ENCCS
EuroCC National Competence Centre Sweden

KTH
VETENSKAP
OCH KONST
ROYAL INSTITUTE
OF TECHNOLOGY

EXCELLERAT

Introductory OpenFOAM Workshop
Day 2

Timofey Mukha

KTH Engineering Mechanics

Further customize your case

- `fvOptions` – Modify governing equations without changing the code of the solver.
  - Lives in `system` (for latest versions also in `constant`).
  - Examples:
    - Introduce a source term.
    - Bound the values of a field.
- "Function objects" – A common name for a diverse set of utilities.
  - Executed at run time or as a post-processing step.
  - Added to the `functions` sub-dictionary in the `controlDict`.
  - Examples:
    - Time averaging, point probes, compute forces, compute y+, compute new field…

Finding what is out there

- Method 1: RTFM ☺
  - https://www.openfoam.com/documentation/guides/latest/doc/

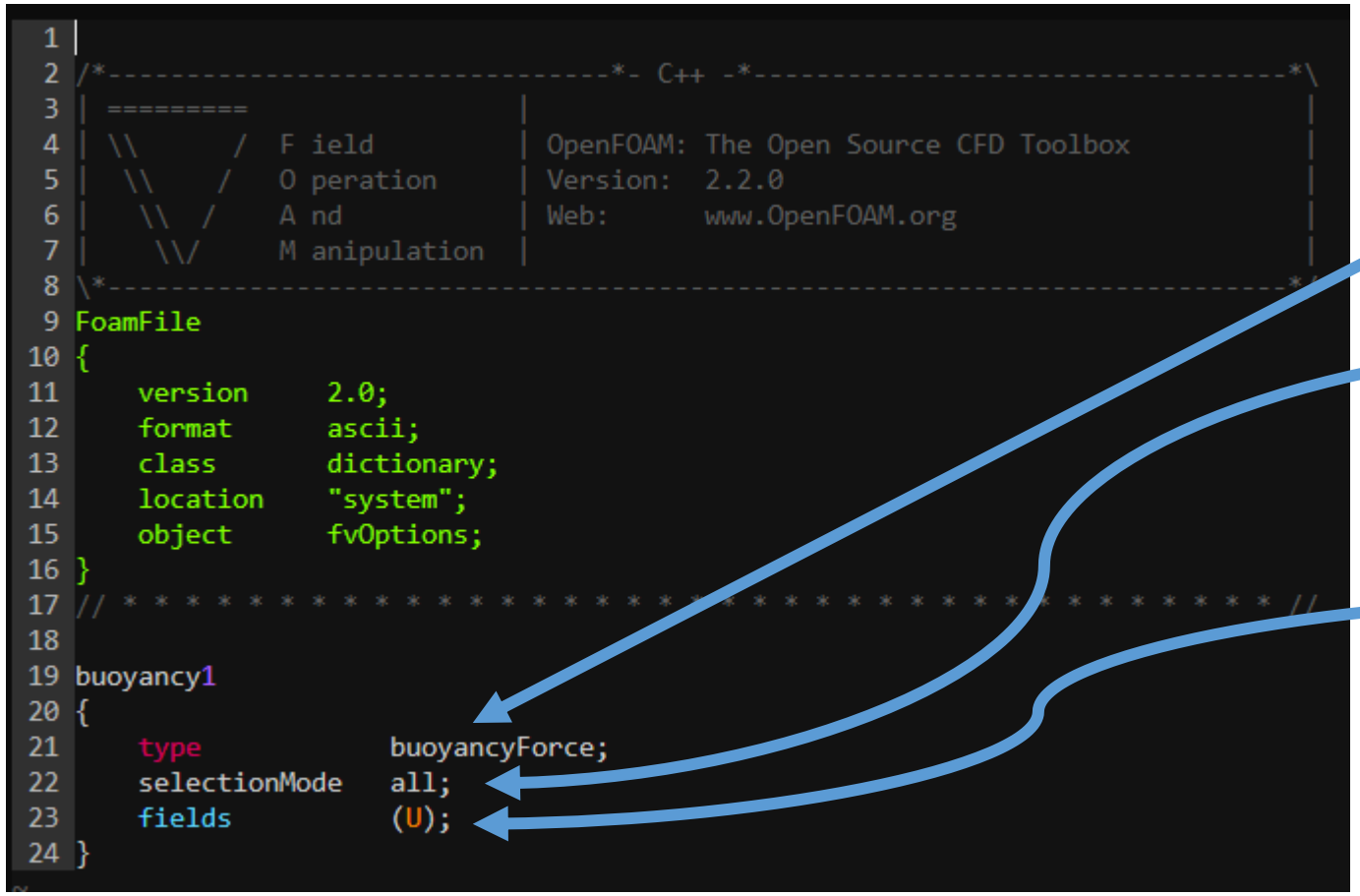
Finding what is out there

- Method 2: the universal banana trick

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvOptions;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

test
{
    type        banana;
}
```

```
--> FOAM FATAL IO ERROR: (openfoam-2106)
Unknown fvOption type banana

Valid fvOption types :

55
(
acousticDampingSource
actuationDiskSource
atmAmbientTurbSource
atmBuoyancyTurbSource
atmCoriolisUSource
atmLengthScaleTurbSource
atmNutSource
atmPlantCanopyTSource
atmPlantCanopyTurbSource
atmPlantCanopyUSource
buoyancyEnergy
```

A simple fvOption



