ACHP Documentation

Release 1.5

Ian Bell

Nov 14, 2017

Contents

1 ACHP Graphical User Interface **1**
- 1.1 Installation and Setup . . . 1
- 1.2 Tutorial . . . 2
- 1.3 Making Modifications . . . 6
- 1.4 Additional Functionality . . . 8

2 ACHP Cycle Model Description **9**
- 2.1 Direct Expansion Cycle Solver . . . 9
- 2.2 Secondary Loop Cycle Solver . . . 20
- 2.3 Cycle Solver Preconditioner . . . 26
- 2.4 General Numerics . . . 32
- 2.5 Frequently Asked Questions . . . 36

3 ACHP Component Models **39**
- 3.1 Compressor . . . 39
- 3.2 Fin-Tube Heat Exchangers . . . 43
- 3.3 Heat Exchangers with and without Dehumidification . . . 53
- 3.4 Condenser . . . 62
- 3.5 Cooling Coil . . . 70
- 3.6 Evaporator . . . 72
- 3.7 Multi-Circuited Evaporator . . . 75
- 3.8 Line Set . . . 80
- 3.9 Plate-Heat-Exchangers . . . 82
- 3.10 Fluid Correlations and Other Calculations . . . 93
- 3.11 Appendices . . . 104

Python Module Index **115**

CHAPTER 1

ACHP Graphical User Interface

1.1 Installation and Setup

1.1.1 Binaries for Windows Users (Easy way):

32-bit binaries including all other required packages are available from https://sourceforge.net/projects/achp/files/. Download the most recent zip file, and expand it somewhere convenient. If you have other configuration files, place them in the *configs* folder which is next to the ACHP.exe file

This is the easiest way to get started!

1.1.2 Using the source for Developers (Not for the faint of heart):

Step 1. Download a distribution of Python. For Windows users I strongly recommend the Python(x,y) distribution. Go to downloads and then pick the full install. I currently use the 2.6.6 version. When you are installing, I recommend going into the *other* section at the bottom and adding SWIG and mingw. This will allow you to recompile CoolProp if necessary. Otherwise the defaults should be fine.

For Mac OSX users, the Enthought Python Distribution (EPD) is recommended. It is free to use in 32-bit mode for academic users.

For masochists, you could download Python, and the other packages that you would need are scipy, numpy, matplotlib, and wx. It is highly recommended to use one of the above distributions.

For linux users, use your package manager to install python and the required packages.

Step 2. Download a copy of CoolProp. For Windows users, you can download an installer directly by going to CoolProp files and downloading the most recent file for your architecture. Pretty much the only architecture that is directly supported is 32bit Windows. But on other architectures, the requisite files are included.

Step 3. Download a subversion client. This allows you to interface with the subversion repository that the code is maintained in. You can use a command line subversion client (http://www.collab.net/downloads/subversion/ and download the command-line client), or use a graphical user interface. The GUI interface is a lot easier to use. I recommend TortoiseSVN. If you successfully installed CoolProp in the previous step, skip to Step 5.

Step 4. Download a copy of the CoolProp source (everyone except 32-bit Windows). If you are using a command line subversion client, type:

```
svn co http://coolprop.svn.sourceforge.net/svnroot/coolprop/trunk coolprop
```

This will pull all the CoolProp source code in the main development trunk from the servers and put it the folder coolprop. If you are using TortoiseSVN, in a Windows explorer window, right click and select *SVN Checkout...* then put https://achp.svn.sourceforge.net/svnroot/achp/trunk in the URL of repository and change the path if you like. Leave it to fully recursive and HEAD revision. Open a command prompt in the folder that the source resides and type:

```
python setup.py install
```

This will install CoolProp and put it in a location that Python will be able to find.

Step 5. Download a copy of the ACHP source code. If you are using TortoiseSVN, open an explorer window and right-click where you want the code to go. Select *SVN Checkout...*, set the URL of repository to https://achp.svn.sourceforge.net/svnroot/achp/trunk and change the path if you like. Leave it to fully recursive and HEAD revision. In a few moments you will then have the most recent set of code on your computer.

Step 6. You can now peruse the code. The components can all be found in the PyACHP subfolder and the GUI is in the GUI folder. If you want to begin development, I recommend the use of Eclipse with the plugin Pydev. The source tree includes a Pydev project file. Eclipse is included with the Python(x,y) distribution. Eclipse can be found in the pythonxy folder in the Windows start menu. Once Eclipse is opened, to get the Eclipse project opened, right click in the Pydev Package Explorer and select *Import...*. Then select *Existing Projects into Workspace*, and in the next window, click on the browse button next to the *Select root directory* option button. Browse to the folder that contains your ACHP source, and it should load up the project.

To run a given component, open the component in the Pydev Package explorer, click on the down-arrow on the button that looks like a play button in the toolbar, and go to Run as... and then select Python Run. This should run the component in stand-alone mode. If you want to run the cycle model, open Cycle.py and run it.

To start the graphical user interface, run GUI/ACHPMainFrame.py using the same Python run method as for a component.

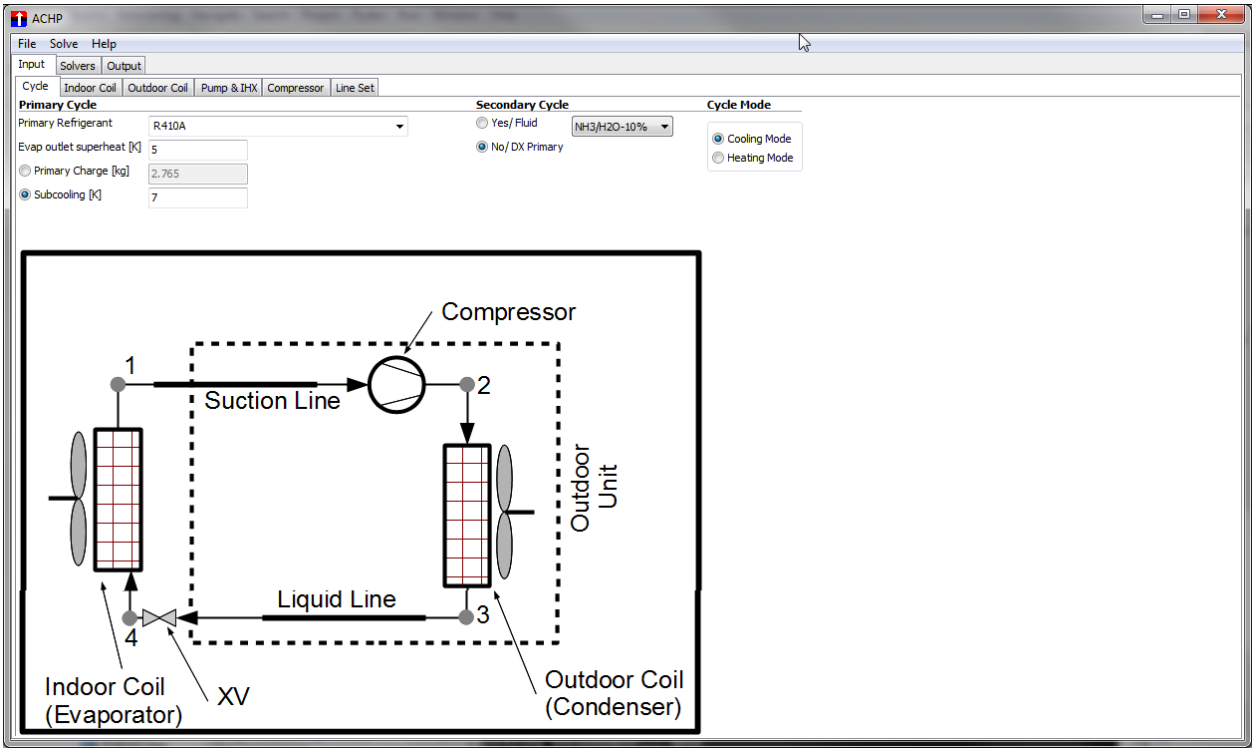
Step 7. To update your code, either right click on the source folder and select SVN Update... if you are using TortoiseSVN, or open a command prompt in the main source folder and type:

```
svn update
```

1.2 Tutorial

1.2.1 Getting Started

In order to get started using ACHP, start the ACHP.exe executable which after showing a splash screen, should boot up into a screen that looks like



This is where a number of the important parameters can be changed.

The default configuration file that ships with ACHP is loaded from the file *Default.cfg* and is a direct expansion air conditioning system that is derived from the experimental and modeling work of Bo Shen[1]. At any time, the file *Default.cfg* can be overwritten which changes the parameters that will be loaded into ACHP when it starts.

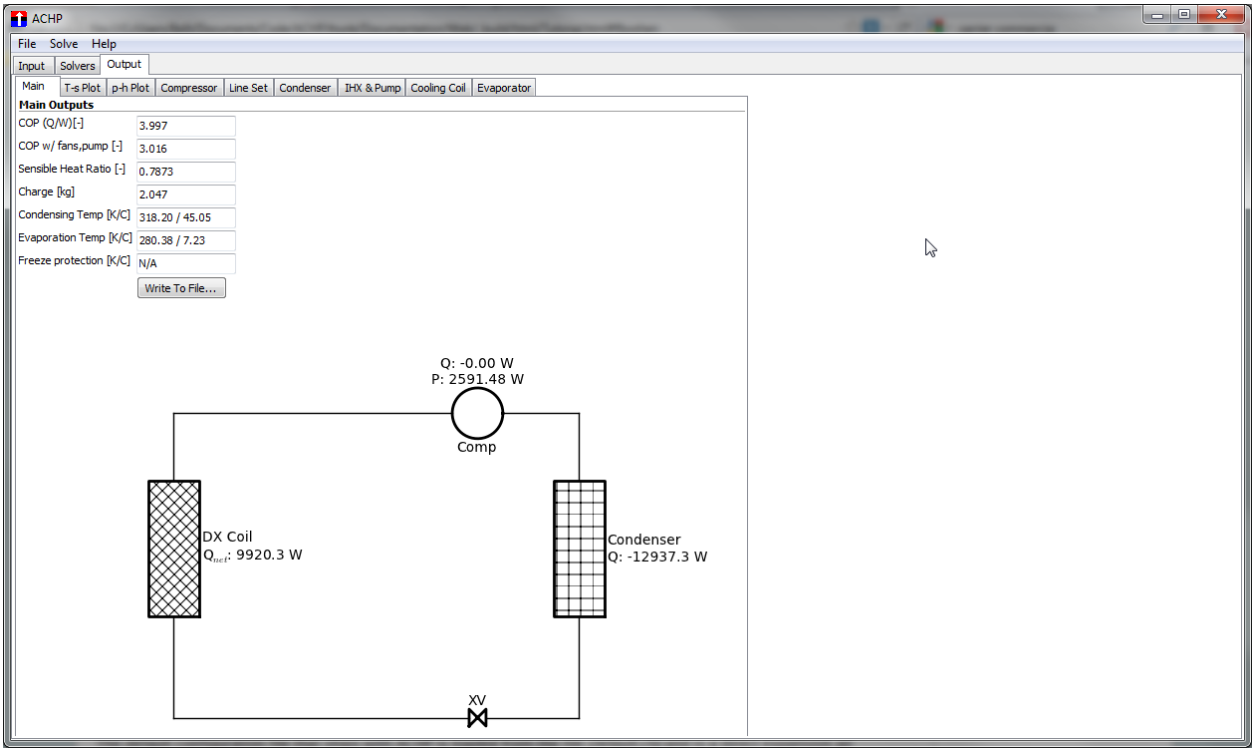
To run the model, press the F5 key, or go into the menu to Solve->Solve (F5)

Once the model has run (you can watch the console for intermediate output while it is running), all the output fields are then populated.

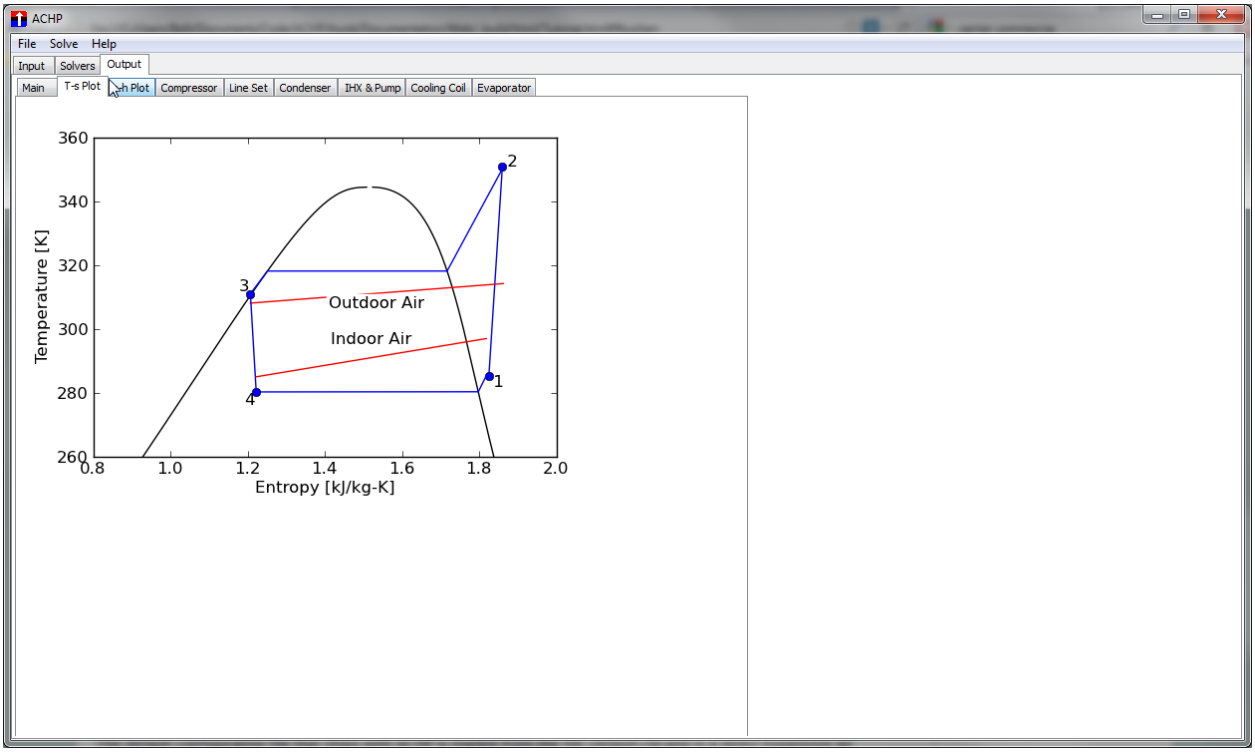
In the output are things like

Main Output Screen

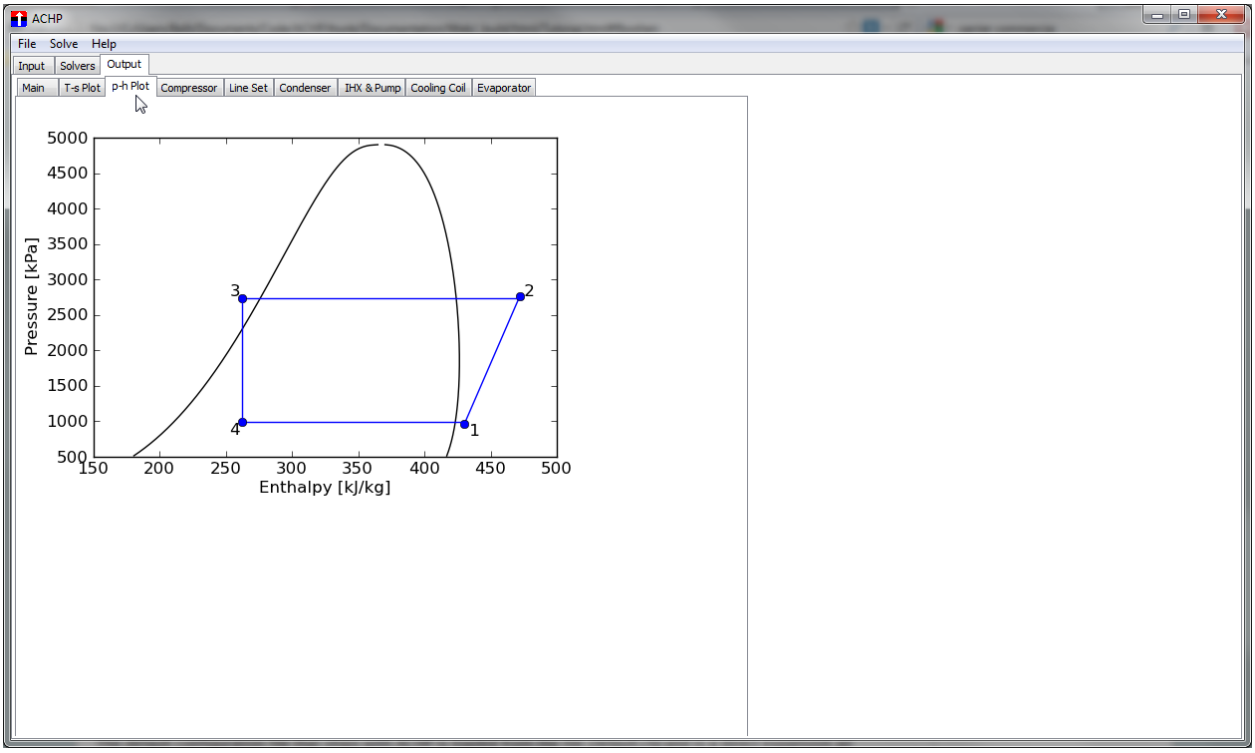
[1] Bo Shen, 2006, "Improvement and Validation of Unitary Air Conditioner and Heat Pump Simulation Models at Off-Design Conditions" Final Report 1173-RP Link to file



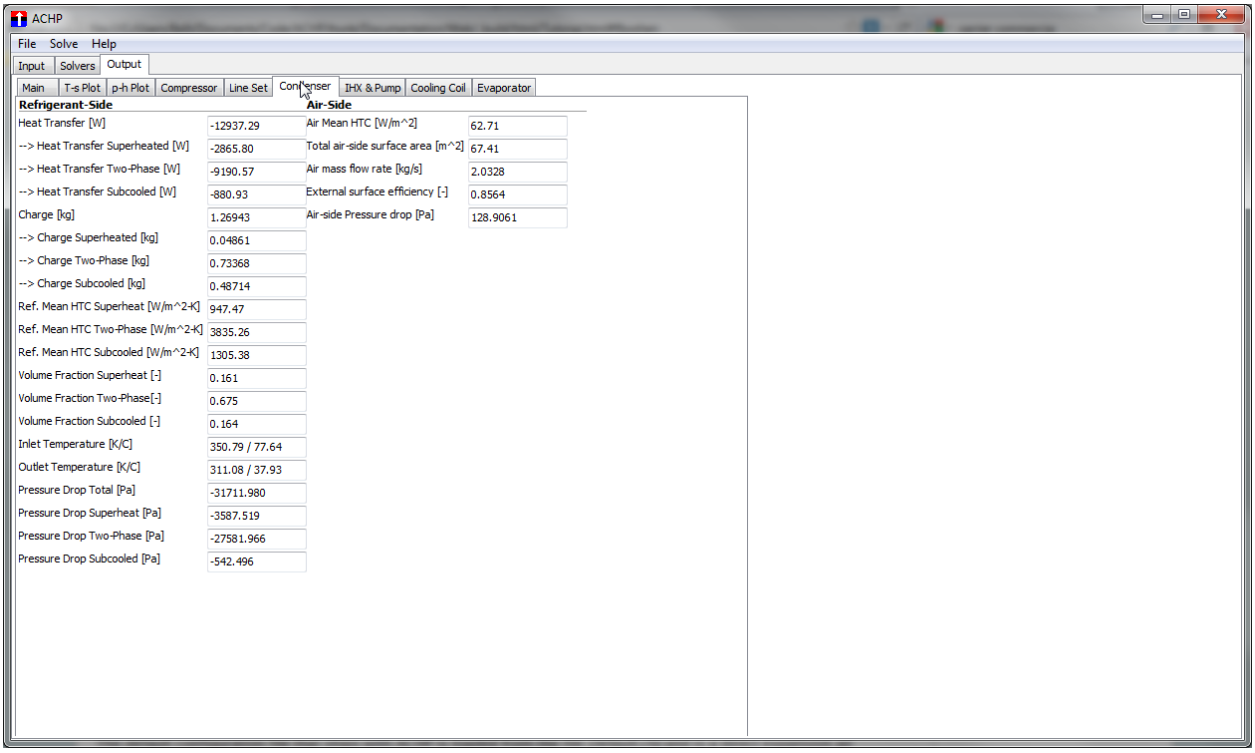
Temperature-Entropy plots



Pressure-Enthalpy plots



Output screens for each component



To save the data from a run to a file, click the **Write to File...** button on the main output screen. It will prompt you for a location to save the file. The file comprises the all the data that is output on the output screens of ACHP. It is a comma-separated file that you can open up in Excel.

1.3 Making Modifications

While it is nice to be able to see the performance of the default system, you probably want to make some changes for your own system. To do that, it is necessary to understand how ACHP is configured.

ACHP was built to be able to handle both conventional four-component air-conditioning and heat pump systems as well as secondary loop systems that use a secondary working fluid between the primary working fluid and the indoor coil. This type of system has been proposed, and is used, in large industrial scale systems.

1.3.1 Comments

In each of the input screens, if you hover your mouse over the text which describes a field, the tooltip that pops up will have further description of the field.

The user interface of ACHP is broken up into three parts - Input, Solvers, and Output. The tabs at the top of the screen allow you to move between the different parts.

1.3.2 Main Inputs

A number of different types of refrigerants can be used. The types of refrigerants that are available in ACHP are

- Working fluids that are implemented in CoolProp. A list of working fluids implmented in CoolProp can be found at the CoolProp website. These refrigerants are listed at the top of the list, before the REFPROP refrigerants
- Pure refrigerants, and defined blends from REFPROP. These fluids are listed as starting with "REFPROP-"
- Arbitrary blends of fluids from REFPROP. This functionality is achieved by creating a string which describes the blend. For instance, the fluid name "REFPROP-MIX:R32[0.697615]&R125[0.302385]" is the refrigerant R410A which is composed of a R32 and R125 mole fractions of 0.697615 and 0.302385 respectively. The syntax of the string must be followed, but can be extended if more fluids comprise the mixture. For instance, the fluid name "REFPROP-MIX:R125[0.35782]&R134a[0.038264]&R143a[0.60392]" would be the fluid name string for R404A

Since no expansion device model is employed in ACHP, the superheat is imposed as a model input. The superheat is that at the outlet of the evaporator, or in the case of the secondary loop system in cooling mode, the outlet of the internal heat exchanger.

Either the subcooling or the charge can be imposed by selecting one of the variables and providing its value, and the other one is then solved for. The convergence characteristics for subcooling imposed are slightly better than for charge imposed due to the formulation of the preconditioner. At the very least, if charge is desired to be imposed, a reasonable value for the charge should be obtained by fixing the subcooling for one point.

On the main screen of ACHP you can select whether the system is operating in cooling mode, or in heating mode, and whether a secondary loop is employed, or whether the system is a direct expansion system. As the mode and secondary loop/DX options are changed, the cycle schematic will also change.

If the cycle is a secondary loop cycle, the secondary working fluid can be selected from the dropdown box.

1.3.3 Heat Exchanger Inputs

In both secondary loop systems and direct expansion systems, there are two heat exchangers that are used to transfer heat with the indoor and outdoor air streams. The input screens for both of the heat exchangers are identical.

Currently, there are three type of fins coded - plain fin, herringbone fin and wavy lanced fin. If other fin types are needed, they must be coded into ACHP.

The inputs are divided into three subgroups - fins, tubes and air. In the fins group, you can select the fin pitch, waviness parameters of the fin, as well as the fin thickness, and fin material thermal conductivity.

In the tubes section, the geometry of the tubes and the circuits is defined. At any time, clicking on the **Show Circuits...** button will pop up a screen that will show an end-on view of the heat exchanger in order to demonstrate the construction of the currently defined heat exchanger. Clicking on the **Select** button will open up a small window that will allow you to select from standard tube dimensions.

1.3.4 Pump and Internal Heat Exchanger Inputs

In the pump and internal heat exchanger inputs pane, it is possible to set inputs for the pump and the internal heat exchanger. The pump inputs are straightforward, but the internal heat exchanger inputs need some description.

In cooling mode, either the coaxial heat exchanger or the plate heat exchanger can be used. In heating mode, the coaxial heat exchanger model has not been modified to handle condensation on the refrigerant side, so therefore it is not possible to use the coaxial heat exchanger in heating mode.

1.3.5 Compressor Inputs

The compressor model is based on a 10-coefficient compressor model as in ANSI/AHRI standard 540-2004

$$\dot{m} = M_1 + M_2 T_s + M_3 T_d + M_4 T_s^2 + M_5 T_s T_d + M_6 T_d^2 + M_7 T_s^3 + M_8 T_d T_s^2 + M_9 T_d^2 T_s + M_{10} T_d^3 \tag{1.1}$$

$$Power = P_1 + P_2 T_s + P_3 T_d + P_4 T_s^2 + P_5 T_s T_d + P_6 T_d^2 + P_7 T_s^3 + P_8 T_d T_s^2 + P_9 T_d^2 T_s + P_{10} T_d^3 \tag{1.2}$$

where T_s and T_d are refrigerant dew point temperatures in degrees Fahrenheit, the mass flow rate $\dot{m}$ is in lbm/hr and the power is in Watts. The coefficients can be read in from a comma-delimited file by clicking the **Load Coefficients** button where the first column in the file are the coefficients $M_1, M_2, ...$ and the second column are the coefficients $P_1, P_2, ...$ If the compressor coefficients are loaded from a file, it will over-write the coefficients for the compressor present in the user interface. A sample compressor coefficient file would be:

```
117.316,-461.3
5.094,-18.6
-1.593,46.9
4.48E-02,-0.21
-2.14E-02,0.43
1.04E-02,-0.44
7.90E-05,2.25E-04
-5.73E-05,2.37E-03
1.79E-04,-3.32E-03
-8.08E-05,2.50E-03
```

Information about the compressor map can be obtained by clicking on the **Info** button

If the primary refrigerant is changed, an appropriate compressor map must be employed for the given primary refrigerant.

1.3.6 Line Set Inputs

The line set allows the condensing unit and the indoor coil to be physically separated.

In the line set inputs pane you can set inputs for the line set. The supply line set goes from the outdoor unit to the indoor coil, and the return line goes from the indoor coil to the outdoor unit.

1.4 Additional Functionality

1.4.1 Using Configuration Files

If you have a configuration that you like and would like to use at a later time in ACHP, you can save a configuration file that can be loaded back into ACHP. Once you have all the parameters as you would like them, go to *File->Save Config File...* or press Ctrl+s (Apple+s for OSX users). This will pop up a file selection dialog, and you can save the file somewhere.

At a later time, you can load the configuration file back into ACHP by going to *File->Load Conf File...* or pressing Ctrl+o (Apple+o for OSX users). Again you will get a file selection dialog and you can open your configuration file.

The configuration files are simple text files that are delimited with the delimiter ":::". The configuration files list each of the items that are in the user interface and their current value (Also see note for developers below).

For Developers: When defining the names of items in the user interface, the first few letters of the object name determine whether it will be written to the configuration file or not, and whether it will be loaded back into the interface. The naming conventions from Visual Basic are used, so names starting with *opt* are option boxes, *cmb* are combo boxes, *lbl* are labels, *txt* are textboxes, *rad* are radio boxes, and *chk* are checkboxes. Refer to `LoadGUI` to see how the user interface is constructed, and the functions ReadConfigFile and WriteConfigFile in `ACHPMainFrame.py`

1.4.2 Parametric Studies

One of the features built into versions 1.3 and onwards of ACHP is the flexible multi-dimensional parametric study solver. With this solver you can run multi-dimensional parametric studies. In order to activate this mode, go to the solvers Tab and switch the *Solver Method* to Parametric Study.

In each of the rows of the parametric study, you can select a variable that you would like to vary by selecting the variable from the combobox, and turn it on by checking the checkbox. You must then provide the values that will be used for the variable. There are two ways of doing this:

- Provide the minimum value in the *Min Value/List* column, the maximum value in the *Max Value* column, and the integer number of linearly spaced steps in the *Number Steps* column. This will yield a linearly spaced set of inputs that will be used.
- Provide a comma-separated list of inputs in the *Min Value/List* column, and put the letter "L" in the *Number Steps* column. This allows you to put your own values in, particularly useful if you are trying to duplicate non-linearly-spaced rating data, which was the motivating factor for this functionality

Not all variables make sense for all configurations. For instance, with a direct expansion air conditioning system, there is no secondary loop, so altering the secondary loop mass flow rate will not change anything, or may raise an error.

For Developers: If you want to add other things to the parametric study, you can open the file *parametric/params.txt* (path relative to ACHP.exe) and add or delete entries. You must know the name of the variable you are trying to modify, so some insight into the naming conventions of ACHP is required.

CHAPTER 2

ACHP Cycle Model Description

2.1 Direct Expansion Cycle Solver

In the cycle solver, the goal is to use all the physical component models together in order to obtain the performance of the combined cycle. The process is slightly different for each of the 4 standard configurations of ACHP, but there are many common themes among the configurations. ACHP has been designed to investigate direct expansion (DX) systems as well as systems employing a secondary working loop, in both cooling and heating modes.

A number of simplifying assumptions are employed at the cycle level:

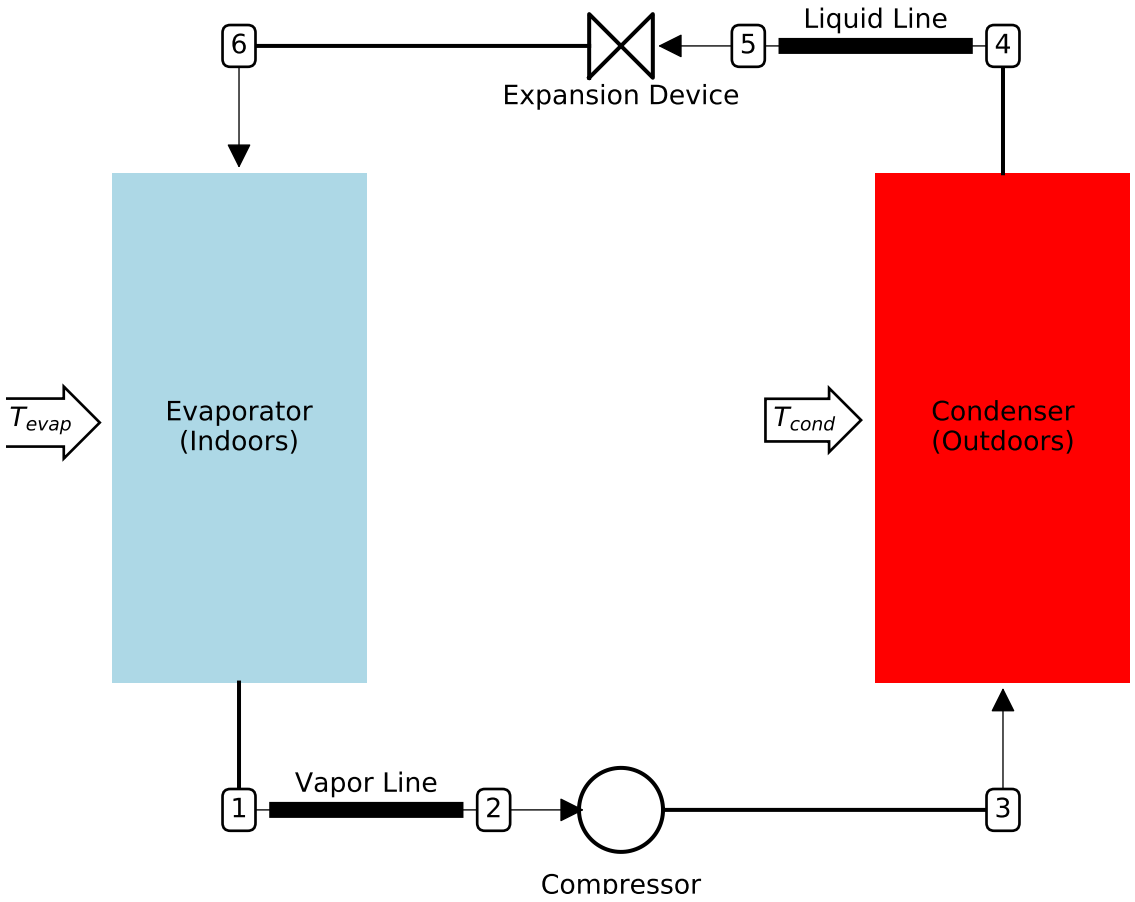
- Pressure drops in each component are calculated, but not used directly(see *Pressure-Drop Correction* for explanation)
- There is no charge inventory in the compressor shell
- The evaporator outlet refrigerant superheat is imposed (implies an expansion device with perfect superheat control)
- No pressure drops except for in the components considered

2.1.1 Direct Expansion Cooling Mode

The most common and straightforward system is the direct expansion cooling mode system. The schematic of this cycle is shown in this figure:

In addition to the geometry of each component, required inputs are the superheat at the outlet of the evaporator $\Delta T_{sh,evap}$, and either the total refrigerant charge m_r or the refrigerant subcooling at the outlet of the condenser $\Delta T_{sc,cond}$.

The primary independent variables in the cycle solver are the saturation temperatures of the refrigerant at evaporation and condensation, given by T_{evap} and T_{cond} respectively. For pure refrigerants the saturation temperature for a given pressure is the same for saturated liquid and saturated vapor. For pseudo-pure fluids and blends, the refrigerant dew temperature is used which corresponds to saturated vapor.


6
5
Liquid Line
4
Expansion Device
T_{evap}
Evaporator
(Indoors)
T_{cond}
Condenser
(Outdoors)
1
Vapor Line
2
3
Compressor

Before the full cycle-solver is run, the preconditioner in section *Cycle Solver Preconditioner* is used to get a good guess for the temperatures T_{evap} and T_{cond} using extremely simple cycle models.

Once the preconditioner has been run, preliminary values for T_{evap} and T_{cond} are known, and the first iteration of the cycle model may begin. Execution of the cycle model follows in the same direction as the refrigerant flow. The pressure drops through low-pressure and high-pressure parts of the system are assumed to be zero initially.

The cycle analysis begins with the vapor line which returns refrigerant vapor from the evaporator that is typically in the air duct back to the condensing unit outdoors. In the first iteration of the cycle solver, the mass flow rate of refrigerant in the vapor line is not known, but the compressor map is used to provide a reasonable guess value for the mass flow rate of refrigerant. The *Line Set* model is used to calculate the process from point 1 to point 2. The model is run with known flow rates and an inlet temperature of $T_{evap} + \Delta T_{sh}$ in order to calculate the state at the inlet to the compressor.

The compressor compresses refrigerant from state point 2 to state point 3. The *Compressor* model is used which is based on an empirical correlation with superheat correction and it yields the outlet state 3 as well as the compressor mass flow rate, compressor electrical power, etc.. The compressor model requires the temperatures T_{evap} and T_{cond} as well as the compressor inlet superheat.

The condenser takes the superheated refrigerant at state point 3 and condenses it to a subcooled refrigerant at state point 4. The *Condenser* model is used to calculate the process, and this condenser model is based on a moving-boundary model. The condenser model requires T_{cond} as an input, among others.

After the condenser, the subcooled refrigerant passes through the liquid-line that takes refrigerant from the condenser to the indoor coil that is inside the ductwork in the home. The *Line Set* model is used to model the flow that passes from state point 4 to state point 5.

The expansion device then expands the refrigerant from the high-pressure side of the system to the low-pressure side of the system. In the current iteration of ACHP, no expansion device is included. Thus the expansion device is just a constant-enthalpy throttling device that takes refrigerant from the high-side pressure at state point 5 to the low-side pressure at state point 6.

At the outlet of the expansion device, the refrigerant passes into the evaporator at some two-phase quality. The *Evaporator* model is used to model the performance of the evaporator. In the evaporator, refrigerant is heated by the air-stream (which cools the air stream) from state point 6 back to state point 1', which should be superheated (note: not necessarily so at intermediate iterations of cycle model) close to state point 1. If the values of the cycle independent variables T_{evap} and T_{cond} have been well selected, the state points 1 and 1' will be coincident.

Once the cycle model has been run around the loop from state point 1 to state point 1', the residuals can be calculated. The residuals are terms that when the cycle model has converged should all be equal to zero.

One of the residuals that is always active is an energy balance over the cycle. You left from state point 1, and you should hopefully arrive back there if energy is conserved in the cycle. Thus, the first residual is given by

$$\vec{\Delta}_1 = \dot{m}_r(h_1 - h_{1'}) \tag{2.1}$$

where h_1 and $h_{1'}$ are the enthalpies of the refrigerant at state points 1 and 1' respectively.

Since there are two independent variables, there must be a second constraint, which in this case is a charge-level constraint. Either the charge is constrained directly with an imposed charge level, or indirectly with an imposed refrigerant subcooling. Thus the second residual can be given by

$$\vec{\Delta}_2 = \begin{cases} m_r - m_{r,target} & \text{(Charge imposed)} \\ \Delta T_{sc,cond} - \Delta T_{sc,cond,target} & \text{(Subcooling imposed)} \end{cases} \tag{2.2}$$

and a numerical solver is used to drive the residual vector $\vec{\Delta}$ to sufficiently close to zero by altering T_{evap} and T_{cond} (see *Multi-dimensional solver*).

Pressure-Drop Correction

After the cycle model has iterated to convergence, the pressure drops are then considered. In reality, the pressure drop in each component results in a lower pressure at the inlet of the next component in the refrigerant loop. In terms of modeling, coupling pressure drop and the component models causes great numerical difficulties. The compromise that is employed instead is to run all the component models without pressure drop, but calculate the high-side pressure drop as the pressure drop of the condenser and liquid-line:

$$\Delta p_{high} = \Delta p_{cond} + \Delta p_{liquid-line} \tag{2.3}$$

and similarly, the low-side pressure drop is defined by

$$\Delta p_{low} = \Delta p_{evap} + \Delta p_{vapor-line} \tag{2.4}$$

These pressure drops are then employed to shift the saturation temperatures used in the compressor map in order to yield less refrigerant mass flow rate, and a higher compressor power.

The new effective compressor suction and discharge pressures are

$$\begin{aligned} p^*_{evap} &= p(T_{evap}) - \Delta p_{low} \\ p^*_{cond} &= p(T_{cond}) + \Delta p_{high} \end{aligned} \tag{2.5}$$

and the new, effective compressor dew temperatures are

$$\begin{aligned} T^*_{evap} &= T(p = p_{evap}, x = 1) \\ T^*_{cond} &= T(p = p_{cond}, x = 1) \end{aligned} \tag{2.6}$$

This calculated pressure drop is used until the model reaches convergence again, at which point the pressure drop tems are updated, and the model is run again. This process continues until the imposed low- and high-side pressure drops are equal to the pressure drop terms calculated from the converged cycle model.

