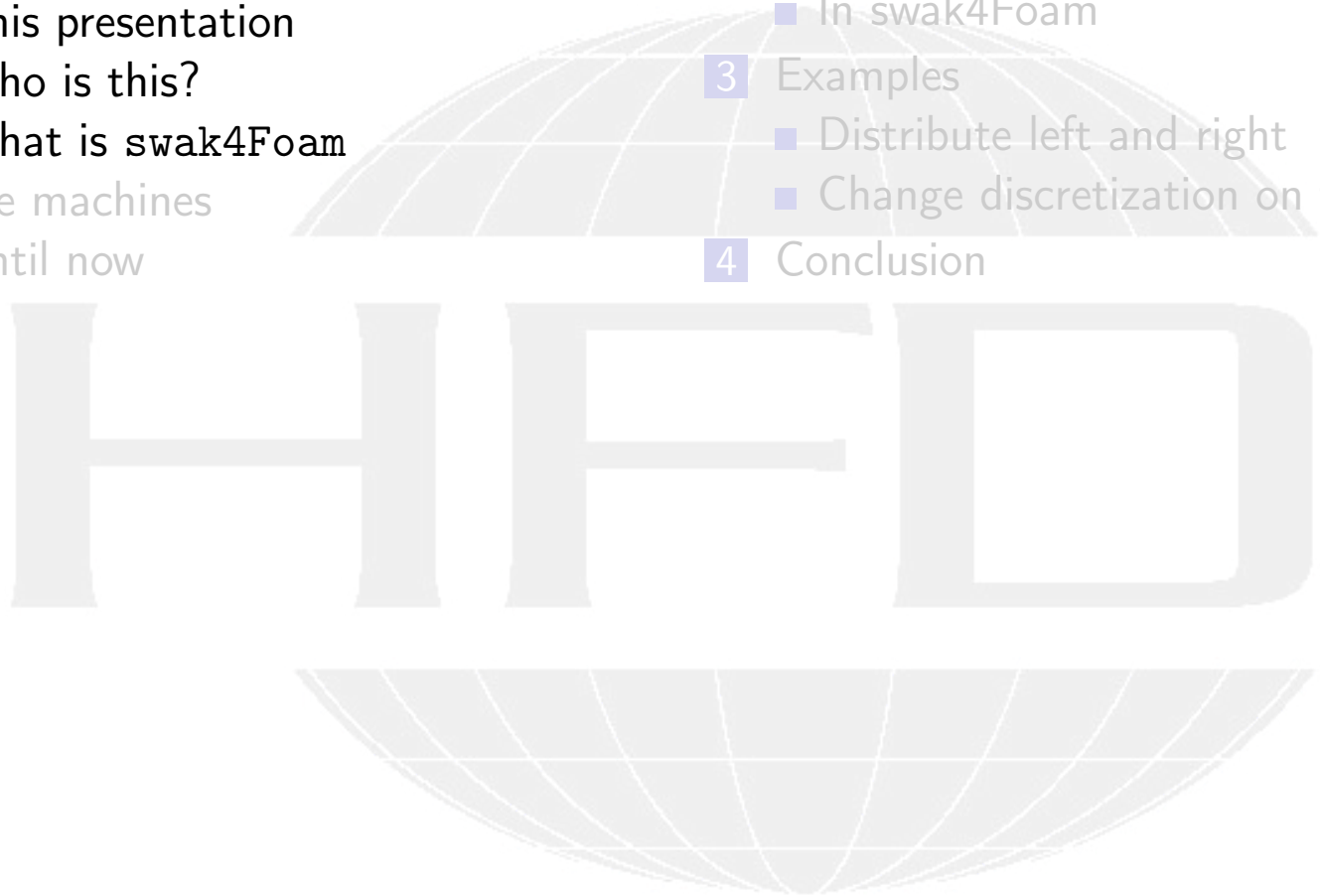
States in swak4Foam

Not: State of swak4Foam

Bernhard F.W. Gschaider

Vienna, Austria
15. November 2016

Outline



Outline

The contents

This presentation presents a new feature in the public *development repository* of `swak4foam`

- State machines
  - store discrete changes
- It also shows some other *undocumented* features:
- `funkyWarpMesh`
  - already in the release
- `groovyACMI`
  - switchable boundary conditions
  - not yet pushed to the repository
- function objects that manipulate `fvSolution` and `fvSchemes` "on the fly"

The contents

This presentation presents a new feature in the public *development repository* of `swak4foam`

- State machines
  - store discrete changes

It also shows some other *undocumented* features:

- `funkyWarpMesh`
  - already in the release
- `groovyACMI`
  - switchable boundary conditions
  - not yet pushed to the repository
- function objects that manipulate `fvSolution` and `fvSchemes` "on the fly"

The contents

This presentation presents a new feature in the public *development repository* of `swak4foam`

- State machines
  - store discrete changes

It also shows some other *undocumented* features:

- `funkyWarpMesh`
  - already in the release
- `groovyACMI`
  - switchable boundary conditions
  - not yet pushed to the repository
- function objects that manipulate `fvSolution` and `fvSchemes` "on the fly"

Outline

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-activies are not my main work but *collateral damage* from my real work at ...

Heinemann Fluid Dynamics Research GmbH

The company

- Subsidary 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

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

Outline

Outline

The problems

- Some machines need more than one boundary conditions
  - Valves open and close
  - Heaters switch on and off
- These boundaries switches may depend on the state of the simulation
  - Pressure/temperature/etc goes above/below a certain threshold
  - Time has passed since an event
  - ...
- Adding such states to a simulation requires programming
  - Special solver
  - elaborate boundary conditions
- Programming should be avoided
  - it only leads to errors

The problems

- Some machines need more than one boundary conditions
  - Valves open and close
  - Heaters switch on and off
- These boundaries switches may depend on the state of the simulation
  - Pressure/temperature/etc goes above/below a certain threshold
  - Time has passed since an event
  - ...
- Adding such states to a simulation requires programming
  - Special solver
  - elaborate boundary conditions
- Programming should be avoided
  - it only leads to errors

Solution in swak4Foam (until now)

Implementing states in swak4Foam involved

- Function objects to create *global variables*
  - Variables that could be read in other function objects and boundary conditions
- Function objects that manipulated these global variables
- Function objects that executed depending on some conditions
- Boundary conditions that read these global variables
- and/or *stored variables*
  - Variables that "remembered" their states

