

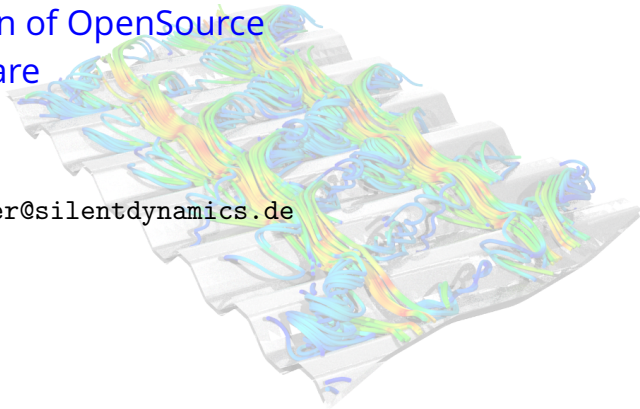
silentdynamics

Automation of OpenSource
CAE Software

Hannes Kröger

`hannes.kroeger@silentdynamics.de`

GOFUN2018



Introduction

Example: OpenFOAM Simulation

- Semi-Manual Setup

- Scripted Setup

- Add GUI

Present solutions for analysis workflow automation

- ▶ with focus on numerical simulations \Rightarrow CFD, FEM
- ▶ using open-source tools \Rightarrow OpenFOAM

Automated workflow,

- ▶ to avoid errors
- ▶ to increase productivity, speed
- ▶ make complicated analyses available to unexperienced users

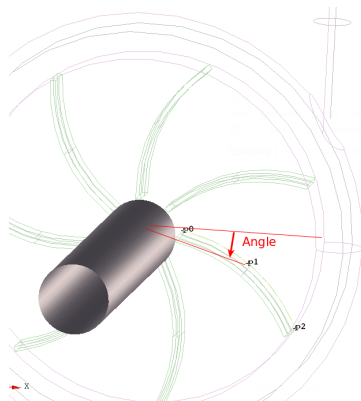
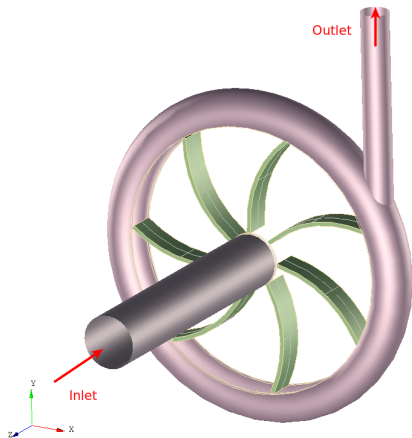
Tools for case setup and manipulation

- ▶ pyFoam
 - ▶ contains several command line tools for case manipulation
 - ▶ in python
- ▶ InsightCAE
 - ▶ also contains library of tools for case manipulation
 - ▶ in C++ with python wrappers
 - ▶ workflow GUI, report creation
 - ▶ CAD handling

We want to go through different steps of escalation

- ▶ create case manually, use supporting tools
- ▶ create a python script for the same task
- ▶ then turn into fully automated workflow with GUI

Simplified centrifugal pump with geometrical parameter



```
1 D=100;
2 d=20;
3 dd=15;
4 h=10;
5 t=1.5;
6 nb=7;
7 Lin=100;
8
9 blade_skel: SplineCurve(
10  0.5*d*EX,
11  rot(0.25*(d+D)*EX by -15*deg around EZ),
12  rot(0.5*D*EX by -30*deg around EZ)
13 );
14
15 blade1=Thicken(Extrusion(blade_skel, h*EZ), -t);
16 blades=
17  CircularPattern(blade1, 0, (360*deg/nb)*EZ, nb);
18
19 inlettube: asModel(
20  Cylinder(0+(h+t)*EZ, (Lin+h)*EZ, d)?faces('isCylinder') );
```



```

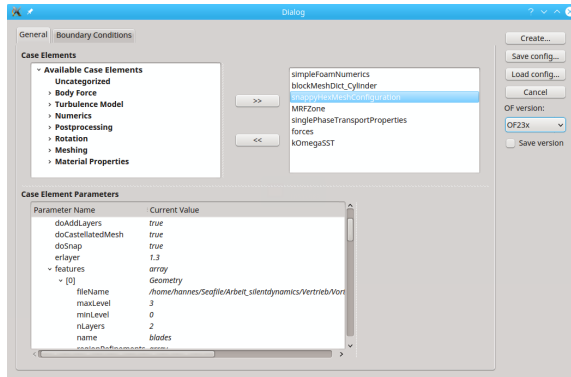
22 rotordomain=Cylinder(0, h*EZ, D);
23
24 rotor:
25 ((Cylinder(0 -t*EZ, (h+t)*EZ, D) - rotordomain) | blades)
26 - Cylinder(h*EZ, (h+t)*EZ, d);
27
28 volute1=
29 (( Cylinder(0, ax D*EY, 0.7*dd) << (0.5*D*EX + 0.5*h*EZ) )
30 | Torus( 0 +0.5*h*EZ, EZ*(D+0.6*dd), dd ))
31 - Cylinder(0, 10*h*EZ, D, centered);
32
33 volute_inlet=volute1?faces('isPlane');
34
35 volute_outlet=volute1
36 ?faces('isCylinder_&&_isCoincident(%0)',
37   rotordomain?allfaces);
38
39 volute: StitchedShell(
40   volute1?faces('!in(%0)_&&_!in(%1)',
41     volute_inlet, volute_outlet));