Cycle Performance Parameters

The common metrics of system efficiency are

$$\begin{aligned} COP &= \frac{\dot{Q}_{evap}}{\dot{W}_{comp}} \\ COSP &= \frac{\dot{Q}_{evap} - \dot{W}_{fan,evap}}{\dot{W}_{comp} + \dot{W}_{fan,evap} + \dot{W}_{fan,cond}} \end{aligned} \tag{2.7}$$

Minimal working example

```
from __future__ import print_function

from ACHP.Cycle import DXCycleClass

#Instantiate the cycle class
Cycle=DXCycleClass()

#-------------------------------------
# Cycle parameters
#-------------------------------------
Cycle.Verbosity = 0 #the idea here is to have different levels of debug output
Cycle.ImposedVariable = 'Subcooling'
```

```
Cycle.DT_sc_target = 7.0
Cycle.Mode='AC'
Cycle.Ref='R410A'
Cycle.Backend='TTSE&HEOS' #Backend for refrigerant properties calculation: 'HEOS',
↪'TTSE&HEOS','BICUBIC&HEOS','REFPROP','SRK','PR'

#---------------------------------------
#       Compressor parameters
#---------------------------------------
#A 3 ton cooling capacity compressor map
M=[217.3163128,5.094492028,-0.593170311,4.38E-02,-2.14E-02,
   1.04E-02,7.90E-05,-5.73E-05,1.79E-04,-8.08E-05]
P=[-561.3615705,-15.62601841,46.92506685,-0.217949552,
   0.435062616,-0.442400826,2.25E-04,2.37E-03,-3.32E-03,2.50E-03]

params={
        'M':M,
        'P':P,
        'Ref':Cycle.Ref, #Refrigerant
        'Backend':Cycle.Backend, #Backend for refrigerant properties calculation
        'fp':0.0, #Fraction of electrical power lost as heat to ambient
        'Vdot_ratio': 1.0, #Displacement Scale factor
        'Verbosity': 0, # How verbose should the debugging be [0-10]
        }

Cycle.Compressor.Update(**params)

#---------------------------------------
#      Condenser parameters
#---------------------------------------
Cycle.Condenser.Fins.Tubes.NTubes_per_bank=24       #number of tubes per bank=row
Cycle.Condenser.Fins.Tubes.Nbank=1                  #number of banks/rows
Cycle.Condenser.Fins.Tubes.Ncircuits=3
Cycle.Condenser.Fins.Tubes.Ltube=2.252
Cycle.Condenser.Fins.Tubes.OD=0.00913
Cycle.Condenser.Fins.Tubes.ID=0.00849
Cycle.Condenser.Fins.Tubes.Pl=0.0191                #distance between center of tubes␣
↪in flow direction
Cycle.Condenser.Fins.Tubes.Pt=0.0254                #distance between center of tubes␣
↪orthogonal to flow direction

Cycle.Condenser.Fins.Fins.FPI=25                    #Number of fins per inch
Cycle.Condenser.Fins.Fins.Pd=0.001                  #2* amplitude of wavy fin
Cycle.Condenser.Fins.Fins.xf=0.001                  #1/2 period of fin
Cycle.Condenser.Fins.Fins.t=0.00011                 #Thickness of fin material
Cycle.Condenser.Fins.Fins.k_fin=237                 #Thermal conductivity of fin␣
↪material

Cycle.Condenser.Fins.Air.Vdot_ha=1.7934             #rated volumetric flowrate
Cycle.Condenser.Fins.Air.Tmean=308.15
Cycle.Condenser.Fins.Air.Tdb=308.15
Cycle.Condenser.Fins.Air.p=101325                   #Condenser Air pressure in Pa
Cycle.Condenser.Fins.Air.RH=0.51
Cycle.Condenser.Fins.Air.RHmean=0.51
Cycle.Condenser.Fins.Air.FanPower=260

Cycle.Condenser.FinsType = 'WavyLouveredFins' #WavyLouveredFins, HerringboneFins,␣
↪PlainFins
```

```
Cycle.Condenser.Ref=Cycle.Ref
Cycle.Condenser.Backend=Cycle.Backend
Cycle.Condenser.Verbosity=0

#-------------------------------------
# Evaporator Parameters
#-------------------------------------
Cycle.Evaporator.Fins.Tubes.NTubes_per_bank=32
Cycle.Evaporator.Fins.Tubes.Nbank=3
Cycle.Evaporator.Fins.Tubes.Ltube=0.452
Cycle.Evaporator.Fins.Tubes.OD=0.00913
Cycle.Evaporator.Fins.Tubes.ID=0.00849
Cycle.Evaporator.Fins.Tubes.Pl=0.0191
Cycle.Evaporator.Fins.Tubes.Pt=0.0254
Cycle.Evaporator.Fins.Tubes.Ncircuits=5

Cycle.Evaporator.Fins.Fins.FPI=14.5
Cycle.Evaporator.Fins.Fins.Pd=0.001
Cycle.Evaporator.Fins.Fins.xf=0.001
Cycle.Evaporator.Fins.Fins.t=0.00011
Cycle.Evaporator.Fins.Fins.k_fin=237

Cycle.Evaporator.Fins.Air.Vdot_ha=0.56319
Cycle.Evaporator.Fins.Air.Tmean=297.039
Cycle.Evaporator.Fins.Air.Tdb=297.039
Cycle.Evaporator.Fins.Air.p=101325              #Evaporator Air pressure in Pa
Cycle.Evaporator.Fins.Air.RH=0.5
Cycle.Evaporator.Fins.Air.RHmean=0.5
Cycle.Evaporator.Fins.Air.FanPower=438

Cycle.Evaporator.FinsType = 'WavyLouveredFins' #WavyLouveredFins, HerringboneFins,␣
↪PlainFins
Cycle.Evaporator.Ref=Cycle.Ref
Cycle.Evaporator.Backend=Cycle.Backend
Cycle.Evaporator.Verbosity=0
Cycle.Evaporator.DT_sh=5

# ---------------------------------
#       Line Set Parameters
# ---------------------------------
params={
        'L':7.6,
        'k_tube':0.19,
        't_insul':0.02,
        'k_insul':0.036,
        'T_air':297,
        'Ref': Cycle.Ref,
        'Backend': Cycle.Backend,
        'h_air':0.0000000001,
        }

Cycle.LineSetSupply.Update(**params)
Cycle.LineSetReturn.Update(**params)
Cycle.LineSetSupply.OD=0.009525
Cycle.LineSetSupply.ID=0.007986
Cycle.LineSetReturn.OD=0.01905
Cycle.LineSetReturn.ID=0.017526
```

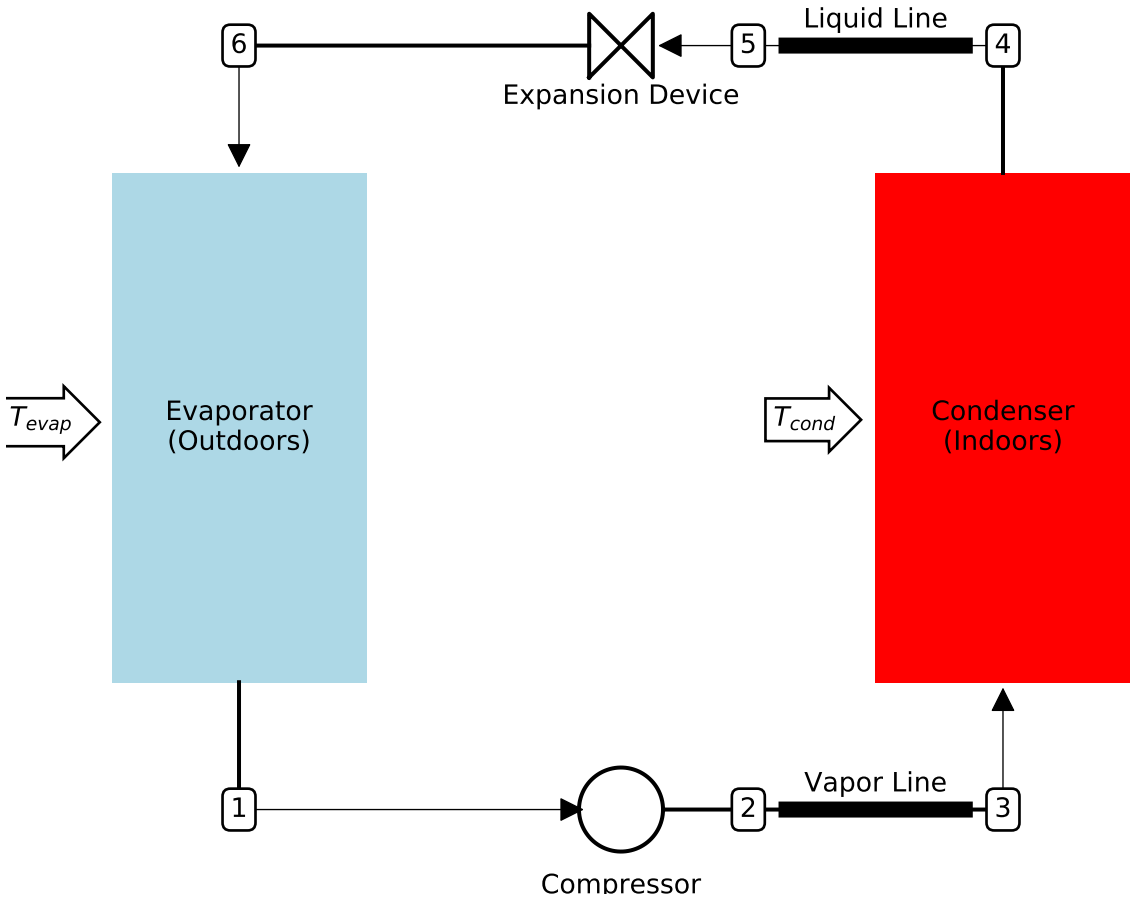
```
#Now solve
from time import time
t1=time()
Cycle.PreconditionedSolve()
print('Took '+str(time()-t1)+' seconds to run Cycle model')
print('Cycle COP is '+str(Cycle.COSP))
print('Cycle refrigerant charge is '+str(Cycle.Charge)+' kg')
```

which should yield the output, when run, of

```
Took 14.908493757247925 seconds to run Cycle model
Cycle COP is 3.20732414824
Cycle refrigerant charge is 2.0542017125183585 kg
```

2.1.2 Direct Expansion Heating Mode

The heat pump configuration of the system is as shown here:



Physically, reversing valves are used to switch the mode of the system and the directions of the flows. What was the condenser of the air conditioning system becomes the evaporator of the heat pump and *vice versa*, and the line sets are configured in a slightly different way. Other than that, the analysis of the heat pump is directly analogous to that of the *Direct Expansion Cooling Mode* system.

A *Preconditioner* is used to get approximate values for T_{evap} and T_{cond}, and using these values (which are iteratively modified using numerical methods), the solution for the cycle performance is found.

As with the cooling mode, the cycle analysis follows the refrigerant flow path around the loop.

Beginning at the outlet of the evaporator, state point 1 is known because T_{evap} and ΔT_{sh} are known. Thus the compressor model is used directly to calculate the electrical power, refrigerant mass flow rate and state point 2.

The *Line Set* model is then applied to the flow from the outlet of the compressor at state point 2 to the inlet of the condenser at state point 3.

The *Condenser* model is used to model the condensing process from state point 3 to a subcooled state at state point 4.

The *Line Set* model is used to model the flow of subcooled refrigerant at state point 4 back to the expansion device at state point 5.

As in cooling mode, the expansion device is assumed to be an ideal expansion device, which means that the working process is a constant-enthalpy expansion from state point 5 to state point 6.

The *Evaporator* model is then used to calculate the evaporation process of refrigerant from state point 6 to state point 1'.

As in cooling mode, the residual vector is given by

$$\vec{\Delta} = [\vec{\Delta}_1, \vec{\Delta}_2] \tag{2.8}$$

with

$$\begin{aligned} \vec{\Delta}_1 &= \dot{m}_r(h_1 - h_{1'}) \\ \vec{\Delta}_2 &= \begin{cases} m_r - m_{r,target} & \text{(Charge imposed)} \\ \Delta T_{sc} - \Delta T_{sc,target} & \text{(Subcooling imposed)} \end{cases} \end{aligned} \tag{2.9}$$

The set of T_{evap} and T_{cond} which solve the residual equations are obtained by a multi-dimensional solver (see *Multi-dimensional solver*).

Cycle Performance Parameters

The common metrics of system efficiency are

$$\begin{aligned} COP &= \frac{\dot{Q}_{cond}}{\dot{W}_{comp}} \\ COSP &= \frac{\dot{Q}_{cond} + \dot{W}_{fan,cond}}{\dot{W}_{comp} + \dot{W}_{fan,evap} + \dot{W}_{fan,cond}} \end{aligned} \tag{2.10}$$

Minimal working example

```
from __future__ import print_function

from ACHP.Cycle import DXCycleClass
from ACHP.convert_units import F2K

#Instantiate the class
Cycle=DXCycleClass()

#-------------------------------------
# Cycle parameters
#-------------------------------------
```

```
Cycle.Verbosity = 0 #the idea here is to have different levels of debug output
Cycle.ImposedVariable = 'Subcooling' #or it could be 'Charge'
Cycle.DT_sc_target = 7.0
#Cycle.Charge_target = 3.3 #uncomment for use with imposed charge
Cycle.Mode='HP'
Cycle.Ref='R410A'
Cycle.Backend='TTSE&HEOS' #Backend for refrigerant properties calculation: 'HEOS',
↪'TTSE&HEOS','BICUBIC&HEOS','REFPROP','SRK','PR'

#-------------------------------------
#       Compressor parameters
#-------------------------------------
#A few 3 ton cooling capacity compressor maps
M=[217.3163128,5.094492028,-0.593170311,4.38E-02,
   -2.14E-02,1.04E-02,7.90E-05,-5.73E-05,1.79E-04,-8.08E-05]
P=[-561.3615705,-15.62601841,46.92506685,-0.217949552,
   0.435062616,-0.442400826,2.25E-04,2.37E-03,-3.32E-03,2.50E-03]

params={
        'M':M,
        'P':P,
        'Ref':Cycle.Ref,        #refrigerant
        'Backend':Cycle.Backend,    #backend for refirgerant properties calculation
        'fp':0.0, #Fraction of electrical power lost as heat to ambient #shell heat␣
↪loss
        'Vdot_ratio': 1.0, #Displacement Scale factor #up- or downsize compressor
        'Verbosity': 0, # How verbose should the debugging statements be [0 to 10]
        }

Cycle.Compressor.Update(**params)

#-------------------------------------
# Condenser parameters
#-------------------------------------
Cycle.Condenser.Fins.Tubes.NTubes_per_bank=32
Cycle.Condenser.Fins.Tubes.Nbank=3
Cycle.Condenser.Fins.Tubes.Ncircuits=6
Cycle.Condenser.Fins.Tubes.Ltube=0.452
Cycle.Condenser.Fins.Tubes.OD=0.009525
Cycle.Condenser.Fins.Tubes.ID=0.0089154
Cycle.Condenser.Fins.Tubes.Pl=0.0254
Cycle.Condenser.Fins.Tubes.Pt=0.0219964

Cycle.Condenser.Fins.Fins.FPI=14.5
Cycle.Condenser.Fins.Fins.Pd=0.001
Cycle.Condenser.Fins.Fins.xf=0.001
Cycle.Condenser.Fins.Fins.t=0.00011
Cycle.Condenser.Fins.Fins.k_fin=237

Cycle.Condenser.Fins.Air.Vdot_ha=0.5663
Cycle.Condenser.Fins.Air.Tmean=F2K(70)
Cycle.Condenser.Fins.Air.Tdb=F2K(70)
Cycle.Condenser.Fins.Air.p=101325               #Condenser Air Pressure in Pa
Cycle.Condenser.Fins.Air.RH=0.51
Cycle.Condenser.Fins.Air.RHmean=0.51
Cycle.Condenser.Fins.Air.FanPower=438

Cycle.Condenser.FinsType = 'WavyLouveredFins' #WavyLouveredFins, HerringboneFins,␣
↪PlainFins
```

```
Cycle.Condenser.Ref=Cycle.Ref
Cycle.Condenser.Backend=Cycle.Backend    #backend for refirgerant properties␣
↪calculation
Cycle.Condenser.Verbosity=0

#-------------------------------------
# Evaporator parameters
#-------------------------------------
Cycle.Evaporator.Fins.Tubes.NTubes_per_bank=41 #number of tubes per bank=row
Cycle.Evaporator.Fins.Tubes.Nbank=1     #number of banks/rows
Cycle.Evaporator.Fins.Tubes.Ncircuits=5
Cycle.Evaporator.Fins.Tubes.Ltube=2.286
Cycle.Evaporator.Fins.Tubes.OD=0.007
Cycle.Evaporator.Fins.Tubes.ID=0.0063904
Cycle.Evaporator.Fins.Tubes.Pl=0.0191   #distance between center of tubes in flow␣
↪direction
Cycle.Evaporator.Fins.Tubes.Pt=0.0222   #distance between center of tubes orthogonal␣
↪to flow direction
Cycle.Evaporator.Fins.Fins.FPI=25       #Number of fins per inch
Cycle.Evaporator.Fins.Fins.Pd=0.001     #2* amplitude of wavy fin
Cycle.Evaporator.Fins.Fins.xf=0.001     #1/2 period of fin
Cycle.Evaporator.Fins.Fins.t=0.00011    #Thickness of fin material
Cycle.Evaporator.Fins.Fins.k_fin=237    #Thermal conductivity of fin material

Cycle.Evaporator.Fins.Air.Vdot_ha=1.7934 #rated volumetric flowrate
Cycle.Evaporator.Fins.Air.Tmean=F2K(47)
Cycle.Evaporator.Fins.Air.Tdb=F2K(47)
Cycle.Evaporator.Fins.Air.p=101325       #Evaporator Air pressure in Pa
Cycle.Evaporator.Fins.Air.RH=0.51
Cycle.Evaporator.Fins.Air.RHmean=0.51
Cycle.Evaporator.Fins.Air.FanPower=160

Cycle.Evaporator.FinsType = 'WavyLouveredFins' #WavyLouveredFins, HerringboneFins,␣
↪PlainFins
Cycle.Evaporator.Ref=Cycle.Ref
Cycle.Evaporator.Backend=Cycle.Backend    #backend for refirgerant properties␣
↪calculation
Cycle.Evaporator.Verbosity=0
Cycle.Evaporator.DT_sh=5

# ----------------------------------
# Line Set parameters
# ----------------------------------
params={
        'L':7.6,
        'k_tube':0.19,
        't_insul':0.02,
        'k_insul':0.036,
        'T_air':297,
        'Ref': Cycle.Ref,
        'Backend': Cycle.Backend,
        'h_air':6,
        }

Cycle.LineSetSupply.Update(**params)
Cycle.LineSetReturn.Update(**params)
Cycle.LineSetSupply.OD=0.01905
Cycle.LineSetSupply.ID=0.017526
```

```
Cycle.LineSetReturn.OD=0.009525
Cycle.LineSetReturn.ID=0.007986

#Now solve
from time import time
t1=time()
Cycle.PreconditionedSolve()

#Outputs
print('Took '+str(time()-t1)+' seconds to run Cycle model')
print('Cycle coefficient of system performance is '+str(Cycle.COSP))
print('Cycle refrigerant charge is '+str(Cycle.Charge)+' kg')
```

which should yield the output, when run, of

```
Took 18.14244508743286 seconds to run Cycle model
Cycle coefficient of system performance is 3.6034037343008345
Cycle refrigerant charge is 1.719562780251362 kg
```

Cycle Solver Code Documentation

class ACHP.Cycle.**DXCycleClass**

Calculate(*DT_evap*, *DT_cond*)

Inputs are differences in temperature [K] between HX air inlet temperature and the dew temperature for the heat exchanger.

Required Inputs:

DT_evap: Difference in temperature [K] between evaporator air inlet temperature and refrigerant dew temperature

DT_cond: Difference in temperature [K] between condenser air inlet temperature and refrigeant dew temperature

OutputList()

Return a list of parameters for this component for further output

It is a list of tuples, and each tuple is formed of items: [0] Description of value [1] Units of value [2] The value itself

PreconditionedSolve()

Solver that will precondition by trying a range of DeltaT until the model can solve, then will kick into 2-D Newton Raphson solve

The two input variables for the system solver are the differences in temperature between the inlet air temperature of the heat exchanger and the dew temperature of the refrigerant. This is important for refrigerant blends with temperature glide during constant-pressure evaporation or condensation. Good examples of common working fluid with glide would be R404A or R410A.

Nomenclature

Variable	Description
COP	Coefficient of Performance [-]
$COSP$	Coefficient of System Performance [-]
h_1	Enthalpy at outlet of evaporator [J/kg]
$h_{1'}$	Enthalpy after going around the cycle [J/kg]
$\dot{m}_r$	Refrigerant mass flow rate [kg/s]
m_r	Model-predicted charge [kg]
$m_{r,target}$	Target refrigerant charge in system [kg]
T_{evap}	Evaporating (dewpoint) temperature [K]
T_{cond}	Condensing (dewpoint) temperature [K]
T^*_{evap}	Effective evaporating temperature [K]
T^*_{cond}	Effective condensing temperature [K]
ΔT_{sh}	Evaporator outlet superheat [K]
$\Delta T_{sc,cond}$	Condenser outlet subcooling [K]
$\Delta T_{sc,cond,target}$	Condenser outlet subcooling target [K]
Δp_{cond}	Pressure drop in condenser [Pa]
Δp_{evap}	Pressure drop in evaporator [Pa]
Δp_{high}	Pressure drop on high-pressure side of system [Pa]
$\Delta p_{liquid-line}$	Pressure drop in liquid line [Pa]
Δp_{low}	Pressure drop on low-pressure side of system [Pa]
$\Delta p_{vapor-line}$	Pressure drop in vapor line [Pa]
p^*_{evap}	Effective evaporation saturation pressure [Pa]
p^*_{cond}	Effective condensing saturation pressure [Pa]
$\dot{Q}_{evap}$	Evaporator heat transfer rate [W]
$\dot{W}_{fan,evap}$	Evaporator fan power [W]
$\dot{W}_{fan,cond}$	Condenser fan power [W]
$\dot{W}_{comp}$	Compressor power input [W]
$\vec{\Delta}_1$	Residual vector [varied]

2.2 Secondary Loop Cycle Solver

2.2.1 Secondary Loop Cooling Mode

For secondary loops in cooling mode, there is an internal heat exchanger which physically separates the secondary loop and the refrigerant loop.

In this case, there are now three inputs, and three residuals (to be defined later). The three inputs are $T_{g,i,cc}$, T_{evap} and T_{cond}.

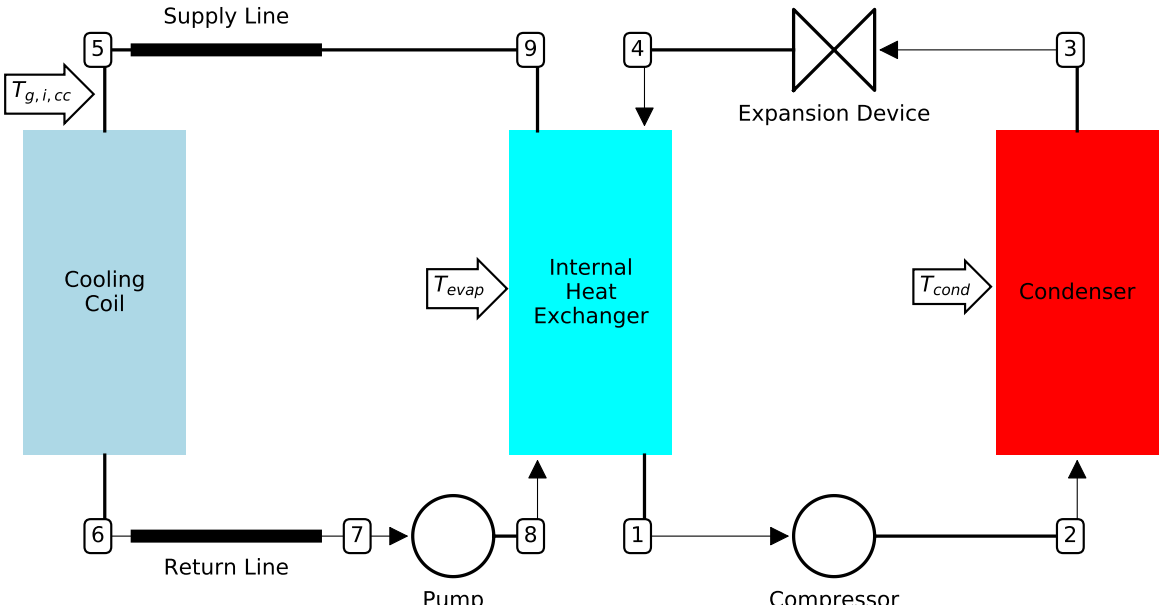
The two loops can be solved separately, where for the secondary loop, the inlet temperature to the cooling coil is known, and the secondary working fluid's properties are independent of pressure.

To begin, the *Cooling Coil* model is employed to calculate the heat transfer rate in the cooling coil, which gives the process from state point 5 to state point 6.

The *Line Set* model is used to determine the heat transfer and pressure drop in the line going from state point 6 to state point 7.

The pump model is run to determine how much electrical power is consumed in the pump, which gives the state point 8, the glycol inlet to the internal heat exchanger.

The refrigerant loop is then solved. The refrigerant superheat at the outlet of the IHX is imposed as an input for the cycle. Thus state point 1 is known, and the *Compressor* model is used to calculate state point 2, the compressor mass



flow rate, electrical power, etc..

The *Condenser* model is then solved using the state point 2 as the inlet, and yielding the (hopefully) subcooled refrigerant outlet state point 3.

Finally, the isenthalpic throttling process is used to determine the state point 4 at the refrigerant inlet to the IHX.

Lastly, the *Plate-Heat-Exchangers* model is run, using the inputs at state points 8 and 4, and yielding the refrigerant outlet state of state point 1'.

The residuals are given by an energy balance between state points 1 and 1', an energy balance on the secondary loop, and either matching the mass or the subcooling on the refrigerant side. As in the DX system analysis, the residual vector is given by

$$\vec{\Delta} = [\vec{\Delta}_1, \vec{\Delta}_2, \vec{\Delta}_3] \tag{2.11}$$

with

$$\begin{aligned} \vec{\Delta}_1 &= \dot{m}_r(h_1 - h_{1'}) \\ \vec{\Delta}_2 &= \begin{cases} m_r - m_{r,target} & \text{(Charge imposed)} \\ \Delta T_{sc,cond} - \Delta T_{sc,cond,target} & \text{(Subcooling imposed)} \end{cases} \\ \vec{\Delta}_3 &= \dot{W}_{pump} + \dot{Q}_{cc} - \dot{Q}_{IHX} + \dot{Q}_{\text{suppplyline}} + \dot{Q}_{\text{returnline}} \end{aligned} \tag{2.12}$$

The set of $T_{g,i,cc}$, T_{evap} and T_{cond} which solve the residual equations are obtained by a multi-dimensional solver (see *Multi-dimensional solver*).

The common metrics of system efficiency are

$$\begin{aligned} COP &= \frac{\dot{Q}_{cc}}{\dot{W}_{comp}} \\ COSP &= \frac{\dot{Q}_{cc} - \dot{W}_{fan,cc}}{\dot{W}_{comp} + \dot{W}_{fan,cc} + \dot{W}_{fan,cond} + \dot{W}_{pump}} \end{aligned} \tag{2.13}$$

```
from __future__ import print_function

from ACHP.Cycle import SecondaryCycleClass

# Instantiate the class
Cycle = SecondaryCycleClass()

#-------------------------------------
#       Cycle parameters
#-------------------------------------
Cycle.Verbosity = 0 #the idea here is to have different levels of debug output
Cycle.ImposedVariable = 'Subcooling' #or this could be 'Charge' for imposed charge
Cycle.DT_sc_target = 7.0
#Cycle.Charge_target = 2.4  #Needed if charge is imposed, not otherwise
Cycle.Ref='R410A'
Cycle.SecLoopFluid = 'MEG'
Cycle.MassFrac_SLF = 0.21 #Mass fraction of incompressible SecLoopFluid [i.e MEG-20%]
Cycle.Backend_SLF = 'INCOMP' #backend of SecLoopFluid
Cycle.IHXType = 'PHE'# or could be 'Coaxial'
Cycle.Mode='AC'

#-------------------------------------
#-------------------------------------
#       Compressor parameters
#-------------------------------------
#-------------------------------------

# A 3 ton cooling capacity compressor map
if Cycle.Ref=='R410A':
    M=[217.3163128,5.094492028,-0.593170311,4.38E-02,-2.14E-02,1.04E-02,
    7.90E-05,-5.73E-05,1.79E-04,-8.08E-05]
    P=[-561.3615705,-15.62601841,46.92506685,-0.217949552,0.435062616,
    -0.442400826,2.25E-04,2.37E-03,-3.32E-03,2.50E-03]

params={
    'M':M,
    'P':P,
    'Ref':Cycle.Ref,    #refrigerant
    'fp':0.15, #Fraction of electrical power lost as heat to ambient
    'Vdot_ratio': 1.0, #Displacement Scale factor to up- or downsize compressor␣
↪(1=original)
    'Verbosity': 0, # How verbose should the debugging statements be [0 to 10]
  }
Cycle.Compressor.Update(**params)

#-------------------------------------
#      Condenser parameters
#-------------------------------------
Cycle.Condenser.Fins.Tubes.NTubes_per_bank=24  #number of tubes per bank=row
Cycle.Condenser.Fins.Tubes.Nbank=1             #number of banks/rows
Cycle.Condenser.Fins.Tubes.Ncircuits=3
Cycle.Condenser.Fins.Tubes.Ltube=2.252
Cycle.Condenser.Fins.Tubes.OD=0.00913
Cycle.Condenser.Fins.Tubes.ID=0.00849
Cycle.Condenser.Fins.Tubes.Pl=0.0191  #distance between center of tubes in flow␣
↪direction
Cycle.Condenser.Fins.Tubes.Pt=0.0254  #distance between center of tubes orthogonal to␣
↪flow direction
```

```
Cycle.Condenser.Fins.Fins.FPI=25      #Number of fins per inch
Cycle.Condenser.Fins.Fins.Pd=0.001    #2* amplitude of wavy fin
Cycle.Condenser.Fins.Fins.xf=0.001    #1/2 period of fin
Cycle.Condenser.Fins.Fins.t=0.00011   #Thickness of fin material
Cycle.Condenser.Fins.Fins.k_fin=237   #Thermal conductivity of fin material

Cycle.Condenser.Fins.Air.Vdot_ha=1.7934 #rated volumetric flowrate
Cycle.Condenser.Fins.Air.Tmean=308.15
Cycle.Condenser.Fins.Air.Tdb=308.15     #Dry Bulb Temperature
Cycle.Condenser.Fins.Air.p=101325       #Air pressure
Cycle.Condenser.Fins.Air.RH=0.51        #Relative Humidity
Cycle.Condenser.Fins.Air.RHmean=0.51
Cycle.Condenser.Fins.Air.FanPower=260

Cycle.Condenser.FinsType = 'WavyLouveredFins' #WavyLouveredFins, HerringboneFins,␣
↪PlainFins
Cycle.Condenser.Ref=Cycle.Ref
Cycle.Condenser.Verbosity=0

#-------------------------------------
#      Cooling Coil parameters
#-------------------------------------
Cycle.CoolingCoil.Fins.Tubes.NTubes_per_bank=32
Cycle.CoolingCoil.Fins.Tubes.Nbank=3
Cycle.CoolingCoil.Fins.Tubes.Ncircuits=5
Cycle.CoolingCoil.Fins.Tubes.Ltube=0.452
Cycle.CoolingCoil.Fins.Tubes.OD=0.00913
Cycle.CoolingCoil.Fins.Tubes.ID=0.00849
Cycle.CoolingCoil.Fins.Tubes.Pl=0.0191
Cycle.CoolingCoil.Fins.Tubes.Pt=0.0254

Cycle.CoolingCoil.Fins.Fins.FPI=14.5
Cycle.CoolingCoil.Fins.Fins.Pd=0.001
Cycle.CoolingCoil.Fins.Fins.xf=0.001
Cycle.CoolingCoil.Fins.Fins.t=0.00011
Cycle.CoolingCoil.Fins.Fins.k_fin=237

Cycle.CoolingCoil.Fins.Air.Vdot_ha=0.56319
Cycle.CoolingCoil.Fins.Air.Tmean=297.039
Cycle.CoolingCoil.Fins.Air.Tdb=297.039
Cycle.CoolingCoil.Fins.Air.p=101325
Cycle.CoolingCoil.Fins.Air.RH=0.5
Cycle.CoolingCoil.Fins.Air.RHmean=0.5
Cycle.CoolingCoil.Fins.Air.FanPower=438

params={
    'Ref_g': Cycle.SecLoopFluid,
    'Backend_g': Cycle.Backend_SLF,
    'MassFrac_g': Cycle.MassFrac_SLF,
    'pin_g': 200,
    'Verbosity':0,
    'mdot_g':0.38,
    'FinsType':'WavyLouveredFins' #WavyLouveredFins, HerringboneFins, PlainFins
}
Cycle.CoolingCoil.Update(**params)

params={
```

```
    'ID_i':0.0278,
    'OD_i':0.03415,
    'ID_o':0.045,
    'L':50,
    'pin_g':300,
    'Ref_r':Cycle.Ref,
    'Ref_g':Cycle.SecLoopFluid,
    'Backend_g': Cycle.Backend_SLF,
    'MassFrac_g': Cycle.MassFrac_SLF,
    'Verbosity':0
    }
Cycle.CoaxialIHX.Update(**params)

params={
    'pin_h':300000,
    'Ref_h':Cycle.SecLoopFluid,
    'Backend_h': Cycle.Backend_SLF,
    'MassFrac_h': Cycle.MassFrac_SLF,
    'Ref_c':Cycle.Ref,

    #Geometric parameters
    'Bp' : 0.117,
    'Lp' : 0.300, #Center-to-center distance between ports
    'Nplates' : 46,
    'PlateAmplitude' : 0.001, #[m]
    'PlateThickness' : 0.0003, #[m]
    'PlateConductivity' : 15.0, #[W/m-K]
    'PlateWavelength' : 0.00628, #[m]
    'InclinationAngle' : 3.14159/3,#[rad]
    'MoreChannels' : 'Hot', #Which stream gets the extra channel, 'Hot' or 'Cold'
    'Verbosity':0,
    'DT_sh':5
}
Cycle.PHEIHX.Update(**params)

params={
    'eta':0.5,  #Pump+motor efficiency
    'mdot_g':0.38, #Flow Rate kg/s
    'pin_g':300000,
    'Ref_g':Cycle.SecLoopFluid,
    'Backend_g': Cycle.Backend_SLF,
    'MassFrac_g': Cycle.MassFrac_SLF,
    'Verbosity':0,
    }
Cycle.Pump.Update(**params)

params={
    'L':5,
    'k_tube':0.19,
    't_insul':0.02,
    'k_insul':0.036,
    'T_air':297,
    'Ref': Cycle.SecLoopFluid,
    'Backend': Cycle.Backend_SLF,
    'MassFrac': Cycle.MassFrac_SLF,
    'pin': 300000,
    'h_air':0.0000000001,
}
```

```
Cycle.LineSetSupply.Update(**params)
Cycle.LineSetReturn.Update(**params)
Cycle.LineSetSupply.OD=0.009525
Cycle.LineSetSupply.ID=0.007986
Cycle.LineSetReturn.OD=0.01905
Cycle.LineSetReturn.ID=0.017526

#Now solve
from time import time
t1=time()
Cycle.PreconditionedSolve()

#Outputs
print('Took '+str(time()-t1)+' seconds to run Cycle model')
print('Cycle coefficient of system performance is '+str(Cycle.COSP))
print('Cycle refrigerant charge is '+str(Cycle.Charge)+' kg')
```

which should yield the output, when run, of

```
Took 8.836137294769287 seconds to run Cycle model
Cycle coefficient of system performance is 2.7891650414848344
Cycle refrigerant charge is 1.4989869734994357 kg
```

2.2.2 Secondary Loop Heating Mode

This section is left intentionally empty

2.2.3 Cycle Solver Class Documentation

class ACHP.Cycle.**SecondaryCycleClass**

Calculate(*DT_evap*, *DT_cond*, *Tin_CC*)
Inputs are differences in temperature [K] between HX air inlet temperature and the dew temperature for the heat exchanger.

Required Inputs:

DT_evap: Difference in temperature [K] between cooling coil air inlet temperature and refrigerant dew temperature

DT_cond: Difference in temperature [K] between condenser air inlet temperature and refrigerant dew temperature

Tin_CC: Inlet "glycol" temperature to line set feeding cooling coil

OutputList()
Return a list of parameters for this component for further output

It is a list of tuples, and each tuple is formed of items: [0] Description of value [1] Units of value [2] The value itself

PreconditionedSolve(*PrecondValues=None*)
PrecondValues = dictionary of values DT_evap, DT_cond and Tin_CC

Nomenclature

Variable	Description
COP	Coefficient of Performance [-]
$COSP$	Coefficient of System Performance [-]
h_1	Enthalpy at outlet of evaporator [J/kg]
$h_{1'}$	Enthalpy after going around the cycle [J/kg]
$\dot{m}_r$	Refrigerant mass flow rate [kg/s]
m_r	Model-predicted charge [kg]
$m_{r,target}$	Target refrigerant charge in system [kg]
T_{cond}	Condensing (dewpoint) temperature [K]
T_{evap}	Evaporating (dewpoint) temperature [K]
$T_{g,i,cc}$	Glycol inlet temperature to cooling coil [K]
ΔT_{sh}	IHX outlet superheat [K]
$\Delta T_{sc,cond}$	Condenser outlet subcooling [K]
$\Delta T_{sc,cond,target}$	Condenser outlet subcooling target [K]
$\dot{Q}_{cc}$	Cooling coil heat transfer rate [W]
$\dot{W}_{fan,cc}$	Cooling coil fan power [W]
$\dot{W}_{fan,cond}$	Condenser fan power [W]
$\dot{W}_{comp}$	Compressor power input [W]
$\vec{\Delta}$	Residual vector [varied]

2.3 Cycle Solver Preconditioner

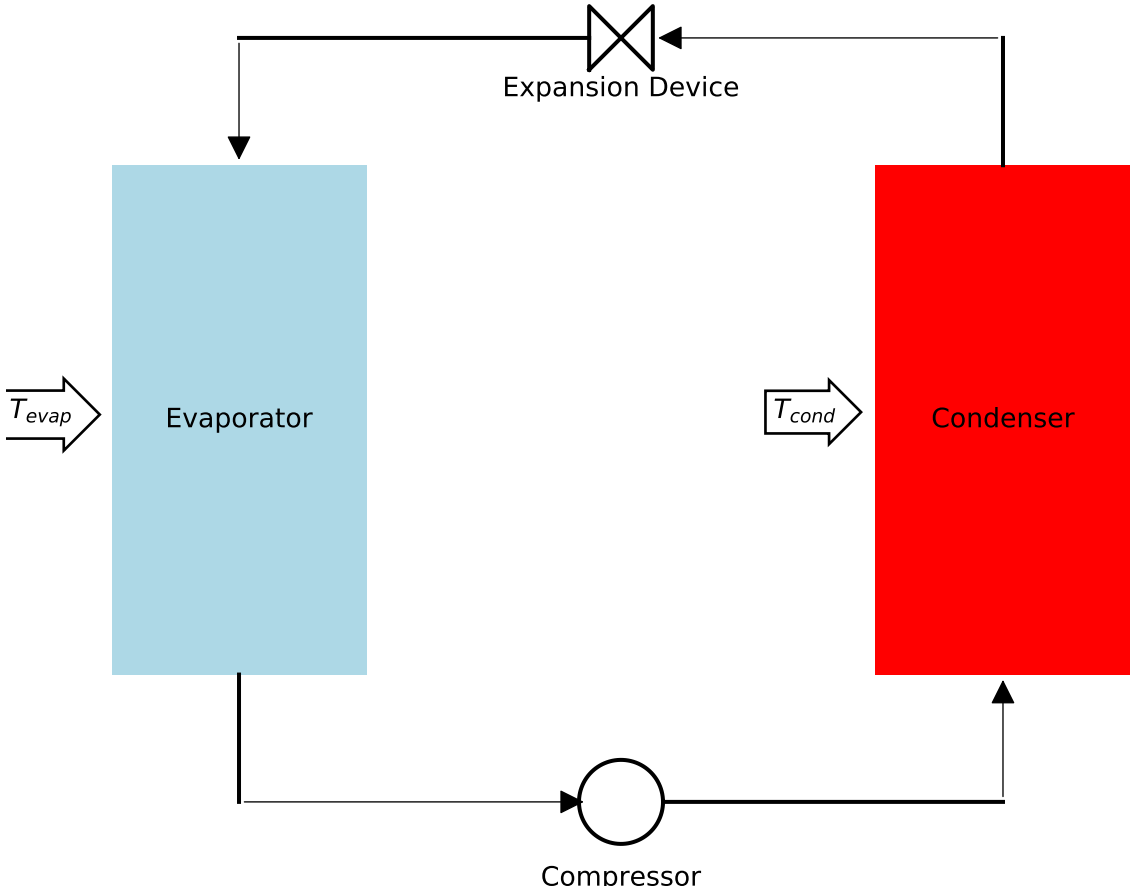
The basic idea of a preconditioner is to use an extremely simple model in order to obtain good initial values for the solver used for the cycle in ACHP.

2.3.1 Conventional system

The following assumptions are employed in the preconditioner employed for the conventional heat pump and air conditioning models:

- Condenser air stream is the minimum capacitance rate in the condenser, and the limiting outlet state for the air in the condenser is T_{cond}. [This further assumes that there is no subcooled section, but this is ok, because there is also a pinch point at the saturated liquid point, so the basic analysis still works. In the end, the preconditioner is only used to get an *approximate* solution anyway, so inaccuracy is acceptable, as long as it is reasonable.]
- Effectiveness of all heat exchangers is known, and fixed
- Compressor inlet superheat is fixed and known
- Compressor is adiabatic
- Evaporator is fully wet, fully dry, or a simple weighted mix of the two
- Line sets are not considered

Essentially the preconditioner operates with the same independent variables as ACHP, which are the dew temperatures of the refrigerant in the evaporator and condenser, given by the variables T_{evap} and T_{cond} respectively.


Expansion Device
Evaporator
T_{evap}
Condenser
T_{cond}
Compressor

Compressor

For a given set of T_{evap}, T_{cond}, and known superheat ΔT_{sh}, the inlet state and outlet pressure for the compressor is known. Therefore, the compressor map can be used to predict the refrigerant mass flow rate as well as the compressor power. This yields the values

$$\begin{aligned} \dot{m}_r &= f_{map}(T_{evap}, T_{cond}, \Delta T_{sh}) \\ \dot{W}_{comp} &= f_{map}(T_{evap}, T_{cond}, \Delta T_{sh}) \end{aligned} \tag{2.14}$$

Condenser

The heat transfer rate in the condenser can therefore be given by the value

$$\dot{Q}_{cond} = \varepsilon_{HX} \rho_{da} \dot{V}_{ha,cond} c_{p,a} (T_{i,a,cond} - T_{cond}) \tag{2.15}$$

The condenser heat transfer rate based on the imposed subcooling can also be given by

$$\dot{Q}_{cond,\Delta h} = \dot{m}_r (h_{r,o,comp} - h(T_{cond} - \Delta T_{sc}, p_{cond})) \tag{2.16}$$

Theoretically these two heat transfer rate terms should match, and consistency is imposed by the numerical solver that is employed.

