



OpenFOAM®

3rd Iberian Meeting

11 & 12 June, 2019 Porto - Portugal

Advanced Courses (A1)

Introduction to swak4Foam and PyFoam

Bruno Santos: bruno.santos@fsdynamics.pt

CC BY-NC license

ceft
TRANSPORT PHENOMENA RESEARCH CENTER

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO


Universidade do Minho


UNIVERSIDADE DE
COIMBRA

 CESGA

 IH Cantabria
INSTITUTO DE HIDRAULICA AMBIENTAL
UNIVERSIDAD DE CANTABRIA

 MARE
centro de
ciencias do mar
e do ambiente
UCoimbra

 dtx
Digital
Transformation
CoLab

 cidem

 blueCaPe
an FS DYNAMICS company

 ineqi
driving science
& innovation

ceft
TRANSPORT PHENOMENA RESEARCH CENTER

3GOMPUTE

OpenVCFD®

FUJITSU

Contents

1. Introduction
2. Getting ready to work
3. swak4Foam – pre-processing utilities
4. swak4Foam – boundary conditions
5. swak4Foam – function objects
6. PyFoam
7. Further information

Target Versions:

- OpenFOAM 6 (might work with OpenFOAM-v1812)
- swak4Foam (branch version_0.4.2_v2.x) and PyFoam 0.6.10

Introduction (1/4)

swak4Foam

- **SW**iss **A**rmey **K**nife **f**or **F**oam.
- Its primary feature is the power of mathematical expressions, no C++ required, e.g.:
 - $10*(1+0.5*\sin(500*time()))$
 - $15*pow(x,2)+20*pow(y,2)$
- Pre-processing utilities
- Boundary conditions
- Function Objects (co-processing)
- openfoamwiki.net/index.php/Contrib/swak4Foam

Introduction (2/4)

Why was **swak4Foam** created:

- OpenFOAM is a CFD toolbox
- It's coded in C++
- Whenever a feature is missing, it's expected the user to code it in C++

swak4Foam aims to bypass the requirement to code in C++, by empowering the user with capabilities that don't exist yet in OpenFOAM, without the need to rely on coding in C++.

Introduction (3/4)

PyFoam

- Helps unleash the power of Python, applied to controlling and manipulating OpenFOAM cases
- Features:
 - Case running utilities
 - Utilities for log files
 - Networking utilities
 - Utilities for manipulating case data
 - Scripted manipulation of dictionaries
 - ParaView related utilities (requires Python in ParaView)
 - GUI-Tools (e.g. **pyFoamDisplayBlockMesh**)
- openfoamwiki.net/index.php/Contrib/PyFoam

Introduction (4/4)

Why was **PyFoam** created:

- OpenFOAM relies on:
 - conventional shell scripting (usually **bash**) for handling cases;
 - the user to either post-process results manually or with one's own scripts.
- PyFoam aims to provide:
 - a common library infrastructure, built on top of Python, for manipulating and processing cases;
 - a common scripting toolkit for managing the cases.

Getting ready to work (1/2)

1. Unpack the cases package, by running the following command:

```
tar -xf swak4Foam_and_PyFoam_FOAMatIBERIA_2019.tar.gz
```

You can quickly type it, by first typing:

```
tar -xf swak |
```

Then press the TAB key while the cursor is right next to the “k” letter, where the red vertical bar is, then the autocomplete functionality should complete the rest of the file name.

Getting ready to work (2/2)

2. Then go into the unpacked folder that has the cases that we will be running in this session and what case folders are there, by running the following commands:

```
cd cases_swak4Foam_PyFoam  
ls -l
```

What each case folder does will be revealed during this session in the next slides!

