

The Knife, the Snake and the Moon

Ongoing developments in swak4Foam

Bernhard F.W. Gschaider

HFD Research GesmbH

Leoben, Austria

11. December 2017

Outline I

- 1 Introduction
 - Who is this?
- 2 The Knife
 - swak4Foam
 - Advanced swak4Foam
 - What doesn't work
- 3 The Snake
 - Python
 - Integration in swak4Foam
 - Limitations
- 4 The Moon
 - Generalization of scripting
 - Lua
 - Examples



Outline II

- 5 Conclusions
 - When to use what
 - The Future



Outline

- 1 Introduction
 - Who is this?
- 2 The Knife
 - swak4Foam
 - Advanced swak4Foam
 - What doesn't work
- 3 The Snake
 - Python
- 4 The Moon
 - Integration in swak4Foam
 - Limitations
 - Generalization of scripting
 - Lua
 - Examples
- 5 Conclusions
 - When to use what
 - The Future

The logo for Heinemann Fluid Dynamics Research GmbH (HFD) is displayed in a large, light gray font. The letters 'H', 'F', and 'D' are stylized and interconnected. The logo is centered on the slide and partially overlaid by a faint globe graphic.



Outline

- 1 Introduction
 - Who is this?
- 2 The Knife
 - swak4Foam
 - Advanced swak4Foam
 - What doesn't work
- 3 The Snake
 - Python
- 4 The Moon
 - Integration in swak4Foam
 - Limitations
 - Generalization of scripting
 - Lua
 - Examples
- 5 Conclusions
 - When to use what
 - The Future

HFD

Bernhard Gschaider

- Working with OPENFOAM™ since it was released
 - Still have to look up things in Doxygen
- I am **not** a core developer
 - But I don't consider myself to be an *Enthusiast*
- My involvement in the OPENFOAM™-community
 - Janitor of the `openfoamwiki.net`
 - Author of two additions for OPENFOAM™
 - `swak4foam` Toolbox to avoid the need for C++-programming
 - `PyFoam` Python-library to manipulate OPENFOAM™ cases and assist in executing them
 - In the admin-team of `foam-extend`
 - Organizing committee for the OPENFOAM™ *Workshop*
- The community-activities are not my main work but *collateral damage* from my real work at ...

Heinemann Fluid Dynamics Research GmbH

The company



- Subsidiary company of *Heinemann Oil*
 - Reservoir Engineering
 - Reservoir management

Description

- Located in Leoben, Austria
- Works on
 - Fluid simulations
 - OPENFOAM™ and Closed Source
 - Software development for CFD
 - mainly OPENFOAM™
- Industries we worked for
 - Automotive
 - Processing
 - ...

Outline

- 1 Introduction
 - Who is this?
- 2 The Knife
 - swak4Foam
 - Advanced swak4Foam
 - What doesn't work
- 3 The Snake
 - Python
- 4 The Moon
 - Integration in swak4Foam
 - Limitations
 - Generalization of scripting
 - Lua
 - Examples
- 5 Conclusions
 - When to use what
 - The Future



HFD

Outline

- 1 Introduction
 - Who is this?
- 2 The Knife
 - swak4Foam
 - Advanced swak4Foam
 - What doesn't work
- 3 The Snake
 - Python
- 4 The Moon
 - Integration in swak4Foam
 - Limitations
 - Generalization of scripting
 - Lua
 - Examples
- 5 Conclusions
 - When to use what
 - The Future

The logo for Heinemann Fluid Dynamics Research GmbH (HFD) is displayed in a large, light gray font. The letters 'H', 'F', and 'D' are stylized and spaced out. Behind the logo is a faint, light gray globe with latitude and longitude lines, centered behind the 'HFD' text.

What is swak4Foam

From <http://openfoamwiki.net/index.php/Contrib/swak4Foam>

swak4Foam stands for SWiss Army Knife for Foam. Like that knife it rarely is the best tool for any given task, but sometimes it is more convenient to get it out of your pocket than going to the tool-shed to get the chain-saw.

