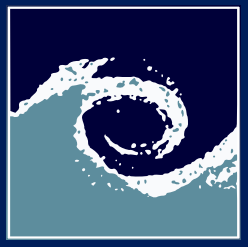
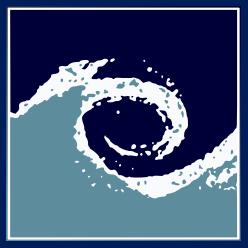


Advaced post-processing

Open-Source CFD Course 2021 – Lab session 9

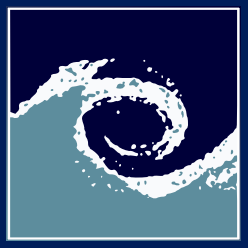
Miklós BALOGH
and
Josh DAVIDSON

2021



Laboratory tasks

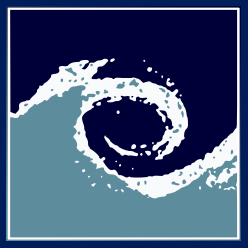
1. Use the original floatingObjectWeek7 case as a donor case: Class Materials > Cases > floatingObjectWeek7.zip
   - Double the mesh resolution in each direction
   - Modify the setup to make it parallel
   - Add the corresponding entries to controlDict
     - Function to monitor rigid body state
     - Function to calculate vorticity in fluid domain
     - Iso-surface for water-air interface with U, p and vorticity fields
     - Cutting planes with x and y normals with the origin of the domain with U, p, vorticity and alpha.water fields
     - Patch surface of the rigid body with U, p, vorticity and alpha.water fields
     - Function to monitor forces acting on the rigid body
   - Create a gnuplot script to monitor quantities in run-time



Doubling the mesh resolution

In the blockMeshDict file

```
...

blocks
(
    hex (0 1 2 3 4 5 6 7) (40 40 60) simpleGrading (1 1 1)
);

...
```



Make it parallel

Check the number of available processors

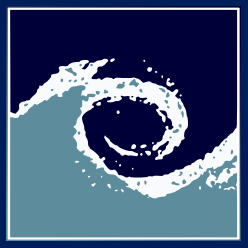
```
# Command line
$ nproc
```

In the decomposParDict file

```
// Based on the available number of processors
// if you have 2 (nproc = 2), set 2
// if you have more, set nproc - 1
numberOfSubdomains 2;

method          scotch;
...
```

In the Allrun script

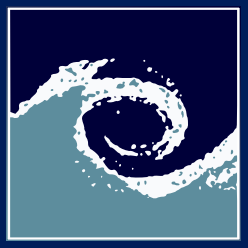
```
# runApplication $application
runApplication decomposePar
runParallel $application
runApplication reconstructPar
```



Include functions to controlDict

Monitoring rigid body state and calculate vorticity

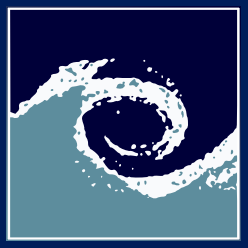
```
...
functions
{
    vorticity
    {
        libs             (fieldFunctionObjects);
        type             vorticity;
        executeControl   timeStep;
        executeInterval  1;
        writeControl     writeTime;
    }

    // The entries on the following slides should be included here
}
```



Include functions to controlDict

Iso-surface for water-air interface

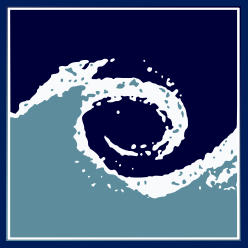
```
waterSurface
{
    type            surfaces;
    libs            (sampling);
    writeControl    adjustableRunTime;
    writeInterval   0.1;

    surfaceFormat   vtk;
    fields          ( p U vorticity );

    interpolationScheme        cellPoint;

    surfaces
    (
      isoAlphaWater05
      {
          type                isoSurface;
          isoField            alpha.water;
          isoValue            0.5;
          interpolate         true;
      }
    );
}
```



Include functions to controlDict

Surfaces for image creation

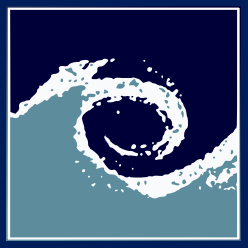
```
cuttingPlaneImage
{
    type            surfaces;
    libs            (sampling);
    writeControl    adjustableRunTime;
    writeInterval   0.1;

    surfaceFormat   vtk;
    fields          ( p U vorticity alpha.water );

    interpolationScheme cellPoint;

    surfaces
    (
      // Definitions on the next slide
    );
}
```



