# Tutorial 11: Natural Circulation in a closed loop

# Problem Description:

A rectangular natural circulation loop made of glass is considered. The loop has an inside diameter of 26 mm and an outside diameter of 28 mm. It is filled with water at an operating pressure of 1 bar. The lengths of the horizontal and vertical legs are 1.415 m and 2.2 m, respectively. A constant power of 1 kW is added to the bottom horizontal leg. The top horizontal leg is cooled by a cooler of tube-in-tube configuration, with the inside diameter of the outer tube being 36 mm. The secondary coolant flow rate is 15 lpm (liters per minute) with an inlet temperature of 20 °C. The steady-state loop flow rate and temperature difference in the loop must be estimated. The friction factor in the pipes can be assumed as 0.05.

# Steps:

(Note the input file for this tutorial problem is available in /deck/example17.py)

Natural circulation in closed loops is to be solved through **transient route only** because the steady state solver would fail without specification of flow boundary conditions in any of the nodes in the loop.

1. Draw the layout and label the components for the problem as shown below:

Diagram

Description automatically generated

Note that a heat slab with a cooling pipe is used to model sink. The inputs for HS1 and Pipe5 are arbitrary and will not affect the steady state flow rate in the loop. However, these inputs affect the time to attain steady state solution. Hence, inputs giving higher heat transfer rates shall be preferably specified for a quick solution.

1. Create two circuits (main and cooling circuits) with the following commands:

circuit1 **=** comp**.**Circuit**(**"circuit1"**,**solveSS**=True)**

circuit1**.**assign\_fluid**(**"Water"**,**"CoolProp"**,**incomp**=True)**

circuit2 **=** comp**.**Circuit**(**"circuit2"**)**

circuit2**.**assign\_fluid**(**"Water"**,**"CoolProp"**,**incomp**=True)**

1. Add nodes to the circuit with the following commands:

node1 = circuit1.add\_node("node1",elevation=0.)

node2 = circuit1.add\_node("node2",elevation=2.2,ttemp\_old=325.)

node3 = circuit1.add\_node("node3",elevation=2.2)

node4 = circuit1.add\_node("node4",elevation=0.)

node5 = circuit2.add\_node("node5")

node6 = circuit2.add\_node("node6")

Note that a guess value is specified in a node to create an initial temperature gradient which would help flow initiation in the first iteration. Note that the guess value is to be specified in a node whose conditions are not fixed with boundary conditions which otherwise would be nullified. Also the value needs to be different from the temperature value in the fixed node for gradient generation.

1. Add pipes to the circuit with the following commands:

pipe1=circuit1.add\_pipe(identifier="pipe1",diameter=0.026,length=2.2,unode="node1",dnode="node2",fricopt=0.05,roughness=30.,ncell=5)

pipe2=circuit1.add\_pipe("pipe2",0.026,1.415,"node2","node3",0.05,30.,5)

pipe3=circuit1.add\_pipe("pipe3",0.026,2.2,"node3","node4",0.05,30.,5)

pipe4=circuit1.add\_pipe("pipe4",0.026,1.415,"node4","node1",0.05,30.,5,heat\_input=1000.)

pipe5=circuit2.add\_pipe(identifier="pipe5",diameter=0.004,length=1.415,unode="node5",dnode="node6",fricopt='DW',roughness=30.E-5,ncell=5,cfarea=0.000188)

1. Note that the pipes are incremented (say, 5 nos.) to have more accurate results since cell average densities are used to calculate the driving head for natural circulation.
2. Attach boundary conditions to the nodes

bc1 = comp.BC("bc1","node5",'P',2.E5)

bc2 = comp.BC("bc2","node5",'T',283.)

bc3 = comp.BC("bc3","node6",'msource',-0.1666667)

bc4 = comp.BC("bc4","node1",'P',70.E5,trans=False)

bc5 = comp.BC("bc5","node1",'T',330.,trans=False)

Note that the boundary conditions bc4 and bc5 are needed for steady state simulation and need to be removed in transient simulation. Hence, the ‘trans’ flag in them is made false. Note that the values fixed are arbitrary and will not affect the final steady state results. Note that the sink pipe inlet is fixed at 283 °C. Hence, the main loop temperatures will be higher than this value and closer to this if high heat transfer rates are specified in the heat slab.

1. Add heat slab

Au **=** math**.**pi**\***0.026**\***0.8

Ad **=** math**.**pi**\***0.028**\***0.8

hslab1 **=** HTcomp**.**HSlab**(**"hslab1"**,**ucomp**=**"pipe2"**,**uvar**=**"pipe"**,**uval**=[**1000.**],**dcomp**=**"pipe5"**,**dvar**=**"pipe"**,**dval**=[**1000.**],**uarea**=**Au**,**config**=**"counter"**,**solveSS**=True)**

layer1 **=** hslab1**.**add\_layer**(**thk\_elem**=**0.002**,**thk\_cros**=**1.415**,**nnodes**=**3**,**darea**=**Ad**,**mat**=**'glass'**)**

The inputs above are arbitrary and will not affect the final steady state flow rate in the loop as discussed in Step 1.

1. Add transient parameters

**import** scheduler

scheduler**.**delt **=** 1.0

scheduler**.**etime **=** 10000.

1. Modify convergence criteria

**import** solver\_settings

solver\_settings**.**temp\_solve **=** **True**

solver\_settings**.**T\_ambient **=** 283.

solver\_settings**.**conv\_crit\_temp\_SS **=** 1.E-7

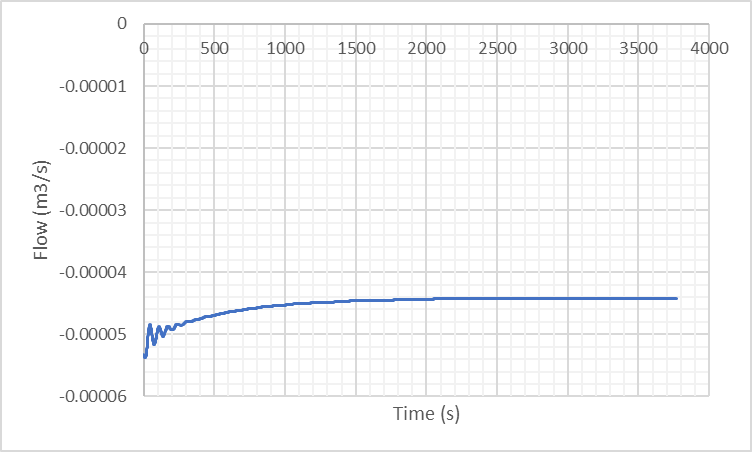
solver\_settings**.**conv\_crit\_flow **=** 1.E-7

Note that for natural circulation, since it is difficult to get convergence with default convergence criteria (residue ~10-10), higher residue values (~10-7) are used in the convergence criteria. However, the results obtained are in general accurate enough for practical purposes.

1. Run steady state simulation

Notice the energy source associated with the node with fixed temperature. Hence, the results are not correct. This energy source will be removed through transient solution to get the correct result.

1. Run transient simulation till the steady state is reached (~ 1 hr). The evolution of volumetric flow rate in a face is shown below:



1. Verify that the mass flow rate (volume flow rate x density) (in any pipe) is equal to 0.04399 kg/s from the output file. The temperature profile in the loop would look as shown below. The temperature difference between the hot and cold leg should be 5.4597 °C.

Chart, line chart

Description automatically generated