- It is the result of the merge of
 - funkySetFields
 - groovyBC
 - simpleFunctionObjects

and has grown since

- The goal of swak4Foam is to make the use of C++ unnecessary
 - Even for complex boundary conditions etc

The core of swak4Foam

- At its heart swak4Foam is a collection of parsers (subroutines that read a string and interpret it) for expressions on OpenFOAM-types
 - fields
 - boundary fields
 - other (faceSet, cellZone etc)
- ... and a bunch of utilities, function-objects and boundary conditions that are built on it
- swak4foam tries to reduce the need for throwaway C++ programs for case setup and postprocessing

Boundary conditions with swak4Foam

A round hot spot moves in circles

Variation of 0.org/T of the hotRoom-tutorial

```

floor
{
    type            groovyBC;
    value           uniform 300;
    variables (
        "center=vector(5,0,5);"
        "radiusFire=0.75;"
        "radiusCircle=1.5;"
        "radiant=2*pi*time()/3600;"
        "middle=center+radiusCircle*vector(sin(radiant),0,cos(radiant));"
        "tHigh=600;"
        "tLow=300;"
    );
    valueExpression "mag(pos()-middle)<radiusFire?tHigh:tLow";
}

```

Building from smaller blocks

- Most of swak4foam are small, dynamically loadable parts
 - function objects
 - boundary conditions
 - fvOptions
- Each of them is quite limited in what it can do
- But they can pass information to each other
 - Through fields
 - Global variables
 - other things
- By using that quite complex applications can be built
 - It is a bit like programming

Outline

- 1 Introduction
 - Who is this?
- 2 The Knife
 - swak4Foam
 - **Advanced swak4Foam**
 - What doesn't work
- 3 The Snake
 - Python
- 4 The Moon
 - Integration in swak4Foam
 - Limitations
 - Generalization of scripting
 - Lua
 - Examples
- 5 Conclusions
 - When to use what
 - The Future

HFD

Global and stored variables

- swak4Foam allows passing data between different parts
 - boundary conditions
 - function objects
 - fvOptions
- This happens through a mechanism called *global variables*
 - "normal" OpenFOAM-fields stored in separate namespaces
 - usually set in special function objects
 - can be accessed like normal variables
- *Stored variables* keep their values between time-steps
 - For instance: store the maximum temperature over the whole simulation
- Combination of these allows complex logics
 - "Measure the temperature at a probe location. If it rises above a limit open the second inlet and remember that decision"

State machines

- Complex logic might need a number of stored and global variables
 - Hard to read because it might be distributed across multiple function objects
- For that *State machines* were added to swak4Foam
 - A mathematical abstraction that can only have one of a finite number of states at a time
 - There are clear rules for the transition between these states



HFD

Outline

- 1 Introduction
 - Who is this?
- 2 The Knife
 - swak4Foam
 - Advanced swak4Foam
 - What doesn't work
- 3 The Snake
 - Python
- 4 The Moon
 - Integration in swak4Foam
 - Limitations
 - Generalization of scripting
 - Lua
 - Examples
- 5 Conclusions
 - When to use what
 - The Future

Limitations of "pure" swak4Foam

- The "natural" swak-expression acts on a whole field
 - Or a single value
 - Everything else is clumsy
- No complex data structures possible
- Can't interact with the outside world
 - Reading and writing files
 - Signaling to other programs
- Lagrangian particles can only be read
 - Would be nice to add "logic" to them ("Did I already cross this surface?")
- Things that can not be expressed by a single expression
 - Particle injectors

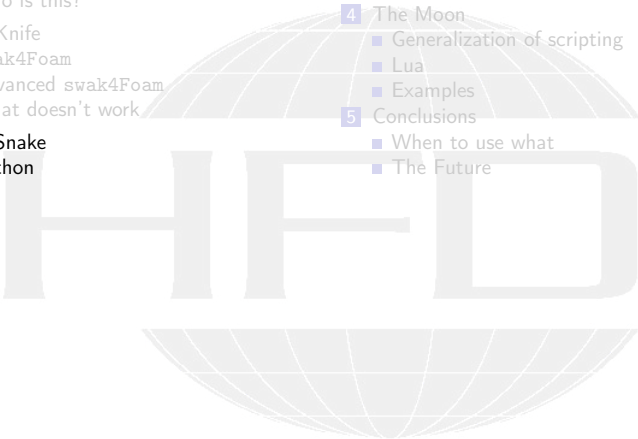
Outline

- 1 Introduction
 - Who is this?
- 2 The Knife
 - swak4Foam
 - Advanced swak4Foam
 - What doesn't work
- 3 The Snake
 - Python
- 4 The Moon
 - Integration in swak4Foam
 - Limitations
 - Generalization of scripting
 - Lua
 - Examples
- 5 Conclusions
 - When to use what
 - The Future