It was a bit of a hack

- Hard to maintain
- Hard to understand

But at least it didn't require C++

Solution in swak4Foam (until now)

Implementing states in swak4Foam involved

- Function objects to create *global variables*
  - Variables that could be read in other function objects and boundary conditions
- Function objects that manipulated these global variables
- Function objects that executed depending on some conditions
- Boundary conditions that read these global variables
- and/or *stored variables*
  - Variables that "remembered" their states

It was a bit of a hack

- Hard to maintain
- Hard to understand

But at least it didn't require C++

Solution in swak4Foam (until now)

Implementing states in swak4Foam involved

- Function objects to create *global variables*
  - Variables that could be read in other function objects and boundary conditions
- Function objects that manipulated these global variables
- Function objects that executed depending on some conditions
- Boundary conditions that read these global variables
- and/or *stored variables*
  - Variables that "remembered" their states

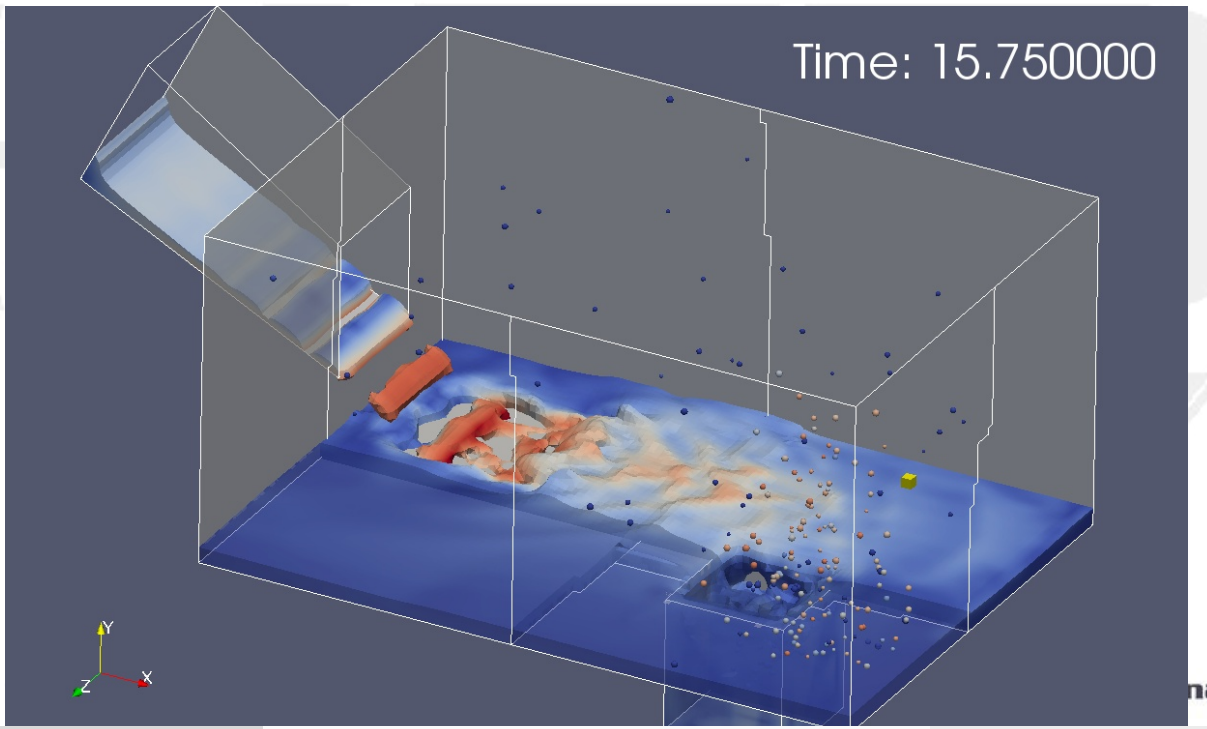
It was a bit of a hack

- Hard to maintain
- Hard to understand

But at least it didn't require C++

Example from OSCIC 2012 in London

- This example switched a number of things on and off with global variables
- In the `swak`-distribution:
  `Examples/FromPresentations/OSCFD_cleaningTank3D` (and `2D`)



Outline

Definition of State machines

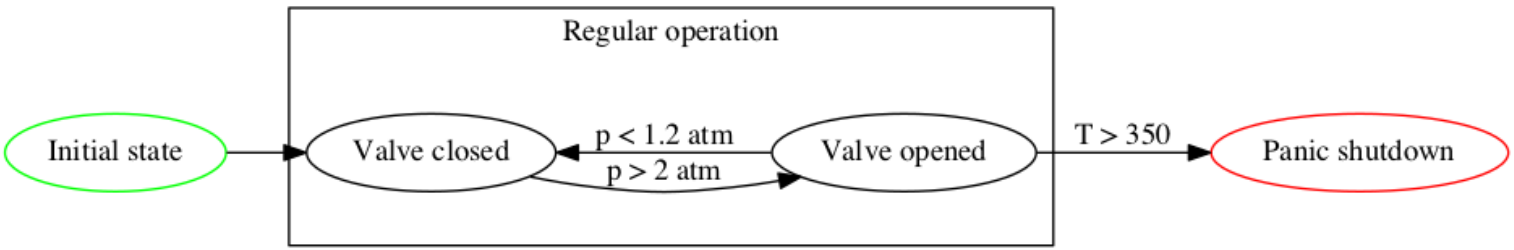
Stolen from Wikipedia:

- A finite-state machine (FSM) or finite-state automaton (FSA, plural: automata), or simply a state machine, is a mathematical model of computation used to design both computer programs and sequential logic circuits.
- It is conceived as an abstract machine that can be in one of a finite number of states.
- The machine is in only one state at a time
  - the state it is in at any given time is called the current state.
- It can change from one state to another when initiated by a triggering event or condition
  - this is called a transition.
- A particular FSM is defined by
  1. a list of its states,
  2. its initial state
  3. the triggering condition for each transition.