```
1
2  /*--------------------------------*- C++ -*----------------------------------*\
3  | =========                 |                                                 |
4  | \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
5  |  \\    /   O peration     | Version:  2.2.0                                 |
6  |   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
7  |    \\/     M anipulation  |                                                 |
8  \*---------------------------------------------------------------------------*/
9  FoamFile
10 {
11     version     2.0;
12     format      ascii;
13     class       dictionary;
14     location    "system";
15     object      fvOptions;
16 }
17 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
18
19 buoyancy1
20 {
21     type            buoyancyForce;
22     selectionMode   all;
23     fields          (U);
24 }
```

Will add ρg as a source term

Where to apply the extra term. Can be limited to a previously defined cell set.

What equation the source term is applied to.

Function object example: Courant number

```
Co
{
    type            CourantNo;
    libs            (fieldFunctionObjects);
    executeControl  writeTime;
    executeInterval 1;
    log             true;
    writeControl    writeTime;
}
```

File Co put into `system`.
NB: Make the file name and the object name consistent!

```

functions
{
    #include Co;
};
```

In the `controlDict`, we include Co

```
~/OpenFOAM/timofey-v1912/run/of_training/pitzDaily_transient » ls 0.01
uniform  Co  epsilon  k  nut  p  phi  U
```

Co field appears in the output

Execution and write control keywords

- `executionControl` – when is the function object run?
- `writeControl` – when are the results saved to disk?
- `excution/writeInterval` – defines the frequency as per the corresponding Control keyword.

Option	Description
none	Trigger is disabled
timeStep	Trigger every 'Interval' time-steps, e.g. every x time steps
writeTime	Trigger every 'Interval' output times, i.e. alongside standard field output
runTime	Trigger every 'Interval' run time period, e.g. every x seconds of calculation time
adjustableRunTime	Currently identical to "runTime"
clockTime	Trigger every 'Interval' clock time period
cpuTime	Trigger every 'Interval' CPU time period
onEnd	Trigger on end of simulation run

Another example: field min and max

```
fieldMinMax1
{
    // Mandatory entries (unmodifiable)
    type            fieldMinMax;
    libs            (fieldFunctionObjects);

    // Mandatory entries (runtime modifiable)
    mode            magnitude;
    fields          (U p);

    location        true;

    writePrecision  8;
    writeToFile     true;
    enabled         true;
    log             true;
    executeControl  timeStep;
    executeInterval 1;
    writeControl    timeStep;
    writeInterval   1;
}
```

The log file

```
fieldMinMax fieldMinMax1 write:
    min(mag(U)) = 0 in cell 522 at location (-0.0198086 0.0254 0)
    max(mag(U)) = 11.9221 in cell 5968 at location (0.0167861 0.00144538 2.27514e-20)
    min(p) = -54.0397 in cell 3268 at location (0.0167857 -0.00810691 -4.49852e-20)
    max(p) = 20.5416 in cell 610 at location (0.0493639 -0.0248581 0)
```

Time-series saved to postProcessing

```
~/OpenFOAM/timofey-v1912/run/of_training/pitzDaily_transient » tree postProcessing
postProcessing
└── fieldMinMax1
    └── 0
        └── fieldMinMax.dat
```

```
# Field minima and maxima
# Time                  field                 min                     location(min)                 max                     location(max)
0.000120482             mag(U)                0.00000000e+00  (-1.98085815e-02 2.54000000e-02 0.00000000e+00) 2.05470462e+01  (2.64479650e-04 1.59940304e-04 2.63956988e-20)
0.000120482             p                     0.00000000e+00  (2.90000000e-01 -1.62925926e-02 0.00000000e+00) 1.52362573e+04  (-1.98085815e-02 2.47383520e-02 0.00000000e+00)
0.000185479             mag(U)                0.00000000e+00  (-1.98085815e-02 2.54000000e-02 0.00000000e+00) 2.23431008e+01  (2.64479650e-04 1.59940304e-04 2.63956988e-20)
```

Executing after the run

- So far our function objects were executed at runtime.
- But for some it makes sense to run them after the simulation is done.
- For simple function objects we can use the `postProcess` command.
  - `postProcess –list :` Look at available function objects by name. Don't have to be defined by us, they already simply exist.
  - Example: `postProcess –func yPlus`
  - `Can also run the objects we defined: postProcess –func fieldMinMax1`

Executing after the run

- However, some stuff won't run with just `postProcess`
  - Example: `postProcess -func CourantNo` will fail.
- For such cases we actually have to run the solver. But with a special flag!
- `pimpleFoam -postProcess -func CourantNo`

Summary

- fvOptions and functionObject practically remove the need for modifying the solver, as long as it captures your physics.

- Lot’s of fvOptions and functionObjects out there. Try and play with them during the hands on!

Special mentions

- There is a `coded` type of `fvOption` and `functionObject`, which allows you to simply write you own C++ to be executed! Will be compiled when the case runs, with no involvment from your side.

- An addon library called `swak4foam` adds a huge number of function objects allowing to do some really cool stuff! Search for slides by Bernhard F.W. Gschaider – excellent for self-study.