Evaporator

As usual, the evaporator is the most complicated component due to the possibility of the evaporator coil being fully wet, fully dry, or partially wet and partially dry. As in the full evaporator model, the evaporator is first considered to be fully dry, yielding the heat transfer rate of

$$\dot{Q}_{evap,dry} = \varepsilon_{HX} \rho_{da} \dot{V}_{ha,evap} c_{p,a} (T_{i,a,evap} - T_{evap}) \tag{2.17}$$

Then using the dry evaporator heat transfer analysis it is possible to determine the surface temperature. The UA values can be obtained from

$$\begin{aligned} \mathrm{UA}_r &= \alpha_r A_{r,total} \\ \mathrm{UA}_a &= \eta_a \alpha_a A_{a,total} \end{aligned} \tag{2.18}$$

The outlet temperature of the air can be given from

$$T_{o,a,evap} = T_{i,a,evap} - \frac{\dot{Q}_{evap,dry}}{\dot{m}_{a,total} c_{p,a}} \tag{2.19}$$

which yields the air inlet surface temperature of

$$T_{s,a,i} = \frac{\mathrm{UA}_a T_{i,a,evap} + \mathrm{UA}_r T_{evap}}{\mathrm{UA}_a + \mathrm{UA}_r} \tag{2.20}$$

and the air outlet surface temperature of

$$T_{s,a,o} = \frac{\mathrm{UA}_a T_{o,a,evap} + \mathrm{UA}_r T_{evap}}{\mathrm{UA}_a + \mathrm{UA}_r} \tag{2.21}$$

If both $T_{s,a,o}$ and $T_{s,a,i}$ are above the dewpoint temperature of the entering air (T_{dp}), the rate of heat transfer in the evaporator is equal to the dry-analysis heat transfer rate. If both $T_{s,a,o}$ and $T_{s,a,i}$ are below the dewpoint of the entering air, the coil is entirely wet, for which the heat transfer rate can be obtained from

$$\dot{Q}_{evap,wet} = \varepsilon_{HX} \rho_{da} \dot{V}_{ha,evap} (h_{a,i} - h_{a,s,evap}) \tag{2.22}$$

where $h_{a,s,evap}$ is the saturated air enthalpy at T_{evap} and $h_{a,i}$ is the enthalpy of the inlet air to the evaporator.

If the dewpoint of the inlet air is somewhere between $T_{s,a,o}$ and $T_{s,a,i}$, the heat transfer rate in the evaporator is given by a simple weighting. This yields the following solution for the evaporator:

$$\dot{Q}_{evap} = \begin{cases} \dot{Q}_{evap,dry} & T_{s,a,o} > T_{dp} \\ \dot{Q}_{evap,wet} & T_{s,a,i} < T_{dp} \\ \frac{T_{s,a,o}-T_{dp}}{T_{s,a,o}-T_{s,a,i}} \dot{Q}_{evap,wet} + \left(1 - \frac{T_{s,a,o}-T_{dp}}{T_{s,a,o}-T_{s,a,i}}\right) \dot{Q}_{evap,dry} & T_{s,a,i} > T_{dp} > T_{s,a,o} \end{cases} \tag{2.23}$$

Solution Method

The residuals to driven to zero are therefore an overall energy balance over the system, as well as matching $\dot{Q}_{cond}$ and $\dot{Q}_{cond,\Delta h}$. So the residual vector as a function of T_{evap} and T_{cond} can be expressed as

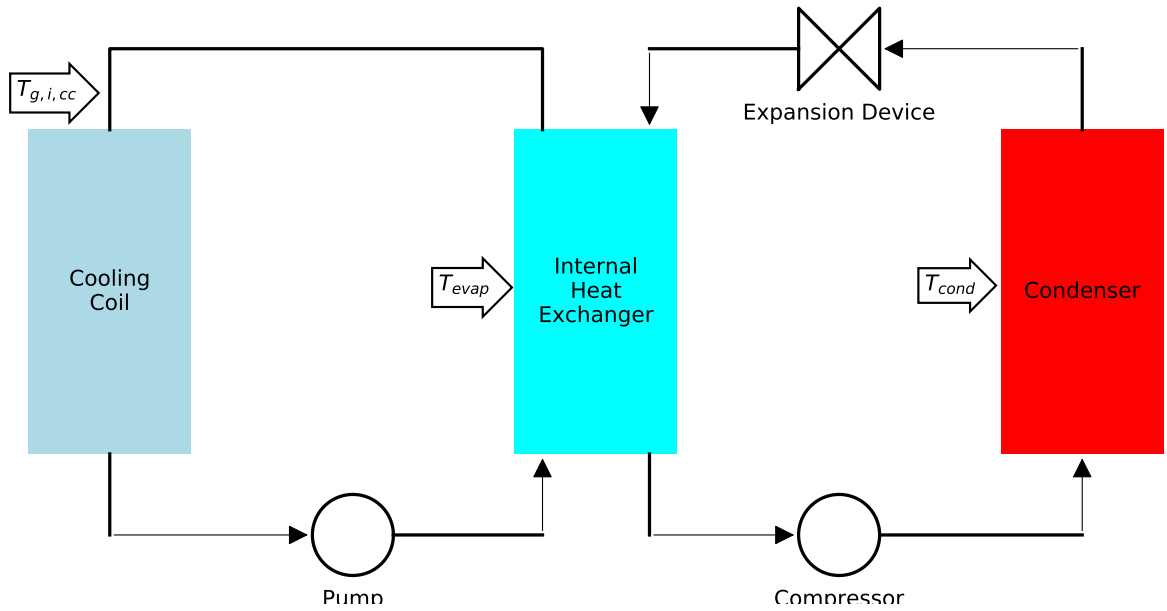
$$\vec{\Delta}(T_{evap}, T_{cond}) = \begin{bmatrix} \dot{Q}_{evap} + \dot{W}_{comp} + \dot{Q}_{cond} \\ \dot{Q}_{cond} + \dot{Q}_{cond,\Delta h} \end{bmatrix} \tag{2.24}$$

and a two-dimensional solver can be used to drive the norm of $\vec{\Delta}$ to sufficiently close to zero by altering T_{evap} and T_{cond}.

Heating Mode

In heating mode, the system schematic remains exactly the same, and the same analysis is used, but the physical geometry of the evaporator and condenser are swapped.

2.3.2 Secondary Loop Systems



The same basic structure is employed for the preconditioner for the secondary loop systems, except that one more variable must be determined by the preconditioner. The preconditioner for the secondary loop system is used to determine the saturation temperatures T_{evap} and T_{cond}, as well as the cooling coil inlet temperature $T_{g,i,cc}$.

The same exact analysis as for the DX preconditioner is employed for the compressor and condenser, and a very similar analysis is used for the cooling coil. The cooling coil analysis mirrors that of the evaporator, as described here.

Cooling Coil

The cooling coil is first considered to be fully dry, yielding the heat transfer rate of

$$\dot{Q}_{cc,dry} = \varepsilon_{HX}\rho_{da}\dot{V}_{ha,cc}c_{p,a}(T_{a,i,cc} - T_{g,i,cc}) \tag{2.25}$$

Then using the dry cooling coil heat transfer analysis it is possible to determine the surface temperature. The UA values can be obtained from

$$\begin{aligned} \mathrm{UA}_g &= \alpha_g A_g \\ \mathrm{UA}_a &= \eta_a \alpha_a A_a \end{aligned} \tag{2.26}$$

The outlet temperature of the air can be given from

$$\begin{aligned} T_{a,o,cc} &= T_{a,i,cc} - \frac{\dot{Q}_{cc,dry}}{\dot{m}_a c_{p,a}} \\ T_{g,o,cc} &= T_{g,i,cc} + \frac{\dot{Q}_{cc,dry}}{\dot{m}_g c_{p,g}} \end{aligned} \tag{2.27}$$

which yields the air inlet surface temperature of

$$T_{s,a,i} = \frac{\mathrm{UA}_a T_{a,i,cc} + \mathrm{UA}_g T_{g,i,cc}}{\mathrm{UA}_a + \mathrm{UA}_g} \tag{2.28}$$

and the air outlet surface temperature of

$$T_{s,a,o} = \frac{\mathrm{UA}_a T_{a,o,cc} + \mathrm{UA}_g T_{g,o,cc}}{\mathrm{UA}_a + \mathrm{UA}_g} \tag{2.29}$$

If both $T_{s,a,o}$ and $T_{s,a,i}$ are above the dewpoint temperature of the entering air (T_{dp}), the rate of heat transfer in the evaporator is equal to the dry-analysis heat transfer rate. If both $T_{s,a,o}$ and $T_{s,a,i}$ are below the dewpoint of the entering air, the coil is entirely wet, for which the heat transfer rate can be obtained from

$$\dot{Q}_{cc,wet} = \varepsilon_{HX}\rho_{da}\dot{V}_{ha,cc}(h_{a,i} - h_{a,s,cc}) \tag{2.30}$$

where $h_{a,s,cc}$ is the saturated air enthalpy at $T_{g,i}$ and $h_{a,i}$ is the enthalpy of the inlet air to the cooling coil.

If the dewpoint of the inlet air is somewhere between $T_{s,a,o}$ and $T_{s,a,i}$, the heat transfer rate in the cooling coil is given by a simple weighting. This yields the following solution for the cooling coil:

$$\dot{Q}_{cc} = \begin{cases} \dot{Q}_{cc,dry} & T_{s,a,o} > T_{dp} \\ \dot{Q}_{cc,wet} & T_{s,a,i} < T_{dp} \\ \frac{T_{s,a,o}-T_{dp}}{T_{s,a,o}-T_{s,a,i}}\dot{Q}_{cc,wet} + \left(1 - \frac{T_{s,a,o}-T_{dp}}{T_{s,a,o}-T_{s,a,i}}\right)\dot{Q}_{cc,dry} & T_{s,a,i} > T_{dp} > T_{s,a,o} \end{cases} \tag{2.31}$$

Internal Heat Exchanger

Once the cooling coil code has been run, the glycol outlet temperature of the cooling coil can be obtained from

$$T_{g,o,cc} = T_{g,i,cc} + \dot{Q}_{cc}/\dot{m}_g \tag{2.32}$$

The heat transfer rate in the internal heat exchanger is then given by

$$\dot{Q}_{IHX} = \varepsilon_{HX}\dot{m}_g c_{p,g}(T_{g,o,cc} - T_{evap}) \tag{2.33}$$

because the glycol is the limiting capacitance rate in the two-phase portion of the IHX.

Solution Method

The residuals to driven to zero are therefore an overall energy balance over the refrigerant loop, matching $\dot{Q}_{cond}$ and $\dot{Q}_{cond,\Delta h}$, and an energy balance over the secondary loop. So the residual vector as a function of T_{evap}, T_{cond}, and $T_{g,i,cc}$ can be expressed as

$$\vec{\Delta}(T_{evap}, T_{cond}, T_{g,i,cc}) = \begin{bmatrix} \dot{Q}_{IHX} + \dot{W}_{comp} + \dot{Q}_{cond} \\ \dot{Q}_{cond} + \dot{Q}_{cond,\Delta h} \\ \dot{Q}_{cc} - \dot{Q}_{IHX} \end{bmatrix} \tag{2.34}$$

and a three-dimensional solver can be used to drive the norm of $\vec{\Delta}$ to sufficiently close to zero by altering T_{evap}, T_{cond}, and $T_{g,i,cc}$.

The code for the preconditioners can be found in `Preconditioners.py`

Nomenclature

Variable	Description
α_g	Mean glycol heat transfer coefficient [W/m^2/K]
α_r	Mean refrigerant heat transfer coefficient [W/m^2/K]
α_a	Mean air heat transfer coefficient [W/m^2/K]
$\vec{\Delta}$	Residual vector [W]
η_a	Overall air-side surface efficiency [-]
ε_{HX}	Effectiveness of heat exchangers [-]
ρ_{da}	Density of humid air [$\mathrm{kg_{da}}$/m^3]
$A_{a,total}$	Total air-side surface area of evaporator (fins+tubes) [m^2]
$A_{r,total}$	Total refrigerant-side surface area of evaporator [m^2]
$c_{p,a}$	Specific heat of humid air [J/$\mathrm{kg_{da}}$/K]
$h_{a,s,cc}$	Enthalpy of air saturated at $T_{g,i,cc}$ [J/$\mathrm{kg_{da}}$]
$h_{a,s,sat}$	Enthalpy of air saturated at T_{evap} [J/$\mathrm{kg_{da}}$]
$h_{a,i}$	Enthalpy of air inlet to evaporator [J/$\mathrm{kg_{da}}$]
$h_{r,o,comp}$	Compressor outlet enthalpy [J/kg]
$T_{a,i,cond}$	Condenser air inlet dry-bulb temperature [K]
$T_{a,i,cc}$	Cooling coil air inlet dry-bulb temperature [K]
$T_{a,o,cc}$	Cooling coil outlet dry-bulb temperature [K]
$T_{a,i,evap}$	Evaporator air inlet dry-bulb temperature [K]
$T_{a,o,evap}$	Evaporator air outlet dry-bulb temperature [K]
$T_{g,i,cc}$	Cooling coil glycol inlet temperature [K]
$T_{g,o,cc}$	Cooling coil glycol outlet temperature [K]
T_{dp}	Dewpoint temperature of humid air [K]
T_{evap}	Evaporator dew temperature [K]
T_{cond}	Condenser dew temperature [K]
$T_{s,a,i}$	Surface temperature of air at air inlet [K]
$T_{s,a,o}$	Surface temperature of air at air outlet [K]
ΔT_{sh}	Compressor suction superheat [K]
ΔT_{sc}	Condenser outlet subcooling [K]
p_{cond}	Condenser pressure [Pa (abs)]
$\dot{m}_g$	Mass flow rate of glycol [kg/s]
$\dot{m}_r$	Mass flow rate of refrigerant [kg/s]
$\dot{m}_{a,total}$	Mass flow rate of dry air through evaporator [$\mathrm{kg_{da}}$/s]
$\dot{Q}_{evap}$	Evaporator heat transfer rate [W]
$\dot{Q}_{evap,dry}$	Evaporator fully-dry heat transfer rate [W]

Continued on next page

Table 2.1 – continued from previous page

Variable	Description
$\dot{Q}_{evap,wet}$	Evaporator fully-wet heat transfer rate [W]
$\dot{Q}_{cc}$	Cooling Coil heat transfer rate [W]
$\dot{Q}_{cc,dry}$	Cooling Coil fully-dry heat transfer rate [W]
$\dot{Q}_{cc,wet}$	Cooling coil fully-wet heat transfer rate [W]
$\dot{Q}_{cond}$	Condenser heat transfer rate [W]
$\dot{Q}_{cond,\Delta h}$	Condenser heat transfer rate from change in enthalpy [W]
$\dot{Q}_{IHX}$	Internal Heat Exchanger heat transfer rate [W]
UA_a	Air-side UA value [W/K]
UA_g	Glycol-side UA value [W/K]
UA_r	Refrigerant-side UA value [W/K]
$\dot{V}_{ha,cond}$	Volumetric flow rate of humid air in condenser [m^3/s]
$\dot{V}_{ha,evap}$	Volumetric flow rate of humid air in evaporator [m^3/s]
$\dot{V}_{ha,cc}$	Volumetric flow rate of humid air in cooling coil [m^3/s]
$\dot{W}_{comp}$	Electrical power of compressor [W]

2.4 General Numerics

2.4.1 One-dimensional Secant solver

If you have a function f, which is a function of a single variable x and the derivative of f cannot be found in a simple fashion, a secant solver can be used to find the value of x that yields the equality $f(x) = 0$.

The secant solver requires two initial guesses which set the search direction of the secant search. It is not necessarily well-behaved, and can struggle to find a solution at times. Also, if there are multiple solutions, you must start quite close to the solution that you want. All that said, it is a simple model to implement, and performs quite admirably most of the time. With the two initial guesses for x of x_0 and x_1, the function is evaluated at these points, yielding the functional values $f_0 = f(x_0)$ and $f_1 = f(x_1)$. The new guess for x is then found from

$$x_2 = x_1 - \frac{f_1}{\frac{f_1 - f_0}{x_1 - x_0}} \tag{2.35}$$

and the values of x_i and f_i are updated and this method is repeated until $|f(x)|$ is small enough.

The Scientific Python package includes an implementation of this method, and a simple example of its use is

```
#Import the newton function (implements secant method if no function derivative␣
↪provided)
In [1]: from scipy.optimize import newton

# newton(function handle, x0), also see
# http://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.newton.html
In [2]: newton(lambda x: x**2-0.4,1.5)
Out[2]: 0.6324555320337425

# The exact solution - should be the same.
# a**b in Python is a to the power of b
In [3]: 0.4**0.5
\\\\\\\\\\\\\\\\\\\\\\\\\\\\Out[3]: 0.6324555320336759
```

which provides a terribly inefficient solution for the square root of 0.4.

2.4.2 One-dimensional Bounded solver

If on the other hand, you know that the solution of $f(x) = 0$ is bounded within some interval between a and b, a host of more powerful and robust solution methods are possible.

Bisection Method

The bisection method is guaranteed to converge so long as the range [a,b] brackets the solution and the function is continuous. In the bisection method, the interval is repetitively chopped in half until the width of the interval $w = |a - b|$ is small enough. First the values of the function are evaluated at a and b, and the midpoint of the interval $m = (a + b)/2$, to yield the functional values of $f(a)$, $f(b)$, and $f(m)$. If $f(a)f(m) > 0$, a and m do not bracket the solution (solution must be between m and b), and the interval bounds are updated by

if $f(a)f(m) > 0$,

$$a = m \tag{2.36}$$

if $f(a)f(m) < 0$,

$$b = m \tag{2.37}$$

and the new midpoint is found from $m = (a + b)/2$. This method is applied until $|a - b|$ is small enough. The same example as for the secant solver yields

```
#Import the bisect function (implements bisection method in 1-D)
In [4]: from scipy.optimize import bisect

# bisect(function handle, a,b),
# http://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.bisect.html
In [5]: bisect(lambda x: x**2-0.4,0,4)
Out[5]: 0.6324555320334184

# The exact solution - should be the same.
# a**b in Python is a to the power of b
In [6]: 0.4**0.5
\\\\\\\\\\\\\\\\\\\\\\\\\\\\Out[6]: 0.6324555320336759
```

Brent's Method

Brent's method,[1] combines linear interpolation and quadratic interpolation with pure bisection if needed. It is computationally efficient, robust, and stable, and used all over the code whenever a solution interval is known. For well-behaved functions, the secant method is almost always faster, but for poorly behaved functions - like many in ACHP - Brent's method is required.

Python code given by:

```
def Brent(f,a,b,macheps,t):
    """
    Using the algorithm from

    Brent, R. P., Algorithms for Minimization Without Derivatives.
    Englewood Cliffs, NJ: Prentice-Hall, 1973. Ch. 3-4.
    """
    fa=f(a)
```

[1] Brent, R. P., Algorithms for Minimization Without Derivatives. Englewood Cliffs, NJ: Prentice-Hall, 1973. Ch. 3-4. Link to book

```
    fb=f(b)
    c=a
    fc=fa
    if abs(fc)<abs(fb):
        # Goto ext: from Brent ALGOL code
        a=b
        b=c
        c=a
        fa=fb
        fb=fc
        fc=fa
    d=b-a
    e=b-a
    m=0.5*(c-b)
    tol=2*macheps*abs(b)+t
    while (abs(m)>tol and fb!=0):
        #See if a bisection is forced
        if abs(e)<tol or abs(fa) <= abs(fb):
            m=0.5*(c-b)
            d=e=m
        else:
            s=fb/fa
            if a==c:
                #Linear interpolation
                p=2*m*s
                q=1-s
            else:
                #Inverse quadratic interpolation
                q=fa/fc
                r=fb/fc
                m=0.5*(c-b)
                p=s*(2*m*q*(q-r)-(b-a)*(r-1))
                q=(q-1)*(r-1)*(s-1)
            if p>0:
                q=-q
            else:
                p=-p
            s=e
            e=d
            m=0.5*(c-b)
            if 2*p<3*m*q-abs(tol*q) or p<abs(0.5*s*q):
                d=p/q
            else:
                m=0.5*(c-b)
                d=e=m
        a=b
        fa=fb
        if abs(d)>tol:
            b+=d
        elif m>0:
            b+=tol
        else:
            b+=-tol
        fb=f(b)
        if fb*fc>0:
            # Goto int: from Brent ALGOL code
            c=a
            fc=fa
```

```
            d=e=b-a
        if abs(fc)<abs(fb):
            # Goto ext: from Brent ALGOL code
            a=b
            b=c
            c=a
            fa=fb
            fb=fc
            fc=fa
        m=0.5*(c-b)
        tol=2*macheps*abs(b)+t
    return b
```

2.4.3 Multi-dimensional solver

At the cycle-solver level, and elsewhere, is is common that multiple equations must be driven to zero by changing multiple parameters. Beginning the consideration with the one-dimensional case, the Newton-Raphson method gives the solution for the next step from

$$x_{new} = x_{old} - \frac{f(x_{old})}{f'(x_{old})} \tag{2.38}$$

referring to the secant method above, you can see that the secant method is just the one-dimensional N-R method with a numerical approximation for the derivative.

In the multi-dimensional case, the problem to be solved is that to enforce the equality for the vector of nonlinear equations (within the convergence criterion), of

$$\mathbf{f}(\mathbf{x}) = 0 \tag{2.39}$$

where each of the functions $f_1, f_2, ...$ can be functions of inputs $x_1, x_2, ...$ The formulation of the problem in multidimensions is given by

$$\mathbf{x}_{new} = \mathbf{x}_{old} - \mathbf{J}'\mathbf{f} \tag{2.40}$$

where the Jacobian matrix $\mathbf{J}$ is given by

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{2.41}$$

and $\mathbf{J}'$ is the matrix inverse of $\mathbf{J}$. If the partial derivatives of the functions $f_1, f_2, ...$ are known, they can be used directly in the calculation of the Jacobian matrix. Otherwise, the Jacobian matix can be built with numeric derivatives. The easiest way to build the numerical derivatives in the Jacobian matrix is to build the matrix by column. If we call the vector of functional values at the iteration $\hat{\mathbf{f}}$, each column is obtained by the formula

$$\mathbf{J}_k = \frac{\partial \mathbf{f}}{\partial x_k} = \frac{\mathbf{f}(x_1, x_2, ..x_k + \delta x, ..., x_n) - \hat{\mathbf{f}}}{\delta x} \tag{2.42}$$

which forms the k-th column of $\mathbf{J}$. Only $n+1$ functional calls are required. Scientific Python includes a slightly more advanced version of this algorithm with Jacobian updating, but the basic idea remains the same. An example of this

method is the set of equations

$$\begin{aligned} x^2 - 2y - 2 &= 0 \\ x + y^2 - 1 &= 0 \end{aligned} \tag{2.43}$$

which requires the python code

```
#Import the fsolve function (implements N-R multi-dimensional solve)
In [7]: from scipy.optimize import fsolve

# create an inline lambda function to return the values of the functions
In [8]: func=lambda x: [x[0]**2-2*x[1]-2, x[0]+x[1]**2-1]

#Actually solve the function
In [9]: x=fsolve(func,[0,0]); print(x)
[  2.04263526e-13  -1.00000000e+00]

#Verify you have the right solution
In [10]: func(x)
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\Out[10]: [-2.4202861936828413e-13, -3.
↪7747582837255322e-14]
```

2.5 Frequently Asked Questions

2.5.1 Working with the Python Path (aka *Argh!! Why can't Python find my files?*)

One of the biggest hurdles for anyone getting started with Python is dealing with the python path. (Author's note: I *still* struggle with this!) Python searches a number of locations when it is looking for a module from an import command. In a script, if you write:

```
from __future__ import print_function
import sys
print(sys.path)
```

it will tell you all the places that Python is looking for your files. From IPython, you might get an output something like this for the first three entries:

```
In [1]: import sys; print(sys.path[0:3])
['/home/docs/checkouts/readthedocs.org/user_builds/achp/conda/latest/bin', '/home/
↪docs/checkouts/readthedocs.org/user_builds/achp/conda/latest/lib/python35.zip', '/
↪home/docs/checkouts/readthedocs.org/user_builds/achp/conda/latest/lib/python3.5']
```

If you have multiple copies of a module, it will use the first one it finds as it searches from left to right in the list of folders.

Any modules that you have installed using either the:

```
python setup.py install
```

command or with an installer executable should be sitting in the Python folder tree, and should ideally not require any further work from you. CoolProp, a required package for ACHP for instance, is installed into the:

```
C:\\Python26\\Lib\\site-packages\\CoolProp
```

folder with Python 2.6.x. And if you open an ipython prompt and type

```
In [2]: import CoolProp; print(CoolProp.__file__)
/home/docs/checkouts/readthedocs.org/user_builds/achp/conda/latest/lib/python3.5/site-
↪packages/CoolProp/__init__.py
```

no errors should be generated, and it will spit out the path to the module files.

Updating Python Path

If you have code in a file that Python cannot find right now, you have a few options:

1. Move the files to somewhere that Python CAN find (i.e. on the python path)
2. Change the PYTHONPATH environmental variable
3. Add the path to sys.path

In general, for most code my preferred method is to update the PYTHONPATH environmental variable for code that I am likely to reference from a number of other scripts (i.e. the ACHP code). To do this in Windows, click on the windows symbol in the taskbar (the old start button), right click on Computer and go to properties, then advanced system settings, the Advanced tab, and then Environment Variables. In the sytem variables window, there are two lists of environmental variables, user variables and system variables. I tend to add the path to the folder that the files are in to both lists just to be sure. It is not clear how python decides which list to use. The list of folders is semicolon delimited. If the environmental variable PYTHONPATH does not exist, just create a new environmental variable.

The nice part of using the PYTHONPATH solution is that you only have to do it once, and then any subsequent times you want the code, python can find it.

If on the other hand you want to just use the sys.path method, to add a path to the python path, just do something like:

```
import sys
sys.path.append('c:\path\to\folder')
```

which will add the file path to the system path for the current execution of code.

CHAPTER 3

ACHP Component Models

3.1 Compressor

3.1.1 Overview

A compressor is the heart of the air-conditioning or refrigeration system, compressing the refrigerant from the low-pressure side of the system up to the high-pressure side of the system. Because it plays such a a significant role in the overall system efficiency, the accuracy of the compressor model is quite important.

3.1.2 Mathematical Description

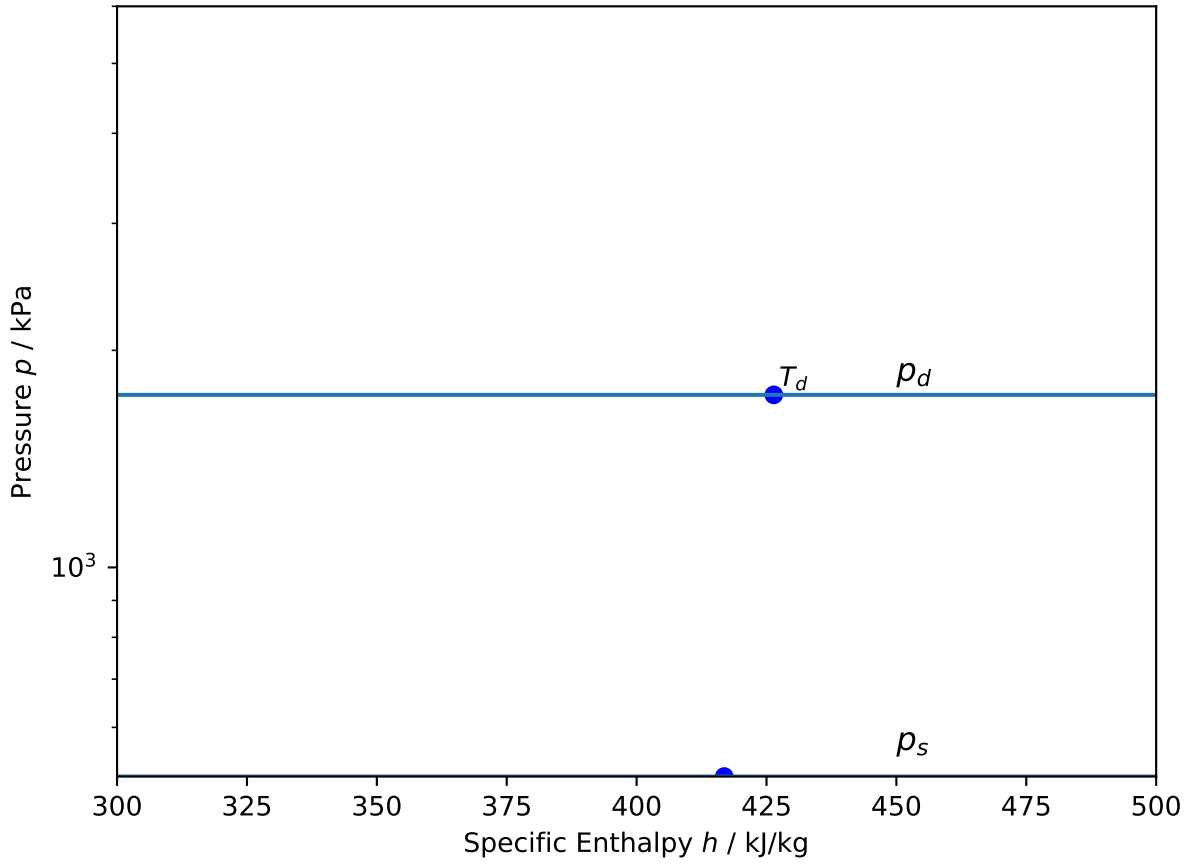
The compressor is modeled based on a 10-coefficient ARI compressor map which is very commonly used to characterize the performance of compressors. The map is based on a given amount of superheat along with input saturated suction and discharge pressures. Though most everything else in the program is based on metric units, the standard in America is to generate the map based on superheat and saturated temperatures in degrees Fahrenheit. Refer to the following figure to see the definitions of the temperatures and pressures:

Thus the map-based mass flow rate (in lbm/hr) and electrical power (in W), can be given by

$$\dot{m}_{map} = M_1 + M_2 T_s + M_3 T_d + M_4 T_s^2 + M_5 T_s T_d + M_6 T_d^2 + M_7 T_s^3 + M_8 T_d T_s^2 + M_9 T_d^2 T_s + M_{10} T_d^3 \quad (3.1)$$

$$\dot{W}_{map} = P_1 + P_2 T_s + P_3 T_d + P_4 T_s^2 + P_5 T_s T_d + P_6 T_d^2 + P_7 T_s^3 + P_8 T_d T_s^2 + P_9 T_d^2 T_s + P_{10} T_d^3 \quad (3.2)$$

where the saturated suction dewpoint temperature T_s and saturated discharge dewpoint temperatures T_d are in degrees Fahrenheit. To be specific, they are the dew temperatures, which are the same as the saturated vapor temperatures for pure fluids. The coefficients $M_1, M_2, ...$ are the mass flow map coefficients and $P_1, P_2, ...$ are the electrical power map coefficients. In practice, the compressor is unlikely to operate at exactly the map superheat. As a result, the map predictions must be corrected to better match the actual operating conditions. The map correction is based on the


Pressure p / kPa
10^3
T_d
p_d
p_s
300
325
350
375
400
425
450
475
500
Specific Enthalpy h / kJ/kg

method of Rice. et al.[1] , which yields

$$\dot{m}_{actual} = \left[1 + 0.75\left(\frac{v_{map}}{v_{actual}} - 1\right)\right]\dot{m}_{map} \tag{3.3}$$

where the subscripts *actual* refer to the properties evaluated at the actual superheat at the suction flange, and the *map* subscripts refer to the properties evaluated at the given map superheat. Similarly, the electrical power correction is given by

$$\dot{W}_{actual} = \dot{W}_{map}\frac{\dot{m}_{actual}}{\dot{m}_{map}}\frac{h_{2s,actual} - h_{1,actual}}{h_{2s,map} - h_{1,map}} \tag{3.4}$$

An energy balance over the compressor yields

$$\dot{W}_{actual} + \dot{Q}_{amb} + \dot{m}_{actual}(h_{1,actual} - h_{2,actual}) = 0 \tag{3.5}$$

Since the electrical power is known from the corrected map, and the heat transfer can be expressed as a fraction of the electrical power by

$$\dot{Q}_{amb} = -f_p\dot{W}_{actual} \tag{3.6}$$

the outlet enthalpy of the compressor can be therefore given by

$$h_{2,actual} = \frac{\dot{W}_{actual}(1 - f_p)}{\dot{m}_{actual}} + h_{1,actual} \tag{3.7}$$

Nomenclature

Variable	Description
f_p	Fraction of electrical power lost at heat transfer [-]
$h_{1,actual}$	Enthalpy of refrigerant at actual superheat [J/kg]
$h_{2s,actual}$	Isentropic enthalpy of refrigerant at discharge pressure using actual superheat [J/kg]
$h_{1,map}$	Enthalpy of refrigerant at map superheat [J/kg]
$h_{2s,map}$	Isentropic enthalpy of refrigerant at discharge pressure using map superheat [J/kg]
$\dot{m}_{actual}$	Actual refrigerant mass flow rate [kg/s]
$\dot{m}_{map}$	Refrigerant mass flow rate from map [lb_m/hr]
$M_1, M_2, ...$	Mass flow map coefficients [varied]
$P_1, P_2, ...$	Electrical power map coefficients [varied]
p_d	Discharge dew pressure [Pa (absolute)]
p_s	Suction dew pressure [Pa (absolute)]
$\dot{Q}_{amb}$	Ambient heat loss [W]
T_d	Discharge dew temperature [°F]
T_s	Suction dew temperature [°F]
v_{actual}	Specific volume of refrigerant at actual superheat [m^3/kg]
v_{map}	Specific volume of refrigerant at map superheat [m^3/kg]
$\dot{W}_{actual}$	Actual compressor electrical power [W]
$\dot{W}_{map}$	Compressor electrical power from map [W]

3.1.3 Compressor Sample Code

Minimal Component Test:

[1] Rice, C. K. and A. E. Dabiri, 1981. "A Compressor Simulation Model with Corrections for the Level of Suction Gas Superheat," ASHRAE Transactions, Vol. 87, Part 2, pp.771-782.

```
from __future__ import print_function

from ACHP.Compressor import CompressorClass
from CoolProp.CoolProp import PropsSI
kwds={
      'M':[217.3163128,5.094492028,-0.593170311,4.38E-02,
        -2.14E-02,1.04E-02,7.90E-05,-5.73E-05,1.79E-04,-8.08E-05],
      'P':[-561.3615705,-15.62601841,46.92506685,-0.217949552,
        0.435062616,-0.442400826,2.25E-04,2.37E-03,-3.32E-03,2.50E-03],
      'Ref':'R134a',
      'Tin_r':280,
      'pin_r':PropsSI('P','T',279,'Q',1,'R134a'),
      'pout_r':PropsSI('P','T',315,'Q',1,'R134a'),
      'fp':0.15, #Fraction of electrical power lost as heat to ambient
      'Vdot_ratio': 1.0 #Displacement Scale factor
      }
Comp=CompressorClass(**kwds)
Comp.Calculate()

print('Electrical power is: ' + str(Comp.W) + ' W')
print('Actual mass flow rate is: ' + str(Comp.mdot_r) + ' kg/s')
print('Isentropic Efficiency is: ' + str(Comp.eta_oi))
print('Discharge Refrigerant Temperature is: ' + str(Comp.Tout_r) + ' K')
```

If you open an IPython shell in the root of the documentation (folder `Documentation/Web` relative to the main trunk), and run the commands below, you should get something like

```
In [1]: %run 'ACHPComponents/ComponentTests/CompressorTest.py'
Electrical power is: 2211.3198584152465 W
Actual mass flow rate is: 0.059501681290018996 kg/s
Isentropic Efficiency is: 0.610793680410131
Discharge Refrigerant Temperature is: 327.76612904368125 K
```

If not, first stop should be the *Frequently Asked Questions*

3.1.4 Component Class Documentation

class ACHP.Compressor.**CompressorClass**(***kwargs*)

Compressor Model based on 10-coefficient Model from ANSI/AHRI standard 540

Required Parameters:

Variable	Units	Description
M	Ibm/hr	A numpy-like list of compressor map coefficients for mass flow
P	Watts	A numpy-like list of compressor map coefficients for electrical power
Ref	N/A	A string representing the refrigerant
Tin_r	K	Refrigerant inlet temperature
pin_r	Pa	Refrigerant suction pressure (absolute)
pout_r	Pa	Refrigerant discharge pressure (absolute)
fp	–	Fraction of electrical power lost as heat to ambient
Vdot_ratio	–	Displacement Scale factor

All variables are of double-type unless otherwise specified

Calculate()

OutputList()
Return a list of parameters for this component for further output

It is a list of tuples, and each tuple is formed of items with indices: [0] Description of value

[1] Units of value

[2] The value itself

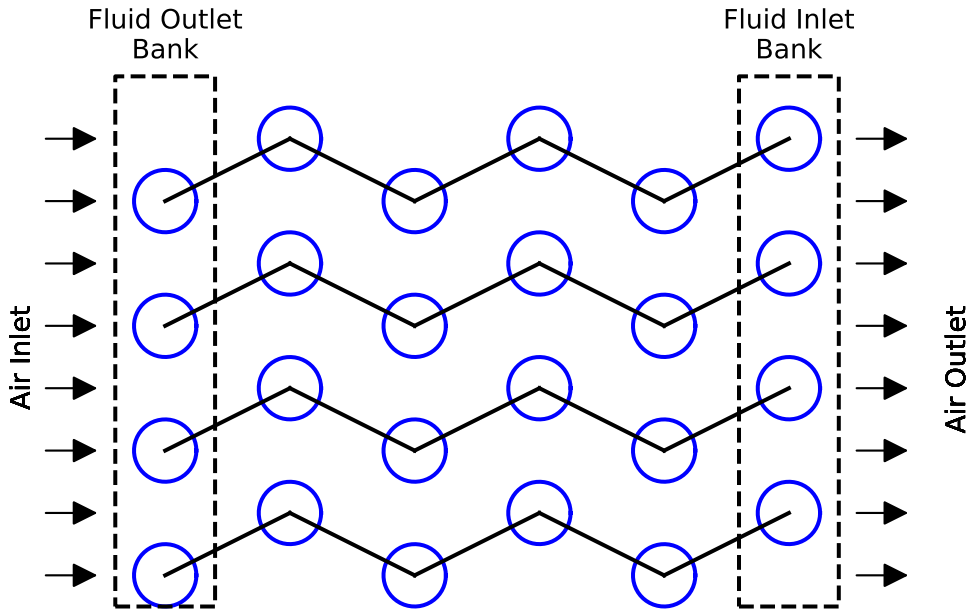
Update(***kwargs*)

3.2 Fin-Tube Heat Exchangers

3.2.1 Overview

In ACHP, all of the heat exchangers that exchange heat with an air stream are of the fin-tube type. In practice, other types of heat exchangers (shell-tube, microchannel, etc.) are also possible, but they are not included in ACHP as of the development of this documentation.

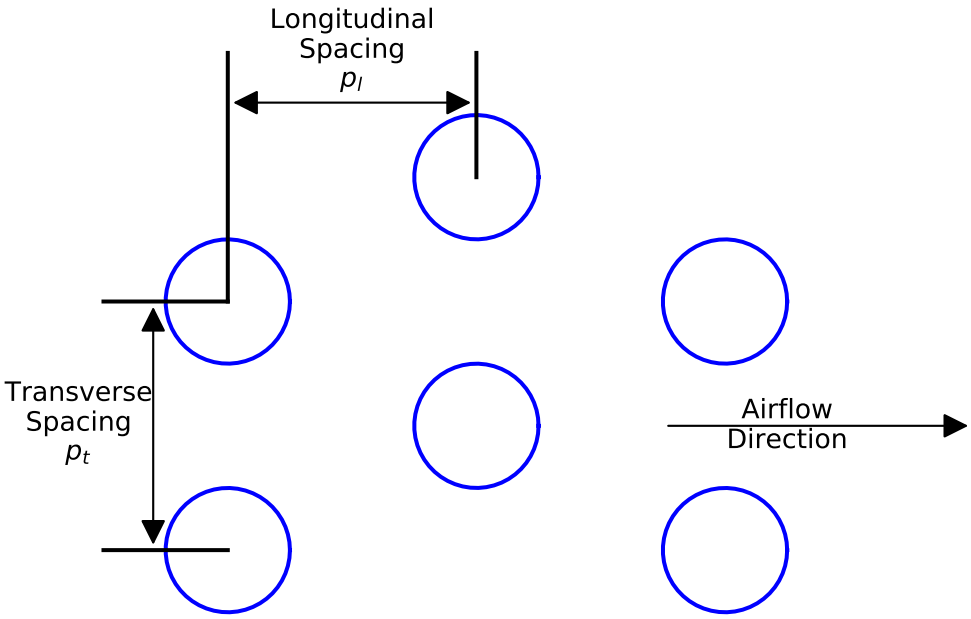
3.2.2 Mathematical Description



There is some disagreement in literature as to the best way to describe the tube layouts. The nomenclature used here is that there are a number of banks of tubes, where the banks are the vertical columns of tubes when viewed end-on

with the air flow passing from left to right. The above heat exchanger has a total of 6 banks of tubes, with 4 tubes per bank; there are 4 refrigerant circuits.