Example

State machine model for a valve

- 4 States: `Initial state`, `Valve opened` `Valve closed` and `Panic shutdown`
  - Represented by the circles
- Initial state is `Initial State`
- Transitions represented by the arrows
  - Condition written next to the arrow (in our case pressure thresholds trigger switches)
- `Panic dump` is a *Final State* (no transitions out of it)
  - Not necessary for a state machine



Outline

Add state machines to swak4Foam

- All things necessary are in one library
  - Names start with `stateMachine`
- Function object to create and update a State machine
- Function plugins to access them in expressions
- Other function objects to manipulate and write the state of the the State machine

controlDict

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

Specification of a state machine

The `stateMachineCreateAndUpdate` function object specifies a state machine

machineName name of the machine

states list of possible states

initialState state to start in

transitions list of dictionaries that specify transitions

- from source state (state the machine is currently in)
- condition expression with the condition that has to be true
- logicalAccumulation does `condition` have to be `true` only once (`or`) or everywhere (`and`)
- to state to move to if `condition` is `true`
- description Text to print if transition "fires"

"Driving" the state machine

- `stateMachineCreateAndUpdate` is "executed" once every timestep
  - `transitions` where `from` is the current state are checked
  - They are evaluated in the order they are in the list
    - The first one that evaluates to `true` is used
    - Transition to state `to`
    - Record time of transition
  - If no `transition` "fires" machine stays in current state
- Function object `stateMachineSetState` unconditionally moves machine to a state
  - To be used in conditional function objects (`executeIf`)
- `stateMachineMachineState` writes the current state of the machine to a file
- State of the machine is written at every output time and will be used for a restart of the simulation

Functions for state machines

These functions can be used everywhere a logical expression is acceptable

`stateMachine_isState(machine,state)` `true` if the machine named `machine` is currently in the state `state`

`stateMachine_timeSinceChange(machine)` time since the `machine` changed into the current state (to implement conditions like "How long has the valve been open")

`stateMachine_stepsSinceChange(machine)` number of time steps since the last state change of `machine`

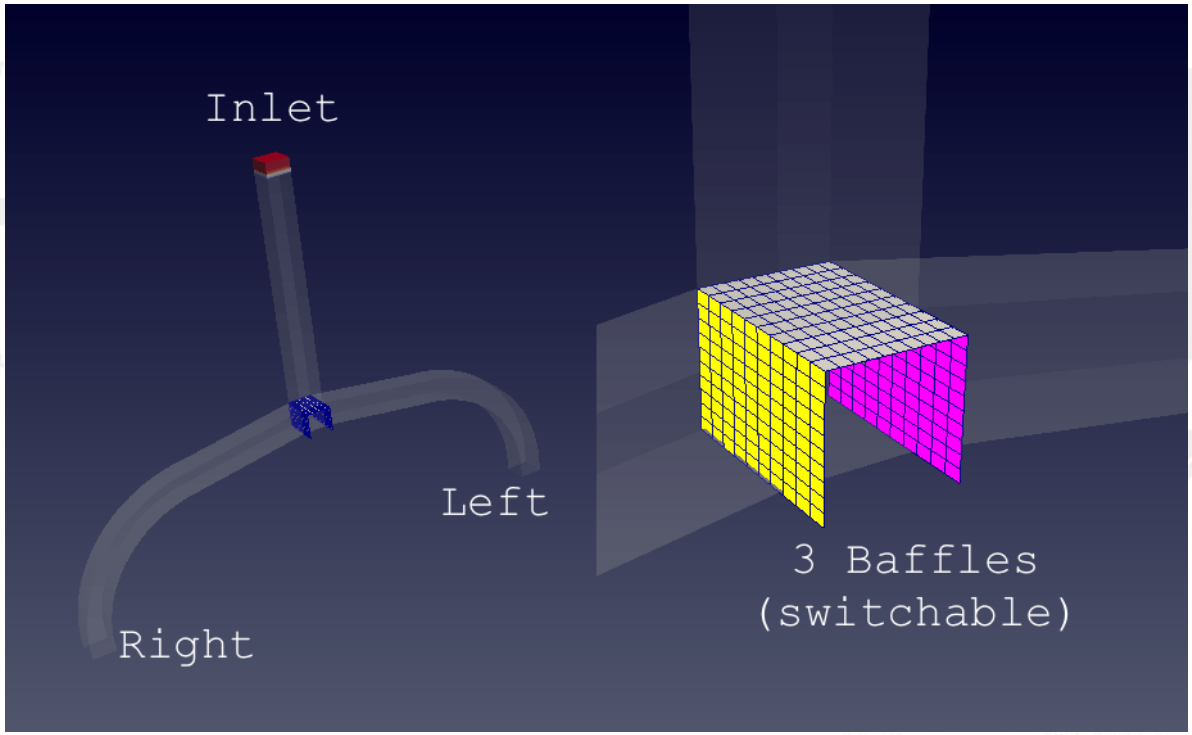
`stateMachine_changedTo(machine,state)` How many times has the `machine` changed to `state` (for conditions like "How often did the valve open")

Outline

Outline

The case

- Liquid comes in from the `inlet`
  - Controlled by state machine `theInlet`
- Is distributed to the outlets `left` and `right`
  - By opening and closing the 3 baffles according to a state machine `valves`



Bending with `funkyWarpMesh`

- The basic mesh was created with `blockMesh`
  - The "outside" pipes were bended down to make the boundary conditons simpler
- For bending the utility `funkyWarpMesh` was used
  - Moves the points according to a `swak`-expression