Note: Inside each case folder is the folder “images”, which has the final result that is also shown in this presentation.

swak4Foam – Pre-Processing

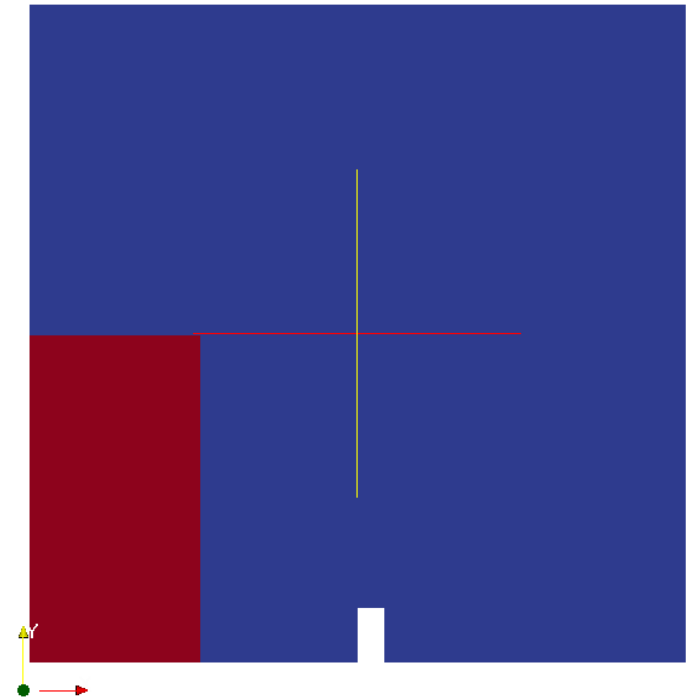
We will be addressing how to use two of several utilities that swak4Foam provides:

- **funkySetFields** – used for setting the initial internal fields, by using mathematical expressions. The name derives from OpenFOAM's own **setFields**, which is limited to preexisting geometries and fixed values.
- **funkySetBoundaryField** – used for setting the values for fixed boundary conditions.

funkySetFields (1/11)

Original tutorial case:

- “multiphase/interFoam/RAS/damBreak/damBreak”
- Static column of water
- Width of column: 0.1461 m
- Height of column: 0.292 m
- Non-moving obstacle at $X = 0.292$ m, width= 0.024 m
- Domain size:
 - width=0.584 m
 - height= 0.584 m



funkySetFields (2/11)

Our example case:

- Case folder: “funkySetFieldsDamBreak”
- Objective is to define the initial internal field:
 - 2D circle of water of 0.05m
 - Centred at $x=0.14$, $y=0.2$ m
 - Added pressure $+100*y$ (in Pascal)
 - Traveling upward at 1.5 m/s

funkySetFields (3/11)

Dictionary file: “system/funkySetFieldsDict”

```
FoamFile
{
    version      2.0;
    format        ascii;
    class         dictionary;
    location      "system";
    object        funkySetFieldsDict;
}

expressions
(
//...
);
```

funkySetFields (4/11)

Basic parameters for each *expression*:

- ***field*** – to specify the name of the field to change.
- ***expression*** – to specify the expression to use for the new field values.
- ***condition*** – to define where the expression is applicable.
- ***keepPatches*** – define *true* or *false*, where *false* will discard the existing boundary conditions.

funkySetFields (5/11)

Expressions to initialize phase and velocity:

```
initFieldAlpha
{
    field alpha.water;
    expression "0";
    keepPatches true;
}
```

```
initFieldU
{
    field U;
    expression "vector(0.0,0.0,0.0)";
    keepPatches true;
}
```

funkySetFields (6/11)

Expression to initialize “pressure - $\rho \cdot g \cdot h$ ”:

```
pressureAir
{
    field p_rgh;
    expression "0";
    keepPatches true;
}
```

funkySetFields (7/11)

Expression to initialize the phase for the water circle:

```
floatingCircle
{
    field alpha.water;
    expression "1";
    condition
        "sqrt(pow((pos().x-0.14),2)+pow((pos().y-0.2),2))<0.05";
    keepPatches true;
}
```


funkySetFields (8/11)

Expression to initialize the added pressure for the water circle:

```
pressureCircle
{
    field p_rgh;
    expression "100.0*pos().y";
    condition
        "sqrt(pow((pos().x-0.14),2)+pow((pos().y-0.2),2))<0.05";
    keepPatches true;
}
```

funkySetFields (9/11)