Include functions to controlDict

Cutting planes and patch surface of the rigid body

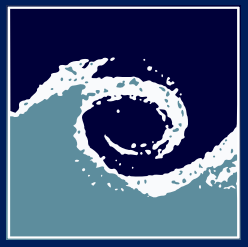
```
yNormal
{
    type            cuttingPlane;
    planeType       pointAndNormal;
    pointAndNormalDict
    {
        point       (0.5 0.5 0.5);
        normal      (0 1 0);
    }
    interpolate     true;
}
xNormal
{
    type            cuttingPlane;
    planeType       pointAndNormal;
    pointAndNormalDict
    {
        point       (0.5 0.5 0.5);
        normal      (1 0 0);
    }
    interpolate     true;
}
floatingObjectWall
{
    type patch;
    patches ( floatingObject );
    interpolate     true;
}
```



Include functions to controlDict

Last entry in function list for forces acting on the rigid body

```
...
forces
{
    type forces;
    libs ( forces );
    patches ( floatingObject );
    rhoInf  998.2;
    log     on;
    writeControl   timeStep;
    writeInterval  1;
    CofR (0.5 0.5 0.5);
}
```

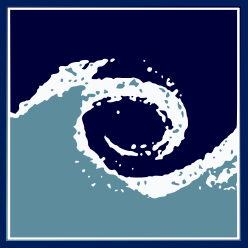


Create and run gnuplot macro

Create e.g. runTimeMonitor.p file with the following content

```
set multiplot layout 3, 1

system("cat log.interFoam | grep 'Time = ' | cut -d' ' -f3- | sed '/ClockTime/d' > t.dat")
system("cat log.interFoam | grep 'Centre of rot' | cut -d' ' -f10 | tr -d ')' > p.dat")
system("cat log.interFoam | grep 'Linear velocity:' | cut -d' ' -f9 | tr -d ')' > .dat")
system("paste t.dat p.dat > tpos.dat")
system("paste t.dat v.dat > tvel.dat")
force="< cat postProcessing/forces/0/force.dat | tr -d '()'"

set title "Position"
set ylabel "Position [m]"
set xlabel "Time [s]"
plot posv using 1:2 title 'z' with lines

set title "Linear velocity"
set ylabel 'Linear velocity [m/s]'
set xlabel 'Time [s]'
plot posv using 1:2 title 'z' with lines

set title "Forces"
set ylabel 'Forces [N]'
set xlabel 'Time [s]'
plot force using 1:4 title 'F_z' with lines

pause 10
reread
```

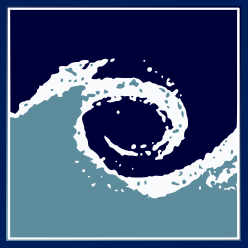
```
# Running the macro (in the case folder while the sim is running)
gnuplot runTimeMonitor.p
```



Collect and rename vtk files

Create a shell script (collectVTK.bash) for collecting images

```
#!/bin/sh

outputDir="VTKfiles"
postProcDir="postProcessing"
surfList="waterSurface cuttingPlaneImage"

if [ -d "$outputDir" ] ; then
  rm -r $outputDir
fi
mkdir $outputDir

for surf in $surfList; do
  timeDir="$postProcDir/$surf/"
  i=0
  for timeStep in $(ls -v $timeDir); do
    timeStepDir="$timeDir$timeStep"
    i=$((i+1))
    for srcFileName in $(ls -v $timeStepDir); do
      extension="${srcFileName##*.}"
      fileName="${srcFileName%.*}_"
      newFileName=$(printf "$fileName%04d.$extension" "$i")
      cp $timeStepDir/$srcFileName $outputDir/$newFileName
    done
  done
done
```