HFD

Outline

- 1 Introduction
 - Who is this?
- 2 The Knife
 - swak4Foam
 - Advanced swak4Foam
 - What doesn't work
- 3 The Snake
 - Python
- 4 The Moon
 - Integration in swak4Foam
 - Limitations
 - Generalization of scripting
 - Lua
 - Examples
- 5 Conclusions
 - When to use what
 - The Future



Enter the snake

Python



is

- An object-oriented scripting language with a simple syntax
- Used as scripting language in a number of software packages
 - Paraview
 - Salome (Preprocessor)

Why Python is widely used in Science and Engineering

- Comes with a number of libraries included
 - Text processing, Internet stuff, etc
- A number of libraries for scientific/numeric purposes exist
 - `numpy` and `matplotlib` give functionality equivalent to MATLAB
 - `scipy` adds numerical methods like ODE-integration and optimization
 - `pandas` adds a layer for data analysis
- The top-level code is easy to read
 - "Executable pseudo-code"
- It has a C-API that allows integrating it into other applications
 - For instance Paraview

Outline

- 1 Introduction
 - Who is this?
- 2 The Knife
 - swak4Foam
 - Advanced swak4Foam
 - What doesn't work
- 3 The Snake
 - Python
- 4 The Moon
 - Generalization of scripting
 - Lua
 - Examples
- 5 Conclusions
 - When to use what
 - The Future
- Integration in swak4Foam
 - Limitations

HFD

Using Python in swak4Foam

- Python-Integration has been a part of swak4Foam for some time
 - Had to be specially configured to compile
- Python integration means
 - A complete Python-interpreter is loaded at the start
 - swak4Foam-data is loaded into the interpreter
 - A python-script is executed
 - Data from Python is used in swak

Function objects that use Python

`pythonIntegration` Executes a Python-script

- can do almost anything

`executeIfPython` Executes other function objects if a Python expression evaluates to True

`writeIfPython` Write data if a Python-expression is True


`writeAndEndIfPython` Stops the run if a Python-expression is True

`setDeltaTWithPython` Python expression returns a value that is used as the new time-step-size

`setEndTimeWithPython` Sets another end-time according to a Python-expression

Also: `dynamicFunctionObjectListProxy` allows generating a list of function objects from a Python-script

Outline

- 
- 1 Introduction
 - Who is this?
 - 2 The Knife
 - swak4Foam
 - Advanced swak4Foam
 - What doesn't work
 - 3 The Snake
 - Python
 - 4 The Moon
 - Integration in swak4Foam
 - Limitations
 - Generalization of scripting
 - Lua
 - Examples
 - 5 Conclusions
 - When to use what
 - The Future

Resource limitations

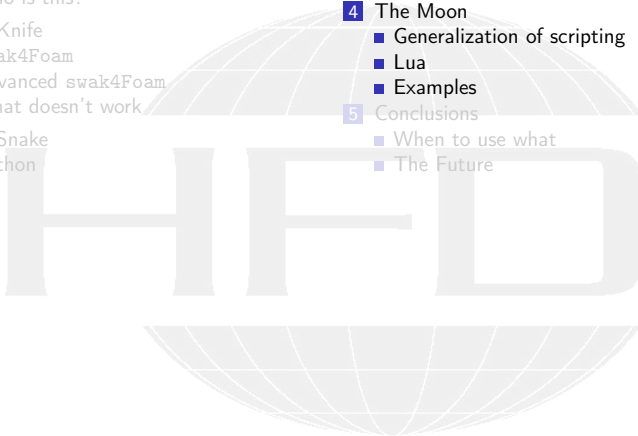
- Python is slower than C++
 - All interpreted languages are
 - `numpy` is almost as fast as equivalent OpenFOAM-operations
- Startup time
 - Starting the Python-interpreter takes some time
 - Switching to it also uses time
- Memory overhead
 - Python needs some memory

Technical problems

- Python is installed differently in different Linux-distros
 - Windows is even worse
- Adds a number of dependencies to the compilation process
- Run-time dependencies might break otherwise running binaries
 - Example:
 - swak4Foam was compiled on a CentOS 7 workstation with Python 2.7
 - Does not work on a cluster with CentOS 6 because that has *only* Python 2.6
- Also: Python 3 is coming into the mainstream
 - API is incompatible with Python 2