```

```
42 @post
43
44 saveAs("rotor.stlb") << rotor;
45 saveAs("volute.stlb") << volute;
46 saveAs("inlettube.stlb") << inlettube;
47 saveAs("rotordomain.stlb") << rotordomain;
```

Create the OpenFOAM by combining case elements with case builder

\$ isofCaseBuilder



Add:

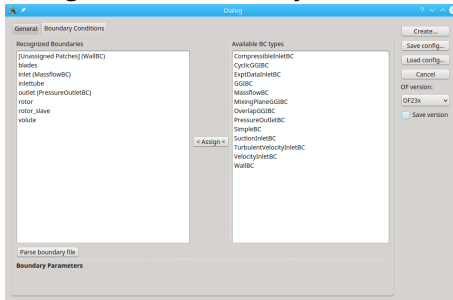
1. simpleFoamNumerics
2. blockMeshDict_Cylinder
3. snappyHexMeshConfiguration
4. MRFZone
5. singlePhaseTransportProperties
6. forces
7. kOmegaSST

Setup Parameters:

1. blockMeshDict_Cylinder: D 0.18, L 0.15, $p_0 (0 \ 0 \ -0.05)^T$, defaultPatchName "outlet", topPatchName "inlet"
2. snappyHexMeshParameters: PiM, add features:
 - 2.1 Geometry: rotor.stlb, name "blades", scale 10^{-3}
 - 2.2 Geometry: inlettube.stlb, name "inlettube", scale 10^{-3}
 - 2.3 Geometry: volute.stlb, name "volute", scale 10^{-3}
 - 2.4 Geometry: rotordomain.stlb, name "rotor", zoneName "rotor", scale 10^{-3}
3. MRFZone: name "rotor", rpm 1000
4. forces: name "forces", patches: add "blades", rhoInf 998.0

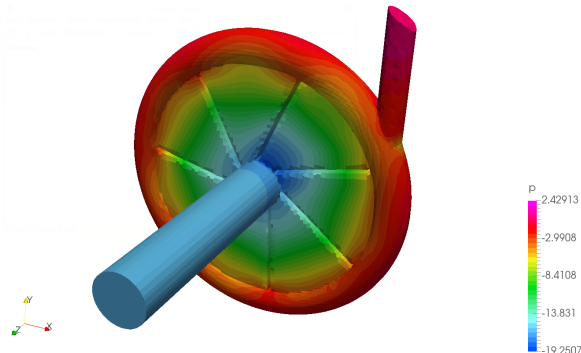
1. Save config, "case.iscb"
2. Create...
3. run blockMesh
4. run snappyHexMesh -overwrite

1. Restart Case Builder: `isofCaseBuilder case.iscb`
2. change to tab "Boundary Conditions"



3. Assign BCs:
 - ▶ inlet: "MassflowBC", massflow 0.0001
 - ▶ outlet: "PressureOutletBC"
 - ▶ [Unassigned Patches]: "WallBC"
4. Save config, Create...

- ▶ `run simpleFoam`
- ▶ then evaluate manually, e.g. `isPV.py`:



Scripted Solution

Repeat after geometry change, goal: self-contained script

- ▶ copy input files into script:

```
cat model.iscad case.iscb > run.py
```

- ▶ edit script

complete the script:

```
1 #!/usr/bin/env python
2
3 from Insight.toolkit import *
4 import subprocess, pprint, numpy, math
5
6 cadscript = """
7 D=100;
8 d=20;
9 dd=15;
10 ...
11 saveAs("rotordomain.stlb") << rotordomain;
12 """
13
14 openfoamcase = """
15 <?xml version="1.0" encoding="utf-8"?>
16 <root>
17 ...
18 </root>
19 """
```

```

20 ## Generate geometry
21 subprocess.Popen(['iscad', '-b', '-'],
22   stdin=subprocess.PIPE).communicate(input=cadscript)
23
24 ## Mesh
25 case=OpenFOAMCase(OFEs_get("OF23x"))
26 case.setFromXML(openfoamcase, workdir, True, True) #skip BCs
27 case.createOnDisk ( workdir );
28 case.modifyCaseOnDisk ( workdir )
29 case.executeCommand( workdir, "blockMesh" )
30 case.executeCommand( workdir, "snappyHexMesh",
31   ["-overwrite"] )
32
33 ## Run
34 case=OpenFOAMCase(OFEs_get("OF23x"))
35 case.setFromXML(openfoamcase, workdir, True) # incl. BCs
36 case.createOnDisk ( workdir );
37 case.modifyCaseOnDisk ( workdir )
38 case.executeCommand( workdir, "simpleFoam" )