There are then a certain number of tubes per bank which form the core of the heat exchanger (not considering the circuiting). The following figure defines the terms which describe the heat exchanger core, including the longitudinal spacing (also sometimes called bank-to-bank spacing),



The empirical correlations for air-side heat transfer and pressure drop of fin-tube heat exchangers can be found in section *Air-Side Correlations and Other Calculations*, and the fluid-side correlations can be found in *Fluid Correlations and Other Calculations*.

In order to use the heat exchanger in ACHP, complexities of circuiting are neglected. In practice, it is uncommon for the number of tubes per bank to be divisible by the number of circuits. As a result, some circuits can be longer than others. For the purposes of ACHP, average circuit lengths are employed. The average circuit length is given by

$$\begin{aligned} L_{tubes,total} &= N_{tubes/bank} N_{bank} L_{tube} \\ \overline{L_{circuit}} &= L_{tubes,total}/N_{circuits} \end{aligned} \tag{3.8}$$

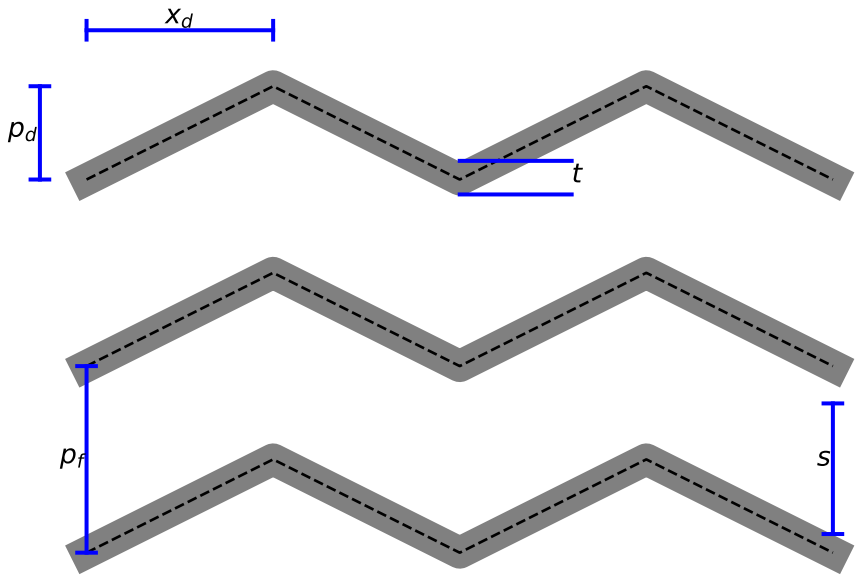
This average circuit length is primarily required for the calculation of the fluid-side pressure drop. Other parameters that are required are the total refrigerant-side volume $V_{r,total}$, which can be given by

$$V_{r,total} = L_{tubes,total} \frac{\pi D_i^2}{4} \tag{3.9}$$

where $\pi D_i^2/4$ is the internal cross-sectional area of the tube. The refrigerant side surface area is given by

$$A_{r,total} = \pi D_i L_{tubes,total} \tag{3.10}$$

3.2.3 Air-Side Correlations and Other Calculations



Geometric Parameters

Fins per meter [1/m]

$$FPM = FPI/0.0254 \tag{3.11}$$

Fin pitch (distance between centerlines of fins)

$$p_f = 1/FPM \tag{3.12}$$

Spacing between fins

$$s = 1/FPM - t \tag{3.13}$$

Height of heat exchanger [m], assuming that fin extends 1/2 P_t above/below last tube in bundle

$$H = P_t(N_{tubes/bank} + 1) \tag{3.14}$$

A_{duct} is the face area [m^2] equivalent to the duct cross-section

$$A_{duct} = HL_{tube} \tag{3.15}$$

Number of fins in the tube sheet [-]

$$N_{fin} = L_{tube} FPM \tag{3.16}$$

Secant of theta is the area enhancement factor [-]. It captures the increase in area due to the waviness of the fins

$$\sec\theta = \frac{\sqrt{x_f^2 + p_d^2}}{x_f} \tag{3.17}$$

Duct cross-sectional area that is not fin or tube [m^2]

$$A_c = A_{duct} - tN_{fin}(H - DN_{tubes/bank}) - N_{tubes/bank} DL_{tube} \tag{3.18}$$

Total outer area of the tubes [m^2]

$$A_{tube} = N_{tubes/bank} N_{bank} \pi D L_{tube} \tag{3.19}$$

Wetted Area of a single fin [m^2], assuming that fin extends 1/2 P_t in front/after last tube in bundle

$$A_{1fin} = 2(HP_l(N_{bank} + 1)\sec\theta - N_{tubes/bank} N_{bank} \pi D^2/4) \tag{3.20}$$

Total wetted area of the fins [m^2]

$$A_f = N_{fin} A_{1fin} \tag{3.21}$$

Total air-side area including tube and fins [m^2]

$$A_{a,total} = A_f + N_{tubes/bank} N_{bank} \pi D(L_{tube} - N_{fin} t) \tag{3.22}$$

Heat Transfer and Pressure Drop Parameters

Evaluate the mass flow rate based on inlet conditions

$$\rho_{ha} = \frac{1 + W}{v_{ha}} \tag{3.23}$$

$$\begin{aligned} \dot{m}_{ha} &= \dot{V}_{ha} \rho_{ha} \\ u_{max} &= \frac{\dot{m}_{ha}}{\rho_{ha} A_c} \end{aligned} \tag{3.24}$$

Specific heat, thermal conductivity and viscosity based on humid air property correlations.

$$\mathrm{Pr} = \frac{c_{p,ha} \mu_{ha}}{k_{ha}} \tag{3.25}$$

Reynolds number based on the tube diameter:

$$\mathrm{Re}_D = \frac{\rho_{ha} u_{max} D}{\mu_{ha}} \tag{3.26}$$

Empirical Overall Correlations

Wavy-Louvered fins from Wang,Tsai,Lu[3]:

Colburn j-Factor:

$$j = 16.06\mathrm{Re}_D^{-1.02(p_f/D)-0.256}\left(\frac{A_{a,total}}{A_{tube}}\right)^{-0.601} N_{bank}^{-0.069}\left(\frac{p_f}{D}\right)^{0.84} \tag{3.27}$$

Air-side mean heat transfer coefficient:

$$\alpha_a = \frac{j\rho_{ha}u_{max}c_{p,a}}{\mathrm{Pr}^{2/3}} \tag{3.28}$$

Air-side pressure drop friction factor:

$$f_{a,total} = 0.264(0.105 + 0.708\exp(-\mathrm{Re}_D/225.0))\mathrm{Re}_D^{-0.637}\left(\frac{A_{a,total}}{A_{tube}}\right)^{0.263}\left(\frac{p_f}{D}\right)^{-0.317} (\mathrm{Re}_D < 1000)$$
$$f_{a,total} = 0.768(0.0494 + 0.142\exp(-\mathrm{Re}_D/1180.0))\left(\frac{A_{a,total}}{A_{tube}}\right)^{0.0195}\left(\frac{p_f}{D}\right)^{-0.121} (\mathrm{Re}_D \geq 1000) \tag{3.29}$$

The air mass flux through the heat exchanger can be defined by

$$G_c = \frac{\dot{m}_{ha}}{A_c} \tag{3.30}$$

which yields the air-side pressure drop (neglecting entrance and exit pressure drops) of

$$\Delta p_a = \frac{A}{A_{tube}}\frac{G_c^2}{2\rho_{ha}}f_{a,total} \tag{3.31}$$

Surface Efficiency

Thermal gradients in the fins result in a finned surface that does not perform as efficiently as if the entire finned surface was at the fluid temperature passing through the tubes. Correlation (or explicit solution where possible) is required to be used to determine the temperature profile in the fins, and from that, the finned surface efficiency.

Circular fins

Schmidt[4] developed a correction term for circular fins to calculate fin efficiency. Using the circular fin correlation of Schmidt yields the solution for the finned surface efficiency of

$$\phi = \left(\frac{r_f}{r} - 1\right)\left[1 + 0.35\ln\left(\frac{r_f}{r}\right)\right]$$
$$\eta_f = \frac{\tanh(mr\phi)}{mr\phi} \tag{3.32}$$

where r is the outer diameter of the tube, and r_f is the outer radius of the circular fin. The parameter m is given by

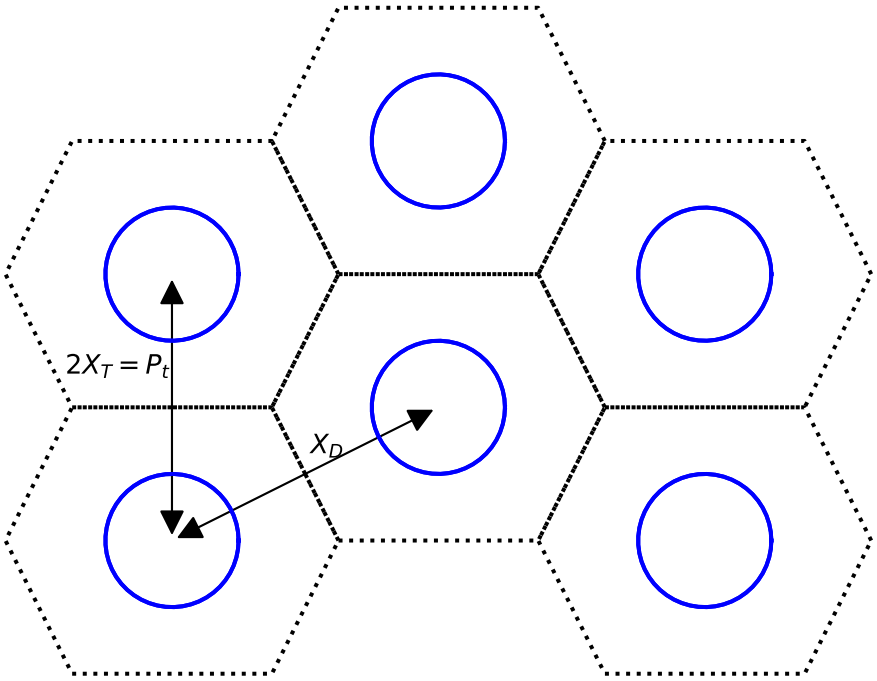
$$m = \sqrt{\frac{2\alpha_a(c_s/c_p)}{k_{fin}t}} \tag{3.33}$$

[3] Chi-Chuan Wang and Yu-Min Tsai and Ding-Chong Lu, 1998, "Comprehensive Study of Convex-Louver and Wavy Fin-and-Tube Heat Exchangers", *Journal of Thermophysics and Heat Transfer*

[4] Schmidt T.E. 1945-46. La Production Calorifique des Surfaces Munies D'ailettes. *Bulletin De L'Institut International Du Froid Annexe G-5*.

Staggered tubes

For the staggered tube bank configurations (like that in ACHP), hexagonal cells with adiabatic boundaries are formed around each tube, and for each cell the fin efficiency can be obtained. Mathematical analysis is used to convert this problem into that like the circular fin.



$$\begin{aligned} r &= \frac{D}{2} \\ X_D &= \frac{\sqrt{P_l^2 + P_t^2/4}}{2} \\ X_T &= \frac{P_t}{2} \end{aligned} \tag{3.34}$$

The effective radius ratio is given by

$$\frac{r_f}{r} = 1.27\frac{X_T}{r}\sqrt{\frac{X_D}{X_T} - 0.3} \tag{3.35}$$

And the m factor is given by

$$m = \sqrt{\frac{2\alpha_a(c_s/c_p)}{k_{fin}t}} \tag{3.36}$$

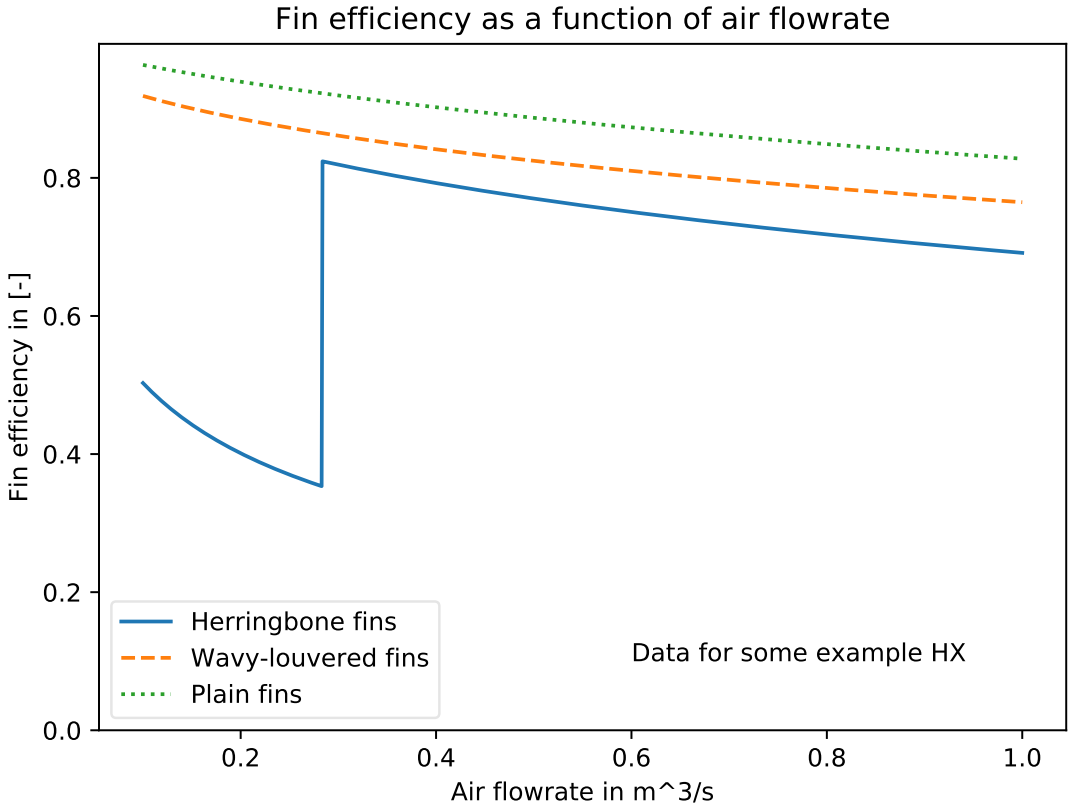
c_s/c_p is the correction for heat/mass transfer for a wetted surface. If the fins are dry, this parameter is set to unity, which yields the standard definition for the parameter m for a fin

Fin efficiency based on analysis by Hong & Webb[2] and Perrotin & CLODIC[1] for wet and dry fins with staggered fins

$$\phi = \left(\frac{r_f}{r} - 1\right)\left[1 + \left(0.3 + \left(\frac{mr(\frac{r_f}{r} - r)}{2.5}\right)^{1.5-\frac{1}{12}\frac{r_f}{r}}\left(0.26\left(\frac{r_f}{r}\right)^{0.3} - 0.3\right)\right)\ln\left(\frac{r_f}{r}\right)\right] \tag{3.37}$$

$$\eta_f = \frac{\tanh(mr\phi)}{mr\phi}\cos(0.1mr\phi) \tag{3.38}$$

Plots



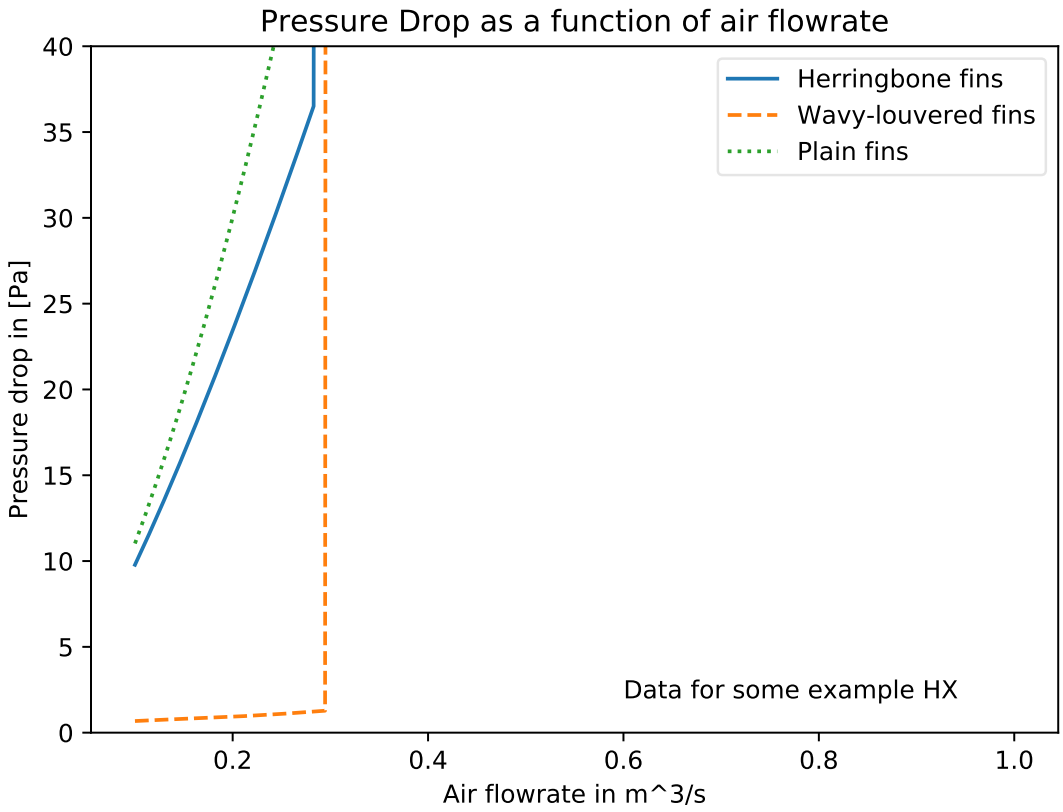
Overall efficiency

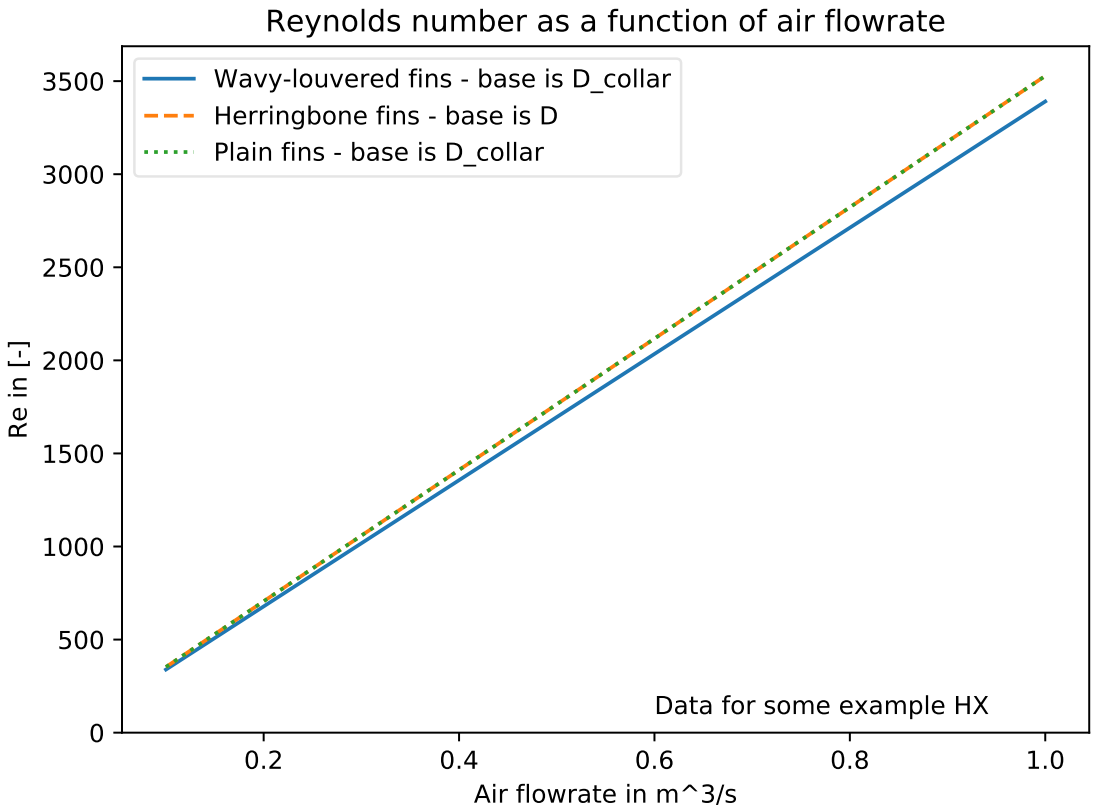
Once the finned surface efficiency (η_f) is known, the overall surface efficiency is given by

$$\eta_a = 1 - \frac{A_f}{A_{a,total}}(1 - \eta_f) \tag{3.39}$$

[2] Kwang Taek Hong and Ralph L. Webb, 1996, "Calculation of Fin Efficiency for Wet and Dry Fins" *HVAC&R Research*,v. 2 Link to File

[1] Thomas Perrotin, Denis Clodic, 2003, "Fin Efficiency Calculation in Enhanced Fin-and-Tube Heat Exchangers in Dry Conditions", *International Congress of Refrigeration*


Pressure Drop as a function of air flowrate
Herringbone fins
Wavy-louvered fins
Plain fins
Data for some example HX
Pressure drop in [Pa]
Air flowrate in m^3/s
0
5
10
15
20
25
30
35
40
0.2
0.4
0.6
0.8
1.0


Reynolds number as a function of air flowrate
Wavy-louvered fins - base is D_collar
Herringbone fins - base is D
Plain fins - base is D_collar
Re in [-]
3500
3000
2500
2000
1500
1000
500
0
0.2
0.4
0.6
0.8
1.0
Air flowrate in m^3/s
Data for some example HX

Nomenclature

Variable	Description
$A_{a,total}$	Total air-side area (fins+tubes) [m^2]
A_f	Air-side area of fins [m^2]
A_{1fin}	Air-side area of 1 fin [m^2]
A_c	Cross-sectional area of duct not filled with fins and tubes [m^2]
$A_{r,total}$	Surface area on the refrigerant (or fluid) side [m^2]
A_{duct}	Duct cross-sectional area [m^2]
A_{tube}	Cross-sectional area of the tubes [m^2]
$c_{p,a}$	Air specific heat [J/kg_{ha}/K]
D_i	Interior diameter of tubes [m]
G_c	Air mass flux [kg/m^2]
$f_{a,total}$	Air-side friction factor [-]
FPI	Fins per inch [1/in]
FPM	Fins per meter [1/m]
k_{fin}	Fin conductivity [W/m/K]
H	Height [m]
j	Colburn j-factor [-]
L_{tube}	Length of one tube [m]
$L_{tubes,total}$	Sum total length of all tubes [m]
$L_{circuit}$	Average length of circuit [m]
m	Non-dimensional group in fin efficiency calculation [-]
$\dot{m}_{ha}$	Mass flow rate of humid air [kg/s]
$N_{tubes/bank}$	Number of tubes per bank [-]
N_{bank}	Number of banks of tubes [-]
$N_{circuits}$	Number of circuits [-]
N_{fin}	Number of fins [-]
p_f	Twice the amplitude of fins corrugation [m]
p_l	Longitudinal bank-bank pitch (in the flow direction)[m]
p_t	Transverse pitch (perpindicular in the flow direction)[m]
Δp_a	Air side pressure drop [Pa]
Pr	Prandtl number [-]
r	Outer radius of tube [m]
r_f	Circular fin radius [m]
Re_D	Reynolds number based on tube OD [-]
s	Spacing between fins [m]
$\sec\theta$	Area increase factor [-]
t	Fin thickness [m]
u_{max}	Maximum air velocity [m/s]
v_{ha}	Density of humid air [m^3/kg]
$\dot{V}_{ha}$	Volumetric flow rate of humid air [m^3/s]
$V_{r,total}$	Open volume on the refrigerant (or fluid) side [m^3]
W	Humidity ratio [-]
x_f	Half-wavelength of fin wave [m]
X_D	Diagonal distance in fin efficiency equation [m]
X_T	Transverse distance in fin efficiency equation [m]
α_a	Air-side mean heat transfer coefficient [W/m^2/K]
ρ_{ha}	Density of air [kg/m^3]
ϕ	Surface efficiency parameter [-]
η_a	Overall finned surface efficiency [-]

Continued on next page

Table 3.1 – continued from previous page

Variable	Description
η_f	Fin efficiency [-]

3.3 Heat Exchangers with and without Dehumidification

In the heat exchanger models for ACHP (for evaporator, condenser, cooling coil, etc.), air and a working fluid exchange heat. In each part of the heat exchanger model, the working fluid phase (single-phase, two-phase) is assumed to be known. Therefore, the given segment of the heat exchanger can be analyzed using the analysis which follows here. In addition, depending on the surface temperatures in the heat exchangers, there exists the possibility that water moisture in the air might condense on the finned surface of the heat exchanger, which must be properly addressed.

Essentially the heat exchanger models can be broken down into these four possibilities:

- Working fluid is two-phase, there is no possibility of condensation of air on coil (always in condenser, possibly in evaporator and cooling coil)
- Working fluid is single-phase, there is no possibility of condensation of air on coil (always in condenser, possibly in evaporator and cooling coil)
- Working fluid is two-phase, there is a possibility of condensation of air on coil (evaporator and cooling coil)
- Working fluid is single-phase, there is a possibility of condensation of air on coil (evaporator and cooling coil)

3.3.1 Heat Exchanger Analysis Fundamentals

The analysis that follows is heavily based on the $\varepsilon - \mathrm{Ntu}$ method of analysis of heat exchangers. This method assumes that the specific heat of both fluids are constant (or a reasonable average value can be found). The beauty of the $\varepsilon - \mathrm{Ntu}$ method is that if the heat transfer coefficients and areas can be obtained explicitly, and the inlet temperatures of both streams are known, the heat transfer rate can be obtained explictly. This proves highly beneficial to the development of computationally-efficient code. In many of the individual heat exchanger solvers, the equations are employed in a slightly different configuration, but the governing equations presented here describe the fundamental models.

Broadly, the way the $\varepsilon - \mathrm{Ntu}$ method works is to first find the heat transfer conductance UA from the heat transfer network analysis, which yields the number of thermal units (Ntu), which gives you the effectiveness ε, which allows for the calculation of everything else.

3.3.2 Analysis of fully-dry HX without Dehumidification

Two-phase on working fluid side

The combination of a two-phase working fluid and no possibility of moisture removal from the air yields the simplest analysis. The capacitance rate of air by definition is the limiting capacitance rate (specific heat of two-phase mixture is infinite), and therefore, the UA are given by

$$\begin{aligned} \mathrm{UA}_a &= \eta_a \alpha_a A_a \\ \mathrm{UA}_r &= \alpha_r A_r \end{aligned} \tag{3.40}$$

the overall heat transfrer conductance by

$$\frac{1}{\mathrm{UA}} = \frac{1}{1/\mathrm{UA}_a + 1/\mathrm{UA}_r} \tag{3.41}$$

and the Ntu from

$$\mathrm{Ntu} = \frac{\mathrm{UA}}{\dot{m}_a c_{p,a}} \tag{3.42}$$

The effectiveness is obtained from

$$\varepsilon = 1 - \exp(-\mathrm{Ntu}) \tag{3.43}$$

because the ratio of capacitance rates is zero ($C_r = 0$). Since the air-side is by definition the limiting capacitance rate, the heat transfer rate is given by

$$\dot{Q} = \varepsilon \dot{m}_a c_{p,a}(T_{h,i} - T_{c,i}) \tag{3.44}$$

where $T_{h,i}$ is the inlet temperature of the hot stream, and $T_{c,i}$ is the inlet temperature of the cold stream. By this definition, the sign of $\dot{Q}$ is defined to be positive.

Single-phase on working fluid side

When there does not exist the possibility that the humid air stream can condense, the analysis of pure-heat transfer is relatively simple. In the heat exchanger, the air will approach the working fluid intlet temperature, and the working fluid will exchange heat with the air, potentially, though not necessarily, bringing the temperature of the working fluid closer to that of the air.

The heat transfer conductances on air- and working fluid-sides are given by

$$\begin{aligned} \mathrm{UA}_a &= \eta_a \alpha_a A_a \\ \mathrm{UA}_w &= \alpha_w A_w \end{aligned} \tag{3.45}$$

where the heat transfer coefficients α and areas A are governed by the analysis in sections *Air-Side Correlations and Other Calculations*, *Fin-Tube Heat Exchangers*, and *Fluid Correlations and Other Calculations*. The overall heat transfer conductance UA can be obtained from

$$\frac{1}{\mathrm{UA}} = \frac{1}{1/\mathrm{UA}_a + 1/\mathrm{UA}_w} \tag{3.46}$$

Thus the number of thermal units can be determined by

$$\mathrm{Ntu} = \frac{\mathrm{UA}}{C_{min}} \tag{3.47}$$

where C_{min} is the minimum capacitance rate, given by

$$\begin{aligned} C_{min} &= \min[\dot{m}_a c_{p,a}, \dot{m}_w c_{p,w}] \\ C_{max} &= \max[\dot{m}_a c_{p,a}, \dot{m}_w c_{p,w}] \\ C_r &= \frac{C_{min}}{C_{max}} \end{aligned} \tag{3.48}$$

Thus with Ntu known, the effectiveness can be obtained based on the geometry of the heat exchanger, shown here

- Counterflow:

$$\varepsilon = \frac{1 - \exp[-\mathrm{Ntu}(1 - C_r)]}{1 - C_r \exp[-\mathrm{Ntu}(1 - C_r)]} \tag{3.49}$$

- Crossflow ($C_a < C_w$ [air-side capacitance rate is minimum and air is treated as being unmixed because the fins block mixing of the air and the fluid is treated as being mixed]):

$$\varepsilon = \frac{1}{C_r}(1 - \exp(-C_r(1 - \exp(-\mathrm{Ntu})))) \tag{3.50}$$

- Crossflow ($C_w < C_a$ [working-fluid-side capacitance rate is minimum and air is treated as being unmixed because the fins block mixing of the air and the fluid is treated as being mixed]):

$$\varepsilon = 1 - \exp\left(-\frac{1}{C_r}[1 - \exp(-C_r \mathrm{Ntu})]\right) \tag{3.51}$$

Which yields the heat transfer rate of

$$\dot{Q} = \varepsilon C_{min}(T_{h,i} - T_{c,i}) \tag{3.52}$$

3.3.3 Analysis of Partial-wet/Partial-dry Heat Exchangers with the possibility of Dehumidification

Two-phase on working fluid side

In the analysis here, averaged heat transfer coefficients on the air-side (from sections *Air-Side Correlations and Other Calculations* and *Fin-Tube Heat Exchangers*) and averaged heat transfer coefficients on the refrigerant side (from two-phase evaporation part of section *Fluid Correlations and Other Calculations*) are used. The slight wrinkle on the refrigerant side is that the average heat transfer coefficient is a function of refrigerant quality, but the analysis here assumes that the inlet and outlet qualities of the refrigerant are known (or are being iterated for).

Another complication is that for refrigerants that experience glide during phase change (azeotropic blends like R404a, R410a, R507c, R407a, etc.), the saturation temperature is not constant during evaporation. If the pressure during phase change is assumed to be constant, there are different temperatures for the saturated liquid (bubble point), and the saturated vapor (dew point) for a given saturation pressure. The analysis which follows for the two-phase section requires an average saturation temperature, which is simply defined to be $T_{sat,r} = (T_{bubble,r} + T_{dew,r})/2$ if the fluid has glide. Otherwise, the conventional definition of the saturation temperature $T_{sat,r}$ is used.

Fully-dry

The heat transfer conductances on inner- (working fluid), and outer-sides (air) are given by

$$\begin{aligned} \mathrm{UA}_o &= \alpha_a \eta_a A_{a,two-phase} \\ \mathrm{UA}_i &= \alpha_{r,two-phase} A_{r,two-phase} \end{aligned} \tag{3.53}$$

and the number of thermal units outside is given by

$$\mathrm{Ntu}_o = \frac{\mathrm{UA}_o}{\dot{m}_{a,two-phase} c_{p,a}} \tag{3.54}$$

In the fully dry analysis for the two-phase section, the UA value is obtained from

$$\mathrm{UA}_{dry} = \frac{1}{\mathrm{UA}_i^{-1} + \mathrm{UA}_o^{-1}} \tag{3.55}$$

which, since C_{min} is on the air-side (because the refrigerant is changing phase), yields

$$\mathrm{Ntu}_{dry} = \frac{\mathrm{UA}_{dry}}{\dot{m}_{a,two-phase} c_{p,a}} \tag{3.56}$$

and the effectiveness

$$\varepsilon_{dry} = 1 - \exp(-\mathrm{Ntu}_{dry}) \tag{3.57}$$

which yields the fully-dry heat transfer rate of

$$\dot{Q}_{dry} = \varepsilon_{dry} \dot{m}_{a,two-phase} c_{p,a}(T_{a,i} - T_{sat,r}) \tag{3.58}$$

which allows to calculate the inlet and outlet surface temperatures from

$$\begin{aligned} T_{s,i} &= \frac{\mathrm{UA}_o T_{a,i} + \mathrm{UA}_i T_{sat,r}}{\mathrm{UA}_o + \mathrm{UA}_i} \\ T_{s,o} &= \frac{\mathrm{UA}_o T_{a,o} + \mathrm{UA}_i T_{sat,r}}{\mathrm{UA}_o + \mathrm{UA}_i} \end{aligned} \tag{3.59}$$

For a given inlet air dewpoint temperature of T_{dp}, if

- $T_{s,o}$ is above T_{dp}, the coil is fully dry
- $T_{s,i}$ is below T_{dp}, the coil is fully wet, and proceed to the analysis in the section *Fully-wet*
- T_{dp} is between $T_{s,i}$ and $T_{s,o}$, the coil is partially wet, and proceed to the analysis in the section *Partially-dry*.

If the coil is fully dry, then the factor f_{dry} is equal to unity, and the outlet air enthalpy can be calculated from

$$h_{a,o} = h_{a,i} - \dot{Q}_{dry}/\dot{m}_{a,two-phase} \tag{3.60}$$

And the heat transfer rate for the two-phase section of the evaporator for the given set of $x_{o,two-phase}$ and $w_{two-phase}$ yields

$$\dot{Q}_{two-phase}(w_{two-phase}, x_{o,two-phase}) = \dot{Q}_{dry} \tag{3.61}$$

Fully-wet

In the fully-wet analysis, the fully-dry section of the two-phase portion of the evaporator doesn't exist. Therefore, the analysis which follows for the partially-wet/ partially-dry case can be employed directly with the inlet to the wet section given by the evaporator air inlet condition. This can be expressed as

$$\begin{aligned} h_{a,x} &= h_{a,i} \\ T_{a,x} &= T_{a,i} \\ f_{dry} &= 0 \end{aligned} \tag{3.62}$$

Partially-dry

In the partially-wet/ partially-dry analysis for a heat exchanger with two-phase working fluid, the configuration for the coil is described by this figure:

To begin the analysis for the dry part (this analysis is skipped if the coil is fully wetted), the temperature $T_{a,x}$ can be obtained directly from an energy balance at the point where the dewpoint is reached (though its location is not known *a priori*).

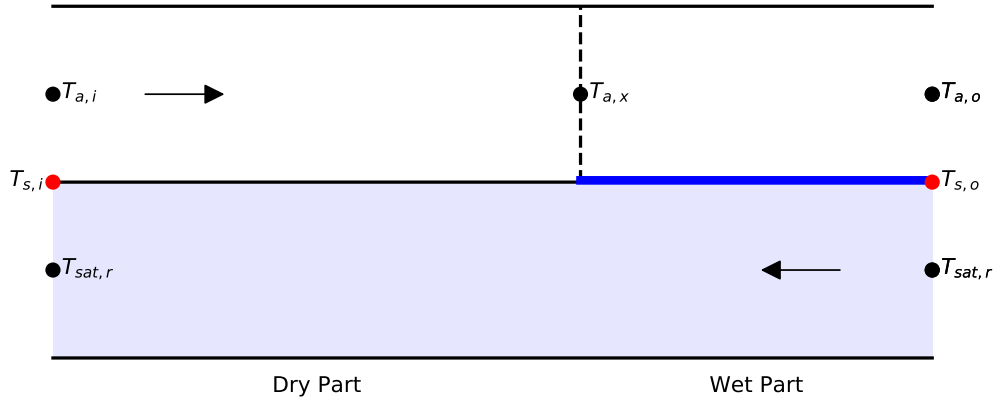
$$T_{a,x} = T_{dp} + \frac{\mathrm{UA}_i}{\mathrm{UA}_o}(T_{dp} - T_{sat,r}) \tag{3.63}$$

where the ratio of UA factors can be obtained from

$$\frac{\mathrm{UA}_i}{\mathrm{UA}_o} = \frac{\alpha_r A_{r,two-phase}}{\eta_a \alpha_a A_{a,two-phase}} \tag{3.64}$$

which yields an explicit solution for the effectiveness (because the inlet and outlet states are known on the air-side) ε_{dry} of

$$\varepsilon_{dry} = \frac{T_{a,i} - T_{a,x}}{T_{a,i} - T_{sat,r}} \tag{3.65}$$



and the f_{dry} factor is obtained from

$$f_{dry} = -\frac{1}{\mathrm{Ntu}_{dry}} \ln(1 - \varepsilon_{dry}) \tag{3.66}$$

by solving the equation

$$\varepsilon_{dry} = 1 - \exp(-f_{dry}\mathrm{Ntu}_{dry}) \tag{3.67}$$

The enthalpy of the air at the temperature $T_{a,x}$ can be obtained from

$$h_{a,x} = h_{air}(T = T_{a,x}, \omega = \omega_{a,i}) \tag{3.68}$$

which yields the heat transfer rate in the dry part of

$$\dot{Q}_{dry} = \dot{m}_{a,two-phase} c_{p,a} (T_{a,i} - T_{a,x}) \tag{3.69}$$

This completes the analysis for the dry part of the partially-wet/ partially-dry analysis. The wet portion begins with the calculation of the factor c_s, which is governed by the equation

$$c_s = \left.\frac{\partial h_{sat}}{\partial T}\right|_{T=T_{sat,r}} \tag{3.70}$$

Using this value for c_s, a new wetted-fin-efficiency (η_a^*) is calculated using the analysis in section *Staggered fin surface efficiency*. This then yields the UA values

$$\begin{aligned} \mathrm{UA}_i &= \alpha_r A_{r,two-phase} \\ \mathrm{UA}_o^* &= \eta_a^* \alpha_a A_{a,two-phase} \end{aligned} \tag{3.71}$$

and the overall wet UA value of

$$\mathrm{UA}_{wet} = \frac{1}{c_s/\mathrm{UA}_i + c_{p,a}/\mathrm{UA}_o^*} \tag{3.72}$$

which yields the Ntu value of

$$\mathrm{Ntu}_{wet} = \frac{\mathrm{UA}_{wet}}{\dot{m}_{a,two-phase}} \tag{3.73}$$

and the wet surface effectiveness of

$$\varepsilon_{wet} = 1 - \exp[-(1 - f_{dry})\mathrm{Ntu}_{wet}] \tag{3.74}$$

With the air saturation enthalpy $h_{a,sat,r}$ at the refrigerant saturation temperature $T_{sat,r}$ of

$$h_{a,sat,r} = h_{sat}(T = T_{sat,r}, \phi = 1) \tag{3.75}$$

the heat transfer rate in the wet section is given by

$$\dot{Q}_{wet} = \varepsilon_{wet}\dot{m}_{a,two-phase}(h_{a,i} - h_{a,sat,r}) \tag{3.76}$$

and the outlet enthalpy can be given by

$$h_{a,o} = h_{a,x} - \dot{Q}_{wet}/\dot{m}_{a,two-phase} \tag{3.77}$$

The air in the wet section interacts with a surface that has an effective surface enthalpy of

$$h_{a,s,s,e} = h_{a,x} - \frac{h_{a,x} - h_{a,o}}{1 - \exp[-(1 - f_{dry})\mathrm{Ntu}_o]} \tag{3.78}$$

which is obtained by considering just the effectiveness of humid-air mass transfer with a fixed-enthalpy $h_{a,s,s,e}$ saturated surface. Which yields an effective surface temperature $T_{s,e}$

$$T_{s,e} = T_{air}(h = h_{a,s,s,e}, \phi = 1) \tag{3.79}$$

This yields the air outlet temperature of

$$T_{a,o} = T_{s,e} + (T_{a,x} - T_{s,e})\exp[-(1 - f_{dry})\mathrm{Ntu_o}] \tag{3.80}$$

where the air is assumed to transfer heat with an isothermal surface at $T_{s,e}$. The total heat transfer rate in the partially-wet/ partially-dry analysis is then given from

$$\dot{Q}_{two-phase} = \dot{Q}_{dry} + \dot{Q}_{wet} \tag{3.81}$$

where $\dot{Q}_{dry}$ comes from Equation (3.69), and $\dot{Q}_{wet}$ comes from Equation (3.76).