Outline

- 1 Introduction
 - Who is this?
- 2 The Knife
 - swak4Foam
 - Advanced swak4Foam
 - What doesn't work
- 3 The Snake
 - Python
- 4 The Moon
 - Integration in swak4Foam
 - Limitations
 - Generalization of scripting
 - Lua
 - Examples
- 5 Conclusions
 - When to use what
 - The Future



Outline

- 1 Introduction
 - Who is this?
 - 2 The Knife
 - swak4Foam
 - Advanced swak4Foam
 - What doesn't work
 - 3 The Snake
 - Python
 - 4 The Moon
 - Generalization of scripting
 - Lua
 - Examples
 - 5 Conclusions
 - When to use what
 - The Future
- Integration in swak4Foam
 - Limitations

HFD

Writing a second scripting interface?

- Plans to move to from Python 2 to Python 3
 - This would have made it hard to compile on older machines
 - Would have broken old scripts
- Alternative: separate Python 3 Integration
 - Would lead to a lot of duplicated code
 - Hard to maintain both in parallel
- Idea: Move all common functionality to a `generalPythonIntegration`-library
 - Each Python-integration implements its special stuff separately
- Flash: why stop at Python
 - One `generalLanguageIntegration`-library to rule them all

What the generalLanguageIntegration does

- Provides a common framework for scripting language integrations
 - Loading global variables
 - Returning results
 - Finding source code
- Makes sure that the "experience" is similar for all scripting languages
 - For instance loading current time into a variable `runTime` in the interpreter
- It tells the interpreter "I've prepared your workspace. Now run the script and return a number"

The dictionary paradox

- One problem with generalizing:
 - The OpenFOAM-library does not know how to interpret dictionaries
 - The application code does!
- Example:
 - The file says

```
a ( 1 2 3 );
```

- The OpenFOAM-library sees it as a series of Tokens: a , (, 1 , 2 , 3 ,) , ;
- Only the application determines whether this is
 - A vector
 - A List<label>
 - or something else
- To allow exchanging dictionaries with the scripting language
 - A general parser had to be written

Outline

- 1 Introduction
 - Who is this?
- 2 The Knife
 - swak4Foam
 - Advanced swak4Foam
 - What doesn't work
- 3 The Snake
 - Python
- 4 The Moon
 - Integration in swak4Foam
 - Limitations
 - Generalization of scripting
 - Lua
 - Examples
- 5 Conclusions
 - When to use what
 - The Future



HFD

Enter "the moon"

Lua

- a lightweight, multi-paradigm programming language designed primarily for embedded systems and clients.



is

- The name is the Portuguese word for *moon*
- written in plain C: very portable
- Has a small library
 - String operations, file handling, math ...
- No object-orientation but facilities to add OO
 - More like JavaScript (prototype based)
- Dynamically typed

More about Lua

- Was developed in 1993 in Brazil
 - To be used as a configuration and "glue" language
 - Embedding was a design target
- Very small footprint
 - Source-Archive needs less than 350 KB
 - Compiled binary **with** the standard-library needs 200-350 KB (on my machine)
- Because of its small footprint it is quite popular as an embedded language
 - Very popular in the gaming industry (for instance *World of Warcraft* uses it)
 - But there is also a variation of \LaTeX with embedded Lua: `luatex`
- But: the simplicity of the language means that sometimes you have to write more code to achieve something

Examples of Lua

- Lua-syntax looks a little bit like simplified Pascal
- Arrays and "structs" are the same data structure
 - Even stranger: array indices start with 1

Adding two vectors

```
a={1,2,3,4,5}
b={5,4,3,2,1}
c={}
for i = 1,#a,1 do
  c[i]=a[i]+b[i]
end
print(table.concat(c,"_"))
```

Prints 6 6 6 6 6

Vector magnitude

```
function mag(x)
  return math.sqrt(x.x*x.x+x.y*x.y+x.z*x.z)
end
pos={x=2,y=3,z=4}
mag(pos)
```

Prints 5.3851648071345

How Lua is integrated into swak4Foam

- The small size makes it possible for swak4Foam to compile its "private" version of Lua
 - Does not have to rely on the one installed on the system
 - Is there any?
 - Configuration: where is it?
 - Is it the right version?
 - Downloading the sources and compilation is done by one script
- In addition to the two Pythons there now is a library `libswakLuaIntegration.so`
 - Brings all the usual scriptable function objects in a Lua flavour