`funkyWarpMeshDict.outletDown`

Maybe not the best example how simple it is to use that utility

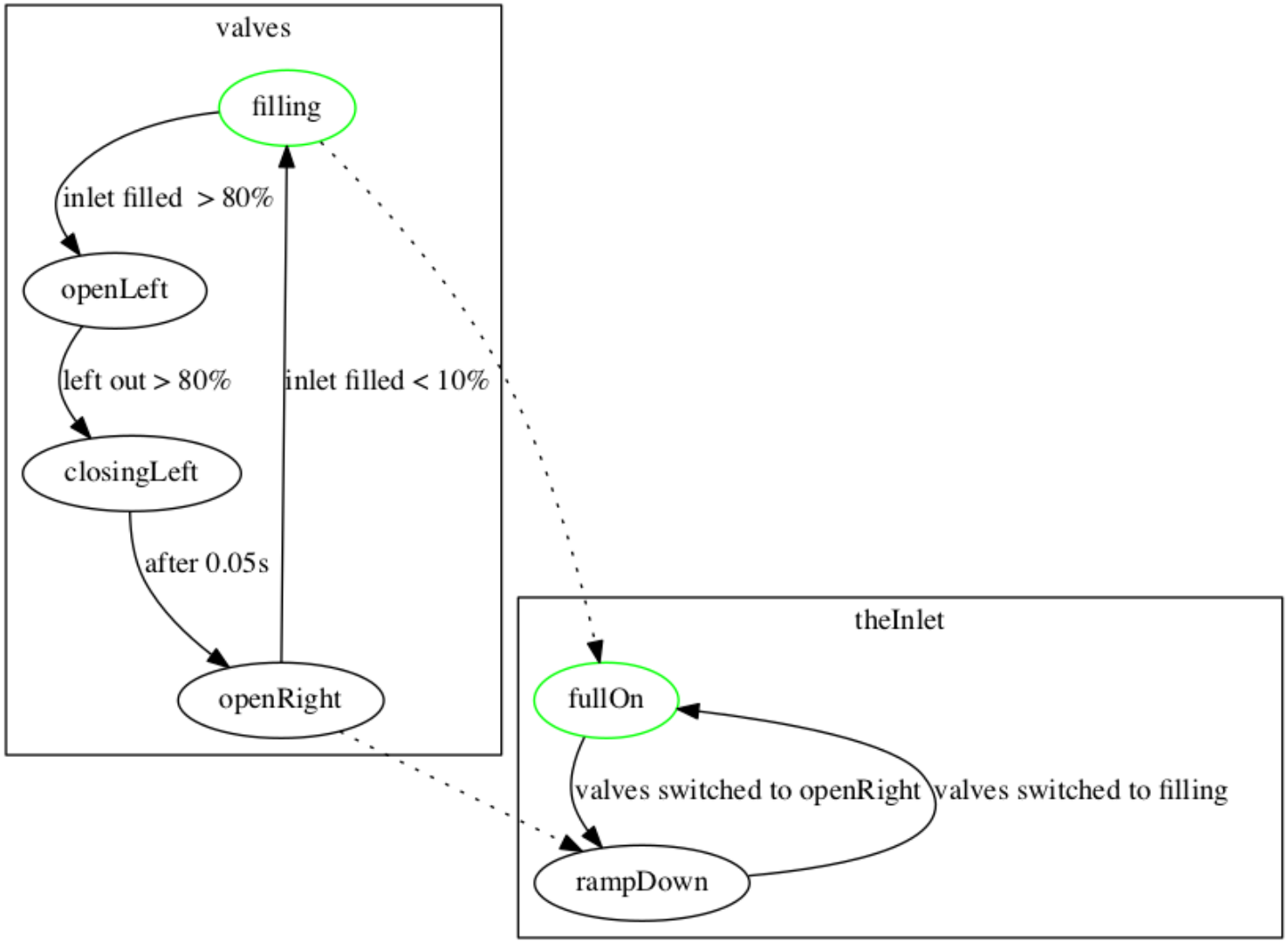
- Only "outer" parts will be bended
- Center of rotation is dynamically calculated

```
relative true;

expression "mag(pts().x)<halfX ? zV : (pts().x>zS ? vector((x0-pts().x)+(pts().y-y0)*sin(angle)<brk>
    <cont>,(y0-pts().y)+(pts().y-y0)*cos(angle),zS) : vector((-x0-pts().x)-(pts().y-y0)*sin(<brk>
    <cont>angle),(y0-pts().y)+(pts().y-y0)*cos(angle),zS))";

variables (
    "maxX=max(pts().x);"
    "halfX=maxX/interpolateToPoint(2);"
    "minY=min(mag(pts().x)<halfX ? pts().y : interpolateToPoint(1e6));"
    "angle=interpolateToPoint(pi/2)*(mag(pts().x)-halfX)/(maxX-halfX);"
    "zV=interpolateToPoint(vector(0,0,0));"
    "zS=interpolateToPoint(0);"
    "x0=halfX;"
    "y0=minY-halfX;"
);
```

What will be simulated

1. Inlet will fill up three quarters
   - Wall on the inlet side will stay half-way open to avoid pressure build-up
2. Door to the right will close. To the left will stay open. Inlet side opens fully
3. When flow on the left is 80% of the flow on the inlet the left will close
   - Wait 0.05 seconds
4. Right will open
   - During that amount of inflow decreases gradually
5. When filling of the inlet channel is below 10% restart the cycle
   - Flow on the inlet turned to full

The state machines



Opening and closing walls

But what about the opening and closing walls?

- No worries: OpenFOAM has a solution
- And `swak` tries to make it more flexible

Opening and closing walls

But what about the opening and closing walls?

- No worries: OpenFOAM has a solution
- And `swak` tries to make it more flexible