Expression to initialize the initial velocity for the water circle:

```
risingCircle
{
    field U;
    expression "vector(0.0,1.5,0.0)";
    condition
        "sqrt(pow((pos().x-0.14),2)+pow((pos().y-0.2),2))<0.05";
    keepPatches true;
}
```

funkySetFields (10/11)

To run the case, simply run:

```
./Allrun
```

Or run manually each step:

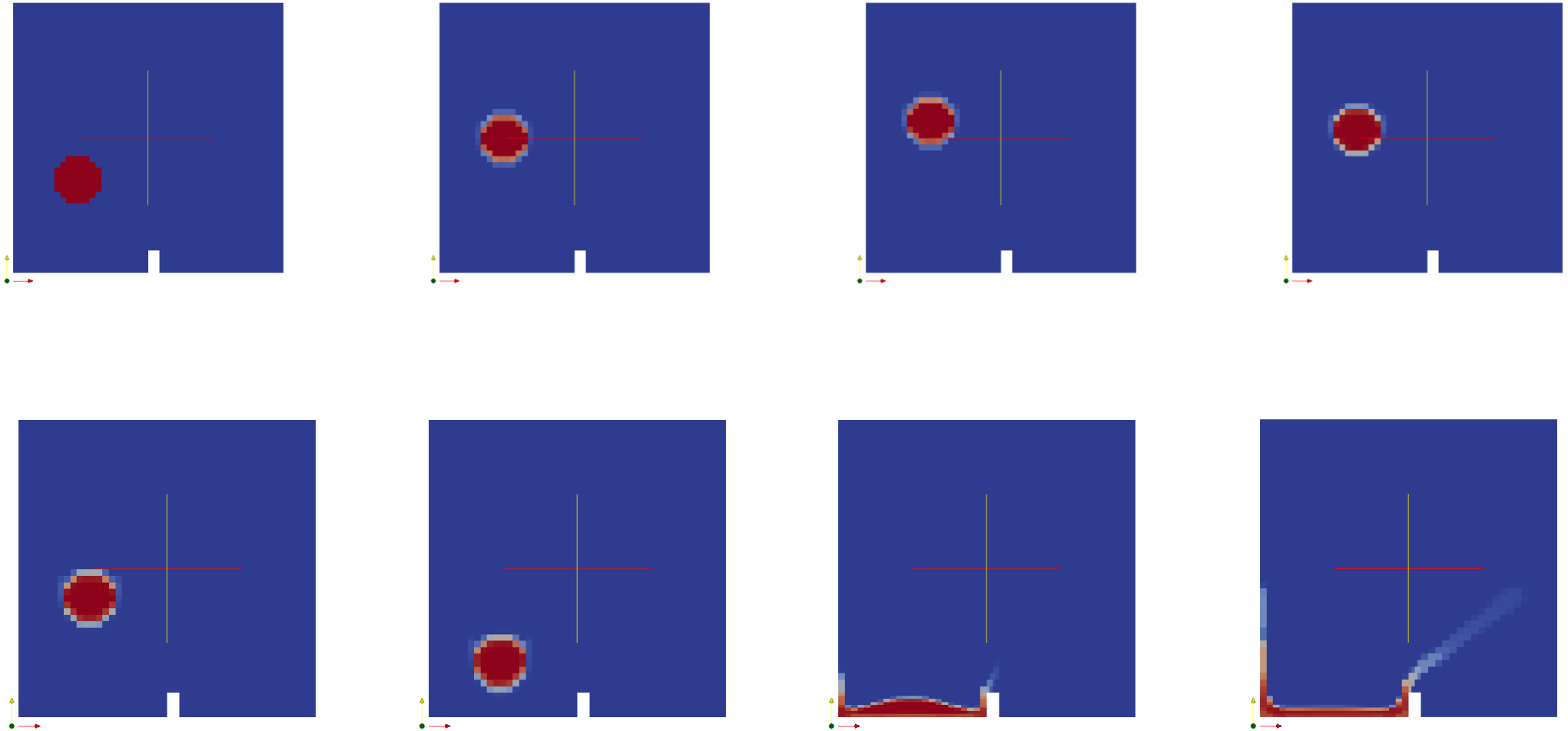
```
cp 0/alpha.water.org 0/alpha.water  
cp 0/U.org 0/U  
cp 0/p_rgh.org 0/p_rgh  
blockMesh  
funkySetFields -time 0  
interFoam
```