Stopping if time-step is too small

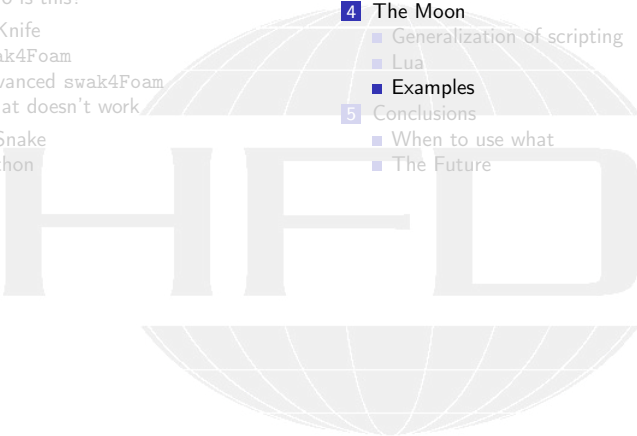
```
endTwoSmall {
    type writeAndEndLua;
    parallelMasterOnly false;
    isParallelized true;
    initCode "prev=100; tiny=1e-6;";
    conditionCode "old=prev; prev=deltaT; return prev < tiny and old < tiny;";
    interactiveAfterException true;
}
```

Scripting particles

- Work in progress is a library to allow scripts with Lagrangian particles
 - injectors
 - cloud function objects
- Generalized for run-time selectable scripting languages
 - library `libswakScriptableLagrangian.so` is needed
 - and at least one concrete scripting language
- Information is exchanged between scripting language and `swak4Foam` with dictionaries
 - That's what the dictionary parser is needed for
- The injectors are finished
 - works by specifying scripts for different informations

Outline

- 1 Introduction
 - Who is this?
- 2 The Knife
 - swak4Foam
 - Advanced swak4Foam
 - What doesn't work
- 3 The Snake
 - Python
- 4 The Moon
 - Integration in swak4Foam
 - Limitations
 - Generalization of scripting
 - Lua
 - Examples
- 5 Conclusions
 - When to use what
 - The Future



Injecting particles in a complex pattern

- This is a modification of the `angledDuct-tutorial`
 - Lagrangian particles were added with a function object
- The injector is used to show the pattern of the flow
 - Particles are injected from a line
 - The line rotates
 - Particles are injected for 20 time-steps and then it pauses for another 20
 - Particles are only injected half of the time

Used like every other injector

```
InjectionModel SwakScriptableInjection;  
  
SwakScriptableInjectionCoeffs {  
    ...  
}
```

Administrative stuff

- Which scripting language to use
- What are the names of the exchange data structures

SwakScriptableInjectionCoeffs

```
languageWrapperType lua;  
interactiveAfterException true;  
  
injectByEvent true;  
  
parameterStructName goingIn;  
resultStructName comingOut;
```

Switching injection on and off

- Same Lua-interpreter for the injector. Scripts can exchange information by setting variables
- init-script is executed only in the beginning
 - All others once per time-step

SwakScriptableInjectionCoeffs

Code could be more readable if we used separate files with doStartInjectionFile instead of doStartInjectionCode

```

initInjectorCode "injectionSteps=0;␣switchInterval=20;";
doStartInjectionCode "print('Waiting␣'.. injectionSteps);injectionSteps=injectionSteps+1;␣<brk>
<cont>injectionDuration=0;␣return␣injectionSteps>switchInterval;";
doStopInjectionCode "print('Injecting␣'.. injectionDuration);injectionSteps=0;␣<brk>
<cont>injectionDuration=injectionDuration+1;␣return␣injectionDuration>switchInterval;\" <brk>
<cont>;
  
```

Setting Particle properties

- prepareParcelDataCode is called once per time-step
- particlePropertiesCode once for every injected particle