Single-phase on working fluid side

Fully Dry Analysis

For the fully dry coil with a single-phase working fluid, (taken to be refrigerant here for the purposes of nomenclature) the internal and external Ntu for the dry analysis are given by

$$\begin{aligned} \mathrm{UA}_i &= \alpha_r A_r \\ \mathrm{Ntu}_i &= \frac{\mathrm{UA}_i}{\dot{m}_r c_{p,r}} \\ \mathrm{Ntu}_o &= \frac{\eta_a \alpha_a A_a}{\dot{m}_a c_{p,a}} \end{aligned} \tag{3.82}$$

where the Ntu are in general defined by $\mathrm{Ntu} = \mathrm{UA}/C$.

The capacitance rate ratio (assuming the minimum capacitance to be on the air side) is

$$C^* = \frac{\dot{m}_a c_{p,a}}{\dot{m}_r c_{p,r}} \tag{3.83}$$

The overall number of thermal units for the entire heat exchanger when dry is

$$\mathrm{Ntu}_{dry} = \frac{\mathrm{Ntu}_o}{1 + C^* \frac{\mathrm{Ntu}_o}{\mathrm{Ntu}_i}} = \frac{\frac{\mathrm{UA}_o}{C_a}}{1 + \frac{\mathrm{UA}_o}{\mathrm{UA}_i}} = \frac{\frac{\mathrm{UA}_o \mathrm{UA}_i}{C_a}}{\mathrm{UA}_i + \mathrm{UA}_o} \tag{3.84}$$

Counterflow effectiveness relations are used for a non-zero C^* since the refrigerant is not changing phase (this is assumed):

$$\varepsilon_{dry} = \frac{1 - \exp(-\mathrm{Ntu}_{dry}(1 - C^*))}{1 - C^* \exp(-\mathrm{Ntu}_{dry}(1 - C^*))} \tag{3.85}$$

The air outlet temperature is given by

$$T_{a,o,dry} = T_{a,i} - \varepsilon_{dry}(T_{a,i} - T_{r,i}) \tag{3.86}$$

and the dry analysis glycol outlet temperature is given by

$$T_{r,o} = T_{r,i} - C^*(T_{a,i} - T_{a,o,dry}) \tag{3.87}$$

The dry air outlet enthalpy is equal to (from overall energy balance over the heat exchanger and assuming constant specific heat on the refrigerant side)

$$h_{a,o} = h_{a,i} - \frac{\dot{m}_r c_{p,r}(T_{r,i} - T_{r,o})}{\dot{m}_a} \tag{3.88}$$

The surface temperature at the air outlet is determined from an energy balance at glycol inlet (air outlet) and solving for surface temperature

$$T_{s,o} = T_{r,i} + C^* \frac{\mathrm{Ntu}_{dry}}{\mathrm{Ntu}_i}(T_{a,o,dry} - T_{r,i}) \tag{3.89}$$

Heat transfer for dry analysis

$$\dot{Q}_{dry} = \varepsilon_{dry} \dot{m}_a c_{p,a}(T_{a,i} - T_{r,i}) \tag{3.90}$$

And dry fraction

$$f_{dry} = 1.0 \tag{3.91}$$

Fully Wet Analysis

If $T_{s,o} < T_{dp}$, then there is at least some portion of the coil which is wetted (potentially all the coil) because some part of the heat exchanger surface is below the dew point of the entering air. The heat exchanger is then assumed to be fully wet in the next step. The possibility still remains that the coil may not be fully wet, but by trying the fully-wet analysis it can be determined whether the coil is fully or partially wetted. Initially, the outlet refrigerant temperature $T_{r,o}$ is not known, but it is determined iteratively, with a first guess given by the fully-dry outlet refrigerant temperature from the above analysis.

The bounding outlet state for the air (the lowest enthalpy it can achieve) is to be saturated at the refrigerant inlet temperature, for which the enthalpy of the saturated air is given by

$$h_{a,sat,r,i} = h(T = T_{r,i}, \phi = 1.0) \tag{3.92}$$

The air saturation specific heat at mean refrigerant temperature is

$$c_s = \left.\frac{\partial h_{a,sat}}{\partial T_r}\right|_{T_r = \frac{T_{r,i} + T_{r,o}}{2}} \tag{3.93}$$

and is polynomial fit based on EES saturated water vapor enthalpy data. The outlet glycol temperature is initially unknown (but is determined iteratively).

The effective humid air mass flow ratio

$$m^* = \frac{\dot{m}_a}{\dot{m}_r \frac{c_{p,r}}{c_s}} \tag{3.94}$$

and the NTU for wet coil

$$\mathrm{Ntu}_{wet} = \frac{\mathrm{Ntu}_o}{1 + m^* \frac{\mathrm{Ntu}_o}{\mathrm{Ntu}_i}} \tag{3.95}$$

counterflow effectiveness

$$\varepsilon_{wet} = \frac{1 - \exp(-\mathrm{Ntu}_{wet}(1 - m^*))}{1 - m^* \exp(-\mathrm{Ntu}_{wet}(1 - m^*))} \tag{3.96}$$

Air outlet enthalpy

$$h_{a,o} = h_{a,i} - \varepsilon_{wet}(h_{a,i} - h_{a,sat,r,i}) \tag{3.97}$$

The amount of heat transfer is

$$\dot{Q}_{wet} = \varepsilon_{wet} \dot{m}_a (h_{a,i} - h_{a,sat,r,i}) \tag{3.98}$$

and the outlet glycol temperature is

$$T_{r,o} = T_{r,i} + \frac{\dot{m}_a}{\dot{m}_r c_{p,r}} (h_{a,i} - h_{a,o}) \tag{3.99}$$

the saturated air enthalpy at the outlet glycol temperature is

$$h_{a,sat,r,o} = h(T = T_{r,o}, \phi = 1.0) \tag{3.100}$$

and the surface temperature at the air inlet is (from local energy balance)

$$T_{s,i} = T_{r,o} + \frac{\dot{m}_a}{\dot{m}_r c_{p,r}} \frac{\mathrm{Ntu}_{wet}}{\mathrm{Ntu}_i} (h_{a,i} - h_{a,sat,r,o}) \tag{3.101}$$

the effective surface enthalpy is given by

$$h_{a,s,sat,e} = h_{a,i} + \frac{h_{a,o} - h_{a,i}}{1 - \exp(-\mathrm{Ntu}_o)} \tag{3.102}$$

from which it it possible to solve

$$h_{a,s,sat,e} = h(T = T_{s,e}, \phi = 1.0) \tag{3.103}$$

for $T_{s,e}$ which makes it possible to solve for the outlet air temperature

$$T_{a,o} = T_{s,e} + (T_{a,i} - T_{s,e}) \exp(-\mathrm{Ntu}_o) \tag{3.104}$$

At this point, if $T_{s,i}$ is also below the dewpoint, the coil is truly full wetted, and the analysis is finished. If on the other hand, $T_{s,i} > T_{dp}$, then the coil is partially-wetted and the analysis in the next section is required.

Partially-wet/Partially-dry

If the cold fluid is water, glycol, or other similar single-phase fluid, the partially-wet/partially-dry analysis is quite a bit more complicated than for two-phase refrigerant. The full derivation is presented in section *Algorithm of partial-wet-partial-dry evaporator*, and the code for this analysis is in `DryWetSegment()`.

Nomenclature

Variable	Description
A_a	Air-side area [m^2]
$A_{a,two-phase}$	Air-side area in two-phase section [m^2]
$A_{r,two-phase}$	Refrigerant-side area in two-phase section [m^2]
A_r	Refrigerant-side area [m^2]
A_w	Water-side area [m^2]
C_{min}	Minimum capacitance rate [W/K]
$c_{p,a}$	Specific heat of humid air per kg dry air [J/kg$_{da}$]
$c_{p,w}$	Specific heat of water [J/kg/K]
c_s	Saturation specific heat of humid air [J//kg$_{da}$/K]
C_r	Ratio of capacitance rates [-]
C^*	Ratio of capacitance rates [-]
f_{dry}	Fraction of coil that is dry [-]
f_{wet}	Fraction of coil that is wet [-]
$h_{a,i}$	Enthalpy of humid air at inlet to coil [J/kg$_{da}$]
$h_{a,o}$	Enthalpy of humid air at outlet from coil [J/kg$_{da}$]
$h_{a,x}$	Enthalpy of humid air at wet-dry interface [J/kg$_{da}$]
$h_{a,s,s,e}$	Effective surface saturated enthalpy [J/kg$_{da}$]
$h_{a,sat,r,i}$	Enthalpy of saturated air at refrigerant inlet temperature [J/kg$_{da}$]
$h_{a,sat,r}$	Enthalpy of saturated air at refrigerant saturation temperature [J/kg$_{da}$]
m^*	Effective wet ratio of mass flow rates [-]
$\dot{m}_a$	Mass flow rate of dry air [kg/s]
$\dot{m}_{a,two-phase}$	Mass flow rate of dry air in two-phase section [kg/s]
Ntu	Number of thermal units [-]
Ntu_o	Outside number of thermal units [-]
Ntu_{dry}	Dry number of thermal units [-]
Ntu_{wet}	Wet number of thermal units [-]
$\dot{Q}$	Heat transfer rate [W]
$\dot{Q}_{dry}$	Heat transfer rate from dry analysis [W]
$\dot{Q}_{two-phase}$	Heat transfer rate in two-phase section [W]
$\dot{Q}_{wet}$	Heat transfer rate from wet analysis [W]
$T_{a,i}$	Air inlet drybulb temperature [K]
$T_{a,o}$	Air outlet drybulb temperature [K]
$T_{a,o,dry}$	Air outlet drybulb temperature from dry analysis [K]
$T_{a,x}$	Air drybulb temperature at wet-dry interface [K]
$T_{bubble,r}$	Bubble-point temperature of refrigerant [K]
$T_{c,i}$	Cold stream inlet temperature [K]
T_{dp}	Dewpoint temperature of air [K]
$T_{dew,r}$	Dew-point temperature of refrigerant [K]
$T_{h,i}$	Hot stream inlet temperature [K]
$T_{r,i}$	Refrigerant inlet temperature [K]
$T_{r,o}$	Refrigerant outlet temperature [K]
$T_{sat,r}$	Saturation temperature of refrigerant [K]
$T_{s,e}$	Effective surface temperature[K]
$T_{s,i}$	Surface temperature at refrigerant inlet [K]
$T_{s,o}$	Surface temperature at refrigerant outlet [K]
UA	Overall surface conductance [W/K]
UA_{dry}	Overall surface conductance for dry analysis [W/K]
UA_{wet}	Overall surface conductance for wet analysis [W/K]
UA_i	Inside thermal conductance [W/K]
UA_o	Outside thermal conductance [W/K]

Continued on next page

Table 3.2 – continued from previous page

Variable	Description
UA_o^*	Effective outside thermal conductance for wetted coil [W/K]
UA_a	Air-side thermal conductance [W/K]
UA_r	Refrigerant-side thermal conductance [W/K]
UA_w	Water-side thermal conductance [W/K]
α_a	Mean air-side heat transfer coefficient [W/m^2/K]
α_w	Mean water-side heat transfer coefficient [W/m^2/K]
α_r	Mean refrigerant-side heat transfer coefficient [W/m^2/K]
$\alpha_{r,two-phase}$	Mean refrigerant-side heat transfer coefficient in two-phase section [W/m^2/K]
η_a	Air-side surface effectivess (dry analysis) [-]
η_a^*	Air-side surface effectivess (wet analysis) [-]
ε	Effectiveness [-]
ε_{dry}	Effectiveness from dry analysis [-]
ε_{wet}	Effectiveness from wet analysis [-]
ω	Humidity ratio [-]
ω_i	Humidity ratio at air inlet [-]

class ACHP.DryWetSegment.**DWSVals**
Empty Class for passing data with DryWetSegment

ACHP.DryWetSegment.**DryWetSegment**(*DWS*)
Generic solver function for dry-wet mixed surface conditions for a given element. Can handle superheated, subcooled and two-phase regions. Does not handle the pressure drops, only HT required to get the dry/wet interface

3.4 Condenser

3.4.1 Overview

The goal of a condenser is to take a refrigerant stream at a superheated state and cool it at nearly constant pressure so that it condenses to a subcooled refrigerant that can then be throttled to the low-pressure side of the system, exiting the throttling valve at low temperature.

3.4.2 Mathematical Description

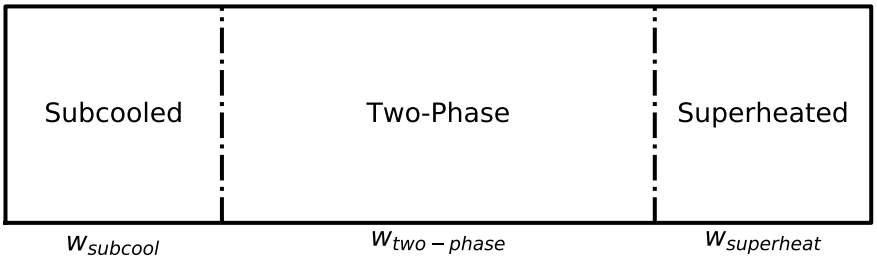
On the refrigerant side, the refrigerant path can be divided into as many as three sections - a superheated section, a two-phase section, and a subcooled section. In practice, operating conditions (particularly low charge), may result in a condenser that does not have a subcooled section, and at extreme conditions, does not have a two-phase section either. The following plot shows the possible configurations for the first two cases described:

In both cases, the schematic represents an averaged circuit on the refrigerant side. Overall, the goal of solver for the condenser is to determine how much of the length of an averaged circuit is in the superheated, two-phase, and subcooled phases. The fraction of the circuit length in each of these segments are given by $w_{superheat}$, $w_{two-phase}$, and $w_{subcool}$ respectively. The sum of these factors must be equal to unity, and if any of the sections do not exist, its respective circuit fraction parameter w is set to zero.

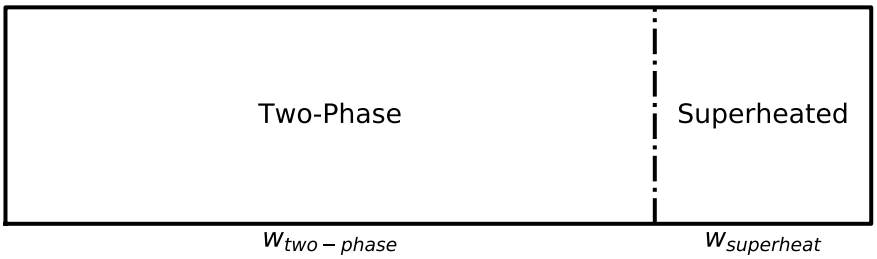
Thus if the average length of one circuit is given by $\overline{L_{circuit}}$, then the lengths of each segment are given by

$$\begin{aligned} L_{superheat} &= w_{superheat}\overline{L_{circuit}} \\ L_{two-phase} &= w_{two-phase}\overline{L_{circuit}} \\ L_{subcool} &= w_{subcool}\overline{L_{circuit}} \end{aligned} \tag{3.105}$$

Superheated, Subcooled and Two-Phase Sections



Superheated and Two-Phase Sections



and for consistency

$$w_{superheat} + w_{two-phase} + w_{subcool} = 1 \tag{3.106}$$

On the air side, the condenser is treated as being pure cross-flow. This is a particularly good assumption for the condition where the condenser is composed of a single bank of tubes, which is a common configuration for condensers in the USA. Thus, since the flow is pure crossflow, the air-side area, as well as the air flow rate for each of the superheated, two-phase, and subcooled sections can be assumed to be proportional to the area for that section (See below).

The analysis here makes a number of assumptions:

1. There is no condensation of moist air on the outside of the tubes since the temperature of refrigerant is above the dewpoint of the air stream
2. The flow is evenly balanced between all the circuits on the refrigerant side
3. The flow is evenly distributed on the air side

Air-side Considerations

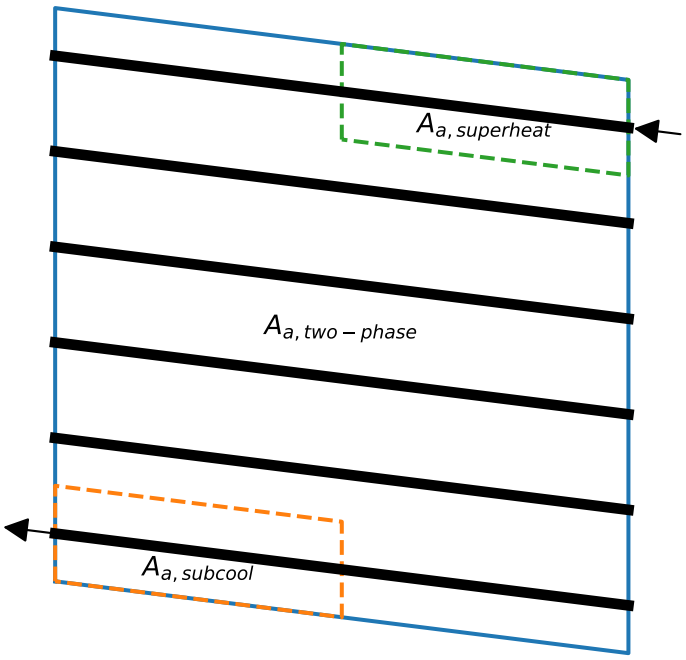
On the air side, the air flow rate flow $\dot{m}_{a,total}$ is assumed to be evenly distributed across the face of the coil which has a total air-side area of $A_{a,total}$. Therefore the areas and mass flow rates for each section are given by

$$\begin{aligned} A_{a,superheat} &= w_{superheat} A_{a,total} \\ A_{a,two-phase} &= w_{two-phase} A_{a,total} \\ A_{a,subcool} &= w_{subcool} A_{a,total} \\ \dot{m}_{a,superheat} &= w_{superheat} \dot{m}_{a,total} \\ \dot{m}_{a,two-phase} &= w_{two-phase} \dot{m}_{a,total} \\ \dot{m}_{a,subcool} &= w_{subcool} \dot{m}_{a,total} \end{aligned} \tag{3.107}$$

Condenser Algorithm

1. Explicitly solve the superheated section to determine $w_{superheat}$
2. First assume that the refrigerant exits the two-phase portion at a quality of 0, and calculate the required $w_{two-phase}$ from the heat transfer analysis for the two-phase section. Then either
   - If the value of $w_{superheat} + w_{two-phase}$ is less than 1, there is a subcooled section
     - The outlet quality of the two-phase section is therefore known to be 0, which yields the value for $w_{two-phase}$
     - Use the remainder of the condenser area for the subcooled section ($w_{subcool} = 1 - w_{two-phase} - w_{superheat}$), solve subcooled section for heat transfer rate, pressure drop, etc.
   - If the value of $w_{superheat} + w_{two-phase}$ is greater than 1, there is no subcooled section
     - Area fraction for two phase section is known to be $w_{two-phase} = 1 - w_{superheat}$, iteratively calculate the outlet quality of the condenser

The analysis for each of these calculations is described in the sections that follow.


$A_{a,\,superheat}$
$A_{a,\,two-phase}$
$A_{a,\,subcool}$

Superheated Section

In the superheated region, the inlet refrigerant temperature and the outlet refrigerant temperature (dew temperature) are known. Therefore, assuming pure crossflow, the fraction of the area required for the superheated region can be explicitly obtained from

$$w_{superheat} = -\frac{\ln(1-\Psi)}{[1-\exp(-\mathrm{UA}_{overall}/(c_{p,a}\dot{m}_{a,total}))]}\frac{\dot{m}_r c_{p,r}}{\dot{m}_{a,total}c_{p,a}} \tag{3.108}$$

where

$$\Psi = \frac{(T_{r,i}-T_{dew,r})}{(T_{r,i}-T_{a,i})} \tag{3.109}$$

and where the derivation of this term can be obtained from *Condenser Area Derivation*. The superheated portion is assumed to exist.

Two-Phase Section

In the two-phase section of the heat exchanger, there are two basic possibilities. Either the outlet of the two-phase section is at some two-phase quality (because there is no subcooled section) or the outlet of the two-phase section is at a quality of 0 (saturated liquid) because there is a subcooled section. The first step is to assume that the outlet of the two-phase section is at a quality of 0 and calculate the required fraction of the circuit length. Before the length fraction can be calculated, the average refrigerant side heat transfer coefficient is required, for which the analysis can be found in section *Shah Condensation*. The average refrigerant-side heat transfer coefficient is a function of refrigerant outlet quality.

In either configuration of the two-phase section (solving for outlet quality or solving for w), the effectiveness is known, since the parameter w cancels out of the solution for Ntu, which means that the Ntu are independent of the length fraction w. Thus the effectiveness can be obtained directly from

$$\varepsilon_{two-phase} = 1-\exp\left(\frac{-\mathrm{UA}_{overall}}{\dot{m}_{a,total}c_{p,a}}\right) \tag{3.110}$$

where the value of $\mathrm{UA}_{overall}$ is given by

$$\mathrm{UA}_{overall} = \frac{1}{(\eta_a\alpha_a A_{a,total})^{-1}+(\alpha_{r,two-phase}A_{r,total})^{-1}} \tag{3.111}$$

For a given outlet quality of the two-phase section ($x_{out,r,two-phase}$), the length fraction $w_{two-phase}$ can be obtained from

$$w_{two-phase} = -\frac{\dot{m}_r h_{fg}(1-x_{out,r,two-phase})}{\dot{m}_{a,total}c_{p,a}(T_{a,i}-T_{sat,r})\varepsilon_{two-phase}} \tag{3.112}$$

Otherwise, if the quality is being iterated for, the residual to be driven to zero by altering the outlet quality of the two phase section ($x_{out,r,two-phase}$) is

$$\Delta = (1-w_{superheat}) - w_{two-phase}(x_{out,r,two-phase}) \tag{3.113}$$

where $w_{two-phase}$ is evaluated from equation (3.112), and $w_{superheat}$ is known from its solution above. Unfortunately, iterative methods are required since the value of the average refrigerant two-phase heat transfer coefficient is dependent on the two-phase section outlet quality.

Subcooled Section

In the subcooled section, if it exists, the available area is known (since $w_{subcool}$ is known), as are the inlet air and refrigerant temperatures. The inlet refrigerant temperature to the subcooled section is the bubble temperature of the refrigerant, and the inlet air temperature is the same inlet air temperature as for all the other sections. Thus the UA value can be given by

$$\mathrm{UA} = \frac{w_{subcool}}{(\eta_a \alpha_a A_{a,total})^{-1} + (\alpha_r A_{r,total})^{-1}} \tag{3.114}$$

and the minimum and maximum capacitance rates of air and subcooled refrigerant can be obtained from

$$\begin{aligned} C_{min} &= \min[\dot{m}_r c_{p,r}, \dot{m}_{a,total} c_{p,a} w_{subcool}] \\ C_{max} &= \max[\dot{m}_r c_{p,r}, \dot{m}_{a,total} c_{p,a} w_{subcool}] \end{aligned} \tag{3.115}$$

which give the Ntu from

$$\begin{aligned} C_r &= \frac{C_{min}}{C_{max}} \\ \mathrm{Ntu} &= \frac{\mathrm{UA}}{C_{min}} \end{aligned} \tag{3.116}$$

which yields the effectiveness, if the minimum capacitance is on the air side, of

$$\varepsilon_{subcool} = \frac{1}{C_r}\left(1 - \exp(-C_r(1 - \exp(-\mathrm{Ntu})))\right) \tag{3.117}$$

or, if the minimum capacitance rate is on the refrigerant side, the effectiveness is given by

$$\varepsilon_{subcool} = 1 - \exp\left(-\frac{1}{C_r}(1 - \exp(-C_r \mathrm{Ntu}))\right) \tag{3.118}$$

The total heat transfer rate in the subcooled section is given by

$$\dot{Q}_{subcool} = -\varepsilon_{subcool} C_{min}(T_{bubble,r} - T_{i,a}) \tag{3.119}$$

which is negative since heat is removed from the refrigerant.

Terminal Calculations

After all the sections of the condenser have been solved and all the requisite terms determined, the results from each section are grouped together, yielding the overall terms

$$\begin{aligned} \dot{Q} &= \dot{Q}_{superheat} + \dot{Q}_{two-phase} + \dot{Q}_{subcool} \\ \Delta p_r &= \Delta p_{r,superheat} + \Delta p_{r,two-phase} + \Delta p_{r,subcool} \\ m_r &= m_{r,superheat} + m_{r,two-phase} + m_{r,subcool} \end{aligned} \tag{3.120}$$

If there is a subcooled section, the condenser outlet subcooling is evaluated from

$$\Delta T_{sc} = T_{bubble,r} - T_{r,o} \tag{3.121}$$

or if there is no subcooled section, an effective subcooling amount is calculated from

$$\Delta T_{sc} = \frac{h_{fg} x_{out,r,two-phase}}{c_{p,dew}} \tag{3.122}$$

where $c_{p,dew}$ is the specific heat of saturated liquid. This effective subcooling parameter is primarily needed to allow for continuous behavior during the solving process. Hopefully the subcooled section exists at cycle model convergence.

3.4.3 Condenser Sample Code

Minimal Component Test:

```
from __future__ import print_function

from CoolProp.CoolProp import PropsSI
from ACHP.Condenser import CondenserClass
from ACHP.FinCorrelations import FinInputs

Fins=FinInputs()
Fins.Tubes.NTubes_per_bank=41    #number of tubes per bank or row
Fins.Tubes.Nbank=1               #number of banks or rows
Fins.Tubes.Ncircuits=5           #number of circuits
Fins.Tubes.Ltube=2.286           #one tube length
Fins.Tubes.OD=0.007
Fins.Tubes.ID=0.0063904
Fins.Tubes.Pl=0.0191             #distance between center of tubes in flow direction
Fins.Tubes.Pt=0.0222             #distance between center of tubes orthogonal to flow␣
↪direction

Fins.Fins.FPI=25                 #Number of fins per inch
Fins.Fins.Pd=0.001               #2* amplitude of wavy fin
Fins.Fins.xf=0.001               #1/2 period of fin
Fins.Fins.t=0.00011              #Thickness of fin material
Fins.Fins.k_fin=237              #Thermal conductivity of fin material

Fins.Air.Vdot_ha=1.7934          #rated volumetric flowrate
Fins.Air.Tdb=308.15              #Dry Bulb Temperature
Fins.Air.p=101325                #Air pressure in Pa
Fins.Air.RH=0.51                 #Relative Humidity
Fins.Air.FanPower=160

params={'Ref': 'R410A',
        'mdot_r': 0.0708,
        'Tin_r': 333.15,
        'psat_r': PropsSI('P','T',323.15,'Q',1.0,'R410A'),
        'Fins': Fins,
        'FinsType': 'WavyLouveredFins',#WavyLouveredFins, HerringboneFins, PlainFins
        'Verbosity': 0
        }

Cond=CondenserClass(**params)
Cond.Calculate()

print('Heat transfer rate in condenser is', Cond.Q,'W')
print('Heat transfer rate in condenser (superheat section) is',Cond.Q_superheat,'W')
print('Heat transfer rate in condenser (twophase section) is',Cond.Q_2phase,'W')
print('Heat transfer rate in condenser (subcooled section) is',Cond.Q_subcool,'W')
print('Fraction of circuit length in superheated section is',Cond.w_superheat)
print('Fraction of circuit length in twophase section is',Cond.w_2phase)
print('Fraction of circuit length in subcooled section is',Cond.w_subcool)
```

If you open an IPython shell in the root of the documentation (folder `Documentation/Web` relative to the main trunk), and run the commands below, you should get something like

```
In [1]: %run 'ACHPComponents/ComponentTests/CondenserTest.py'
Heat transfer rate in condenser is -13289.948674749348 W
```

```
Heat transfer rate in condenser (superheat section) is -1349.2720895090915 W
Heat transfer rate in condenser (twophase section) is -9648.207719103715 W
Heat transfer rate in condenser (subcooled section) is -2292.4688661365417 W
Fraction of circuit length in superheated section is 0.05695832811673864
Fraction of circuit length in twophase section is 0.38007068990354076
Fraction of circuit length in subcooled section is 0.5629709819797206
```

If not, first stop should be the *Frequently Asked Questions*

Nomenclature

Variable	Description
$A_{a,total}$	Total air side area (fins+tubes) [m^2]
$A_{a,superheat}$	Air-side area for superheated portion of circuit [m^2]
$A_{a,two-phase}$	Air-side area for two-phase portion of circuit [m^2]
$A_{a,subcool}$	Air-side area for subcooled portion of circuit [m^2]
$A_{r,total}$	Total refrigerant side area of all tubes [m^2]
$c_{p,r}$	Specific heat of refrigerant [J/kg/K]
$c_{p,dew}$	Specific heat of refrigerant at dewpoint temperature [J/kg/K]
$c_{p,a}$	Specific heat of humid air per kg of dry air [J/kg$_{da}$/K]
C_{min}	Minimum capacitance rate [W/K]
C_{max}	Maximum capacitance rate [W/K]
C_r	Ratio of capacitance rates [-]
h_{fg}	Refrigerant latent heat [J/kg]
$\overline{L_{circuit}}$	Average circuit length [m]
$\dot{m}_{a,total}$	Mass flow rate of dry air [kg/s]
$\dot{m}_{a,superheat}$	Mass flow rate of dry air in superheated section [kg/s]
$\dot{m}_{a,two-phase}$	Mass flow rate of dry air in two-phase section [kg/s]
$\dot{m}_{a,subcool}$	Mass flow rate of dry air in subcooled section [kg/s]
$\dot{m}_r$	Mass flow rate of refrigerant [kg/s]
m_r	Mass of refrigerant charge (total) [kg]
$m_{r,superheat}$	Mass of refrigerant charge in superheated section [kg]
$m_{r,two-phase}$	Mass of refrigerant charge in two-phase section [kg]
$m_{r,subcool}$	Mass of refrigerant charge in subcooled section [kg]
Ntu	Number of thermal units [-]
$L_{superheat}$	Length of superheated portion of circuit [m]
$L_{two-phase}$	Length of two-phase portion of circuit [m]
$L_{subcool}$	Length of subcooled portion of circuit [m]
Δp_r	Refrigerant-side pressure drop (total) [Pa]
$\Delta p_{r,superheat}$	Refrigerant-side pressure drop in superheated section [Pa]
$\Delta p_{r,two-phase}$	Refrigerant-side pressure drop in two-phase section [Pa]
$\Delta p_{r,subcool}$	Refrigerant-side pressure drop in subcooled section [Pa]
$\dot{Q}$	Total heat transfer rate [W]
$\dot{Q}_{superheat}$	Heat transfer rate in superheated section [W]
$\dot{Q}_{two-phase}$	Heat transfer rate in two-phase section [W]
$\dot{Q}_{subcool}$	Heat transfer rate in subcooled section [W]
ΔT_{sc}	Subcooling amount [K]
$T_{a,i}$	Air inlet temperature [K]
$T_{bubble,r}$	Bubble-point temperature of refrigerant [K]
$T_{dew,r}$	Dew-point temperature of refrigerant [K]
$T_{r,i}$	Refrigerant inlet temperature [K]

Continued on next page

Table 3.3 – continued from previous page

Variable	Description
$T_{sat,r}$	Refrigerant saturation temperature [K]
$\mathrm{UA}_{overall}$	Overall heat transfer conductance [W/K]
$w_{superheat}$	Fraction of length of superheated portion of circuit [-]
$w_{two-phase}$	Fraction of length of two-phase portion of circuit [-]
$w_{subcool}$	Fraction of length of subcooled portion of circuit [-]
$x_{out,r,two-phase}$	Quality of refrigerant at outlet of two-phase section [-]
η_a	Overall surface effectiveness on air side [-]
α_a	Air side heat transfer coefficient [W/m^2/K]
$\alpha_{r,two-phase}$	Refrigerant side heat transfer coefficient [W/m^2/K]
Δ	Residual term [-]
$\varepsilon_{two-phase}$	Effectiveness of heat transfer in the two-phase section [-]
$\varepsilon_{subcool}$	Effectiveness of heat transfer in the subcooled section [-]
Ψ	Effectiveness term from derivation [-]

3.4.4 Component Class Documentation

class ACHP.Condenser.**CondenserClass**(***kwargs*)

Calculate()

OutputList()
Return a list of parameters for this component for further output

It is a list of tuples, and each tuple is formed of items: [0] Description of value [1] Units of value [2] The value itself

Update(***kwargs*)

3.5 Cooling Coil

3.5.1 Overview

The goal of a cooling coil is to cool and/or dehumidify an air stream by passing the air stream over a finned set of tubes that have cooler glycol (or similar liquid) flowing through them. In principle the exact same physical heat exchanger can be used to heat an air stream.

3.5.2 Mathematical Description

The cooling coil is a particular case of the Heat Exchanger modeling presented in section *Heat Exchangers with and without Dehumidification*. The water (or similar liquid) passing through the heat exchanger is single phase, and the humid air is assumed to be moist enough that condensation on the heat exchanger surfaces is possible. Therefore, the partially-wet/partially-dry analysis of section *Single-phase on working fluid side* must be used. The use of this analysis still allows for the possibility that the coil is fully wet or fully dry. The entire area of the heat exchanger is employed in order to calculate $A_{a,total}$ and $\dot{m}_{a,total}$.

3.5.3 Cooling Coil Sample Code

Minimal Component Test:

```
from __future__ import print_function

from ACHP.CoolingCoil import CoolingCoilClass
from ACHP.FinCorrelations import FinInputs

FinsTubes=FinInputs()
FinsTubes.Tubes.NTubes_per_bank=32
FinsTubes.Tubes.Nbank=3
FinsTubes.Tubes.Ncircuits=5
FinsTubes.Tubes.Ltube=0.452
FinsTubes.Tubes.OD=0.009525
FinsTubes.Tubes.ID=0.0089154
FinsTubes.Tubes.Pl=0.0254
FinsTubes.Tubes.Pt=0.0219964

FinsTubes.Fins.FPI=14.5
FinsTubes.Fins.Pd=0.001
FinsTubes.Fins.xf=0.001
FinsTubes.Fins.t=0.00011
FinsTubes.Fins.k_fin=237

FinsTubes.Air.Vdot_ha=0.5663
FinsTubes.Air.Tmean=299.8
FinsTubes.Air.Tdb= 299.8
FinsTubes.Air.p=101325              #Air pressure in Pa
FinsTubes.Air.RH=0.51
FinsTubes.Air.RHmean=0.51
FinsTubes.Air.FanPower=438          #fan power in Watts

# Here are two equivalent methods for setting parameters
# 1. Create an empty instance of the class, then set parameters CC=CoolingCoilClass()
CC=CoolingCoilClass()
CC.Fins = FinsTubes
CC.FinsType = 'WavyLouveredFins'    #Choose fin Type: 'WavyLouveredFins' or
↪'HerringboneFins'or 'PlainFins'
CC.Ref_g = 'Water'
CC.mdot_g = 0.15
CC.Tin_g = 278
CC.pin_g = 300000                   #Refrigerant vapor pressure in Pa
CC.Verbosity = 3
CC.Calculate()

print("Method 1:")
print("Cooling Coil Q: " + str(CC.Q) + " W")
print("Cooling Coil SHR: " + str(CC.SHR) + " ")

# 2. Build a dictionary of values, then use that to initialize the class
kwds={'Fins':FinsTubes,
      'FinsType': 'WavyLouveredFins',   #Choose fin Type: 'WavyLouveredFins' or
↪'HerringboneFins'or 'PlainFins'
      'Ref_g': 'Water',
      'mdot_g': 0.15,
      'Tin_g': 278,
      'pin_g': 300000,                #Refrigerant vapor pressure in Pa
      'Verbosity':3}
CC2=CoolingCoilClass(**kwds)
CC2.Calculate()
```

```
print("Method 2:")
print("Cooling Coil Q: " + str(CC2.Q) + " W")
print("Cooling Coil SHR: " + str(CC2.SHR) + " ")
```

If you open an IPython shell in the root of the documentation (folder `Documentation/Web` relative to the main trunk), and run the commands below, you should get output similar to

```
In [1]: %run 'ACHPComponents/ComponentTests/CoolingCoilTest.py'
Method 1:
Cooling Coil Q: 9696.412470349105 W
Cooling Coil SHR: 0.8047125117004815
Method 2:
Cooling Coil Q: 9696.412470349105 W
Cooling Coil SHR: 0.8047125117004815
```

If not, first stop should be the *Frequently Asked Questions*

3.5.4 Component Class Documentation

class ACHP.CoolingCoil.**CoolingCoilClass**(***kwargs*)

The module that implements a cooling coil. See documentation for further information

Calculate()

This function is now simply a wrapper around the DryWetSegment() function in order to decrease the amount of code replication

Initialize()

OutputList()

Update(***kwargs*)

Update the parameters passed in using the dictionary

3.6 Evaporator

3.6.1 Overview

In the evaporator, refrigerant enters at some vapor quality between 0% and 100%, and typically exits as a superheated vapor. The analysis for the evaporator shares many features with the analysis employed for the *Condenser* and the *Cooling Coil*, but the evaporator's analysis is significantly more complicated due to the fact that there are moving boundaries on both the refrigerant and air sides. On the refrigerant-side, the moving boundary is between two-phase refrigerant and superheated refrigerant, and on the air-side, there is a moving boundary between wet and dry parts of the coil.

3.6.2 Mathematical Description

The evaporator is treated as being cross-counter-flow. What this means practically is that the mass flow of air to and total air side surface area for each of the superheated and two-phase portions (assuming each exists), are given by

$$\begin{aligned} A_{a,superheat} &= w_{superheat} A_{a,total} \\ A_{a,two-phase} &= w_{two-phase} A_{a,total} \\ \dot{m}_{a,superheat} &= w_{superheat} \dot{m}_{a,total} \\ \dot{m}_{a,two-phase} &= w_{two-phase} \dot{m}_{a,total} \end{aligned} \tag{3.123}$$

which is analogous to the analysis employed for the condenser. On the refrigerant-side, the surface area for superheated and two-phase sections are given by

$$\begin{aligned} A_{r,superheat} &= w_{superheat} A_{r,total} \\ A_{r,two-phase} &= w_{two-phase} A_{r,total} \end{aligned} \tag{3.124}$$

3.6.3 Two-Phase Section

In the two-phase section, the target heat transfer rate is given by

$$\dot{Q}_{target} = \dot{m}_r (x_{r,o} - x_{r,i}) h_{fg} \tag{3.125}$$

and the heat transfer rate from the partially-wet/partially-dry analysis in section *Two-phase on working fluid side* is given the name $\dot{Q}_{PWPD}$. To begin with, the first guess is that all the heat exchanger is in the two-phase region. If using the outlet quality of saturated vapor ($x_{r,o} = 1$) with all the heat exchanger in the two-phase region, and $\dot{Q}_{PWPD}$ is greater than $\dot{Q}_{target}$, too much of the heat exchanger area was given to the two-phase section, and there must be a superheated section as well. Thus, the length fraction $w_{two-phase}$ is iteratively altered using a bounded solver ($w_{two-phase}$ is bounded between 0 and 1). The remaining area is given to the superheated section, so the superheated section circuit length fraction $w_{superheat}$ is given by

$$w_{superheat} = 1 - w_{two-phase} \tag{3.126}$$

3.6.4 Superheated Section

In the superheated section, for a given $w_{superheat}$, the analysis is exactly the same as the *Cooling Coil*, and given by the section *Single-phase on working fluid side*. As in the cooling coil, the working fluid is single-phase, and the air-side surface may be fully wet, partially wet, or fully dry, and all the inlet conditions for the superheated section are known. The air inlet state to the superheated portion is assumed to be the same as for the two-phase portion because the superheated portion is typically rather small, and both superheated and two-phase portions should see approximately the same inlet air state. The refrigerant properties are calculated using the same single-phase correlations as for the cooling coil.