```

```
39 ## Evaluate
40 f_vs_t=numpy.array(
41     forces_readForces(case, workdir, "forces"))
42 Qfinal=f_vs_t[-1,8]+f_vs_t[-1,12]
43
44 patch_in = numpy.array(
45     patchIntegrate(case, workdir, "p", "inlet"))
46 patch_out = numpy.array(
47     patchIntegrate(case, workdir, "p", "outlet"))
48
49 p_in=patch_in[-1,1]/patch_in[-1,2]
50 p_out=patch_out[-1,1]/patch_out[-1,2]
51 delta_p=(p_out-p_in)*rho
52 H=delta_p/rho/9.81
```

Add a GUI

Complete automation by adding GUI and Report Creation

- ▶ Insight supports Python analysis modules
go into `$HOME/.insight/share/python_modules`
- ▶ copy script to `$HOME/.insight/share/python_modules/
Centrifugal\ Pump.py`
- ▶ Do following modifications

Define Parameters

Introduction | Example: OpenFOAM Simulation

```
1 #!/usr/bin/env python
2
3 from Insight.toolkit import *
4 import subprocess, pprint, numpy, math
5 import matplotlib.pyplot as plt
6
7 def category():
8     return "Centrifugal_Pump"
9
10 def defaultParameters():
11     p=ParameterSet([
12         ("rpm", DoubleParameter(1000.0, "[rpm]_Rotation_speed")),
13         ("Q", DoubleParameter(0.0001, "[m^3/s]_Volume_flux")),
14         ("bladeAngle", DoubleParameter(15., "[deg]_Blade_sweep_angle"))
15     ])
16     return p
```

```
18 def executeAnalysis(ps, workdir):
19
20     rpm=ps.getDouble("rpm")
21     Q=ps.getDouble("Q")
22     bladeAngle=ps.getDouble("bladeAngle")
23
24     cadscript=""""
25 D=100;
26 ...
27 blade_skel:
28 SplineCurve(
29     0.5*d*EX,
30     rot(0.25*(d+D)*EX by -%g*deg around EZ),
31     rot(0.5*D*EX by -30*deg around EZ)
32 );
33 ...
34 saveAs("rotordomain.stlb") << rotordomain;
35 """"%bladeAngle
```

```

36   openfoamcase=""
37   <?xml version="1.0" encoding="utf-8"?>
38   <root>
39   ...
40       <vector name="rotationCentre" value="0 0 0"/>
41       <double name="rpm" value="%g"/>
42   </OpenFOAMCaseElement>
43   ...
44       <Patch patchName="inlet" BCtype="MassflowBC">
45           <double name="T" value="300"/>
46           <string name="UName" value="U"/>
47           <double name="gamma" value="1"/>
48           <double name="massflow" value="%g"/>
49       ...
50   </root>
51   """"%(rpm,Q)

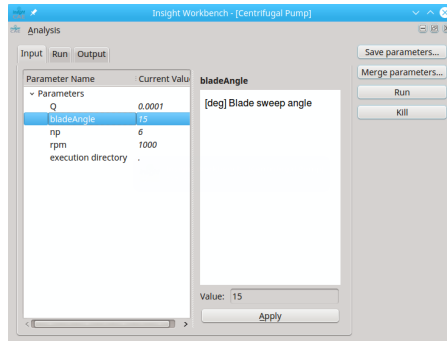
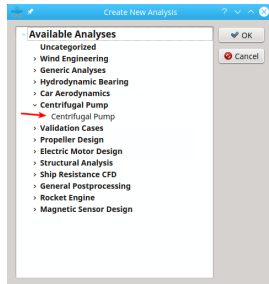
```



```

52  ## Generate geometry
53  subprocess.Popen(['iscad', '-b', '-'],
54  ...
55  H=delta_p/rho/9.81
56
57  res=ResultSet(ps, "Centrifugal_Pump", "CFD_Results")
58  res.insert("Q", ScalarResult(Qfinal, "Torque", "", "Nm"))
59  res.insert("P", ScalarResult(Qfinal*2.*math.pi*rpm/60.,
60  "Power", "", "W"))
61  res.insert("delta_p", ScalarResult(delta_p,
62  "Pressure_increase", "", "Pa"))
63  res.insert("H", ScalarResult(H, "Head", "", "m"))
64  res.insert( "forceConvergence",
65  Chart( "iter", "F",
66  [PlotCurve(f_vs_t[:,0], f_vs_t[:,3], "axialForce",
67  "w_l_t_ '$F_{ax}$'")], "", "", "" ) )
68  res.insert( "torqueConvergence",
69  Chart( "iter", "Q",
70  [PlotCurve(f_vs_t[:,0], f_vs_t[:,9], "torque",
71  "w_l_t_ '$Q$'")], "", "", "" ) )
72  return res

```



Thank you for your attention!

Dr.-Ing. Hannes Kröger

Email: hannes.kroeger@silentdynamics.de

Tel.: +49 381 36 77 98 53

<http://silentdynamics.de>

<http://sourceforge.net/projects/insightcae>