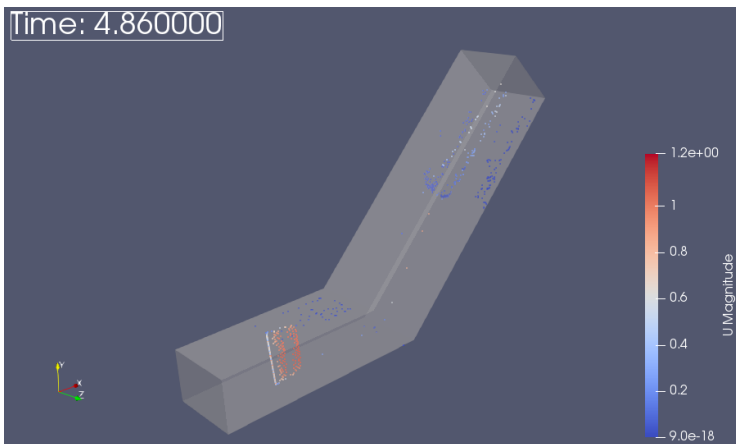
SwakScriptableInjectionCoeffs

```

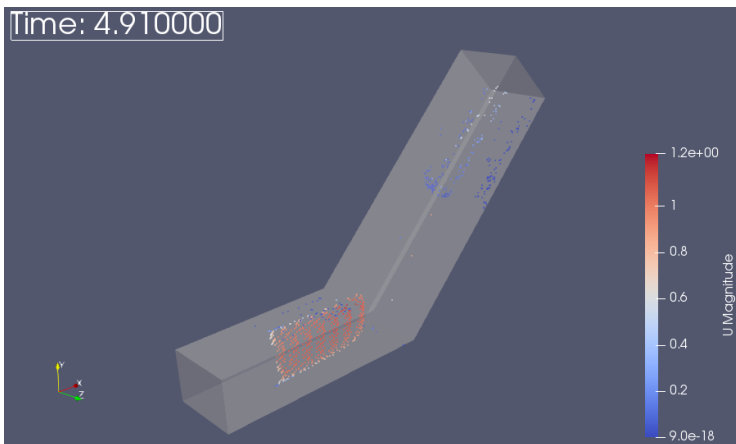
parcelsToInjectCode "if(goingIn.time0%0.1)<0.05,return,100,else,return,0,end";
volumeToInjectCode "return,0.000000001";
prepareParcelDataCode "zVals={};n=goingIn.newParcels;for(i=1,n,1)do,zVals[i<br>
<cont>]=-0.025+0.05*(i-0.5)/n,end";
particlePropertiesCode "n=goingIn.nParcels;i=goingIn.parcelI+1;comingOut={position<br>
<cont>={-0.1,0.02+zVals[i]*math.sin(runTime),math.cos(runTime)*zVals[i]},diameter<br>
<cont>=0.001e-3,U0={0,0,0}}";

```

One injection starts



Particles evolve



Injection depends on the solution

- The infamous drivenCavity tutorial is modified
 - Every time-step one particle is injected from the center of the swirl
- Finding the center:
 - Function object calculates the *stream function*
 - Stream function has its minimum at the center
 - Location of the center is put into a global variable
 - Variable is passed to the injector

Getting the center

In the functions in the controlDict:

Calculating the stream function

Requires the
libswakVelocityPluginFunctions.so

```
streamFunction {
    type expressionField;
    fieldName streamF;
    expression "interpolateToCell(<brk>
        <cont>streamFunction(phi))";
    autowrite true;
    outputControl timeStep;
    outputInterval 1;
}
```

Putting it into a global variable

```
minLocation {
    type calculateGlobalVariables;
    outputControl timeStep;
    outputInterval 1;
    valueType internalField;
    toGlobalNamespace injectorData;
    toGlobalVariables (
        centerPoint
    );
    variables (
        "centerPoint=minPosition(<brk>
            <cont>streamF);"
    );
    noReset true;
}
```


Administrative

In SwakScriptableInjectionCoeffs

Make sure centerPoint is known to Lua

```
swakToLuaNamespaces (
    injectorData
);
```

Inject only one particle

```
initInjectorCode "";
doStartInjectionCode "return_␣runtime>0.05;";
doStopInjectionCode "return_␣false;";

parcelsToInjectCode "return_␣1;";
volumeToInjectCode "return_␣0.000000001";
prepareParcelDataCode "";
particlePropertiesFile "$FOAM_CASE/particle.lua"
```