Then open it in ParaView:

```
paraFoam
```

And choose to see the “alpha.water” field.

funkySetFields (11/11)



Left as a future exercise: Check the differences for the results reached in the past with OpenFOAM 3.0.x, as shown above and inside the “images” folder.

funkySetBoundaryField (1/4)

It's essentially **funkySetFields**, for manipulating only the boundary fields on the surface mesh.

Specifically, it can operate on dictionary entries like this one:

```
value                uniform (0 0 0);
```

We will also use the previous case and add a new dictionary file...

funkySetBoundaryField (2/4)

... “system/funkySetBoundaryDict”:

```
blowerLeftWall
{
    field U;
    expressions
    (
        {
            target value;
            patchName leftWall;
            variables "maxY=max(pts().y);thres=0.5*maxY;";
            expression
            "(pos().y<thres)?vector(3,3,0)*(maxY-pos().y):vector(0,0,0)";

        }
    );
}
```

funkySetBoundaryField (3/4)

To run the case, simply run:

```
./Allrun
```

Or run manually each step:

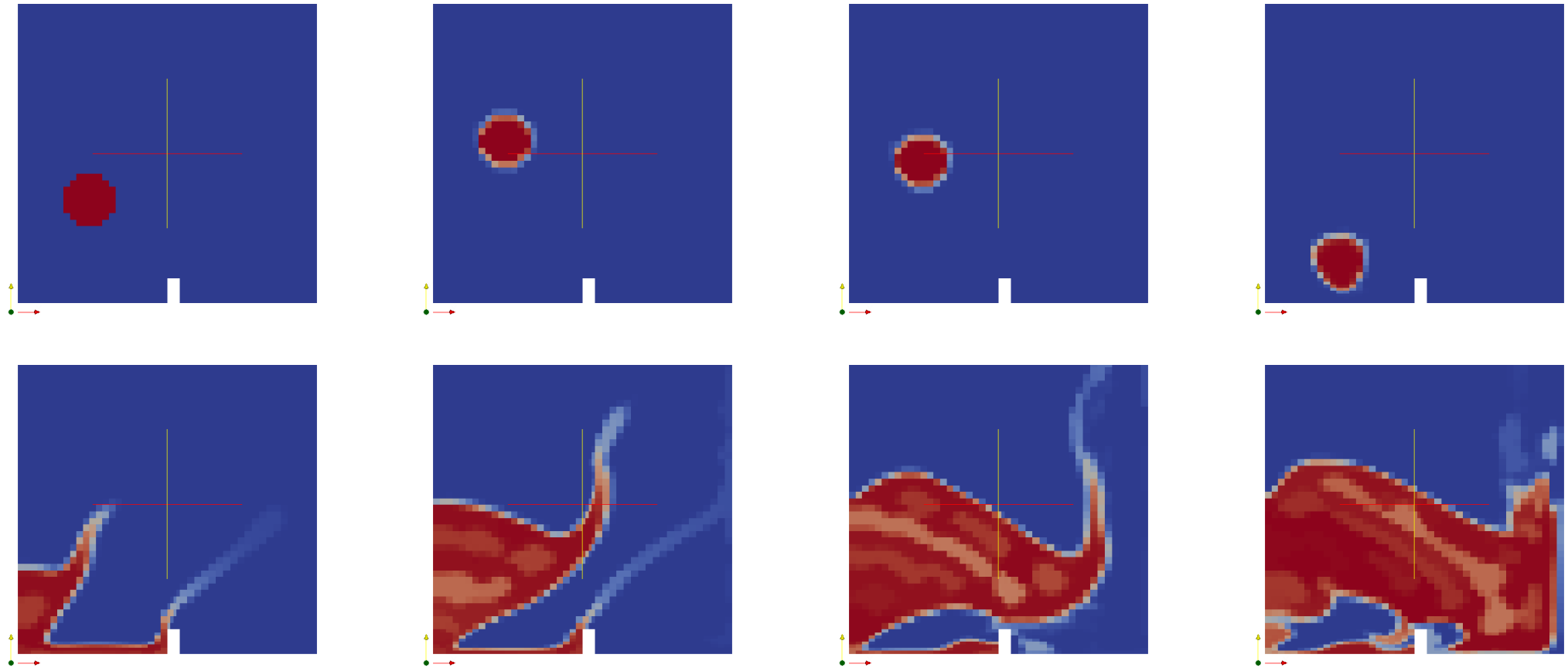
```
cp 0/alpha.water.org 0/alpha.water  
cp 0/U.org 0/U  
cp 0/p_rgh.org 0/p_rgh  
blockMesh  
funkySetFields -time 0  
funkySetBoundaryField -time 0  
interFoam
```