3.6.5 Minimal Example Code

```
from __future__ import print_function

from CoolProp.CoolProp import PropsSI
from ACHP.FinCorrelations import FinInputs
from ACHP.Evaporator import EvaporatorClass

FinsTubes=FinInputs()
```

```
FinsTubes.Tubes.NTubes_per_bank=32
FinsTubes.Tubes.Ncircuits=5
FinsTubes.Tubes.Nbank=3
FinsTubes.Tubes.Ltube=0.452
FinsTubes.Tubes.OD=0.009525
FinsTubes.Tubes.ID=0.0089154
FinsTubes.Tubes.Pl=0.0254
FinsTubes.Tubes.Pt=0.0219964

FinsTubes.Fins.FPI=14.5
FinsTubes.Fins.Pd=0.001
FinsTubes.Fins.xf=0.001
FinsTubes.Fins.t=0.00011
FinsTubes.Fins.k_fin=237

FinsTubes.Air.Vdot_ha=0.5663
FinsTubes.Air.Tdb=299.8
FinsTubes.Air.p=101325
FinsTubes.Air.RH=0.51
FinsTubes.Air.FanPower=438

Ref = 'n-Propane'

kwargs={'Ref': Ref,
        'mdot_r': 0.0708,
        'psat_r': PropsSI('P','T',282,'Q',1.0,Ref),
        'Fins': FinsTubes,
        'FinsType': 'WavyLouveredFins',#WavyLouveredFins, HerringboneFins, PlainFins
        'hin_r': PropsSI('H','T',282,'Q',0.15,Ref), #[J/kg]
        'Verbosity': 0
        }

Evap=EvaporatorClass(**kwargs)
Evap.Update(**kwargs)
Evap.Calculate()

print('Evaporator heat transfer rate is',Evap.Q,'W')
print('Evaporator capacity (less fan power) is',Evap.Capacity,'W')
print('Evaporator fraction of length in two-phase section',Evap.w_2phase,'W')
print('Evaporator sensible heat ratio',Evap.SHR)
```

If you open an IPython shell in the root of the documentation (folder `Documentation/Web` relative to the main trunk), and run the commands below, you should get output something like

```
In [1]: %run 'ACHPComponents/ComponentTests/EvaporatorTest.py'
Evaporator heat transfer rate is 15338.089884663588 W
Evaporator capacity (less fan power) is 14900.089884663588 W
Evaporator fraction of length in two-phase section 1.0 W
Evaporator sensible heat ratio 0.6748897722684061
```

If not, first stop should be the *Frequently Asked Questions*

Nomenclature

Variable	Description
$A_{a,total}$	Total air side area (fins+tubes) [m^2]
$A_{a,superheat}$	Air-side area for superheated portion of circuit [m^2]
$A_{a,two-phase}$	Air-side area for two-phase portion of circuit [m^2]
$A_{r,superheat}$	Refrigerant-side area for superheated portion of circuit [m^2]
$A_{r,two-phase}$	Refrigerant-side area for two-phase portion of circuit [m^2]
h_{fg}	Latent heat of refrigerant [J/kg]
$\dot{m}_{a,total}$	Mass flow rate of dry air [kg/s]
$\dot{m}_{a,superheat}$	Mass flow rate of dry air in superheated section [kg/s]
$\dot{m}_{a,two-phase}$	Mass flow rate of dry air in two-phase section [kg/s]
$\dot{m}_r$	Mass flow rate of refrigerant [kg/s]
$\dot{Q}_{target}$	Target heat transfer rate [W]
$\dot{Q}_{PWPD}$	Heat transfer rate from the partially wet/partially-dry analysis [W]
$x_{r,i}$	Refrigerant inlet quality [-]
$x_{r,o}$	Refrigerant outlet quality [-]

3.6.6 Component Class Documentation

class ACHP.Evaporator.**EvaporatorClass**(***kwargs*)

AirSideCalcs()

Calculate()

Initialize()

OutputList()
Return a list of parameters for this component for further output

It is a list of tuples, and each tuple is formed of items: [0] Description of value [1] Units of value [2] The value itself

Update(***kwargs*)

3.7 Multi-Circuited Evaporator

In an evaporator, there is always the possibility that the distribution of refrigerant or air may be unequal between the circuits, which usually results in lower evaporator performance. In the case of equal distribution between circuits, the analysis of the *Evaporator* can be used, otherwise the multi-circuited evaporator is needed. The multi-circuited evaporator (MCE) analyzed as being a set of evaporators, each comprising one circuit, that are then each fed some (not necessarily even) distribution of refrigerant and air. The tubes per banks are distributed among the different circuits. If the tubes per bank is divisible by number of circuits, all the circuits will have the same number of tubes. Otherwise, the circuits are ordered from fewer to more if not evenly distributed.

3.7.1 Mathematical Description

The MCE model is capable of handling the following types of maldistribution:

- Volumetric air flow mal-distribution
- Air-side inlet condition mal-distribution

- Refrigerant mass-flow-per-circuit mal-distribution
- Refrigerant quality-per-circuit mal-distribution

For the MCE, there are $N_{circuits}$ circuits, and each circuit is treated as an individual evaporator, since then the analysis for each circuit can be provided by the analysis for the conventional evaporator with one circuit.

For the MCE, the total refrigerant mass flow rate $\dot{m}_r$ is known as an input (which arises from the compressor map). The total refrigerant mass flow rate per circuit can be given by

$$\dot{m}_{r,i} = \gamma_i \dot{m}_r \tag{3.127}$$

where γ_i is the mass flow distribution factor for the i-th circuit of the evaporator. The sum of the indices is given by

$$\sum_{i=0}^{N_{circuits}-1} [\gamma_i] = 1 \tag{3.128}$$

and if the flow is equally distributed, all the terms γ_i are equal.

If the inlet refrigerant quality is not balanced between circuits, the refrigerant vapor mal-distribution can be given by a set of weighting parameters that distribute refrigerant vapor among the circuits. The total amount of vapor entering the evaporator can be given by

$$\dot{m}_v = x \dot{m}_r \tag{3.129}$$

and the amount of vapor entering the i-th circuit can be given by

$$\dot{m}_{v,i} = \xi_i \dot{m}_v \tag{3.130}$$

where the factors ξ_i also must sum to unity. The inlet quality for each circuit is then equal to

$$x_i = \frac{\dot{m}_{v,i}}{\dot{m}_{r,i}} \tag{3.131}$$

where all the x_i values must be greater than zero. The evaporator component model takes enthalpy as the inlet, which can be calculated from

$$h_i = h(p_{evap}, x_i) \tag{3.132}$$

using the CoolProp property routines.

3.7.2 MCE Sample Code

Minimal Component Test:

```
from __future__ import print_function

from ACHP.FinCorrelations import FinInputs
from ACHP.Evaporator import EvaporatorClass
from ACHP.MultiCircuitEvaporator import MultiCircuitEvaporatorClass
from CoolProp.CoolProp import PropsSI
import numpy as np

FinsTubes=FinInputs()

FinsTubes.Tubes.NTubes_per_bank=32
FinsTubes.Tubes.Ncircuits=5
```

```
FinsTubes.Tubes.Nbank=3
FinsTubes.Tubes.Ltube=0.452
FinsTubes.Tubes.OD=0.009525
FinsTubes.Tubes.ID=0.0089154
FinsTubes.Tubes.Pl=0.0254
FinsTubes.Tubes.Pt=0.0219964

FinsTubes.Fins.FPI=14.5
FinsTubes.Fins.Pd=0.001
FinsTubes.Fins.xf=0.001
FinsTubes.Fins.t=0.00011
FinsTubes.Fins.k_fin=237

FinsTubes.Air.Vdot_ha=0.5663
FinsTubes.Air.Tmean=299.8
FinsTubes.Air.Tdb=299.8
FinsTubes.Air.p=101325
FinsTubes.Air.RH=0.51
FinsTubes.Air.RHmean=0.51
FinsTubes.Air.FanPower=438

# This uses the normal baseline evaporator model
Ref = 'R32'
kwargs={'Ref': Ref,
        'mdot_r': 0.0708,
        'psat_r': PropsSI('P','T',282.0,'Q',1.0,Ref),
        'Fins': FinsTubes,
        'FinsType': 'WavyLouveredFins',
        'hin_r': PropsSI('H','T',282.0,'Q',0.15,Ref),
        'Verbosity':0
        }
Evap=EvaporatorClass(**kwargs)
Evap.Calculate()
print('Evap Q=' + str(Evap.Q) + ' W')

#This uses the multi-circuited evaporator model but with no mal-distribution
kwargs={'Ref': Ref,
        'mdot_r': 0.0708,
        'psat_r': PropsSI('P','T',282.0,'Q',1.0,Ref),
        'Fins': FinsTubes,
        'FinsType': 'WavyLouveredFins',
        'hin_r': PropsSI('H','T',282.0,'Q',0.15,Ref),
        'Verbosity':0
        }
MCE=MultiCircuitEvaporatorClass(**kwargs)
MCE.Calculate()
print('MCE Q='+str(MCE.Q)+' W w/o mal-distribution')

#Not exactly the same since
# This uses the multi-circuited evaporator model with mal-distribution of
# refrigerant, refrigerant quality, and air volumetric flow rate
kwargs={'Ref': Ref,
        'mdot_r': 0.0708,
        'psat_r': PropsSI('P','T',282.0,'Q',1.0,Ref),
        'mdot_r_coeffs': [0.3,0.2,0.1,0.2,0.2],
        'mdot_v_coeffs': [0.4,0.2,0.1,0.2,0.1],
        'Vdot_ha_coeffs': [0.3,0.2,0.2,0.2,0.1],
        'Fins': FinsTubes,
```

```
        'FinsType': 'WavyLouveredFins',
        'hin_r': PropsSI('H','T',282.0,'Q',0.15,Ref),
        'Verbosity':0
        }
MCE=MultiCircuitEvaporatorClass(**kwargs)
MCE.Calculate()
print('MCE Q='+str(MCE.Q)+' W w/ mal-distribution')


#Another way to express air maldirtribution for the last example,
#using a vector of Vdot_ha and m_dot_r instead of vector of coefficients
FinsTubes.Air.Vdot_ha=0.5663*np.array([0.3,0.2,0.2,0.2,0.1])
kwargs={'Ref': Ref,
        'mdot_r': 0.0708*np.array([0.3,0.2,0.1,0.2,0.2]),
        'mdot_v_coeffs': [0.4,0.2,0.1,0.2,0.1],
        'psat_r': PropsSI('P','T',282.0,'Q',1.0,Ref),
        'Fins': FinsTubes,
        'FinsType': 'WavyLouveredFins',
        'hin_r': PropsSI('H','T',282.0,'Q',0.15,Ref),
        'Verbosity':0
        }
MCE=MultiCircuitEvaporatorClass(**kwargs)
MCE.Calculate()
print('MCE Q='+str(MCE.Q)+' W w/ mal-distribution')
```

If you open an IPython shell in the root of the documentation (folder `Documentation/Web` relative to the main trunk), and run the commands below, you should get output similar to

```
In [1]: %run 'ACHPComponents/ComponentTests/MCETest.py'
Evap Q=15431.23873425192 W
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~/checkouts/readthedocs.org/user_builds/achp/checkouts/latest/Documentation/Web/
↪ACHPComponents/ComponentTests/MCETest.py in <module>()
     56         }
     57 MCE=MultiCircuitEvaporatorClass(**kwargs)
---> 58 MCE.Calculate()
     59 print('MCE Q='+str(MCE.Q)+' W w/o mal-distribution')
     60

~/checkouts/readthedocs.org/user_builds/achp/conda/latest/lib/python3.5/site-packages/
↪ACHP-1.5-py3.5.egg/ACHP/MultiCircuitEvaporator.py in Calculate(self)
    120             # Make a deep copy to break all the links between the Fins structs
    121             # of each of the evaporator instances
--> 122             ED=copy.deepcopy(EvapDict)
    123             #Create new evaporator instanciated with new deep copied␣
↪dictionary
    124             E=EvaporatorClass(**ED)

~/checkouts/readthedocs.org/user_builds/achp/conda/latest/lib/python3.5/copy.py in␣
↪deepcopy(x, memo, _nil)
    153     copier = _deepcopy_dispatch.get(cls)
    154     if copier:
--> 155         y = copier(x, memo)
    156     else:
    157         try:

~/checkouts/readthedocs.org/user_builds/achp/conda/latest/lib/python3.5/copy.py in _
↪deepcopy_dict(x, memo)
```

```
    241     memo[id(x)] = y
    242     for key, value in x.items():
--> 243         y[deepcopy(key, memo)] = deepcopy(value, memo)
    244     return y
    245 d[dict] = _deepcopy_dict

~/checkouts/readthedocs.org/user_builds/achp/conda/latest/lib/python3.5/copy.py in␣
↪deepcopy(x, memo, _nil)
    180                                 raise Error(
    181                                     "un(deep)copyable object of type %s" % cls)
--> 182                 y = _reconstruct(x, rv, 1, memo)
    183
    184     # If is its own copy, don't memoize.

~/checkouts/readthedocs.org/user_builds/achp/conda/latest/lib/python3.5/copy.py in _
↪reconstruct(x, info, deep, memo)
    290     if deep:
    291         args = deepcopy(args, memo)
--> 292     y = callable(*args)
    293     memo[id(x)] = y
    294

~/checkouts/readthedocs.org/user_builds/achp/conda/latest/lib/python3.5/copyreg.py in␣
↪__newobj__(cls, *args)
     86
     87 def __newobj__(cls, *args):
---> 88     return cls.__new__(cls, *args)
     89
     90 def __newobj_ex__(cls, args, kwargs):

CoolProp/AbstractState.pyx in CoolProp.CoolProp.AbstractState.__cinit__ (CoolProp/
↪CoolProp.cpp:11120)()

TypeError: __cinit__() takes exactly 2 positional arguments (0 given)
```

If not, first stop should be the *Frequently Asked Questions*

3.7.3 Component Class Documentation

class ACHP.MultiCircuitEvaporator.**MultiCircuitEvaporatorClass**(***kwargs*)

Calculate()

OutputList()
Return a list of parameters for this component for further output

It is a list of tuples, and each tuple is formed of items: [0] Description of value [1] Units of value [2] The value itself

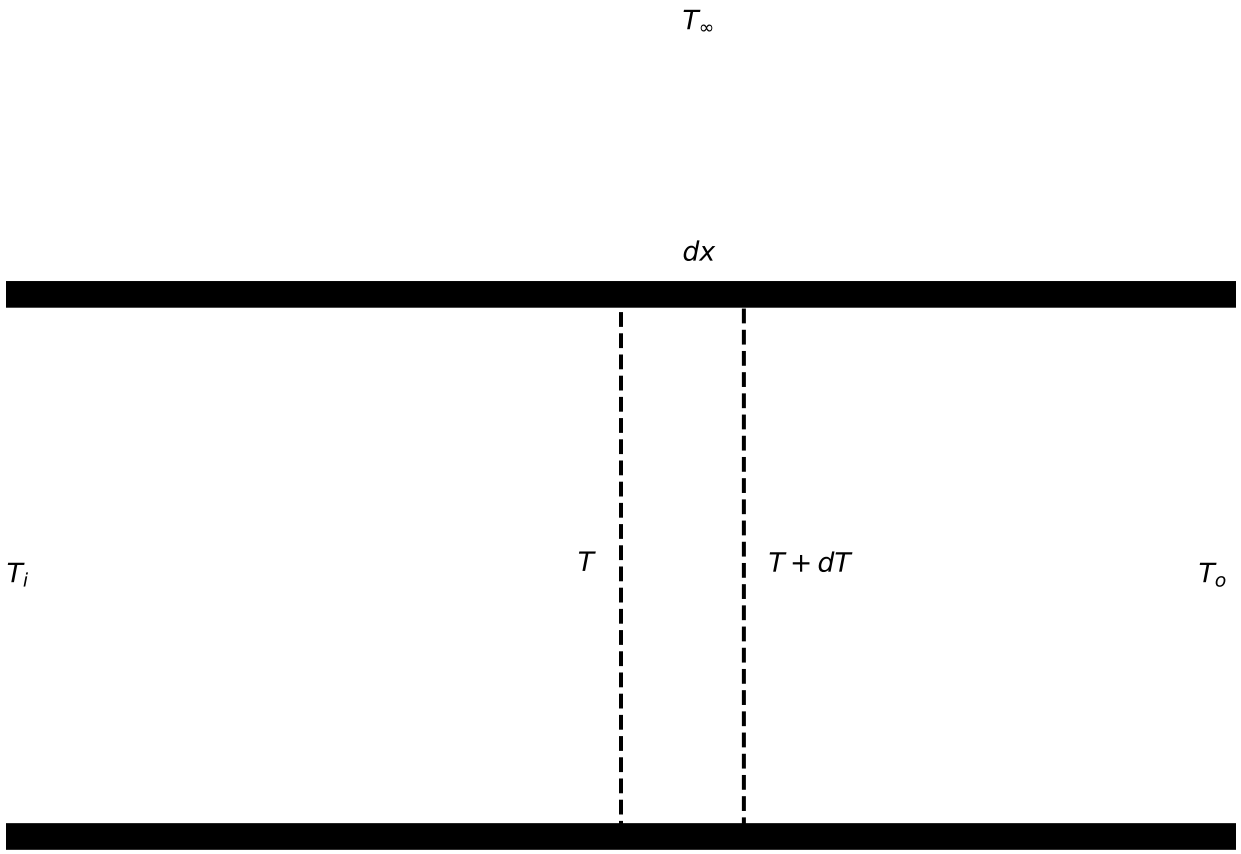
Update(***kwargs*)

3.8 Line Set

3.8.1 Overview

A line set is essentially just a set of tubes of a reasonably long length that allows for fluid to be moved from one place to another. It is commonly insulated in order to decrease heat transfer between the tube and ambient.

3.8.2 Mathematical Description



An energy balance over the control volume in the tube gives

$$\dot{m}_r c_{p,r} dT = U dA (T_\infty - T) \tag{3.133}$$

where if T_∞ is greater than T, the differential dT is positive. Therefore separation of variables yields

$$\frac{dT}{T_\infty - T} = \frac{U dA}{\dot{m}_r c_{p,r}} \tag{3.134}$$

and upon integration (with a u-substitution)

$$-\ln(T_\infty - T)|_{T_i}^{T_o} = \frac{UA}{\dot{m}_r c_{p,r}} \tag{3.135}$$

finally yields

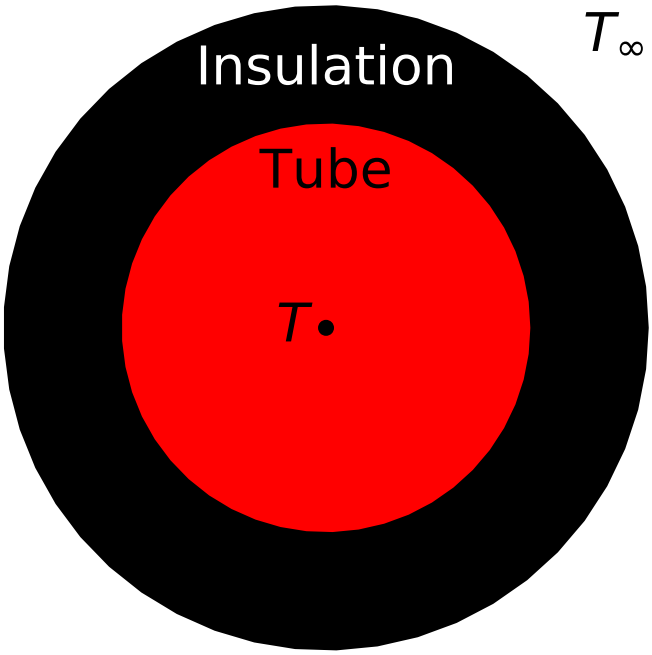
$$\frac{T_o - T_\infty}{T_i - T_\infty} = \exp\left(-\frac{UA}{\dot{m}_r c_{p,r}}\right) \tag{3.136}$$

and the outlet temperature from

$$T_o = T_\infty + (T_i - T_\infty)\exp\left(-\frac{UA}{\dot{m}_r c_{p,r}}\right) \tag{3.137}$$

heat transfer

$$Q = \dot{m}_r c_{p,r}(T_o - T_i) \tag{3.138}$$



Based on the concentric geometry, the heat transfer network terms are given by

$$R_{tube} = \frac{\ln(D_o/D_i)}{2\pi L k_{tube}} \tag{3.139}$$

$$R_{insul} = \frac{\ln[(D_o + 2t_{insul})/D_o]}{2\pi L k_{insul}} \tag{3.140}$$

$$\begin{aligned} \mathrm{UA}_i &= h_i \pi D_i L \\ \mathrm{UA}_o &= h_o \pi (D_o + 2t_{insul}) L \end{aligned} \tag{3.141}$$

and the overall heat conductance is given by

$$\mathrm{UA} = \frac{1}{\mathrm{UA}_i^{-1} + R_{tube} + R_{insul} + \mathrm{UA}_o^{-1}} \tag{3.142}$$

The pressure drop and refrigerant charge are evaluated from section *Single-phase refrigerant pressure drop, heat transfer and charge*.

Nomenclature

Variable	Description
$c_{p,r}$	Refrigerant specific heat [J/kg/K]
D_i	Inner diameter of tube [m]
D_o	Outer diameter of tube [m]
k_{insul}	Thermal conductivity of insulation [W/m/K]
L	Length of tube [m]
$\dot{m}_r$	Refrigerant mass flow rate [kg/s]
R_{tube}	Thermal resistance from tube [K/W]
R_{insul}	Thermal resistance from insulation [K/W]
t_{insul}	Thickness of insulation [m]
T_i	Inlet temperature [K]
T_o	Outlet temperature [K]
T_∞	Ambient temperature [K]
UA	Overall heat transfer conductance [W/K]
$\mathrm{UA_i}$	Inner heat transfer conductance [W/K]
$\mathrm{UA_o}$	Outer heat transfer conductance [W/K]
α_i	Inner heat transfer coefficient [W/m^2/K]
α_o	Outer heat transfer coefficient [W/m^2/K]

3.8.3 Component Class Documentation

class ACHP.LineSet.**LineSetClass**(***kwargs*)

Calculate()

OutputList()
Return a list of parameters for this component for further output

It is a list of tuples, and each tuple is formed of items: [0] Description of value [1] Units of value [2] The value itself

Update(***kwargs*)

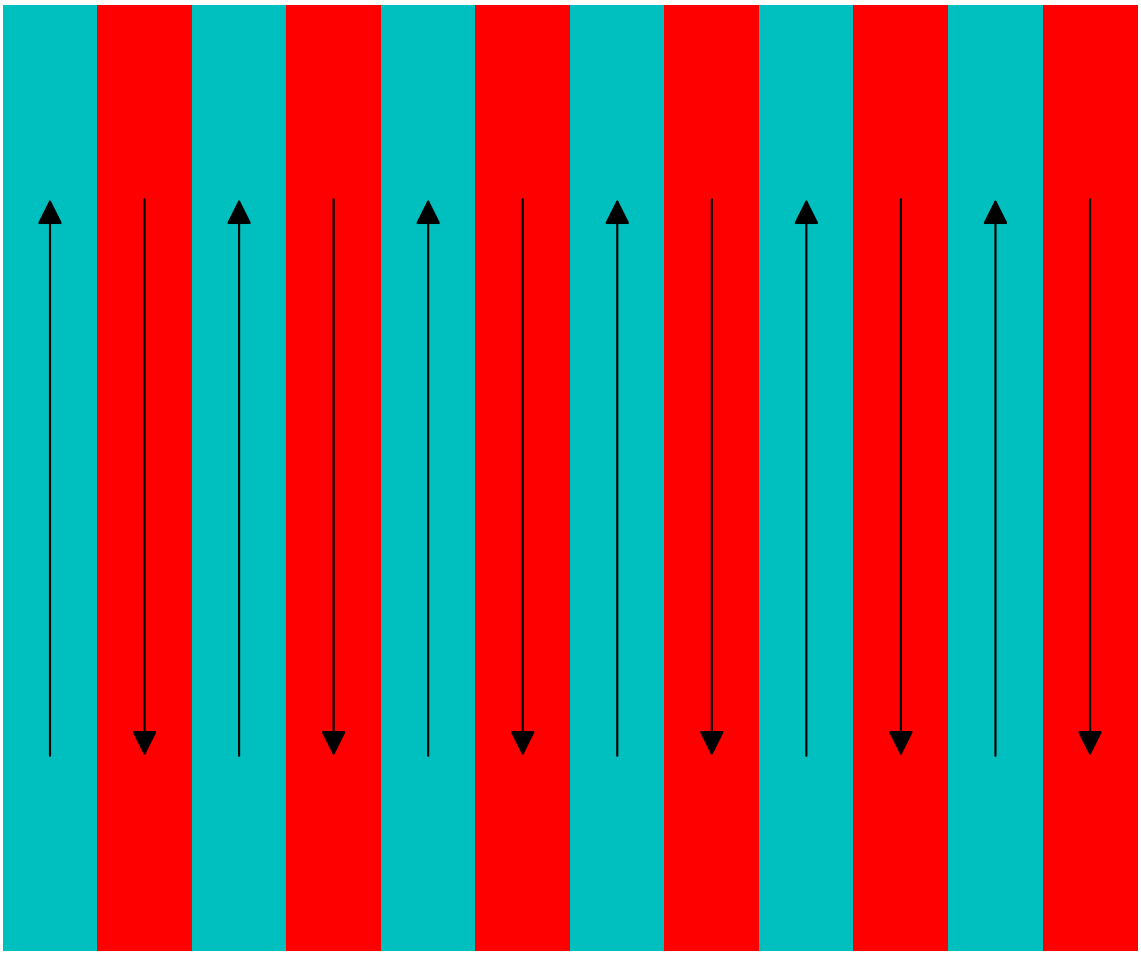
3.9 Plate-Heat-Exchangers

3.9.1 Overview and Geometry

The motivating factor that drives the use of brazed-plate heat exchangers is that they are a highly-compact heat exchanger that allows for excellent heat transfer between two fluids with very well controlled pressure drop. They tend

to be slightly more expensive than equivalent coaxial type heat exchangers due to their more exacting manufacturing requirements. But they can be easily altered to add more plates to give more surface area for increased heat transfer area and lower pressure drop. The trade-off as usual is that adding plates to decrease the pressure drop also results in a decrease in heat transfer coefficient, which means that each m^2 of surface area in the HX becomes less useful.

In the most basic configuration of a plate heat exchanger, hot and cold streams in pure counterflow alternate through the stack of plates. From the side view, a simplified schematic of the BPHE is something like this:

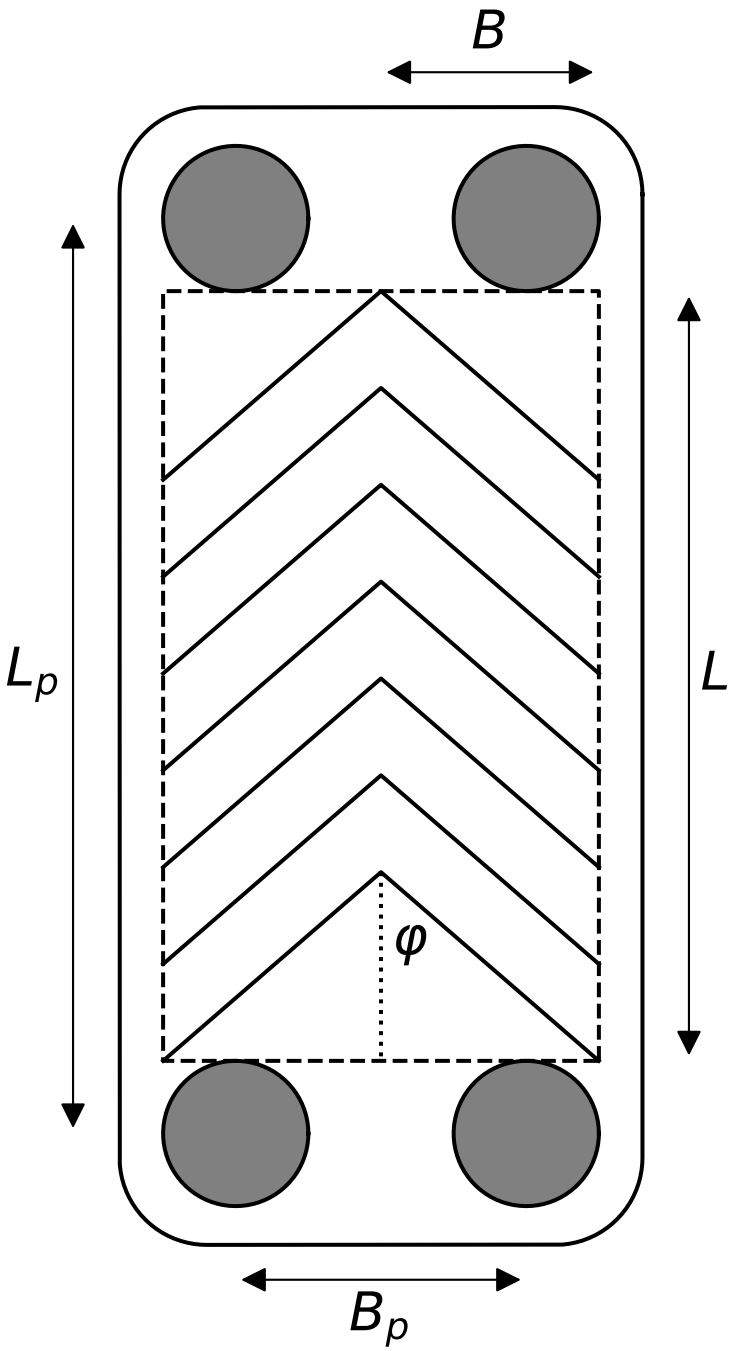


In practice, it is sometimes useful to have one of the stream do multiple passes for one pass of the other stream, but this capability is not included in the BPHE model as of this time. This is commonly used when the capacitance rates are very different, a sufficient heat transfer rate cannot be achieved for one fluid, or when one of the fluids is particularly sensitive to pressure drop. All of these issues are particularly strongly felt for the flow of gases.

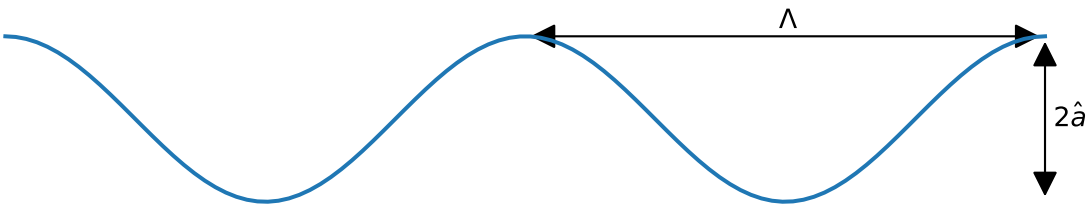
In practice, what you get when you put it all together is a heat exchanger that looks something like from face-on:

Robust gasketing of the plates is required to ensure that the fluid phases do not mix at the inlets and outlets of the plates.

Each of the plates that form the internal surface of the BPHE are formed of plates with a wavy shape, and the edge of


B
L_p
L
φ
B_p

the plate looks something like this:



Based on this geometry, the hydraulic diameter d_h is defined by

$$d_h = \frac{4\hat{a}}{\Phi} \tag{3.143}$$

where the parameter Φ is the ratio of the actual area to the planar area enclosed by the edges of the plate (bounded by L and B in the figure above). Typically the plates do not have exactly a sinusoidal profile, but, to a decent approximation, their profile is sinusoidal, which results in the value of Φ of

$$\Phi = \frac{1}{6}\left(1 + \sqrt{1+X^2} + 4\sqrt{1+X^2/2}\right) \tag{3.144}$$

where the wavenumber X is given by

$$X = 2\pi\hat{a}/\Lambda \tag{3.145}$$

When the plates are put together to form a stack, the plates are alternated, and as a result, chevron-shaped flow paths are formed, which have the effect of yielding highly mixed flow, resulting in good heat transfer coefficients

A stack of N_{plates} plates form the heat exchanger. There are a total of $N_{channels}$ channels formed between the plates. If $N_{channels}$ is evenly divisible by 2, both fluids have the same number of channels. If $N_{channels}$ is not evenly divisible by 2, one stram must have one extra channel. The outer two plates don't provide any heat transfer as they are just used to maintain the channel structure for the outermost channels. There are therefore a total of $N_{plates} - 2$ active plates, for which the active area of one side of one plate is equal to

$$A_{1p} = B_p L_p \Phi \tag{3.146}$$

Each channel of a fluid gets two sides of this area, which yields the cold- and hot-side wetted areas of

$$\begin{aligned} A_h &= 2N_{channels,h}A_{1p} \\ A_c &= 2N_{channels,c}A_{1p} \end{aligned} \tag{3.147}$$

$$\begin{aligned} V_h &= N_{channels,h}B_p 2\hat{a} \\ V_c &= N_{channels,c}B_p 2\hat{a} \end{aligned} \tag{3.148}$$

and the mass flow rate of the hot and cold fluids per circuit (needed for correlations), is equal to

$$\begin{aligned} \dot{m}_{h,ch} &= \dot{m}_h/N_{channels,h} \\ \dot{m}_{c,ch} &= \dot{m}_c/N_{channels,c} \end{aligned} \tag{3.149}$$

3.9.2 Heat Transfer and Pressure Drop Correlations

Single-Phase Flow

When there is single-phase flow on one side of the heat exchanger, the analysis of Martin[6] from the VDI Heat Atlas is employed. The nomenclature used here mirrors that of Martin. Using this analysis, the pressure drop and heat transfer coefficients for the fluid flowing between the plates can be calculated.

The Reynolds number for the flow through the channel between two plates is given by

$$\mathrm{Re} = \frac{\rho w d_h}{\eta} \tag{3.150}$$

where the velocity per channel is given by

$$w = \frac{\dot{m}_{ch}}{2\hat{a}\rho B_p} \tag{3.151}$$

The pressure drop and heat transfer coefficients are usual a function of the Reynolds number, and if the flow is laminar ($\mathrm{Re} < 2000$), the factors ζ_0 and $\zeta_{1,0}$ are given by

$$\begin{aligned} \zeta_0 &= \frac{64}{\mathrm{Re}} \\ \zeta_{1,0} &= \frac{597}{\mathrm{Re}} + 3.85 \end{aligned} \tag{3.152}$$

and if the flow is turbulent ($\mathrm{Re} \geq 2000$), the factors ζ_0 and $\zeta_{1,0}$ are given by

$$\begin{aligned} \zeta_0 &= \frac{1}{(1.8\ln(\mathrm{Re}) - 1.5)^2} \\ \zeta_{1,0} &= \frac{39}{\mathrm{Re}^{0.289}} \end{aligned} \tag{3.153}$$

The friction factor ζ is obtained from

$$\frac{1}{\sqrt{\zeta}} = \frac{\cos\varphi}{\sqrt{b\tan\varphi + c\sin\varphi + \zeta_0/\cos\varphi}} + \frac{1-\cos\varphi}{\sqrt{\zeta_1}} \tag{3.154}$$

where the factor ζ_1 is given by

$$\zeta_1 = a\zeta_{1,0} \tag{3.155}$$

and the factors a and b and c given by Martin are

$$\begin{aligned} a &= 3.8 \\ b &= 0.18 \\ c &= 0.36 \end{aligned} \tag{3.156}$$

The Hagen number is defined by

$$\mathrm{Hg} = \frac{\zeta \mathrm{Re}^2}{2} = \frac{\rho \Delta p d_h^3}{\eta^2 L_p} \tag{3.157}$$

which gives the value for the pressure drop

$$\Delta p = \mathrm{Hg}\frac{\eta^2 L_p}{\rho d_h^3} \tag{3.158}$$

[6] Holger Martin, VDI Heat Atlas 2010, Chapter B6: Pressure Drop and Heat Transfer in Plate Heat Exchangers

The Nusselt number is obtained from

$$\mathrm{Nu} = c_q \mathrm{Pr}^{1/3} (\eta/\eta_w)^{1/6} [2\mathrm{Hg}\sin(2\varphi)]^q \tag{3.159}$$

where the recommended values of the constants c_q and q from Martin are 0.122 and 0.374 respectively. Finally the overall heat transfer coefficient is obtained from

$$\alpha = \frac{k\mathrm{Nu}}{d_h} \tag{3.160}$$

Two-Phase Evaporating Flow

When the fluid flow is evaporating, it is quite a bit more difficult to determine the best model to use. There are contradictory conclusions drawn in literature as to what type of heat transfer is occurring.

It seems like the most accepted view, though is open to debate, is that the flow is governed by nucleate boiling within the channels, and as a result, nucleate pool boiling relations are employed in order to calculate the heat transfer coefficient. This model has some features which are not well-suited to implementation into the PHE model. For one, there is no quality dependence on heat transfer coefficient, which yields un-physically high values of heat transfer coefficient at high quality (should go to the saturated vapor gas heat transfer coefficient at pure vapor).

In spite of these shortcomings, the pool boiling correlation of Cooper[5] was used. This yields a simple form of the solution for the heat transfer coefficient. The heat transfer coefficient is obtained from

$$\alpha = K55(p^*)^{0.12-0.2\log_{10}(R_p)}(-\log_{10}(p^*))^{-0.55}(q")^{0.67}M^{-0.5} \tag{3.161}$$

where

$$\begin{aligned} p^* &= p/p_{crit} \\ q" &= \dot{Q}/A \end{aligned} \tag{3.162}$$

and M is the molar mass (kg/kmol) of the fluid and R_p is the relative roughness of the surface and K is set to unity. However, Claesson[4] suggested a correction of 1.5.

In order to calculate the pressure drop in evaporating flow in the PHE channel, the frictional pressure drop is calculated using the Lockhart-Martinelli two-phase pressure drop correlation from section *Lockhart-Martinelli* with the value of the parameter C of 4.67 as recommended by Claesson[4]. The accelerational pressure change is given from the same section.

Two-Phase Condensing Flow

The available models for condensing flow in PHE share many of the shortcomings of the evaporating flow models. There is a paucity of good data available, since most of the know-how is controlled by the major PHE manufacturers. That said, many researchers have studied this topic, but the parameter space (geometrically and thermodynamically) is quite vast. This is still a topic that could do with further study.

Longo has conducted studies[1][2][3] that looked at condensation in PHE, and from these studies it can be seen that at low equivalent Reynolds number ($\mathrm{Re}_{eq} < 1750$), the j-factor is nominally constant at a value of 60, and above that, it is

[5] Cooper, M.G., 1984. Heat Flow Rates in Saturated Nucleate Pool Boiling-A Wide-Ranging Examination Using Reduced Properties. Advances in Heat Transfer, 16, pp.157-239.

[4] Claesson, J., 2004. Thermal and Hydraulic Performance of Compact Brazed Plate Heat Exchangers Operating as Evaporators in Domestic Heat Pumps. PhD Thesis. KTH.

[1] Longo, G.; Gasparella, A. & Sartori, R. (2004), Experimental heat transfer coefficients during refrigerant vaporisation and condensation inside herringbone-type plate heat exchangers with enhanced surfaces, *International Journal of Heat and Mass Transfer* 47, 4125-4136.

[2] Longo, G. (2010), Heat transfer and pressure drop during HFC refrigerant saturated vapour condensation inside a brazed plate heat exchanger, *International Journal of Heat and Mass Transfer* 53, 1079-1087.

[3] Longo, G. (2010), Heat transfer and pressure drop during hydrocarbon refrigerant condensation inside a brazed plate heat exchanger, *International Journal of Refrigeration* 33, 944-953.

linear with equivalent Reynolds number, so the j-factor can be given by

$$j = \begin{cases} 60 & \mathrm{Re}_{eq} < 1750 \\ \frac{75-60}{3000-1750}(\mathrm{Re}_{eq} - 1750) + 60 & \mathrm{Re}_{eq} \geq 1750 \end{cases} \tag{3.163}$$

where the equivalent Reynolds number is defined by

$$\mathrm{Re}_{eq} = \frac{G\left[(1-\overline{x}) + \overline{x}\sqrt{\frac{\rho_L}{\rho_V}}\right] d_h}{\eta_L} \tag{3.164}$$

which finally yields the heat transfer coefficient

$$\alpha = \frac{jk\mathrm{Pr}^{1/3}}{d_h} \tag{3.165}$$

3.9.3 Mathematical Description

With the set of required correlations defined, it is now possible to analyze the plate heat exchanger for a range of different configurations. The PHE model is constructed to be general enough that it can handle any phase of fluids entering into the heat exchanger. The basic idea behind the PHE model is a two-step process:

1. Determine the bounding heat transfer rate (100% effectiveness) limited by taking each fluid to the inlet temperature of the other fluid. This is the most amount of heat transfer possible. Also watch out for internal pinch points
2. Since the heat transfer rate is now bounded between zero and the maximum, iterate to find the actual heat transfer rate in the heat exchanger in order to yield the "right-size" heat exchanger (see description below)

Bounds on Heat Transfer Rate

Since the PHE is pure counterflow, the coldest possible temperature that the hot stream can achieve is the inlet temperature of the cold stream, and similarly, the hottest temperature that the cold stream can achieve is the inlet temperature of the hot stream. The inlet enthalpies of the hot stream $h_{h,i}$ and the cold stream $h_{c,i}$ allow to calculate a preliminary value for the upper bound on the heat transfer rate:

$$\begin{aligned} \dot{Q}_{max,h} &= \dot{m}_h[h_{h,i} - h(T = T_{c,i}, p = p_{h,i}, Ref_h)] \\ \dot{Q}_{max,c} &= \dot{m}_c[h(T = T_{h,i}, p = p_{c,i}, Ref_c) - h_{c,i}] \\ \dot{Q}_{max,\varepsilon=1} &= \max[\dot{Q}_{max}, \dot{Q}_{min}] \end{aligned} \tag{3.166}$$

Using this preliminary bound on the heat transfer rate, it is then possible to determine the enthalpies and temperatures of both fluids at each of their phase transitions(if they exist).

In the case of an evaporator that cools a water stream, there is no possibility of temperature inversion within the heat exchanger because the refrigerant enters at some quality greater than zero, and there is no possibility that the isobar of the refrigerant could intersect the isobar of the hot water. The following figure shows this configuration: Since the heat transfer rate is known, it can be readily determined whether any of the phase transitions can be physically reached. In the configuration shown here, there are two regions; in cell 1 the hot fluid (water) is single phase and the cold fluid (refrigerant) is evaporating, and in cell 2, the hot fluid is still single-phase, and the cold fluid is now single phase as well.