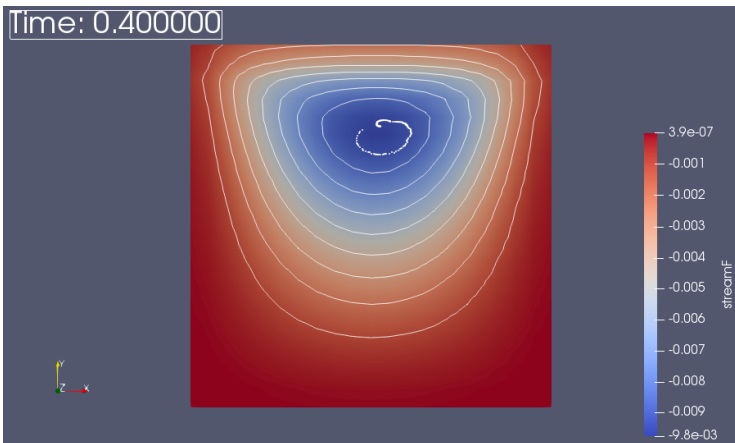
Setting the particle position

Adding a little bit of randomness

particle.lua

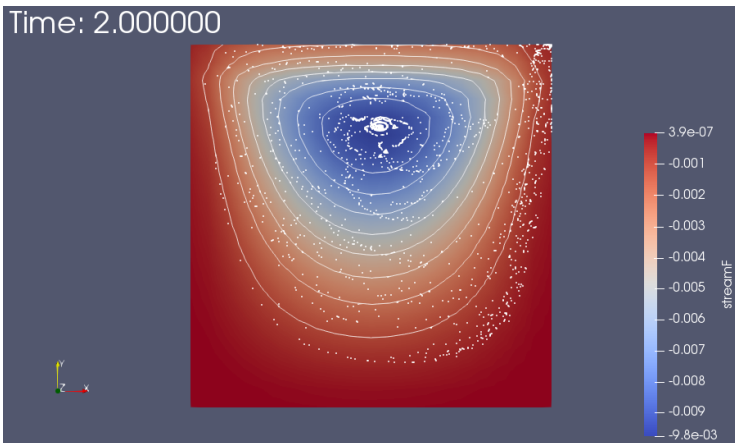
```
l=0.1e-3;
offX=l*(math.random()-0.5);
offY=l*(math.random()-0.5);
comingOut={
  position={
    centerPoint.x+offX,
    centerPoint.y+offY,
    0.005
  },
  diameter=0.1e-3,
  U0={0,0,0}
}
```

Particles from the center of the vortex



Particles transported away from the center

Time: 2.000000



Outline

- 1 Introduction
 - Who is this?
- 2 The Knife
 - swak4Foam
 - Advanced swak4Foam
 - What doesn't work
- 3 The Snake
 - Python
- 4 The Moon
 - Integration in swak4Foam
 - Limitations
 - Generalization of scripting
 - Lua
 - Examples
- 5 Conclusions
 - When to use what
 - The Future



Outline

- 1 Introduction
 - Who is this?
- 2 The Knife
 - swak4Foam
 - Advanced swak4Foam
 - What doesn't work
- 3 The Snake
 - Python
- 4 The Moon
 - Integration in swak4Foam
 - Limitations
 - Generalization of scripting
 - Lua
 - Examples
- 5 Conclusions
 - When to use what
 - The Future

Choice is confusing

- "But doesn't scripting make global variables obsolete?"
- "Shouldn't I now do everything with Lua?"

swak4Foam can make your life easier **if** you choose the right tool

- And you don't have to use C++

When to choose what

- If you evaluate the same expression in every cell/face stay with "regular" swak-expressions
 - 90% of the time this is the case
- For simple exchanges of information use global variables
 - Avoid them if there is logic involved
- If your system switches between discrete states use state-machines
- If everything else fails use scripting integration
 - Use Python if you want to work with big arrays or need some of the libraries
 - Use Lua otherwise or if you want to be completely independent
- If you have to interact with the internals of OpenFOAM and need the speed: use C++

Outline

- 1 Introduction
 - Who is this?
- 2 The Knife
 - swak4Foam
 - Advanced swak4Foam
 - What doesn't work
- 3 The Snake
 - Python
- 4 The Moon
 - Integration in swak4Foam
 - Limitations
 - Generalization of scripting
 - Lua
 - Examples
- 5 Conclusions
 - When to use what
 - The Future

HFD

Still some stuff missing

Some things are currently in development

- Scriptable cloud function objects
- Adding native support for arrays to Lua
 - Currently getting large fields from `swak4Foam` is slow (this has to do with the data structure)
 - A "native" interface has to be added
- The general dictionary parser allows parallelizing the scripting interface
 - Information is put into dictionaries and dictionaries can be sent to other CPUs

Next release

- The general scripting interface will be in the next release of swak4Foam
 - Which should come out beginning of next year
- Will include scripting integration for
 - Python 2
 - Python 3
 - Lua