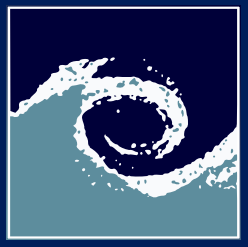
Create video from the files

Create a shell script for collecting images

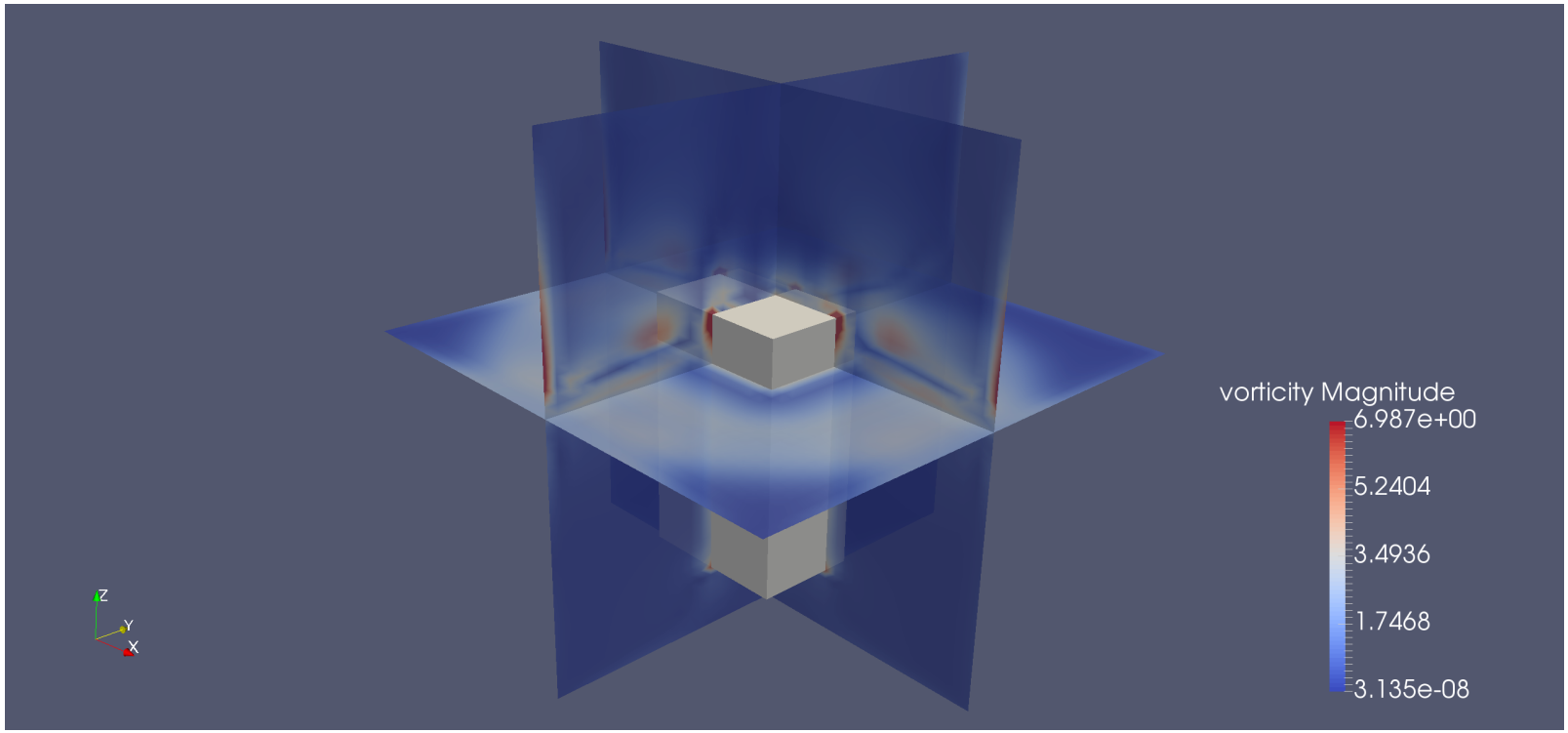
```
# Run shell script
chmod +x collectVTK.bash
./collectVTK.bash
```

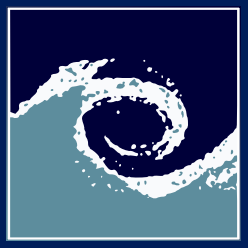
Read them into paraview Save animation (image sequence)
Create video with ffmpeg

```
ffmpeg -framerate 24 -i <images_name>.%04d.png -c:v libx264 <video_name>.mp4
```



Resulted video





Assignments

1. Create a plot from the time history of position, velocity and force in png format! (1 bonus mark)
2. Upload nice visualization images in png format from the case! (1 bonus mark for each)
3. Upload a video similar to the reference! (5 bonus mark)