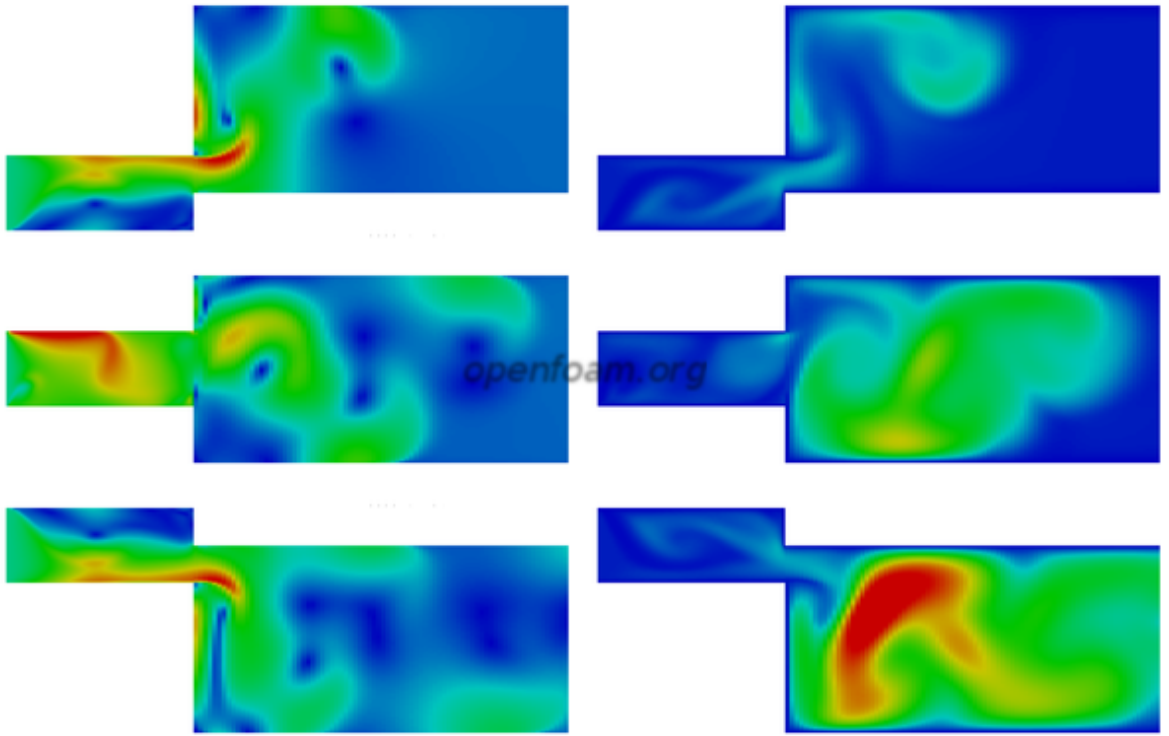
Opening and closing walls

But what about the opening and closing walls?

- No worries: OpenFOAM has a solution
- And `swak` tries to make it more flexible

The ACMI Interface of OpenFOAM

- Added in OpenFOAM 2.3
- Switches between wall and coupled according to overlap



groovyACMI in swak4Foam

- swak4Foam implements a specialization of ACMI
- Does the usual overlap calculation
- But the overlapping faces are not necessarily open
  - In the boundary specification an additional field is specified
  - Only some patches are relevant - depends on the order in the boundary-file
    - Faces on these patches with value 1 (and overlap) are open
    - All others are closed (either value 0 or no overlap)
- Values on the switches can be easily calculated with a groovyBC

Loading the switching field in controlDict

```
getTheValve {
    type readAndUpdateFields;
    fields (
        valveField
    );
}
```

Specification of the valves state machine

controlDict

```
valveStates {
    type stateMachineCreateAndUpdate;
    valueType internalField;
    states (
        filling openLeft closingLeft openRight
    );
    machineName valves;
    initialState filling;
    variables (
        "filledIn{cellZone'inletBlock}=sum(vol()*frac)/sum(vol());"
        "pressureIn{cellZone'inletBlock}=sum(vol()*p)/sum(vol());"
        "inletFlow{inlet}=-sum(phi*frac);"
        "leftFlow{left}=sum(phi*frac);"
        "rightFlow{right}=sum(phi*frac);"
    );
    transitions (
        {
            from filling; to openLeft;
            description "Fill up inlet pipe";
            condition "filledIn>0.75"; logicalAccumulation and;
        }
        {
            from openLeft; to closingLeft;
            description "More goes out than in";
            condition "leftFlow>0.8*inletFlow"; logicalAccumulation and;
        }
        {
```

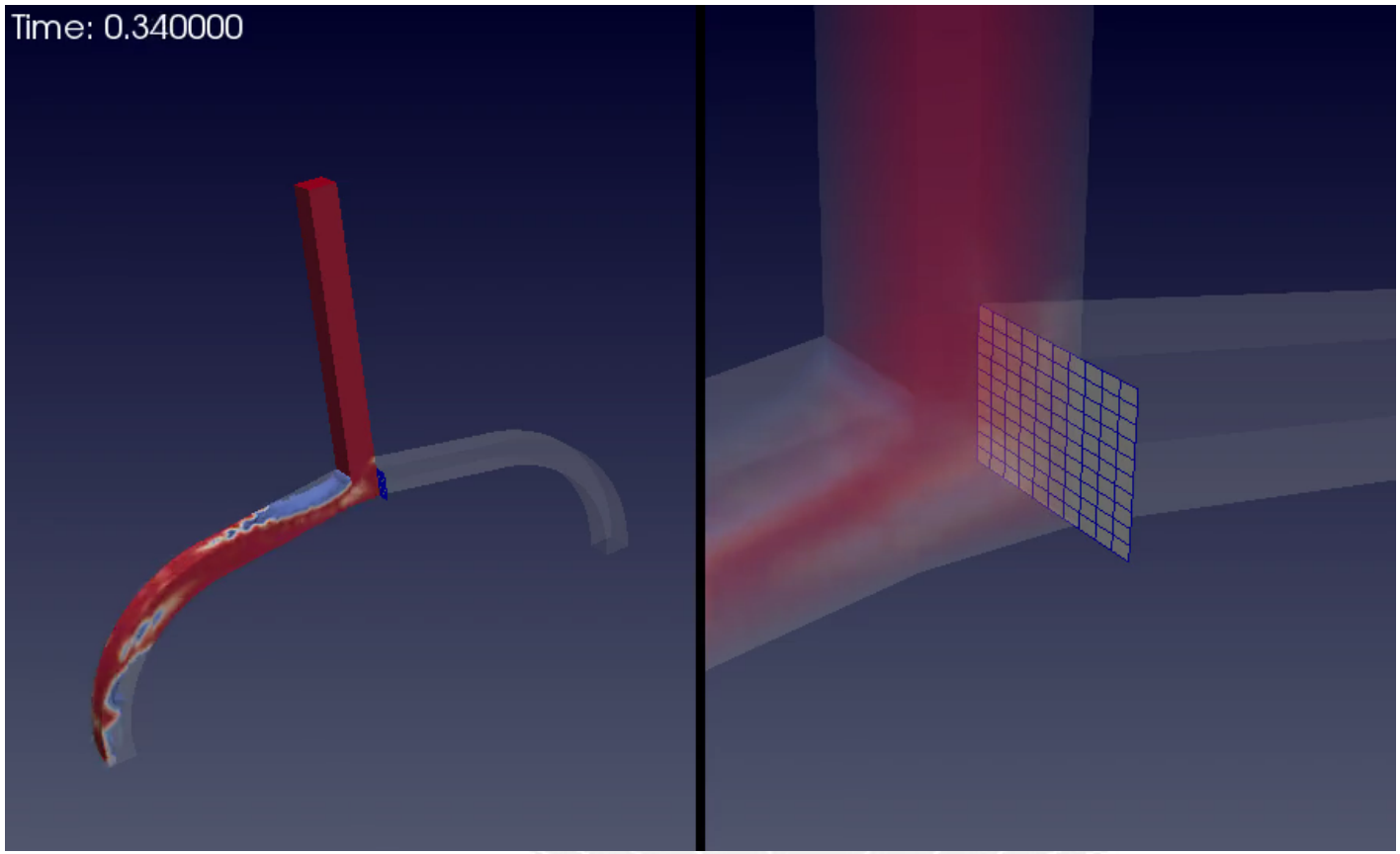
Switching a inlet

- Switch faces on the inlet on and off
  - Depending on the machine state
  - The actual position of the face