Then open it in ParaView:

```
paraFoam
```

(And choose to see the “alpha.water” field.)

funkySetBoundaryField (4/4)



Check the differences for the results reached in the past with OpenFOAM 3.0.x, as shown above and inside the “images” folder.

Because you can see that the results with OpenFOAM 6 are different than the above, are either one preserving mass properly?

swak4Foam – groovyBC (1/7)

funkySetBoundaryField can initialize fields, but what if we need them to be time/iteration dependant?

This is where *groovyBC* comes in!

Objective:

1. We use the case from the **funkySetFields** slides.
2. Use groovy BC for applying an air jet at 20 m/s.
3. Air jet works within the 0.1 and 0.2 second range.
4. Location is in the lower wall, with X within 0.12 and 0.16 metre.
5. The resulting case is in folder “groovyBCDamBreak”.

swak4Foam – groovyBC (2/7)

Edit the file “0/U.org”, scroll down to the end of the file and find “lowerWall”. Replace it with this:

```
lowerWall
{
    type                groovyBC;
    value                uniform (0 0 0);
    variables
    (
        "vel=20.0;"
        "minX=0.12;"
        "maxX=0.16;"
    );
    valueExpression
    "(0.1<=time() && time() <=0.2) && (minX<=pos().x) && (pos().x<=
maxX)?vector(0,vel,0):vector(0,0,0)";
}
```

swak4Foam – groovyBC (3/7)

Edit the file “system/controlDict”, scroll down to the end of the file and add this line:

```
libs ( "libgroovyBC.so" );
```

Notes:

- Make sure you only have 1 entry named “libs”.
- For loading more than one library, list them, e.g.:

```
libs ( "libgroovyBC.so" "libOpenFOAM.so" );
```

swak4Foam – groovyBC (4/7)

To run the case, simply run:

```
./Allrun
```

Or run manually each step:

```
cp 0/alpha.water.org 0/alpha.water  
cp 0/U.org 0/U  
cp 0/p_rgh.org 0/p_rgh  
blockMesh  
funkySetFields -time 0  
interFoam
```

Then open it in ParaView:

```
paraFoam
```

swak4Foam – groovyBC (5/7)

In ParaView (1/2):

1. Select “groovyBCDamBreak” (Pipeline Browser).
2. Change representation to the “alpha.water” field.
3. Menu: Filters → Common → Stream Tracer
4. Turn on the advanced options for “StreamTracer1” (it’s the button with the little gear symbol).
5. “Seed Type” → “High Resolution Line Source”
6. Click on the “X Axis” button.
7. “Resolution” → 50
8. Click on the “Apply” button.

swak4Foam – groovyBC (6/7)

In ParaView (2/2):

