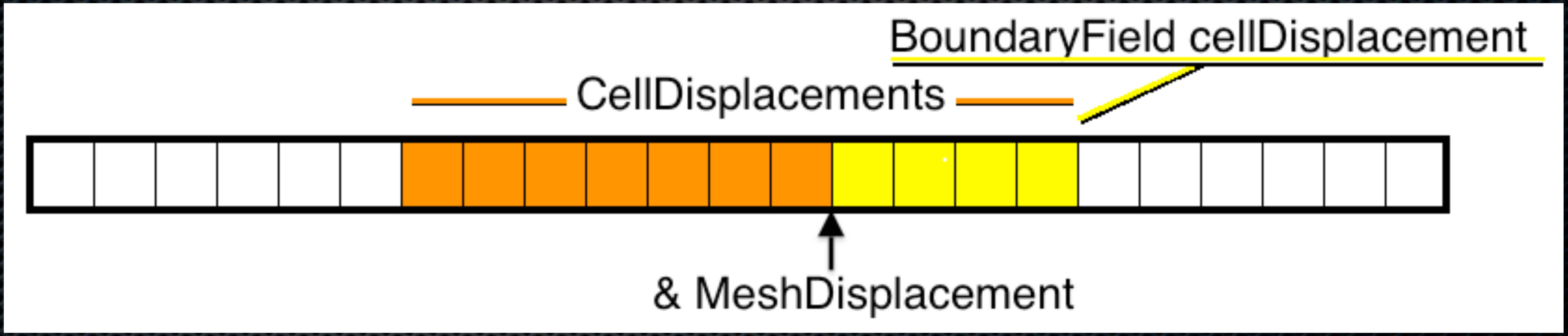
From the cellDisplacement vector, searches the entries corresponding to the boundaryField (inlet, outlet, walls, sail...)

Read the cell displacements from file /1:

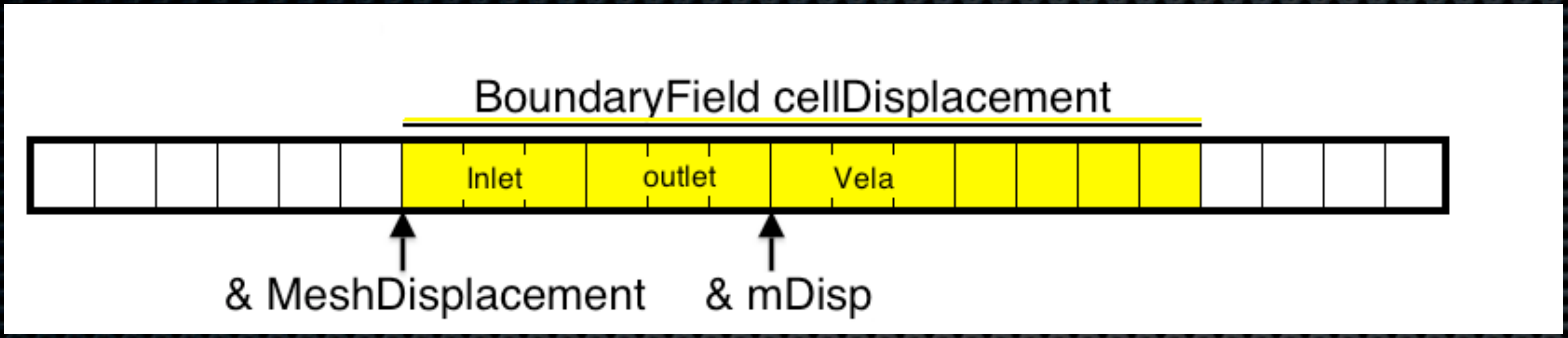
```
#include "readCellDisp.h"

volVectorField::GeometricBoundaryField &meshDisplacement =
        const_cast<volVectorField&>
  (mesh.objectRegistry::lookupObject<volVectorField>("cellDisplacement"))
                        .boundaryField();
```

Read the cell displacements from file /1:

```
#include "readCellDisp.h"

volVectorField::GeometricBoundaryField &meshDisplacement =
        const_cast<volVectorField&>
  (mesh.objectRegistry::lookupObject<volVectorField>("cellDisplacement"))
                    .boundaryField();
```



Read the cell displacements from file /2:



We need now to find the adress of the moving patch (identified by the variable "sail") in the global cellDisplacement Boundary-Field

```
label fluidSideI = mesh.boundaryMesh().findPatchID(sail);

const polyPatch &fluidMesh=mesh.boundaryMesh()[fluidSideI];

labelList interfacePointLabels = mesh.boundaryMesh()[fluidSideI].meshPoints();

vectorField &mDisp=refCast<vectorField>(meshDisplacement[fluidSideI]);
```

Read the cell displacements from file /3:

```
std::ifstream fin("Cell_Displ_out");
int pSize=fluidMesh.size();
vector move;
fin>>pSize;

double u;
double v;
double w;

List<vector> dispVals(fluidMesh.size());
```

Read the cell displacements from file /3:

Read the cell displacements from file /3:

```
forAll(fluidMesh,i) {
    fin >> u;
    fin >> v;
    fin >> w;  // Values from the structrual solver

    dispVals[i][0]=u;
    dispVals[i][1]=v;
    dispVals[i][2]=w;

    vector dd = dispVals[i];;

    mDisp[i]=dd;
  }
```