boundaryField in file valveField

```
ACMI_InletCenter_blockage
{
    type groovyBC;
    value uniform 0;
    variables (
        "minX=min(pts().x);"
        "maxX=max(pts().x);"
        "halfX=(minX+maxX)/2;"
    );
    valueExpression "(stateMachine_isState(valves,filling))␣?␣(pos().x<<brk>
        <cont>halfX␣?␣0␣:␣1)␣:␣1";
}
```

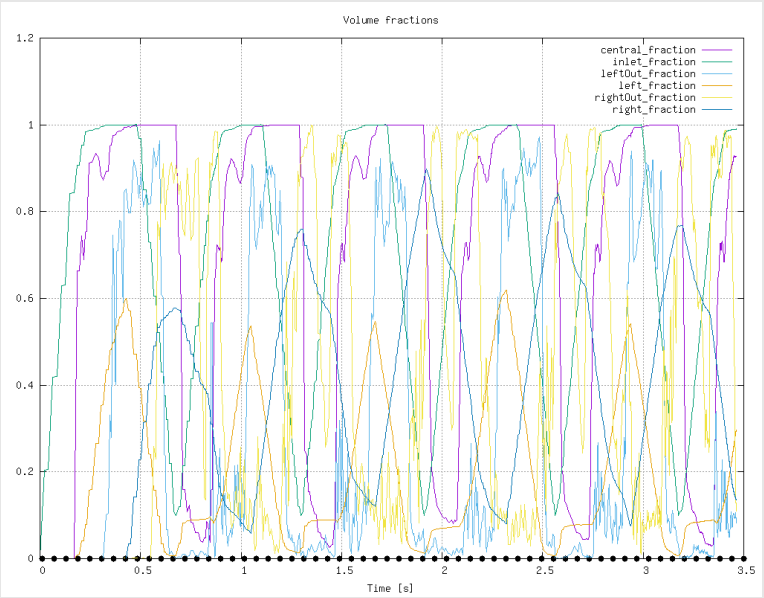
Running compressibleInterDyMFoam



States and volume fractions

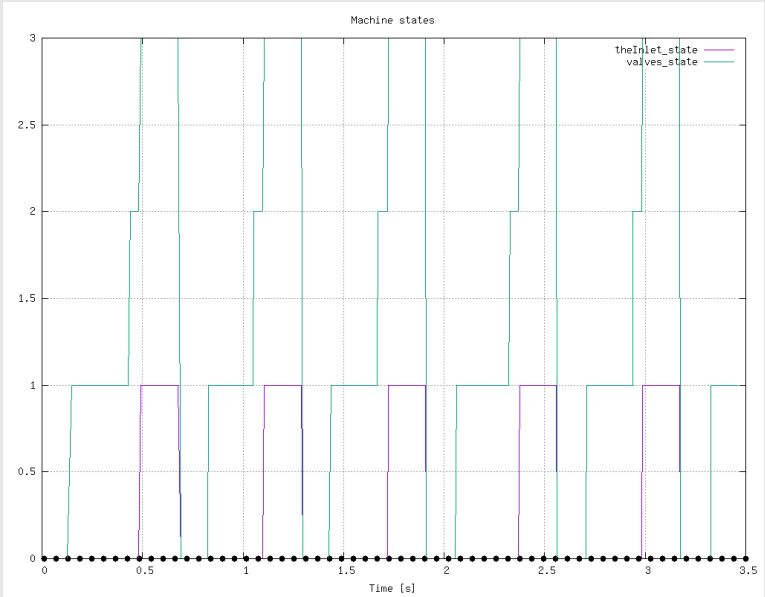
Volume fractions

- Average volume fractions
  - In the three "arms" and the central cube
  - On the outlets and inlets



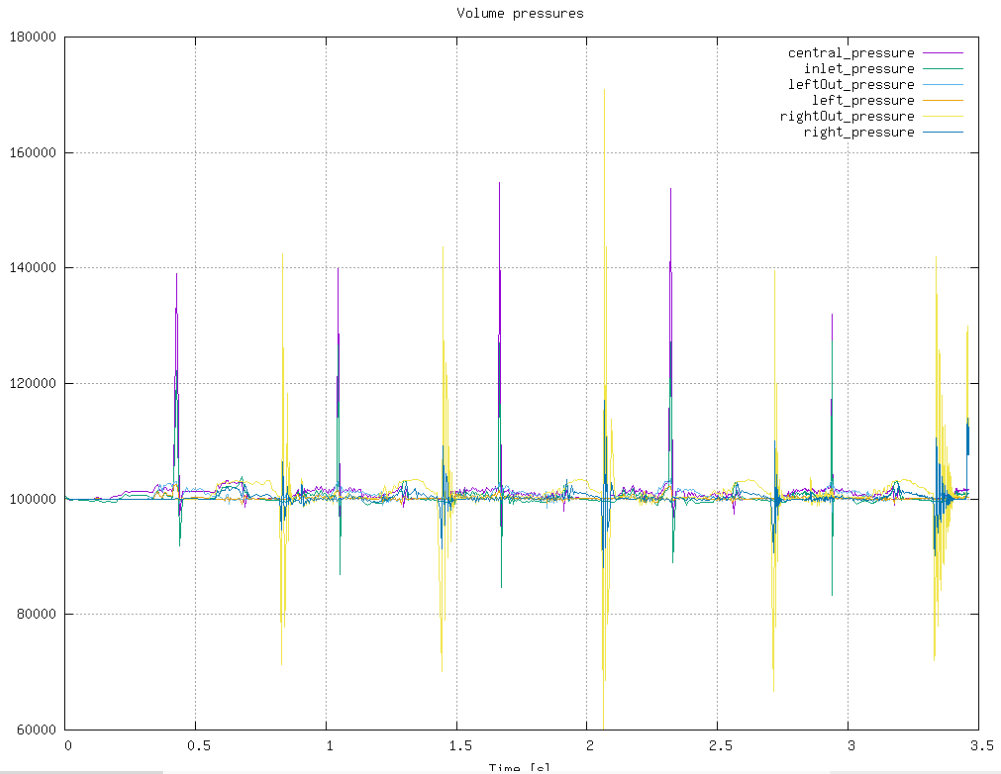
State machine states

- Every machine state is encoded by an integer value
- These are the values for our two machines