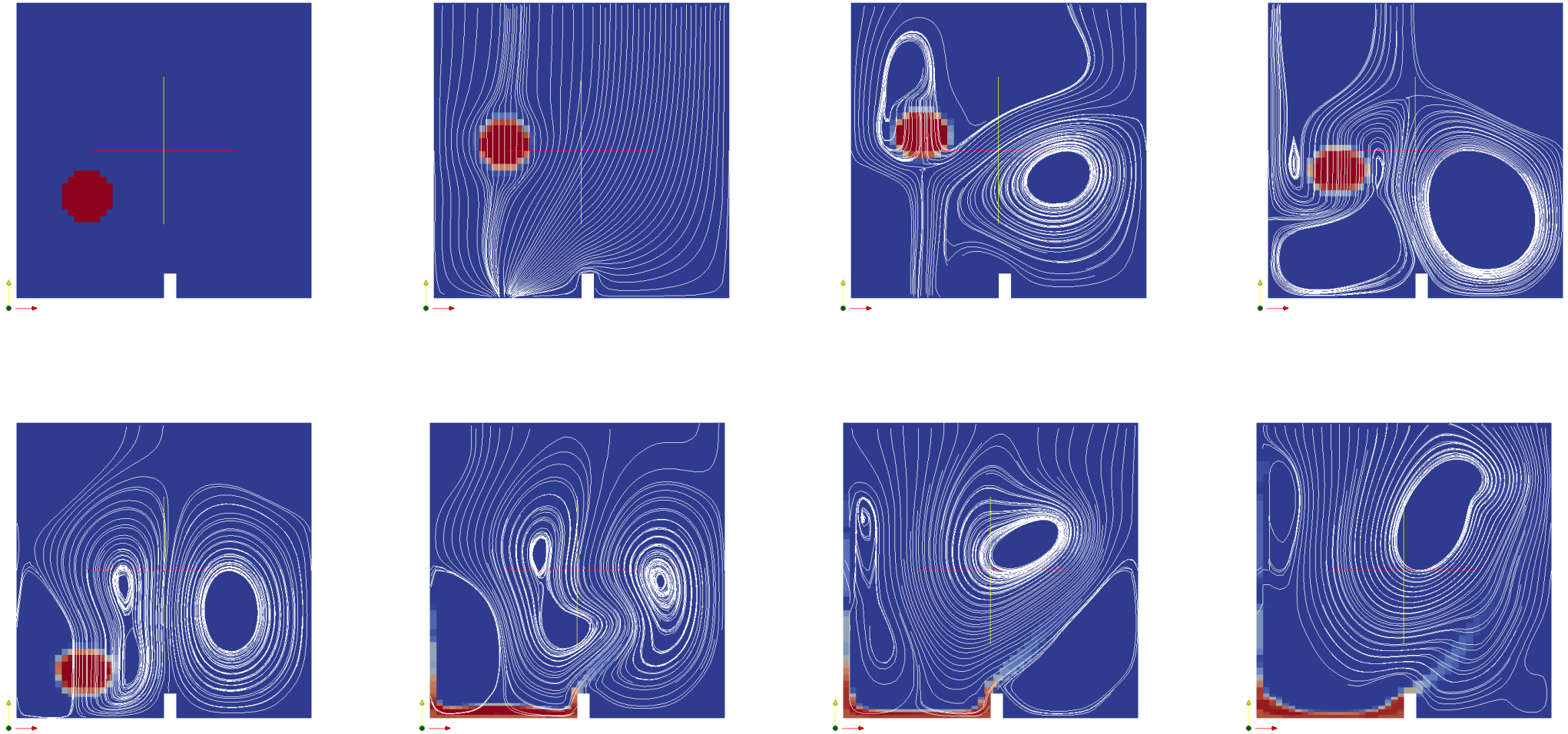
9. Menu: Filters → Alphabetical → Transform

10. “Translation”: 0 0 0.15

11. Click on the “Apply” button.

12. Go to the next or last time step, to check if the stream lines appear.

swak4Foam – groovyBC (7/7)



swak4Foam – Function Objects

As a reminder: function objects provide post-processing capabilities while the simulation is running, which is somewhat commonly known as co-processing.