On the other hand, if the refrigerant were condensing, and entering at some subcooling amount greater than zero, for instance 10 K, the analysis is slightly different. In this case, it is entirely possible that there could be temperature inversion at the heat transfer rate given by $\dot{Q}_{max,\varepsilon=1}$, as shown in the folllowing physically impossible figure:

Thus, a new maximum heat transfer rate $\dot{Q}_{max}$ can be determined that is less than $\dot{Q}_{max,\varepsilon=1}$ whereby the temperatures of the two streams are equated at the possible pinch point, which resembles something like the following figure:

In this case, the water is limiting the heat transfer rate, and the maximum heat transfer rate can be given by taking the water all the way to the dew temperature of the refrigerant, and using the known heat transfer rate in cell 3. The cold-stream pinch enthalpy is given by

$$h_{pinch} = h(T = T_{dew,h}, p = p_c, Ref_c) \tag{3.167}$$

Since the inlet enthalpy and outlet enthalpy (saturated vapor) of the hot refrigerant are known in cell 3, the heat transfer rate in cell 3 is known from

$$\dot{Q}_{cell3} = \dot{m}_h(h_{h,i} - h(T = T_{dew,h}, x = 1, Ref_h)) \tag{3.168}$$

and the new limiting heat transfer rate can be given by

$$\dot{Q}_{max} = \dot{m}_c(h_{pinch} - h_{c,i}) + \dot{Q}_{cell3} \tag{3.169}$$

where the contribution $\dot{m}_c(h_{pinch} - h_{c,i})$ is from heating up the cold fluid to the pinch point temperature.

Calculation of Heat Transfer Rate

Now that the physical bounds on the heat transfer rate in the PHE have been determined, it is now possible to finish analyzing the PHE performance. For a given $\dot{Q} < \dot{Q}_{max}$, there are a number of different cells, and in each one, at least one of the fluids has a phase transition. In the degenerate case that both fluids are single-phase throughout the PHE, there is only one cell, and no phase transitions anywhere in the heat exchanger.

The discussion that follows here assumes that the heat transfer rate $\dot{Q}$ is known, but in practice, it is iteratively obtained by a bounded 1-D solver because $\dot{Q}$ is known to be between 0 and $\dot{Q}_{max}$.

For a given $\dot{Q}$, the outlet enthalpies are known, which begins the process of buiding enthalpy vectors for both streams. The outlet enthalpies for each stream are given by

$$\begin{aligned} h_{h,o} &= h_{h,i} - \dot{Q}/\dot{m}_h \\ h_{c,o} &= h_{c,i} + \dot{Q}/\dot{m}_c \end{aligned} \tag{3.170}$$

which yields the initial enthalpy vectors (ordered from low to high enthalpy) of

$$\begin{aligned} \vec{h}_h &= [h_{h,o}, h_{h,i}] \\ \vec{h}_c &= [h_{c,i}, h_{c,o}] \end{aligned} \tag{3.171}$$

To these enthalpy vectors are now added any phase transitions that exist; a phase transition exists if its corresponding saturation enthalpy is between the inlet and outlet enthalpies of the fluid. With each phase transition enthalpy comes a partner enthalpy of the other stream. This set of enthalpy vectors then define the enthalpies of both streams at each cell edge. For instance, in the case shown in Figure *Evaporator*, there is one phase transition where the refrigerant transitions between two-phase and superheated vapor. The enthalpy of the cold stream at the transition point is given by

$$h_{PT} = h(T = T_{dew,c}, x = 1, \mathrm{Ref}_c) \tag{3.172}$$

and the enthalpy of the hot stream at the phase transition h^*_{PT} can be obtained by an energy balance over cell 2, which yields

$$\dot{m}_h(h_{h,i} - h_{PT}) = \dot{m}_c(h_{c,o} - h^*_{PT}) \tag{3.173}$$

or

$$h^*_{PT} = h_{c,o} - \frac{\dot{m}_h(h_{h,i} - h_{PT})}{\dot{m}_c} \tag{3.174}$$

and now the enthalpy vectors are given by the values

$$\begin{aligned}\vec{h}_h &= [h_{h,o}, h_{PT}, h_{h,i}] \\ \vec{h}_c &= [h_{c,i}, h^*_{PT}, h_{c,o}]\end{aligned} \tag{3.175}$$

If there are multiple phase transitions on each side, the same method is applied, where the phase transition enthalpies and their partner enthalpies are obtained by an energy balance on the new cell that is formed, working from the outer edges of the enthalpy vectors towards the inside since the outlet enthalpies of both streams are known and can be used in the energy balances to back out partner enthalpies.

For a given value of $\dot{Q}$, each of the enthalpy vectors has the same length of $N_{cell}+1$, which then form the enthalpy boundaries for the N_{cell} cells.

In each cell, first the phase of each fluid must be determined. Each fluid will have the same phase throughout the entire cell (that was the whole point in the first place!). The average enthalpy of each fluid in the cell can be used to determine the phase of each fluid in the cell. Our goal now is to determine how much of the physical length of the heat exchanger is required to obtain the given duty in each cell. The required physical heat exchanger length of the cell w can be given by

$$L_i = w_i L \tag{3.176}$$

where all the w_i parameters must sum to unity (1.0).

In a given cell, the heat transfer rate is known because this is how the enthalpy vectors have been constructed. The heat transfer rate in the cell can be given by

$$\dot{Q}_i = \dot{m}_r(\vec{h}_{h,i+1} - \vec{h}_{h,i}) \tag{3.177}$$

So long as at least one of the fluids in the cell is single-phase, the effectiveness in the cell can be defined by

$$\varepsilon = \frac{\dot{Q}_i}{C_{min}(T_{h,i,cell} - T_{c,i,cell})} \tag{3.178}$$

where $T_{h,i,cell}$ and $T_{c,i,cell}$ are the hot fluid and cold fluid inlet temperatures to the cell. The minimum capacitance rate C_{min} is by definition on the single-phase-fluid side. In the single-phase/two-phase cell case, the minimum capacitance rate is given by

$$C_{min} = \dot{m}_{\mathrm{single-phase}} c_{p,\mathrm{single-phase}} \tag{3.179}$$

and since the flow is pure counter-flow, the Ntu can be obtained directly from

$$\mathrm{Ntu} = -\ln(1-\varepsilon) \qquad (C_r = 0) \tag{3.180}$$

If both fluids are single phase, the minimum capacitance rate can be obtained from

$$\begin{aligned}C_{min} &= \min[\dot{m}_h c_{p,h}, \dot{m}_c c_{p,c}] \\ C_{max} &= \max[\dot{m}_h c_{p,h}, \dot{m}_c c_{p,c}] \\ C_r &= C_{min}/C_{max}\end{aligned} \tag{3.181}$$

which yields the Ntu for the single-phase/single-phase cell with pure counterflow of

$$\mathrm{Ntu} = \frac{1}{C_r - 1}\ln\left(\frac{\varepsilon - 1}{\varepsilon C_r - 1}\right) \qquad (C_r < 1) \tag{3.182}$$

and the required heat conductance can be obtained from

$$\mathrm{UA}_{req} = \mathrm{Ntu} C_{min} \tag{3.183}$$

The actual heat transfer conductance in the call can be given by

$$\mathrm{UA}_{actual} = \frac{1}{\alpha_c A_c} + \frac{t}{kA} + \frac{1}{\alpha_h A_h} \tag{3.184}$$

where the areas are based on the total wetted area of the heat exchanger and local heat transfer coefficients (α_h, α_c) for the cell are employed. The fraction of the heat exchanger that would be required for the given thermal duty in the cell can be obtained from

$$w_i = \frac{\mathrm{UA}_{req}}{\mathrm{UA}_{actual}} \tag{3.185}$$

Determination of Thermal Duty

Finally, the heat transfer rate in the PHE is obtained through iterative methods. The value of $\dot{Q}$ is known to be between zero and $\dot{Q}_{max}$, and the residual to be driven to zero by a numerical solver is

$$\Delta = 1 - \sum_i [w_i] \tag{3.186}$$

which will be zero if $\dot{Q}$ has been appropriately found.

Nomenclature

Variable	Description
$\hat{a}$	Amplitude of plate corrugation [m]
a	Constant for equation [-]
b	Constant for equation [-]
c	Constant for equation [-]
c_q	Constant for equation [-]
α	Heat transfer coefficient [W/m^2/K]
A_{1p}	Area of one plate [m^2]
A	Area on given side [m^2]
A_h	Area on hot side [m^2]
A_c	Area on cold side [m^2]
B	Width of wetted section [m]
B_p	Port-port centerline distance [m]
$c_{p,single-phase}$	Specific heat of single-phase fluid [J/kg/K]
$c_{p,c}$	Specific heat of cold fluid [J/kg/K]
$c_{p,h}$	Specific heat of hot fluid [J/kg/K]
C_{min}	Minimum capacitance rate [W/K]
C_{max}	Maximum capacitance rate [W/K]
C_r	Ratio of capacitance rates [W/K]
G	Refrigerant mass flux [kg/m^2/s]
Hg	Hagen number [-]
d_h	Hydraulic diameter [m]
$h_{c,i}$	Enthalpy of cold stream at inlet [J/kg/K]
$h_{h,i}$	Enthalpy of hot stream at inlet [J/kg/K]
$h_{c,o}$	Enthalpy of cold stream at outlet [J/kg/K]
$h_{h,o}$	Enthalpy of hot stream at outlet [J/kg/K]
h_{pinch}	Enthalpy of stream at pinch [J/kg/K]
$\vec{h}_c$	Vector of cold stream enthalpies [J/kg/K]
$\vec{h}_h$	Vector of hot stream enthalpies [J/kg/K]

Continued on next page

Table 3.4 – continued from previous page

Variable	Description
h_{PT}	Enthalpy at phase transition [J/kg/K]
h^*_{PT}	Complementary enthalpy at phase transition [J/kg/K]
j	Colburn j-factor [-]
k	Thermal conductivity of fluid [W/m/K]
L_i	Length of a given cell [m]
L	Length of wetted section [m^2]
L_p	Port-port centerline distance [m]
$\dot{m}_c$	Total mass flow rate of cold fluid [kg/s]
$\dot{m}_h$	Total mass flow rate of hot fluid [kg/s]
$\dot{m}_{c,ch}$	Mass flow rate of cold fluid per channel [kg/s]
$\dot{m}_{h,ch}$	Mass flow rate of hot fluid per channel [kg/s]
M	Molar mass [kg/kmol]
$N_{channels}$	Number of channels [-]
$N_{channels,c}$	Number of channels on the cold side [-]
$N_{channels,h}$	Number of channels on the hot side [-]
N_{plates}	Number of plates [-]
N_{cell}	Number of cells [-]
Nu	Nusselt number [-]
p^*	Reduced pressure [-]
p_{crit}	Critical pressure [kPa]
p	Saturation pressure [kPa]
p_c	Pressure of cold stream [kPa]
$p_{c,i}$	Inlet pressure of cold stream [kPa]
$p_{h,i}$	Inlet pressure of hot stream [kPa]
Pr	Prandtl number [-]
q	Constant for equation [-]
q"	Heat transfer flux [W/m^2]
$\dot{Q}$	Heat transfer rate [W]
$\dot{Q}_i$	Heat transfer rate in the given cell [W]
$\dot{Q}_{max,c}$	Cold stream max heat transfer rate [W]
$\dot{Q}_{cell3}$	Heat transfer rate in the highest-enthalpy cell [W]
$\dot{Q}_{max,h}$	Hot stream max heat transfer rate [W]
$\dot{Q}_{max}$	Maximum heat transfer rate [W]
$\dot{Q}_{max,\varepsilon=1}$	Max heat transfer rate taking each stream to inlet temp of opposite fluid [W]
Re	Reynolds number [-]
Re_{eq}	Equivalent Reynolds number [-]
$T_{h,i}$	Hot stream inlet temperature to PHE [K]
$T_{c,i}$	Cold stream inlet temperature to PHE [K]
$T_{h,i,cell}$	Hot stream inlet temperature to cell [K]
$T_{c,i,cell}$	Cold stream inlet temperature to cell [K]
$T_{dew,h}$	Dewpoint temperature of refrigerant [K]
UA_{reg}	Required conductace [W/K]
UA_{actual}	Actual conductance available [W/K]
V_c	Total volume on cold side [m^3]
V_h	Total volume on hot side [m^3]
w	Velocity of fluid in channel [m/s]
w_i	Fraction of total length for given cell [-]
x	Refrigerant quality [-]
$\bar{x}$	Average refrigerant quality [-]

Continued on next page

Table 3.4 – continued from previous page

Variable	Description
X	Wave number [-]
η	Viscosity of the fluid [Pa-s]
η_L	Viscosity of saturated liquid [Pa-s]
η_w	Viscosity at the wall temperature [Pa-s]
φ	Plate inclination angle [rad]
Λ	Plate corrugation wavelength [m]
Φ	Area increase factor [-]
ρ	Fluid density [kg/m^3]
ρ_L	Saturated liquid fluid density [kg/m^3]
ρ_V	Saturated vapor fluid density [kg/m^3]
ζ	Friction factor [-]
ζ_0	Friction factor [-]
$\zeta_{1,0}$	Friction factor [-]
ζ_1	Friction factor [-]
ε	Effectiveness [-]

3.10 Fluid Correlations and Other Calculations

A model is only as good as the correlations it is based on. A number of heat transfer and pressure drop correlations are needed for various components in the cycle. The table below summarizes the correlations used, and references are available for all the correlations.

Summary of correlations employed

Parameter	Reference
Single-phase pressure drop	Churchill[1]
Single-phase heat transfer	Gnielinski[2]
Two-phase pressure drop	Lockhart-Martinelli[3]
Two-phase evaporation heat transfer	Shah[4]
Two-phase condensation heat transfer	Shah[5]
Two-phase void fraction	Zivi[6]

3.10.1 Single-phase refrigerant pressure drop, heat transfer and charge

The Churchill correlation (based on a Darcy friction factor for which the laminar friction factor is $f = 64/\mathrm{Re}_D$) is

$$A = \left(-2.457 \log \left[\left(\frac{7}{\mathrm{Re}_D} \right)^{0.9} + 0.27(\varepsilon/D) \right] \right)^{16} \tag{3.187}$$

[1] Churchill, S.W., 1977, Friction-factor equation spans all fluid-flow regimes, *Chemical Engineering* v. 84, n. 24 91-92

[2] Gnielinski, V., 1976, New Equation for Heat and Mass Transfer in Turbulent Pipe and Channel Flow, *Int. Chemical Engineering* v. 16, 359-368.

[3] Lockhart, R.W., Martinelli, R.C., 1949, Proposed Correlation of Data for Isothermal Two-Phase Two-Component Flow in Pipes. Chemical Engineering Progress. v. 45, 39-48

[4] Shah, M., 1976. A New Correlation for Heat Transfer During Boiling Flow Through Pipes. ASHRAE Transactions 82, 66-86.

[5] Shah, M., 1979. A general correlation for heat transfer during film condensation inside pipes. International Journal of Heat and Mass Transfer 22, 547-556.

[6] Zivi, S., 1964. Estimation of Steady-State Void-Fraction by Means of the Principle of Minimum Entropy Production. Journal of Heat Transfer 86, 247-252.

$$B = \left[\frac{37530.0}{\mathrm{Re}_D}\right]^{16} \tag{3.188}$$

$$f = 8\left[\left(\frac{8}{\mathrm{Re}_D}\right)^{12} + \frac{1}{(A+B)^{1.5}}\right]^{1/12} \tag{3.189}$$

with the Reynolds number defined by

$$\mathrm{Re}_D = \frac{\rho \bar{U} D}{\mu} = \frac{4\dot{m}}{\pi \mu D} \tag{3.190}$$

With the known friction factor, the pressure gradient is given by

$$\frac{dp}{dz} = \frac{-fvG^2}{2D} \tag{3.191}$$

with the mass flux defined by

$$G = \frac{\dot{m}}{(\pi D^2/4)} \tag{3.192}$$

and assuming the gradient to be constant over the length L because averaged properties are used, the total pressure drop is

$$\Delta p = \frac{-fvG^2 L}{2D} \tag{3.193}$$

The Gnielinski correlation, good for smooth tubes and 0.5 < Pr < 2000 and 3000 < Re_D < 5x10^6, gives the single-phase heat transfer coefficient as

$$\alpha = \frac{k}{D}\frac{(f/8)(\mathrm{Re}_D - 1000)\,\mathrm{Pr}}{1 + 12.7(f/8)^{1/2}(\mathrm{Pr}^{2/3} - 1)} \tag{3.194}$$

The refrigerant charge for a single-phase volume is equal to

$$m = \rho V \tag{3.195}$$

where the density ρ is based on the average temperature and pressure.

In the case that given circuit of a heat exchanger is being analyzed, the value of L is equal to the length of the circuit (or average length if there are multiple circuits). In addition, the mass flow rate $\dot{m}$ is therefore given as the mass flow rate per circuit.

3.10.2 Two-phase refrigerant pressure drop, heat transfer and charge

In the two-phase portion, the pressure drop components which are non-zero are the frictional pressure drop and the accelerational pressure drop. The gravitational pressure drop is also assumed to be negligible. In the case of evaporation, both frictional and acceleration result in a decrease in pressure. The Lockhart-Martinelli correlation is used to find the frictional pressure drop gradient, but it varies with quality. The total pressure drop is then found by integrating the pressure drop gradient over the range of qualities of interest.

Lockhart-Martinelli frictional pressure drop

The Lockhart-Martinelli two-phase pressure drop gradient is based on the following algorithm:

1. Find the Reynolds Number for each phase based on the actual flow rate of the individual phase

$$\mathrm{Re}_g = \frac{GxD}{\mu_g} \tag{3.196}$$

$$\mathrm{Re}_f = \frac{G(1-x)D}{\mu_f} \tag{3.197}$$

2. Friction factor for each phase

$$f_f = \begin{cases} \dfrac{16.0}{\mathrm{Re}_f} & \mathrm{Re}_f < 1000 \\ \dfrac{0.046}{\mathrm{Re}_f^{0.2}} & \mathrm{Re}_f > 2000 \\ (1-w)\dfrac{16.0}{\mathrm{Re}_f} + w\dfrac{0.046}{\mathrm{Re}_f^{0.2}} & 1000 < \mathrm{Re}_f < 2000 \end{cases} \tag{3.198}$$

where $w = (\mathrm{Re}_f - 1000)/(2000 - 1000)$ which results in a linear interpolation for the transitional Reynolds number region

$$f_g = \begin{cases} \dfrac{16.0}{\mathrm{Re}_g} & \mathrm{Re}_g < 1000 \\ \dfrac{0.046}{\mathrm{Re}_g^{0.2}} & \mathrm{Re}_g > 2000 \\ (1-w)\dfrac{16.0}{\mathrm{Re}_g} + w\dfrac{0.046}{\mathrm{Re}_g^{0.2}} & 1000 < \mathrm{Re}_g < 2000 \end{cases} \tag{3.199}$$

where $w = (\mathrm{Re}_g - 1000)/(2000 - 1000)$ which results in a linear interpolation for the transitional Reynolds number region

3. Frictional pressure drop based on actual flow rate of each phase

$$-\left(\frac{dp}{dz}\right)_f = \frac{2f_f G^2(1-x)^2 v_f}{D} \tag{3.200}$$

$$-\left(\frac{dp}{dz}\right)_g = \frac{2f_g G^2 x^2 v_g}{D} \tag{3.201}$$

4. Lockhart-Martinelli parameter

$$X = \sqrt{\frac{\left(\dfrac{dp}{dz}\right)_f}{\left(\dfrac{dp}{dz}\right)_g}} \tag{3.202}$$

5. Find the L-M Constant based on the flow Re of each phase (using 1500 as the transitional Re to ensure continuity)

$$C = \begin{cases} 20 & \mathrm{Re}_f > 1500 \ \& \ \mathrm{Re}_g > 1500 \\ 12 & \mathrm{Re}_f < 1500 \ \& \ \mathrm{Re}_g > 1500 \\ 10 & \mathrm{Re}_f > 1500 \ \& \ \mathrm{Re}_g < 1500 \\ 5 & \mathrm{Re}_f < 1500 \ \& \ \mathrm{Re}_g < 1500 \end{cases} \tag{3.203}$$

6. Two-phase multipliers for each phase

   Gas multiplier

$$\phi_g = 1 + CX + X^2 \tag{3.204}$$

   Fluid multiplier

$$\phi_f = 1 + \frac{C}{X} + \frac{1}{X^2} \tag{3.205}$$

7. Find gradient for a given value of x

$$-\left(\frac{dp}{dz}\right)_{f,2\phi} = \begin{cases} -\left(\frac{dp}{dz}\right)_g \phi_g & -\left(\frac{dp}{dz}\right)_g \phi_g > -\left(\frac{dp}{dz}\right)_f \phi_f \\ -\left(\frac{dp}{dz}\right)_f \phi_f & -\left(\frac{dp}{dz}\right)_g \phi_g < -\left(\frac{dp}{dz}\right)_f \phi_f \end{cases} \tag{3.206}$$

8. Lockhart-Martinelli Void fraction

$$\epsilon = 1 - \frac{X}{X^2 + 20X + 1} \tag{3.207}$$

9. Average pressure drop gradient

$$: label : eqFC22$$
$$\overline{\Delta p_{f,2\phi}} = \frac{\int_{x_1}^{x_2} -\left(\frac{dp}{dz}\right)_{f,2\phi} dx}{x_2 - x_1} \tag{3.208}$$

10. Frictional pressure drop: using Simpson's rule to solve the integration

Accelerational pressure drop

From the consideration of two-phase flow analysis, the accelerational presssure drop can be obtained. It is caused by the change in velocity of the vapor and liquid phases due to phase change, which in boiling creates vapor and accelerates the vapor, or in the case of condensation, reduces the vapor velocity, resulting in a pressure increase.

$$-\left(\frac{\partial p}{\partial z}\right)_A = G^2 \frac{d}{dz}\left[\frac{x^2 v_g}{\epsilon} + \frac{(1-x)^2 v_f}{1-\epsilon}\right] \tag{3.209}$$

where ϵ is the refrigerant vapor void fraction (typically the symbol α is used for void fraction, but here we are using that for heat transfer coefficient). Integrating over the length where the quality goes from x_1 to x_2 yields

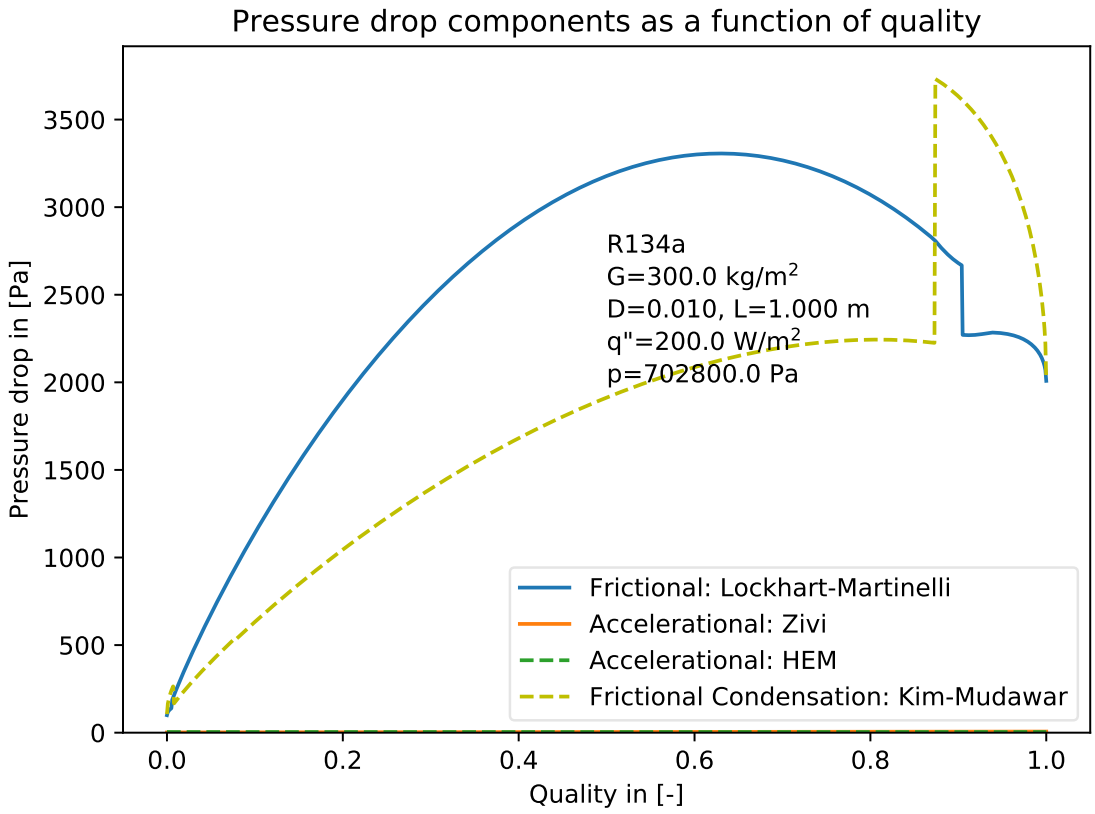
$$\Delta p_A = \int_0^L \left[-\left(\frac{\partial p}{\partial z}\right)_A dz\right] \tag{3.210}$$

$$\Delta p_A = G^2 L \left[\left(\frac{x_2^2 v_g}{\epsilon_2} + \frac{(1-x_2)^2 v_f}{1-\epsilon_2} \right) - \left(\frac{x_1^2 v_g}{\epsilon_1} + \frac{(1-x_1)^2 v_f}{1-\epsilon_1} \right) \right] \tag{3.211}$$

where Δp_A is positive if the pressure is dropping. If the quality in the term

$$\left(\frac{x^2 v_g}{\epsilon} + \frac{(1-x)^2 v_f}{1-\epsilon} \right) \tag{3.212}$$

is 0 or 1, one part is zero and the other is an indeterminate form of 0/0. One evaluation of L'Hopital's rule can be used to show that if the quality is zero, the term in Equation (3.212) is equal to v_f, or if the quality is 1, this term is equal to v_g.



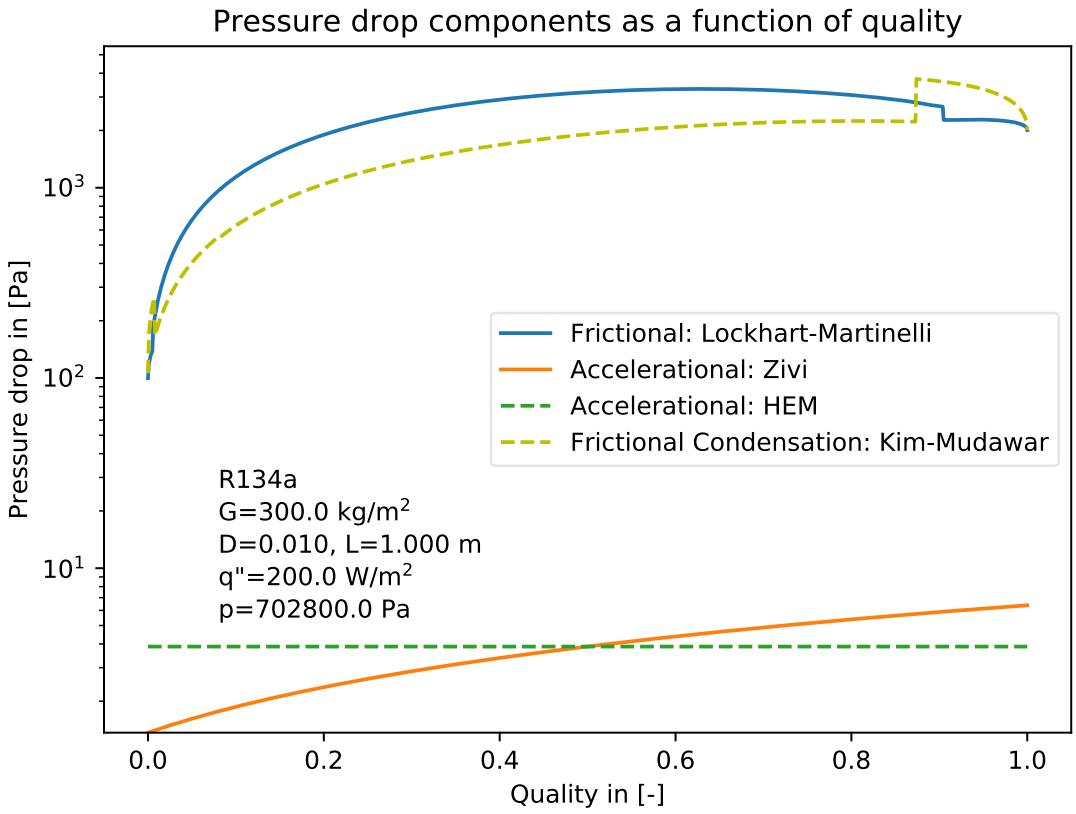
Shah Condensation

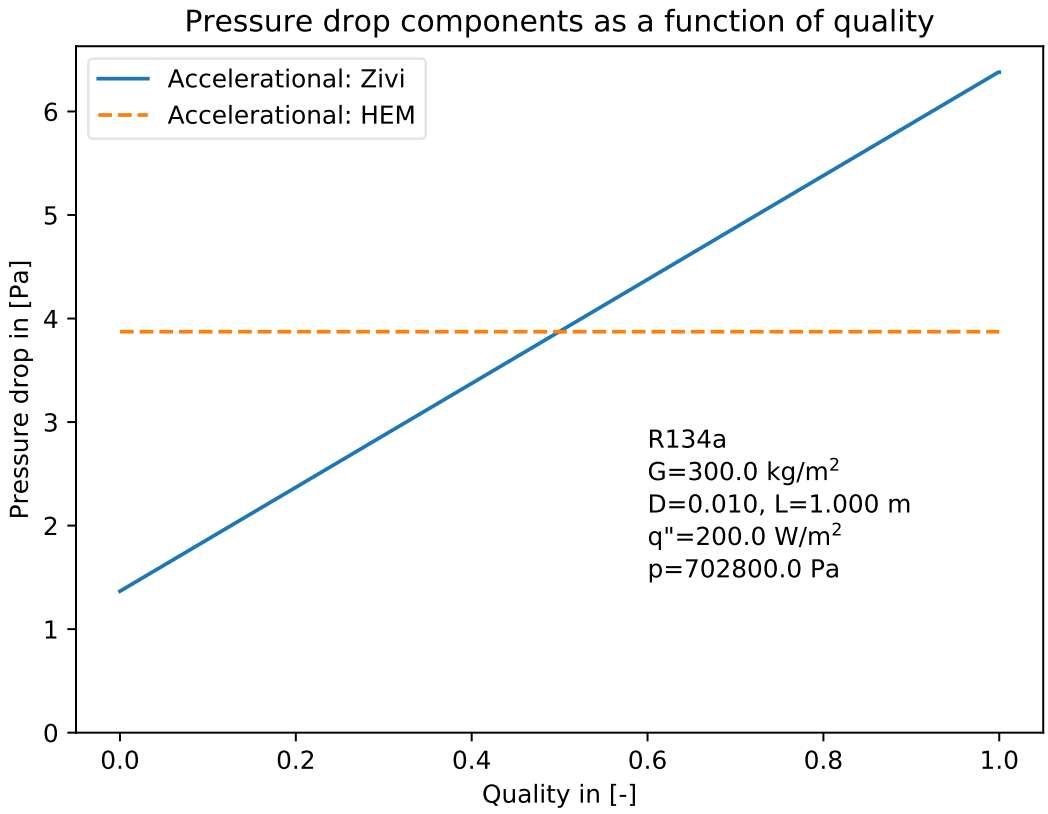
The liquid-only heat transfer coefficient is given by

$$\alpha_L = 0.023 \left(\frac{GD}{\mu_f} \right)^{0.8} \mathrm{Pr}_f^{0.4} \frac{k_f}{D} \tag{3.213}$$

And the overall heat transfer coefficient for a given quality x is given by

$$\alpha_{2\phi}(x) = \alpha_L \left((1-x)^{0.8} + \frac{3.8 x^{0.76} (1-x)^{0.04}}{(p^*)^{0.38}} \right) \tag{3.214}$$

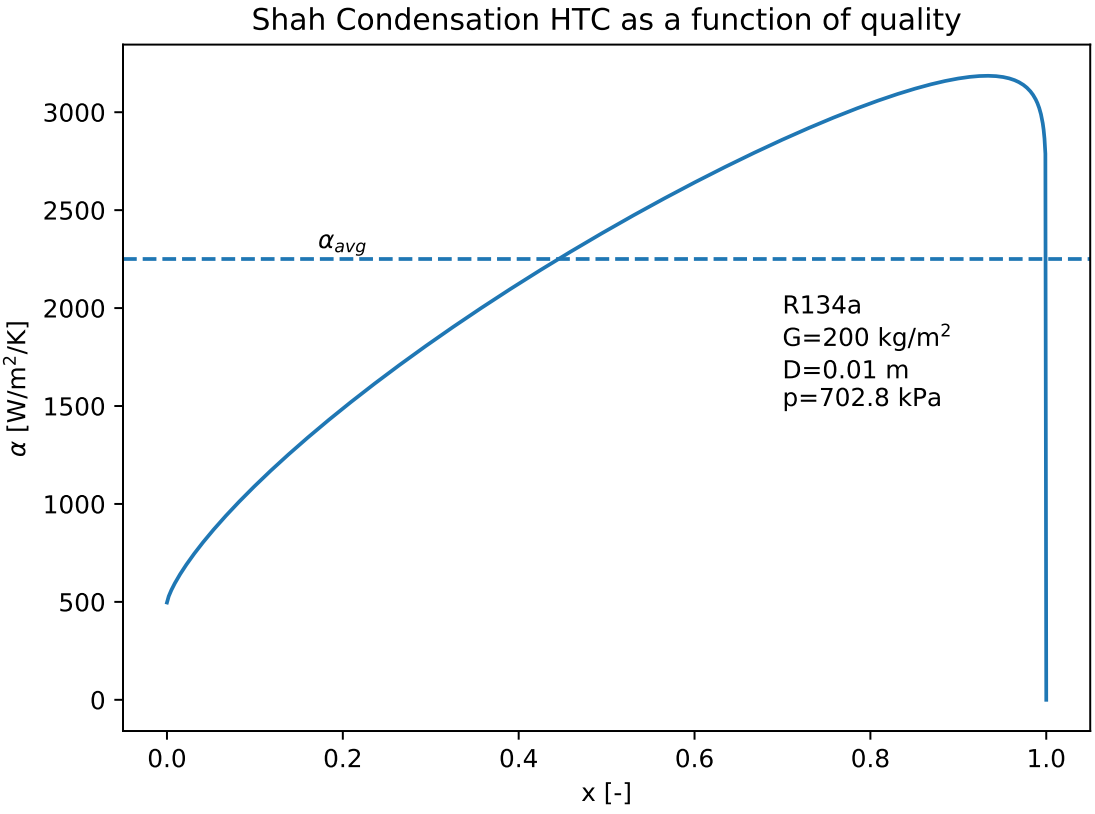

Pressure drop components as a function of quality
Pressure drop in [Pa]
10^3
10^2
10^1
Frictional: Lockhart-Martinelli
Accelerational: Zivi
Accelerational: HEM
Frictional Condensation: Kim-Mudawar
R134a
G=300.0 kg/m^2
D=0.010, L=1.000 m
q"=200.0 W/m^2
p=702800.0 Pa
0.0
0.2
0.4
0.6
0.8
1.0
Quality in [-]


Pressure drop components as a function of quality
Accelerational: Zivi
Accelerational: HEM
R134a
G=300.0 kg/m^2
D=0.010, L=1.000 m
q"=200.0 W/m^2
p=702800.0 Pa
Pressure drop in [Pa]
Quality in [-]
0
1
2
3
4
5
6
0.0
0.2
0.4
0.6
0.8
1.0

where $p^* = p_{sat}/p_{crit}$. The average condensation heat transfer coefficient between a quality of x_1 and x_2 is given by

$$\overline{\alpha_{2\phi}} = \frac{\int_{x_1}^{x_2}[\alpha_{2\phi}(x)dx]}{x_2 - x_1} \tag{3.215}$$

where the integral is evaluated numerically using adaptive quadrature. A sample plot of the heat transfer coefficient as a function of quality is shown here:



Shah Evaporation

This correlation is used to model the heat transfer coefficient for boiling fluid in a tube.