The problem with the pressure

- The immediate switching of the "valves" causes pressure spikes
  - And boundary conditions have to be switched as well
- This causes the simulation to crash after some cycles
  - Didn't bother to fix it because this is only a demonstration



Outline

Stable in the beginning. Exact in the end

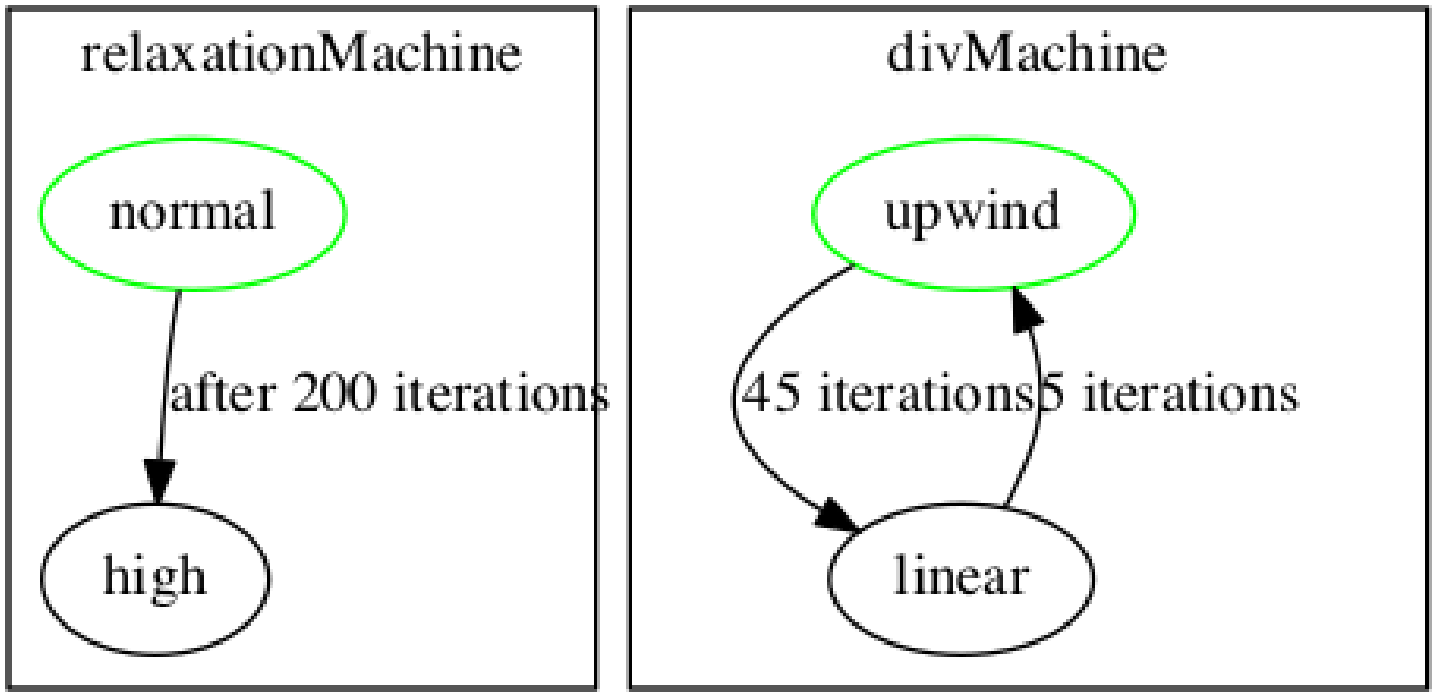
- Sometimes one wants different values in `fvSolution` or `fvSchemes` during the simulation
  - Stable schemes and low relaxation in the beginning
    - To avoid crashes due to unphysical initial conditions
  - Higher order schemes and high relaxation
    - For higher accuracy and/or faster convergence
- The way this is usually done in OpenFOAM
  1. Stop the simulation
  2. Change `fvSchemes` and `fvSolution`
  3. Restart
- . . . or edit the files "on the fly" in the text editor
  - Crashes the simulation if there is a syntax error in the edits
  - Not very "controlled"

Manipulating fvSchemes and fvSolution

- swak4Foam has function objects to manipulate theses files "in memory"
  - names end with FvSolutionFvSchemes
  - One that is controlled by state machines stateMachineFvSolutionFvSchemes
- fvSolution / fvSchemes have the usual content
  - Additional sub-dictionaries (for instance foo)
- when triggered all the content in foo is used to override the regular content
  - Things that have no corresponding content stay the same

Switching the pitzDaily-case

- This is in `Examples/manipulateFvSolutionFvSchemes/pitzDailyStateSwi`
- Switch to higher relaxation after some time
- Alternate between `upwind` and `linear` (not a good idea)



Different discretization schemes

Regular schemes in `fvSchemes`

```
divSchemes
{
    default           none;
    div(phi,U)        bounded Gauss <brk>
          <cont>upwind;
    div(phi,k)        bounded Gauss <brk>
          <cont>upwind;
    div(phi,epsilon) bounded Gauss <brk>
          <cont>upwind;
    div(phi,R)        bounded Gauss <brk>
          <cont>upwind;
    div(R)            Gauss linear;
    div(phi,nuTilda) bounded Gauss <brk>
          <cont>upwind;
    div((nuEff*dev(T(grad(U))))) <brk>
          <cont>Gauss linear;
    div((nuEff*dev2(T(grad(U))))) <brk>
          <cont>Gauss linear;
}
```