We will be addressing how to use three of several function objects that swak4Foam provides:

- **patchExpression** – Calculate any mathematical expression for a particular patch.
- **swakExpression** – Calculate any mathematical expression for any type of region for a new field.
- **expressionField** – Simply calculate a completely new field.

patchExpression (1/11)

Need to calculate the mass flow rate going through a patch?

- OpenFOAM 6 has the “surfaceFieldValue” function object.

Need to calculate the mass flow rate going through a patch, in pound per hour (lb/h)?

- **patchExpression** can do that and a lot more!

patchExpression (2/11)

Example case:

- Original: “incompressible/simpleFoam/pitzDaily”
- Case folder: “patchExpressionPitzDaily”
- Objective:
 - Calculate the volumetric flow in m^3/s
 - Calculate the mass flow in kg/s
 - Calculate the mass flow in lb/h
 - Get the maximum, minimum and average volumetric/mass flow values on the faces of the “inlet” and “outlet” patches.

patchExpression (3/11)

Create a copy of the tutorial and go into that folder:

```
cp -r $FOAM_TUTORIALS/incompressible/simpleFoam/pitzDaily \
    MyPatchExpressionPitzDaily
cd MyPatchExpressionPitzDaily
```

(Or if you are feeling lazy, it's already fully prepared in the folder "patchExpressionPitzDaily".)

Edit the file "system/controlDict", scroll down to the end of the file and add these lines:

```
libs (
    "libsimpleSwakFunctionObjects.so"
    "libswakFunctionObjects.so"
);
```

patchExpression (4/11)

Still in the file “system/controlDict”, find this block:

```
functions
{
    #includeFunc streamlines
}
```

We will add the next blocks inside the block “functions”, after the end of the block “streamLines”.

patchExpression (5/11)

The first block (accurate volumetric flow rate):

```
volumetricFlowSurfaceField
{
    type patchExpression;
    outputControlMode    outputTime;
    verbose true;
    accumulations (
        sum max min average
    );
    patches (
        inlet
        outlet
    );
    expression "phi";
}
```

patchExpression (6/11)

The second block (less accurate):

```
volumetricFlowVolumeField
{
    type patchExpression;
    outputControlMode    outputTime;
    verbose true;
    accumulations (
        sum max min average
    );
    patches (
        inlet
        outlet
    );
    expression "U&Sf() ";
}
```

patchExpression (7/11)

The third block (mass flow rate kg/s):

```
massFlowSurfaceField
{
    $volumetricFlowSurfaceField;
    patches (
        inlet
        outlet
    );
    variables (
        "rhoAir=1.2041;"
    );
    expression "phi * rhoAir";
}
```

patchExpression (8/11)

The fourth block (mass flow rate lb/s):

```
massFlowSurfaceFieldInPoundPerHour
{
    $volumetricFlowSurfaceField;
    patches (
        inlet
        outlet
    );
    variables (
        "rhoAir=1204.1;"
        "poundPerHour=2.20462*3600.0;"
    );
    expression "U&Sf() * rhoAir * poundPerHour";
}
```


patchExpression (9/11)

For running the case:

```
foamRunTutorials
```

Location of the results (formatted with fixed width):

```
ls -l postProcessing/patchExpression_*/*
```

For later clean up the case:

```
foamCleanTutorials
```

patchExpression (10/11)

The results are also available in the file “log.simpleFoam”.
Example:

```
Expression volumetricFlowSurfaceField on outlet:  
sum=0.000254001 max=6.97636e-006 min=7.27091e-007  
average=4.45616e-006
```

```
Expression volumetricFlowSurfaceField on inlet:  
sum=-0.000254 max=-3.13389e-006 min=-1.78262e-005  
average=-8.46667e-006
```

```
Expression volumetricFlowVolumeField on outlet:  
sum=0.00025273 max=6.9322e-006 min=7.08664e-007  
average=4.43386e-006
```

```
Expression volumetricFlowVolumeField on inlet:  
sum=-0.000254 max=-3.13389e-006 min=-1.78262e-005  
average=-8.46667e-006
```

patchExpression (11/11)