The non-dimensional groups of interest are the convection number

$$\mathrm{Co} = \left(\frac{1}{x} - 1\right)^{0.8} \sqrt{\frac{\rho_g}{\rho_f}} \tag{3.216}$$

the Froude number

$$\mathrm{Fr}_l = \frac{G^2}{\rho_f^2 g D} \tag{3.217}$$

and the boiling number

$$\mathrm{Bo} = \frac{q"}{G h_{fg}} \tag{3.218}$$

The pure-liquid heat transfer coefficient is given by

$$\alpha_l = 0.023\left(\frac{G(1-x)D}{\mu_f}\right)^{0.8}\mathrm{Pr}_f^{0.4}\frac{k_f}{D} \tag{3.219}$$

If Bo > 0.0011 then F = 14.7, otherwise F = 15.43

If $\mathrm{Fr}_l \geq 0.04$ then N = Co, else $N = 0.38\mathrm{Fr}_l^{-0.3}Co$

$$\psi_{cb} = \frac{1.8}{N^{0.8}} \tag{3.220}$$

If N is between 0.1 and 1.0 inclusive

$$\begin{aligned}\psi_{bs} &= F\sqrt{\mathrm{Bo}}\exp(2.74N^{-0.1})\\ \psi &= \max(\psi_{bs},\psi_{cb})\end{aligned} \tag{3.221}$$

If N<0.1

$$\begin{aligned}\psi_{bs} &= F\sqrt{\mathrm{Bo}}\exp(2.47N^{-0.15})\\ \psi &= \max(\psi_{bs},\psi_{cb})\end{aligned} \tag{3.222}$$

If N is *very* small in magnitude, $\exp(2.47N^{-0.15})$ blows up to infinity, so to correct, at high vapor quality, the value for the heat transfer coefficient between quality of 0.999 and 1.0 is linearly interpolated to give better behavior at very high vapor quality (which yields very small values of N). The pure vapor (x=1) heat transfer coefficient is given by

$$\alpha_g = 0.023\left(\frac{GD}{\mu_g}\right)^{0.8}\mathrm{Pr}_g^{0.4}\frac{k_g}{D} \tag{3.223}$$

If N > 1.0 and Bo > 0.00003

$$\begin{aligned}\psi_{nb} &= 230\sqrt{\mathrm{Bo}}\\ \psi &= \max(\psi_{nb},\psi_{cb})\end{aligned} \tag{3.224}$$

If N > 1.0 and Bo < 0.00003

$$\begin{aligned}\psi_{nb} &= 1.0 + 46.0\sqrt{\mathrm{Bo}}\\ \psi &= \max(\psi_{nb},\psi_{cb})\end{aligned} \tag{3.225}$$

$$\alpha_{2\phi}(x) = \psi\alpha_l \tag{3.226}$$

The average evaporation heat transfer coefficient between a quality of x_1 and x_2 is given by

$$\overline{\alpha_{2\phi}} = \frac{\int_{x_1}^{x_2}[\alpha_{2\phi}(x)dx]}{x_2 - x_1} \tag{3.227}$$

where the integral is evaluated numerically. A sample plot of the heat transfer coefficient as a function of quality is shown here:

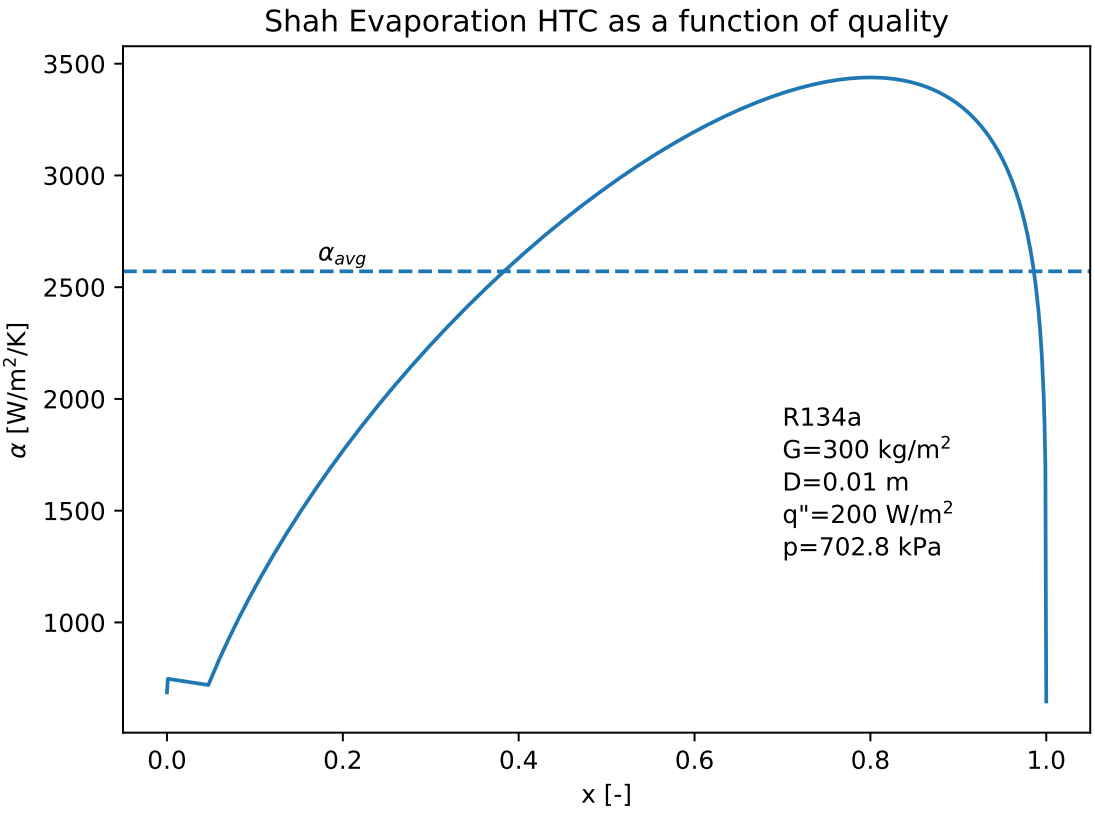
Refrigerant Charge

Using the Zivi slip flow model, the slip ratio is equal to

$$S = \left(\frac{v_g}{v_f}\right)^{1/3} \tag{3.228}$$

which yields the void fraction for a given quality of

$$\epsilon = \frac{1}{1 + \frac{\rho_g}{\rho_f}S\left(\frac{1-x}{x}\right)} \tag{3.229}$$


Shah Evaporation HTC as a function of quality
3500
3000
2500
2000
1500
1000
α [W/m²/K]
α_{avg}
R134a
G=300 kg/m²
D=0.01 m
q"=200 W/m²
p=702.8 kPa
0.0
0.2
0.4
0.6
0.8
1.0
x [-]

and the average void fraction between qualities of x_1 and x_2 can be given by

$$\overline{\epsilon} = -\frac{C_\epsilon \left(\log\left(\frac{(x_2-1)\,C_\epsilon - x_2}{(x_1-1)\,C_\epsilon - x_1} \right) + x_2 - x_1 \right) - x_2 + x_1}{(x_2 - x_1)\; C_\epsilon^2 + (2\,x_1 - 2\,x_2)\; C_\epsilon + x_2 - x_1} \tag{3.230}$$

where the term C_ϵ is given by

$$C_\epsilon = \frac{\rho_g}{\rho_f} S \tag{3.231}$$

which yields the average density in the two-phase portion of

$$\overline{\rho} = \rho_g \overline{\epsilon} + \rho_f (1 - \overline{\epsilon}) \tag{3.232}$$

Thus the total mass contained in the two-phase section is equal to

$$m = \bar{\rho} V \tag{3.233}$$

Nomenclature

Variable	Description
A	Coefficient for friction factor equation [-]
B	Coefficient for friction factor equation [-]
Bo	Boiling number [-]
C	Coefficient in L-M equation [-]
C_ϵ	Coefficient in L-M equation [-]
Co	Convection number [-]
D	Diameter [m]
f	Friction factor [-]
f_f	Friction factor [-]
f_g	Friction factor [-]
F	Coefficient in Shah Evaporation [-]
Fr_l	Froude number [-]
g	Gravitational constant [m/s^2]
G	Mass flux [kg/m^2/s]
k	Thermal conductivity [W/m/K]
k_f	Saturated liquid thermal conductivity [W/m/K]
L	Length [m]
$\dot{m}$	Mass flow rate [kg/s]
m	Mass [kg]
N	Coefficient in Shah Evaporation [-]
p	Pressure [kPa]
p^*	Reduced pressure [-]
q"	Heat flux [W/m^2]
Pr	Prandtl Number
Pr_f	Prandtl number of saturated liquid [-]
Re_D	Reynolds number based on diameter [-]
Re_f	Reynolds number of saturated liquid [-]
Re_g	Reynolds number of saturated vapor [-]
S	Slip ratio [-]
U	Average velocity [m/s]
V	Volume [m^3]
v	Specific volume [m^3/kg]

Continued on next page

Table 3.5 – continued from previous page

Variable	Description
v_f	Specific volume of saturated liquid [m^3/kg]
v_g	Specific volume of saturated vapor [m^3/kg]
w	L-M weighting parameter in transitional region [-]
x	Quality [-]
X	Lockhart-Martinelli parameter [-]
z	Position [m]
α	Heat transfer coefficient [W/m^2/K]
α_L	Liquid heat transfer coefficient [W/m^2/K]
$\alpha_{2\phi}$	Two-phase heat transfer coefficient [W/m^2/K]
ε	Surface roughness [m]
ϵ	Void fraction [-]
ϕ_g	Frictional multiplier [-]
ϕ_f	Frictional multiplier [-]
μ	Viscosity [Pa-s]
μ_f	Viscosity of saturated liquid [Pa-s]
μ_g	Viscosity of saturated vapor [Pa-s]
ψ_{bs}	Coefficient [-]
ψ_{nb}	Nucleate boiling coefficient [-]
ψ_{cb}	Convective boiling coefficient [-]
ρ	Density [kg/m^3]
ρ_f	Density of saturated liquid [kg/m^3]
ρ_g	Density of saturated vapor [kg/m^3]

3.11 Appendices

3.11.1 Condenser Area Derivation

For a configuration like the cross-flow Condenser, the area on both refrigerant and air sides is proportional to a parameter w, the length fraction for a given circuit. In addition, the air mass flow rate is proportional to the parameter w. As a result, independent of whether the minimum capacitance rate is on the air- or refrigerant-side, the same result for the area fraction w is obtained, as shown from the derivation below.

For this derivation, the inlet temperatures of both streams are known, and the outlet stream of the refrigerant is known. In addition, the mass flow rates of refrigerant and air are known. Therefore, the actual amount of heat transfer is also known.

$$\mathrm{UA} = w\frac{1}{(\eta_a\alpha_a A_{a,total})^{-1} + (\alpha_r A_{r,total})^{-1}} = w\mathrm{UA}_{overall}$$
$$C_{min} = \min[\dot{m}_r c_{p,r}, w\dot{m}_{a,total}c_{p,a}] \tag{3.234}$$

Minimum capacitance rate on air side

If C_{min} on air side:

$$\mathrm{Ntu} = \frac{\mathrm{UA}}{wc_{p,a}\dot{m}_{a,total}} = \frac{\mathrm{UA}_{overall}}{c_{p,a}\dot{m}_{a,total}} \tag{3.235}$$

w cancels out, leaving Ntu independent of w. Energy balance yields

$$\dot{m}_r c_{p,r}(T_{r,i} - T_{r,o}) = \varepsilon C_{min}(T_{r,i} - T_{a,i}) \tag{3.236}$$

$$\dot{m}_r c_{p,r}(T_{r,i} - T_{r,o}) = \varepsilon w c_{p,a} \dot{m}_{a,total}(T_{r,i} - T_{a,i}) \tag{3.237}$$

$$\frac{(T_{r,i} - T_{r,o})}{(T_{r,i} - T_{a,i})} = \varepsilon C_r \tag{3.238}$$

LHS is constant, call it Ψ. The minimum capacitance rate is on the air side ($C_{min} = w\dot{m}_{a,total}c_{p,a}$), cross-flow with C_{max} mixed (ref.) and C_{min} unmixed (air) yields

$$\varepsilon = \frac{1}{C_r}(1 - \exp(-C_r(1 - \exp(-\text{Ntu})))) \tag{3.239}$$

Now solve for C_r

$$\Psi = (1 - \exp(-C_r(1 - \exp(-\text{Ntu})))) \tag{3.240}$$

$$\exp(-C_r(1 - \exp(-\text{Ntu}))) = 1 - \Psi \tag{3.241}$$

$$-C_r(1 - \exp(-\text{Ntu})) = \ln(1 - \Psi) \tag{3.242}$$

$$C_r = -\frac{\ln(1 - \Psi)}{(1 - \exp(-\text{Ntu}))} \tag{3.243}$$

Coming back to the definition of C_r as the ratio of capacitance rates, you can get w from

$$w = C_r \frac{\dot{m}_r c_{p,r}}{\dot{m}_{a,total} c_{p,a}} \tag{3.244}$$

and since C_r is already known, you obtain

$$w = -\frac{\ln(1 - \Psi)}{(1 - \exp(-\text{Ntu}))} \frac{\dot{m}_r c_{p,r}}{\dot{m}_{a,total} c_{p,a}} \tag{3.245}$$

$$w = -\frac{\ln(1 - \Psi)}{(1 - \exp(-\text{UA}_{overall}/(c_{p,a}\dot{m}_{a,total})))} \frac{\dot{m}_r c_{p,r}}{\dot{m}_{a,total} c_{p,a}} \tag{3.246}$$

Minimum capacitance rate on refrigerant side

If C_{min} on refrigerant side:

$$\text{Ntu} = \frac{\text{UA}}{\dot{m}_r c_{p,r}} = \frac{w\text{UA}_{overall}}{\dot{m}_r c_{p,r}} \tag{3.247}$$

$$C_r\text{Ntu} = \frac{\dot{m}_r c_{p,r}}{w\dot{m}_{a,total} c_{p,a}} \frac{w\text{UA}_{overall}}{\dot{m}_r c_{p,r}} \tag{3.248}$$

$$C_r \mathrm{Ntu} = \frac{\mathrm{UA}_{overall}}{\dot{m}_{a,total} c_{p,a}} \tag{3.249}$$

Energy balance yields

$$\dot{m}_r c_{p,r}(T_{r,i} - T_{r,o}) = \varepsilon C_{min}(T_{r,i} - T_{a,i}) \tag{3.250}$$

$$\dot{m}_r c_{p,r}(T_{r,i} - T_{r,o}) = \varepsilon \dot{m}_r c_{p,r}(T_{r,i} - T_{a,i}) \tag{3.251}$$

$$\varepsilon = \frac{(T_{r,i} - T_{r,o})}{(T_{r,i} - T_{a,i})} \tag{3.252}$$

Right-hand-side is also equal to Ψ from above. Effectiveness with C_{min} mixed (ref.) and C_{max} unmixed (air) yields

$$\varepsilon = 1 - \exp(-\frac{1}{C_r}(1 - \exp(-C_r \mathrm{Ntu}))) \tag{3.253}$$

$$\exp(-\frac{1}{C_r}(1 - \exp(-C_r \mathrm{Ntu}))) = 1 - \varepsilon \tag{3.254}$$

$$-\frac{1}{C_r}(1 - \exp(-C_r \mathrm{Ntu})) = \ln(1 - \epsilon) \tag{3.255}$$

$$C_r = -\frac{(1 - \exp(-C_r \mathrm{Ntu}))}{\ln(1 - \varepsilon)} = \frac{\dot{m}_r c_{p,r}}{w \dot{m}_{a,total} c_{p,a}} \tag{3.256}$$

$$w = -\frac{\ln(1 - \varepsilon)\dot{m}_r c_{p,r}}{(1 - \exp(-C_r \mathrm{Ntu}))\dot{m}_{a,total} c_{p,a}} \tag{3.257}$$

$$w = -\frac{\ln(1 - \Psi)\dot{m}_r c_{p,r}}{[1 - \exp(-\mathrm{UA}_{overall}/(c_{p,a}\dot{m}_{a,total}))]\dot{m}_{a,total} c_{p,a}} \tag{3.258}$$

Thus both assuming that the minimum capacitance rate is on the air- or refrigerant-sides yields exactly the same solution, which conveniently allows for an explicit solution independent of whether the air-side is the limiting capacitance rate or not

3.11.2 Algorithm of partial-wet-partial-dry evaporator

This derivation was provided by Howard Cheung but was originally carried out by Jim Braun. Editorial modifications have been made by Ian Bell.

Introduction

Partial-wet-partial-dry analysis is the technique given in Braun[1] used to estimate the performance of an air-to-refrigerant evaporating coil with a higher accuracy for estimating the latent heat transfer than completely wet or dry analysis. The methodology is to divide the heat exchanger into two sections by locating the point on the heat exchanger surface where the dewpoint is reached. For the part with surface temperature below dewpoint, they are lumped together for a completely wet analysis. The remaining one is lumped together as a dry section.

To illustrate the idea, the document gives an example on a steady counterflow air-to-refrigerant heat exchanger in which air is cooled and moisture is condensed on the heat exchanger surface. The governing equations are listed based on the assumptions, and an algorithm is derived to solve the implicit mathematical model for the analysis.

Background

In the following analysis, the following assumptions are employed:

- Steady state
- Counterflow heat exchanger
- Single-phase fluid flow on both side
- Constant specific heat throughout the entire heat exchanger
- Constant heat transfer coefficient for air-to-surface and surface-to-refrigerant heat transfer
- Coil completely covered with condensate for wet section
- Unity Lewis number

The analysis can be divided into three parts:

1. Completely dry analysis
2. Completely wet analysis
3. Partial-wet-partial-dry analysis

Initially, the dry analysis is implemented. Should the surface temperature at the air outlet be higher than the dewpoint of the air, the dry coil assumption is accepted. Otherwise the wet analysis is carried out on the entire heat exchanger and the surface temperature at the air inlet is examined. If the temperature is lower than dewpoint, the completely wet coil assumption is accepted. If both assumptions are rejected, there must exist a point on the heat exchanger where the temperature is at the dewpoint and the partial-wet-partial-dry analysis is implemented.

Completely Dry Analysis

To conduct the competely dry analysis, a simple $\varepsilon - \mathrm{Ntu}$ method on a counterflow heat exchanger is used. The governing equations of the $\varepsilon - \mathrm{Ntu}$ method are listed as Equations (3.259) to (3.266).

$$\dot{Q}_{dry} = \dot{m}_r c_{p,r}(T_{r,out} - T_{r,in}) \tag{3.259}$$

$$\dot{Q}_{dry} = \varepsilon C_{min}(T_{a,in} - T_{r,in}) \tag{3.260}$$

$$\varepsilon = \frac{1 - \exp(-\mathrm{Ntu}_{dry}(1 - C_{ratio}))}{1 - C_{ratio}\exp(-\mathrm{Ntu}_{dry}(1 - C_{ratio}))} \tag{3.261}$$

[1] Braun, J. E., 1988. Methodologies for the Design and Control of Central Cooling Plants. Ph.D. thesis, University of Wisconsin - Madison.

$$\mathrm{Ntu}_{dry} = \frac{\mathrm{UA}_o}{C_{min}} \tag{3.262}$$

$$\frac{1}{\mathrm{UA}_o} = \frac{1}{U_a A_a} + \frac{1}{U_r A_r} \tag{3.263}$$

$$\begin{aligned} C_{min} &= \min[\dot{m}_a c_{p,a}, \dot{m}_r c_{p,r}] \\ C_{max} &= \max[\dot{m}_a c_{p,a}, \dot{m}_r c_{p,r}] \end{aligned} \tag{3.264}$$

$$C_{ratio} = \frac{C_{min}}{C_{max}} \tag{3.265}$$

$$\dot{Q} = \dot{m}_a c_{p,a}(T_{a,in} - T_{a,out}) \tag{3.266}$$

These equations can be solved analytically for the heat exchanger performance. After solving the heat exchanger, one may find the temperature on the surface of the heat exchanger at the air outlet by Equation (3.267).

$$U_a A_a (T_{a,out} - T_{s,a,out}) = U_r A_r (T_{s,a,out} - T_{r,in}) \tag{3.267}$$

If the temperature $T_{s,a,out}$ is higher than dewpoint of inlet air, the coil is said to be dry and the heat exchanger performance analysis is completed. Otherwise completely wet analysis is conducted.

Completely Wet Analysis

In the case of wet analysis, the unity Lewis number is used such that the temperatures in the $\varepsilon - \mathrm{Ntu}$ method are converted to the corresponding air-water mixture enthalpies to account for the condensation of moisture from air on the heat exchanger surface. The $\varepsilon - \mathrm{Ntu}$ method is modified to form governing equations listed from Equation (3.268) to (3.277).

$$\dot{Q}_{wet} = \varepsilon^* \dot{m}_{min}(h_{a,in} - h_{sat,r,in}) \tag{3.268}$$

$$\varepsilon^* = \frac{1 - \exp(-Ntu^*(1 - \dot{m}_{ratio}))}{1 - \dot{m}_{ratio} \exp(-Ntu^*(1 - \dot{m}_{ratio}))} \tag{3.269}$$

$$Ntu^* = \frac{UA_o^*}{\dot{m}_{min}} \tag{3.270}$$

$$\dot{m}_{min} = min(\dot{m}_a, \dot{m}_r \frac{c_{p,r}}{c_s}) \tag{3.271}$$

$$\dot{m}_{ratio} = \frac{\dot{m}_{min}}{max(\dot{m}_a, \dot{m}_r \frac{c_{p,r}}{c_s})} \tag{3.272}$$

$$c_s = \frac{dh_{sat}}{dT}|_{T=\frac{T_{r,in}+T_{r,out}}{2}} \tag{3.273}$$

$$\frac{1}{UA_o^*} = \frac{c_{p,a}}{U_a^* A_a} + \frac{c_s}{U_r A_r} \tag{3.274}$$

$$h_{s,sat,eff} = h_{a,in} + \frac{h_{a,out} - h_{a,in}}{1 - \exp(-\frac{U_a^* A_a}{\dot{m}_a})} \tag{3.275}$$

$$T_{a,out} = T_{s,eff} + (T_{a,in} - T_{s,eff}) \exp(-\frac{U_a A_a}{\dot{m}_a c_{p,a}}) \tag{3.276}$$

$$\dot{Q}_{wet} = \dot{m}_a (h_{a,in} - h_{a,out}) \tag{3.277}$$

These equations are formed explicitly and can be solved analytically for the performance of the heat exchanger. The wet coil assumption is verified by comparing the heat exchanger surface temperature with the dewpoint of inlet air. The temperature can be calculated by Equation (3.278) which the definition of $c_{s,local}$ is given in Equation (3.279).

$$U_r A_r (T_{s,in} - T_{r,out}) = UA_o^* (c_{s,local})(h_{a,in} - h_{sat,r,out}) \tag{3.278}$$

$$c_{s,local} = \frac{dh_{sat}}{dT}|_{T=\frac{T_{a,in}+T_{r,out}}{2}} \tag{3.279}$$

If the temperature is lower than the dewpoint, the completely wet coil assumption is accepted. Otherwise the calculation will proceed to the next part: partial-wet-partial-dry analysis.

Partial-Wet-Partial-Dry Analysis

The partial-wet-partial-dry analysis divides the heat exchanger into two regions: a wet region and a dry region. The heat transfer rate can be divided into two parts as shown in Equation (3.280):

$$\dot{Q} = \dot{Q}_{f,dry} + \dot{Q}_{f,wet} \tag{3.280}$$

Both sections can be addressed based on $\varepsilon - \mathrm{Ntu}$ method except that a separate set of governing equations are used for each section. The dry section can be described as lists of equations from Equation (3.281) to (3.284).

$$\dot{Q}_{f,dry} = \varepsilon_{f,dry} C_{min} (T_{a,in} - T_{r,x}) \tag{3.281}$$

$$\dot{Q}_{f,dry} = \dot{m}_a c_{p,a} (T_{a,in} - T_{a,x}) \tag{3.282}$$

$$\dot{Q}_{f,dry} = \dot{m}_r c_{p,r} (T_{r,out} - T_{r,x}) \tag{3.283}$$

$$\varepsilon_{f,dry} = \frac{1 - \exp(-f_{dry} \mathrm{Ntu}_{dry} (1 - C_{ratio}))}{1 - C_{ratio} \exp(-f_{dry} \mathrm{Ntu}_{dry} (1 - C_{ratio}))} \tag{3.284}$$

The wet region is governed by a similar set of equation with $\varepsilon - \mathrm{Ntu}$ method as listed from Equation (3.285) to (3.290).

$$\dot{Q}_{f,wet} = \varepsilon_{f,wet}\dot{m}_{min}(h_{a,x} - h_{sat,r,in}) \tag{3.285}$$

$$\dot{Q}_{f,wet} = \dot{m}_a(h_{a,x} - h_{a,out}) \tag{3.286}$$

$$\dot{Q}_{f,wet} = \dot{m}_r c_{pr}(T_{r,x} - T_{r,in}) \tag{3.287}$$

$$h_{f,s,sat,eff} = h_{a,x} + \frac{h_{a,out} - h_{a,x}}{1 - \exp(-\frac{(1-f_{dry})U_a^* A_a}{\dot{m}_a})} \tag{3.288}$$

$$T_{a,out} = T_{f,s,eff} + (T_{a,x} - T_{f,s,eff})\exp(-\frac{U_a A_a}{\dot{m}_a c_{p,a}}) \tag{3.289}$$

$$\varepsilon_{f,wet} = \frac{1 - \exp(-(1-f_{dry})\mathrm{Ntu}_{wet}(1-\dot{m}^*))}{1 - \dot{m}^*\exp(-(1-f_{dry})\mathrm{Ntu}_{wet}(1-\dot{m}^*))} \tag{3.290}$$

At the intersection between the dry and wet region, the heat transfer is governed as Equation (3.291).

$$\mathrm{UA}_o(T_{a,x} - T_{r,x}) = U_a A_a(T_{a,x} - T_{dp}) \tag{3.291}$$

Unlike the previous analyses, these equations cannot be solved analytically because only the inlet conditions of refrigerant and air are known. To solve equations iteratively, a bounded solver on f_{dry} can be used because f_{dry} is proved to be between 0 and 1 from the previous analysis. One way is to calculate the refrigerant outlet temperature based on the dry region only and on both wet and dry region and compare the error between the two methods. Should the error close to zero, the heat exchanger is solved. In this case, one can derive a function which when the solution is reached, the function equals to zero as Equation (3.292).

$$g(x_{true}) = 0 \tag{3.292}$$

The function g is defined as Equation (3.293) as a function of f_{dry} and a solution of f_{dry} is said to be found if g equals some value very close to zero.

$$g(f_{dry}) = T_{r,out}(f_{dry})|_{\text{from dry region only}} - T_{r,out}(f_{dry})|_{\text{from both regions}} \tag{3.293}$$

The following subsections describe how to find $T_{r,out}(f_{dry})|_{\text{from dry region only}}$ and $T_{r,out}(f_{dry})|_{\text{from both regions}}$.

Solving Refrigerant Outlet Temperature with Dry Region Only

With the dry region only, a simplification is first conducted on Equation (3.284).

$$B = \exp(-f_{dry}\mathrm{Ntu}_{dry}(1 - C_{ratio})) \tag{3.294}$$

$$B = \frac{1 - \varepsilon_{f,dry}}{1 - C_{ratio}\varepsilon_{f,dry}} \tag{3.295}$$

A combination of of Equations (3.281) and (3.283) is done.

$$\dot{m}_r c_{p,r}(T_{r,out} - T_{r,x}) = \varepsilon_{f,dry} C_{min}(T_{a,in} - T_{r,x}) \tag{3.296}$$

$$T_{r,x} = \frac{\dot{m}_r c_{p,r} T_{r,out} - \varepsilon_{f,dry} C_{min} T_{a,in}}{\dot{m}_r c_{p,r} - \varepsilon_{f,dry} C_{min}} \tag{3.297}$$

The terms in Equation (3.291) can be rearranged to Equation (3.298).

$$T_{a,x} = \frac{U_a A_a T_{r,x} - \mathrm{UA}_o T_{dp}}{U_a A_a - \mathrm{UA}_o} \tag{3.298}$$

The Equations (3.281) and (3.282) can also be combined together to form Equation (3.300) through Equation (3.299).

$$\dot{m}_a c_{p,a}(T_{a,in} - T_{a,x}) = \varepsilon_{f,dry} C_{min}(T_{a,in} - T_{r,x}) \tag{3.299}$$

$$\varepsilon_{f,dry} = \frac{\dot{m}_a c_{p,a}}{C_{min}} \frac{T_{a,in} - T_{a,x}}{T_{a,in} - T_{r,x}} \tag{3.300}$$

Equation in (3.297) is re-arranged to form Equation (3.302) through the step in Equation (3.301).

$$T_{a,in} - T_{r,x} = T_{a,in} - \frac{\dot{m}_r c_{p,r} T_{r,out} - \varepsilon_{f,dry} C_{min} T_{a,in}}{\dot{m}_r c_{p,r} - \varepsilon_{f,dry} C_{min}} \tag{3.301}$$

$$T_{a,in} - T_{r,x} = \frac{\dot{m}_r c_{p,r}}{\dot{m}_r c_{p,r} - \varepsilon_{f,dry} C_{min}}(T_{a,in} - T_{r,out}) \tag{3.302}$$

Equation (3.298) can also be arranged in a similar form as Equation (3.304).

$$T_{a,in} - T_{a,x} = T_{a,in} - \frac{U_a A_a T_{r,out} - \mathrm{UA}_o T_{dp}}{U_a A_a - \mathrm{UA}_o} \tag{3.303}$$

$$T_{a,in} - T_{a,x} = \frac{U_a A_a (T_{a,in} - T_{r,x}) - \mathrm{UA}_o (T_{a,in} - T_{dp})}{U_a A_a - \mathrm{UA}_o} \tag{3.304}$$

Dividing Equation (3.304) by Equation (3.302) can construct Equation (3.305).

$$\frac{T_{a,in} - T_{a,x}}{T_{a,in} - T_{r,x}} = \frac{U_a A_a - \mathrm{UA}_o \frac{T_{a,in} - T_{dp}}{T_{a,in} - T_{r,x}}}{U_a A_a - \mathrm{UA}_o} \tag{3.305}$$

The terms in Equation (3.302) can be arranged again as Equation (3.306) so the left-hand side of the equation is the same as Equation (3.305).

$$\frac{T_{a,in} - T_{a,x}}{T_{a,in} - T_{r,x}} = \frac{\dot{m}_r c_{p,r} - \varepsilon_{f,dry} C_{min}}{\dot{m}_r c_{p,r}} \frac{T_{a,in} - T_{dp}}{T_{a,in} - T_{r,out}} \tag{3.306}$$

An Ntu_o can be defined with UA_o as Equation (3.307).

$$\mathrm{Ntu}_o = \frac{\mathrm{UA}_o}{\dot{m}_a c_{p,a}} \tag{3.307}$$

The Ntu_o from Equation (3.307) is substituted into Equation (3.305) together with (3.262) to form (3.308) so that the equation can be defined in terms of Ntus rather than UAs.

$$\frac{T_{a,in}-T_{a,x}}{T_{a,in}-T_{r,x}}=\frac{C_{min}\mathrm{Ntu}_{dry}-\dot{m}_a c_{p,a}\mathrm{Ntu}_o\frac{T_{a,in}-T_{dp}}{T_{a,in}-T_{r,x}}}{C_{min}\mathrm{Ntu}_{dry}-\dot{m}_a c_{p,a}\mathrm{Ntu}_o} \tag{3.308}$$

For convenience, another dimensionless variable Γ is defined in Equation (3.309).

$$\Gamma=\frac{T_{a,in}-T_{a,x}}{T_{a,in}-T_{r,x}} \tag{3.309}$$

By combining Equations (3.306) and (3.308) together, one can express Γ in Equation (3.309) as Equation (3.310).

$$\Gamma=\frac{C_{min}\mathrm{Ntu}_{dry}-\mathrm{Ntu}_o\dot{m}_a c_{p,a}\frac{\dot{m}_r c_{p,r}-\varepsilon_{f,dry}C_{min}}{\dot{m}_r c_{p,r}}\frac{T_{a,in}-T_{dp}}{T_{a,in}-T_{r,out}}}{C_{min}\mathrm{Ntu}_{dry}-\mathrm{Ntu}_o\dot{m}_a c_{p,a}} \tag{3.310}$$

Equation (3.309) can also be formulated as Equation (3.311) with Equation (3.302).

$$\Gamma=\frac{C_{min}\varepsilon_{f,dry}}{\dot{m}_a c_{p,a}} \tag{3.311}$$

Equations (3.310) and (3.311) can be equated together and by rearranging the subject as $\varepsilon_{f,dry}$, one can form Equation (3.312).

$$\varepsilon_{f,dry}=\frac{\dot{m}_r c_{p,r}\dot{m}_a c_{p,a}[C_{min}\mathrm{Ntu}_{dry}(T_{a,in}-T_{r,out})-\dot{m}_a c_{p,a}\mathrm{Ntu}_o(T_{a,in}-T_{dp})]}{\dot{m}_r c_{p,r}C_{min}^2\mathrm{Ntu}_{dry}(T_{a,in}-T_{r,out})-C_{min}\mathrm{Ntu}_o\dot{m}_a c_{p,a}[\dot{m}_r c_{p,r}(T_{a,in}-T_{r,out})+\dot{m}_a c_{p,a}(T_{a,in}-T_{dp})]} \tag{3.312}$$

Further calculation can change the subject of Equation (3.312) as the form of the right hand side of Equation (3.295) to establish Equation (3.313).

$$\begin{aligned}
\frac{1-\varepsilon_{f,dry}}{1-C_{ratio}\varepsilon_{f,dry}}&=\frac{\Xi_1+\Xi_{02}}{\Xi_{03}+\Xi_{04}}\\
\Xi_{01}&=(\dot{m}_a c_{p,a})^2\mathrm{Ntu}_o(\dot{m}_r c_{p,r}-C_{min})(T_{a,in}-T_{dp})\\
\Xi_{02}&=C_{min}\dot{m}_r c_{p,r}[\mathrm{Ntu}_{dry}(C_{min}-\dot{m}_a c_{p,a})+\mathrm{Ntu}_o\dot{m}_a c_{p,a}](T_{a,in}-T_{r,out})\\
\Xi_{03}&=C_{min}\dot{m}_r c_{p,r}[\mathrm{Ntu}_{dry}(C_{min}-C_{ratio}\dot{m}_a c_{p,a})-\mathrm{Ntu}_o\dot{m}_a c_{p,a}](T_{a,in}-T_{r,out})\\
\Xi_{04}&=(\dot{m}_a c_{p,a})^2\mathrm{Ntu}_o[C_{ratio}\dot{m}_r c_{p,r}-C_{min}](T_{a,in}-T_{dp})
\end{aligned} \tag{3.313}$$

Equation (3.295) can also be rearranged as Equation (3.314).

$$f_{dry}=-\frac{1}{(1-C_{ratio})\mathrm{Ntu}_{dry}}\ln B \tag{3.314}$$

Another term can be defined as K in Equation (3.315).

$$K=(1-C_{ratio})\mathrm{Ntu}_{dry} \tag{3.315}$$

The subject in Equation (3.313) can be written as B from Equation (3.295) to Equation (3.316).

$$B=\frac{\Xi_1+\Xi_{02}}{\Xi_{03}+\Xi_{04}} \tag{3.316}$$

using the definitions from Equation (3.313). Equation (3.316) can be simplified should C_{ratio} and C_{min} be known. If $C_{min}=\dot{m}_a c_{p,a}$, Equation (3.316) can be written as Equation (3.317).

$$B=\frac{(T_{dp}-T_{r,out})+C_{ratio}(T_{a,in}-T_{dp})}{[1-\frac{\mathrm{Ntu}_{dry}}{\mathrm{Ntu}_o}(1-C_{ratio})](T_{a,in}-T_{r,out})} \tag{3.317}$$

With the definitions of K and f_{dry} in Equations (3.314) and (3.315), one can write Equation (3.317) into Equation (3.318).

$$f_{dry} = -\frac{1}{K} \ln \frac{(T_{dp} - T_{r,out}) + C_{ratio}(T_{a,in} - T_{dp})}{[1 - \frac{K}{\text{Ntu}_o}](T_{a,in} - T_{r,out})} \tag{3.318}$$

The refrigerant outlet temperature in this case can be calculated as Equation (3.319) from Equation (3.318).

$$T_{r,out} = \frac{T_{dp} - \exp(-K f_{dry})(1 - \frac{K}{\text{Ntu}_o})T_{a,in} + C_{ratio}(T_{a,in} - T_{dp})}{1 - \exp(-K f_{dry})(1 - \frac{K}{\text{Ntu}_o})} \tag{3.319}$$

Similarly, when $C_{min} = \dot{m}_r c_{p,r}$, Equation (3.316) will be written as Equation (3.320).

$$B = \frac{C_{ratio}[1 + \frac{\text{Ntu}_{dry}}{\text{Ntu}_o}(1 - C_{ratio})](T_{ain} - T_{r,out})}{C_{ratio}(T_{dp} - T_{r,out}) + (T_{a,in} - T_{dp})} \tag{3.320}$$

Similar derivation can be made on f_{dry} and $T_{r,out}$ in this case to form Equations (3.321) and (3.322).

$$f_{dry} = -\frac{1}{K} \ln \frac{C^*[1 + \frac{K}{\text{Ntu}_o}](T_{a,in} - T_{r,out})}{C_{ratio}(T_{dp} - T_{r,out}) + (T_{a,in} - T_{dp})} \tag{3.321}$$

$$T_{r,out} = \frac{\exp(-K f_{dry})[T_{a,in} + (C_{ratio} - 1)T_{dp}] - C_{ratio}(1 + \frac{K}{\text{Ntu}_o})T_{a,in}}{C_{ratio} \exp(-K f_{dry}) - C_{ratio}(1 + \frac{K}{\text{Ntu}_o})} \tag{3.322}$$

$T_{r,out}(f_{dry})|_{\text{from dry region only}}$ in Equation (3.293) can then be solved by Equations (3.315), (3.319) and (3.322), depending on the value of C_{min}.

Solving Refrigerant Outlet Temperature with Both Regions

While the method to find $T_{r,out}(f_{dry})|_{\text{from dry region only}}$ is depicted in the previous section, the solution of $T_{r,out}(f_{dry})|_{\text{from both regions}}$ is described below. Equations (3.281) and (3.282) can be combined to yield

$$T_{a,x} = T_{a,i} - \varepsilon_{f,dry} \frac{C_{min}}{\dot{m}_a c_{p,a}} (T_{a,in} - T_{r,x}) \tag{3.323}$$

Because the specific heat of air is taken to be constant, $\Delta h = c_p \Delta T$. Thus, the enthalpy of the air at the wet-dry interface can be given by

$$h_{a,x} = h_{a,i} - \varepsilon_{f,dry} \frac{C_{min}}{\dot{m}_a} (T_{a,in} - T_{r,x}) \tag{3.324}$$

Through the use of Equation (3.324), the $\varepsilon - \text{Ntu}$ equation on the wet region Equation (3.285) can be expressed as

$$\dot{Q}_{f,wet} = \varepsilon_{f,wet} \dot{m}_{min} (h_{a,in} - \varepsilon_{f,dry} \frac{C_{min}}{\dot{m}_a} (T_{a,in} - T_{r,x}) - h_{sat,r,in}) \tag{3.325}$$

The value of $\dot{Q}_{f,wet}$ can be substituted from Equation (3.287) which yields the value for $T_{r,x}$ of

$$T_{r,x} = \frac{T_{r,in} + \varepsilon_{f,wet} \frac{\dot{m}_{min}}{\dot{m}_r c_{p,r}} (h_{a,in} - h_{s,r,in} - \varepsilon_{f,dry} \frac{C_{min}}{\dot{m}_a} T_{a,in})}{1 - \varepsilon_{f,wet} \varepsilon_{f,dry} \frac{C_{min} \dot{m}_{min}}{\dot{m}_r c_{p,r} \dot{m}_a}} \tag{3.326}$$

$T_{r,out}$ from this method can be written (by combining Equations (3.281) and (3.283)) as

$$T_{r,out} = \varepsilon_{f,dry} \frac{C_{min}}{\dot{m}_r c_{p,r}} T_{a,in} + (1 - \varepsilon_{f,dry} \frac{C_{min}}{\dot{m}_r c_{p,r}}) T_{r,x} \tag{3.327}$$

By solving Equations (3.284), (3.290), (3.326) and (3.327), one can find $T_{r,out}(f_{dry})|_{\text{from both regions}}$ in Equation (3.293). The $g(f_{dry})$ in Equation (3.293) can be found for diffferent f_{dry} and the one which gives a function value close to zero is the numerical solution of f_{dry}. With the value of f_{dry}, all other variables in the partial-dry-partial-wet analysis can be computed and the heat exchanger performance can be solved.

Nomenclature

Variable	Description
A	Surface Area [m^2]
B	Dimensionless variable [–]
C	Capacity Rate [W/K]
c_p	Specific Heat Capacity [J/kg/K]
c_s	Analogous specific heat capacity for air-water enthalpy [J/kg/K]
f	Proportion of dry section [–]
g	Function to be solved [any]
h	Air-water mixture enthalpy [J/kg_{ha}]
K	Dimensionless variable [–]
$\dot{m}$	Mass Flow Rate [kg/s]
Ntu	Number of transfer unit [–]
$\mathrm{Ntu_o}$	Overall Number of transfer units [–]
$\dot{Q}$	Heat Transfer Rate [W]
T	Temperature [K]
U	Heat Transfer Coefficient [$W/m^2/K$]
UA_o	Overall Heat Conductance [W/K]
UA_o^*	Overall Heat and Mass Transfer Conductance [W/K]
x_{true}	Solution of $g(x) = 0$ [any]
ε	Heat Exchanger Effectiveness [–]
Γ	Dimensionless variable [–]

Sub-script/Superscript	Description
a	Of air side
dp	Of dewpoint
dry	Of dry region
eff	Effective
f	Of partial condition
in	At inlet
$local$	Localized
min	Minimum
out	At outlet
r	Of refrigerant side; in the case of h, it means the air-water enthalpy at the temperature of the refrigerant
$ratio$	Of ratio
s	Of surface
sat	At saturation
wet	Of wet region
x	At intersection
$*$	Of/Adjusted for mass transfer

Python Module Index

a

ACHP.Compressor, 42
ACHP.Condenser, 70
ACHP.CoolingCoil, 72
ACHP.Cycle, 25
ACHP.DryWetSegment, 62
ACHP.Evaporator, 75
ACHP.LineSet, 82
ACHP.MultiCircuitEvaporator, 79

Index

A

ACHP.Compressor (module), 42
ACHP.Condenser (module), 70
ACHP.CoolingCoil (module), 72
ACHP.Cycle (module), 19, 25
ACHP.DryWetSegment (module), 62
ACHP.Evaporator (module), 75
ACHP.LineSet (module), 82
ACHP.MultiCircuitEvaporator (module), 79
AirSideCalcs() (ACHP.Evaporator.EvaporatorClass method), 75

C

Calculate() (ACHP.Compressor.CompressorClass method), 42
Calculate() (ACHP.Condenser.CondenserClass method), 70
Calculate() (ACHP.CoolingCoil.CoolingCoilClass method), 72
Calculate() (ACHP.Cycle.DXCycleClass method), 19
Calculate() (ACHP.Cycle.SecondaryCycleClass method), 25
Calculate() (ACHP.Evaporator.EvaporatorClass method), 75
Calculate() (ACHP.LineSet.LineSetClass method), 82
Calculate() (ACHP.MultiCircuitEvaporator.MultiCircuitEvaporatorClass method), 79
CompressorClass (class in ACHP.Compressor), 42
CondenserClass (class in ACHP.Condenser), 70
CoolingCoilClass (class in ACHP.CoolingCoil), 72

D

DryWetSegment() (in module ACHP.DryWetSegment), 62
DWSVals (class in ACHP.DryWetSegment), 62
DXCycleClass (class in ACHP.Cycle), 19

E

EvaporatorClass (class in ACHP.Evaporator), 75

I

Initialize() (ACHP.CoolingCoil.CoolingCoilClass method), 72
Initialize() (ACHP.Evaporator.EvaporatorClass method), 75

L

LineSetClass (class in ACHP.LineSet), 82

M

MultiCircuitEvaporatorClass (class in ACHP.MultiCircuitEvaporator), 79

O

OutputList() (ACHP.Compressor.CompressorClass method), 42
OutputList() (ACHP.Condenser.CondenserClass method), 70
OutputList() (ACHP.CoolingCoil.CoolingCoilClass method), 72
OutputList() (ACHP.Cycle.DXCycleClass method), 19
OutputList() (ACHP.Cycle.SecondaryCycleClass method), 25
OutputList() (ACHP.Evaporator.EvaporatorClass method), 75
OutputList() (ACHP.LineSet.LineSetClass method), 82
OutputList() (ACHP.MultiCircuitEvaporator.MultiCircuitEvaporatorClass method), 79

P

PreconditionedSolve() (ACHP.Cycle.DXCycleClass method), 19
PreconditionedSolve() (ACHP.Cycle.SecondaryCycleClass method), 25

S

SecondaryCycleClass (class in ACHP.Cycle), 25

U

Update() (ACHP.Compressor.CompressorClass method), 43
Update() (ACHP.Condenser.CondenserClass method), 70
Update() (ACHP.CoolingCoil.CoolingCoilClass method), 72
Update() (ACHP.Evaporator.EvaporatorClass method), 75
Update() (ACHP.LineSet.LineSetClass method), 82
Update() (ACHP.MultiCircuitEvaporator.MultiCircuitEvaporatorClass method), 79