The linear schemes in the same file

These will override the `upwind` values if `linearDiv` is triggered

```
linearDiv {
    divSchemes
    {
        div(phi,U)        bounded <brk>
              <cont>Gauss linear;
        div(phi,k)        bounded <brk>
              <cont>Gauss linear;
        div(phi,epsilon) bounded <brk>
              <cont>Gauss linear;
    }
}
```

State machines to switch

The usual in controlDict

```
solutionMachine {
    type stateMachineCreateAndUpdate;
    valueType patch;
    patchName inlet;
    machineName relaxationMachine;
    states (
        normal
        high
    );
    initialState normal;
    transitions (
        {
            from normal;
            to high;
            condition "time()>200";
            logicalAccumulation and;
            description "Try higher relaxation";
        }
    );
}
schemesMachine {
    $solutionMachine;
    machineName divMachine;
    states (
        upwind
        linear
    );
```

Switching the schemes

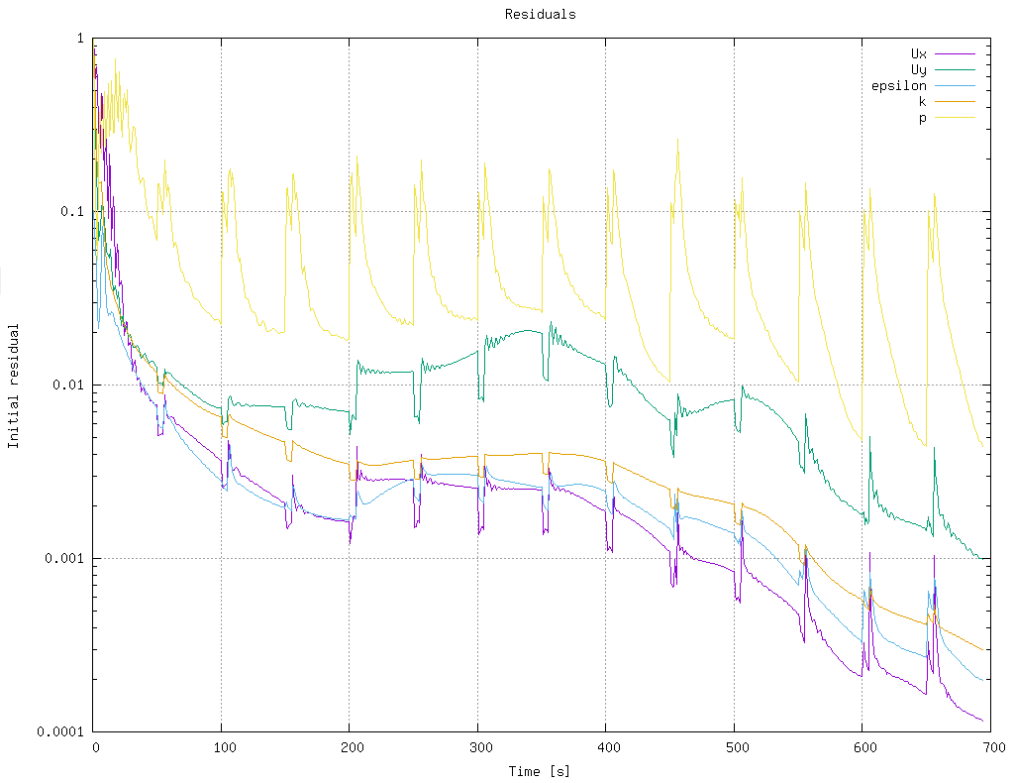
The actual switching in `controlDict`

- The `stateTo` entries describe the mapping of machine states to replacement dictionary
  - If a state has no entry the dictionary is reset to the original state

```
switchSolverSettings
{
    type stateMachineFvSolutionFvSchemes;
    outputControlMode timeStep;
    outputInterval 1;
    solutionStateMachine relaxationMachine;
    stateToSolution (
        high higherRelax
    );
    schemesStateMachine divMachine;
    stateToSchemes (
        linear linearDiv
    );
    resetBeforeTrigger true;
}
```

Residuals (not improved)

- Influence of the relaxation and the change of the schemes can be clearly seen
  - `simpleFoam` doesn't seem to like the `linear` scheme



Outline

Limited machines with unlimited possibilities

- State machines add simple control logic to `swak4Foam`
  - Machines can only have finite number of states
    - But something like gradual opening can be easily implemented with the other stuff in `swak4Foam`
- Applications:
  - Implement actual sensor and changing boundary conditions
  - Control the numerics
  - Usually people come up with applications I haven't thought of
- Advantage to "real" programming
  - No C++
  - Integrates nicely with the rest of `swak4Foam`
  - Things like proper restarts are already taken care of
  - No C++

Limited machines with unlimited possibilities

- State machines add simple control logic to `swak4Foam`
  - Machines can only have finite number of states
    - But something like gradual opening can be easily implemented with the other stuff in `swak4Foam`
- Applications:
  - Implement actual sensor and changing boundary conditions
  - Control the numerics
  - Usually people come up with applications I haven't thought of
- Advantage to "real" programming
  - No C++
  - Integrates nicely with the rest of `swak4Foam`
  - Things like proper restarts are already taken care of
  - No C++

Good bye

Thanks for listening

I'm not procrastinating. I'm delegating to future selves.
("Shit Academics Say" on Twitter: @academicsSay)
Me on Twitter:

@swakPyFoam News on `swak4Foam` and `PyFoam`

@ofwiki News about `openfoamwiki.net`

- Used to announce downtimes

@bgschaid Don't follow this (strange kind of humor and politics)

Good bye

Thanks for listening

I'm not procrastinating. I'm delegating to future selves.
("Shit Academics Say" on Twitter: @academicsSay)

Me on Twitter:

@swakPyFoam News on `swak4Foam` and `PyFoam`

@ofwiki News about `openfoamwiki.net`

- Used to announce downtimes

@bgschaid Don't follow this (strange kind of humor and politics)

Good bye

Thanks for listening

I'm not procrastinating. I'm delegating to future selves.
("Shit Academics Say" on Twitter: @academicsSay)
Me on Twitter:

@swakPyFoam News on `swak4Foam` and `PyFoam`

@ofwiki News about `openfoamwiki.net`

- Used to announce downtimes

@bgschaid Don't follow this (strange kind of humor and politics)