What else can it do? A lot more! One last example:

```
deltaP
{
    type patchExpression;
    accumulations (
        min max average
    );
    patches (
        inlet
    );
    variables "pOut{outlet}=average(p) ; ";
    expression "p-pOut";
    verbose true;
}
```

Source: “swak4Foam/Examples/groovyBC/pulsedPitzDaily”

swakExpression (1/4)

The function object *patchExpression* is essentially derived from *swakExpression*, which is able to operate on following types of mesh domains:


- cellSet
- cellZone
- faceSet
- faceZone
- internalField
- patch
- set
- surface

swakExpression (2/4)

Copy the previous folder and replace all other function objects with just this one:

```
absolutePressureStats
```

```
{  
    type swakExpression;  
    valueType internalField;  
    variables (  
        "rhoAir=1.2041;"  
        "refP=101325;"  
    );  
    expression "p*rhoAir + refP";  
    verbose true;  
  
    outputControlMode    outputTime;  
}
```



```
accumulations (  
    average  
    weightedAverage  
    median  
    weightedMedian  
    quantile0.50  
    weightedQuantile0.50  
    quantile0.75  
    weightedQuantile0.75  
);
```

swakExpression (3/4)

Notes:

- Quantile 50% is the median
- The weighted variants are based on the volumes of each cell
- The number next to the name of an *accumulation* is the argument for it:
 - quantile0.75 → quantile 75%
- Running the case is done the same way as the previous example, i.e.: **foamRunTutorials**

swakExpression (4/4)

The tabulated results are in this file:

```
postProcessing/swakExpression_absolutePressureStats/0/absolutePressureStats
```

The file “log.simpleFoam” also has these values, e.g.:

```
Expression absolutePressureStats :  
average=101324 weightedAverage=101328  
median=101319 weightedMedian=101328  
quantile0.5=101319 weightedQuantile0.5=101328  
quantile0.75=101332 weightedQuantile0.75=101337
```

You can compare your modified case with the one we prepared for this session: `swakExpressionPitzDaily`

expressionField

If you ever need to quickly create a new field for sampling or for common use with other function objects, this function object can do it for you. Example:

```
velocityMagSquared
{
    type expressionField;
    outputControl timeStep;
    outputInterval 1;
    fieldName UMag2;
    expression "U&U";
    autowrite true;
}
```


pyFoamPlotRunner (1/2)

Next, it's time for PyFoam to shine.

- Feel the need to easily keep track of the residuals while the solver is running?
- What about keeping track of the residuals and launch the solver in a single command?

Then on the latest case, try the following commands:

```
foamCleanTutorials  
blockMesh  
pyFoamPlotRunner.py simpleFoam
```

pyFoamPlotRunner (2/2)

What else can it do? Try running:

```
pyFoamPlotRunner.py -help
```

e.g., remove time steps + run 200 iterations only + 0.2s
refresh plotting:

```
pyFoamPlotRunner.py --clear-case --run-until=200 \  
--frequency=0.2 --persist simpleFoam
```

pyFoamPlotWatcher

pyFoamPlotRunner is nice, but what if the simulation is already finished? Then use **pyFoamPlotWatcher**!

Examples:

```
pyFoamPlotWatcher.py PyFoamRunner.simpleFoam.logfile  
pyFoamPlotWatcher.py log.simpleFoam
```

Best of all? You can use this script while the solver is running!

Note: Use the key combination *Ctrl+C* in the terminal window to terminate it when you no longer need it to be running.

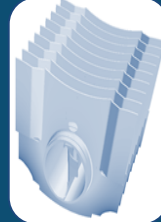
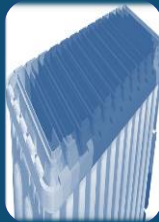
Further Information

This presentation was only a fraction of the tip of the iceberg.

Several presentations are available in the wiki pages of each respective project:

- openfoamwiki.net/index.php/Contrib/swak4Foam:
 - 7.2 Further information
- openfoamwiki.net/index.php/Contrib/PyFoam:
 - 1.3 Other material

Thank you for your time.



bluecape

Computer Applications
in Science & Engineering

an FS  DYNAMICS company