A Python-based code for modeling the thermodynamics of the vapor compression cycle applied to residential heat pumps

Rebecca Allen[a,]*, Eirik Svortevik[a], Henrik Bergersen[a]

[a]*Oslo Metropolitan University, Norway*
*rebeccaa@oslomet.no

Abstract

Heat pumps are an attractive heating system in residential buildings. They operate based on the vapor compression cycle used in refrigeration systems. Design questions surrounding heat pumps can be investigated and answered using modelling tools that incorporate the necessary thermodynamics, fluid mechanics, and machinery component efficiency. Several modelling tools are available, however there is a need for more open-source, script-based programs that are competitive to those already available. This work presents a Python-based code for modeling the thermodynamics of the vapor compression cycle (VCC) in typical heat pumps. The main contribution of this work is an openly available online code, complete with a few examples to show its functionality, that provides the basic thermodynamic model of a heat pump for researchers or development engineers to use, modify, and extend. Its current features include choice of refrigerant, heat exchanger size and characteristics, compressor, and other design parameters such as heating load, and fluid temperatures in and out of the heat exchangers. Simulation outputs include the P-h and T-s diagrams and coefficient of performance (COP). The code is flexible and suggestions for future code development are given.

1. Introduction

Residential heat pumps are an attractive alternative to electrical space heating because more units of heat energy can be transferred while consuming the same amount of electrical power. Indeed, heat pump technology is not new: decades worth of research efforts have been done by, for example, the International Energy Agency (IEA) Heat Pumping Technologies (HPT). Since the heat pump cycle and refrigeration cycle are essentially the same cycle (just with different objective: one for heating, the other for cooling), it is more concise to use the term vapor-compression cycle, or VCC. The VCC is comprised of 4 main thermodynamic processes, and the central part is to transfer heat energy from a low-temperature reservoir to a high-temperature reservoir. An excellent sketch of the components is presented in Figure 2 of Jensen et al. (2018). Heat is "pumped" by first compressing a working fluid (also called refrigerant) in its vapor phase which increases its pressure and temperature, and then exchanging heat energy to another fluid in a secondary loop via both temperature drop and phase change through a condenser. The refrigerant, in its liquid phase, is then throttled, which means it passes through a throttle valve (also known as expansion valve), reducing its pressure and temperature until it becomes a saturated mixture. The mixture then passes through an evaporator where it absorbs heat energy from another secondary fluid loop and comes out in its initial vapor phase state.

The thermodynamics are well-understood in the VCC, however there is a need for flexible tools that can model these thermodynamic processes with the purpose of answering research questions such as optimal cycle "position" and "lift" given certain operating conditions. Other programs and codes have been developed and are available (Aulicino and Bakrania, 2022; Bell *et al.*, 2014; Vering *et al.*, 2022). However there is a need for more models that are open-source, script-based (for example, JavaScript or Python) with a framework that allows for application to one's own engineering design problem in addition to extendibility. The programming tools cannot be like a "black-box". And good documentation of code, and guidance on how to extend or modify the code ought to be available.

The objective of this work is to start an open-source repository for VCC modelling using Python scripting language. There is a great deal of potential when coding in Python, given its wide online community, wealth of libraries and modules or packages such as optimization and machine learning ones, and the fact that some engineering companies choose to develop heat pump dimensioning tools using Python.

The rest of this paper is organized as follows: the methodology section presents the typical modelling equations used to simulate the refrigeration cycle and a description of how the model is implemented in Python, the results section presents three examples to demonstrate the use of the code and gives suggestions for how it can be extended, and the summary section offers ways this code could be developed further and applied to research on heat pump modelling.

2. Methodology

This work includes three examples that illustrate how a user could run the code with different objectives. Example 1 is based on the assumption that condenser temperature (Tcond), evaporation temperature (Tevap), degree of superheating (SH), degree of subcooling (SC), refrigerant (fluid), and compressor efficiency (n_{comp}) are known input values. From these 6 parameters, the state points in the vapor compression cycle is solved directly by the VCC-calculator. Example 2 is similar except that compressor efficiency is not assumed to be known but rather more realistically depends on the pressure ratio between the 2 state points partially defined by Tcond and Tevap. Example 3 is different from the first two examples in that it shows how the VCC-calculator can be used in a conditional while-loop that repeatedly gets called until the energy transfer from a source fluid, energy transfer across the heat pump, and energy transfer to a sink fluid all converge in the sense that the cycle yields a desired heat transfer $\dot{Q}_{sink}$ while satisfying all thermodynamic modelling equations within some set tolerance.

Programming of the VCC-calculator is based on a set of assumptions which are typically used in literature (for example, Ouadha et al., 2008, Camdali 2015, Madessa et al., 2017, Jensen et al., 2018, Wang et al., 2022) when modelling the thermodynamics of a refrigeration cycle, either in heating or cooling mode. These assumptions are:

- A steady-state, closed system with only energy (heat, work) transfer
- No heat losses between system and surroundings
- Isobaric heat exchange in the condenser and evaporator
- Perfect heat transfer between the refrigerant in the primary loop and the fluids in the secondary loops
- Isenthalpic process through the expansion (or throttle) valve
- An irreversible process through the compressor, quantified in terms of an isentropic efficiency value, n_s

The steady-state assumption implies that the mass flow rate of the refrigerant, $\dot{m}$, is constant at any point in the system. This is why the subscripts for this variables is dropped since it does not differ across the cycle's state points. No heat losses between the system and the surroundings imply that:

$$\dot{Q}_{cond} = \dot{W}_{comp} + \dot{Q}_{evap} \tag{1}$$

according to the first law of thermodynamics. The isobaric assumption means that heat transfer across the heat exchangers is given by:

$$\dot{Q}_{cond} = \dot{m}\Delta h_{cond} = \dot{m}(h_2 - h_3) \tag{2}$$

and

$$\dot{Q}_{evap} = \dot{m}\Delta h_{evap} = \dot{m}(h_1 - h_4) \tag{3}$$

Perfect heat transfer between the refrigerant and the secondary fluids (air, water, or brine) implies that $\dot{Q}_{evap} = \dot{Q}_{source}$ and $\dot{Q}_{cond} = \dot{Q}_{source}$, where the source and sink fluids release and absorb the heat according to:

$$\dot{Q}_{source} = \dot{m}_{source} c_p \Delta T \tag{4}$$

and

$$\dot{Q}_{sink} = \dot{m}_{sink} c_p \Delta T \tag{5}$$

Since the process through the expansion (or throttle) valve is assumed to be isenthalpic, the enthalpies at the state points before and after the expansion valve are equal, i.e.,

$$h_4 = h_3 \tag{1}$$

The degree of irreversibility of the process through the compressor is directly related to the compressor efficiency. According to the second-law of thermodynamics, an irreversible process is one in which the entropy difference between the initial and final state is greater and zero. That is, $\Delta s_{comp} > 0$. The specific enthalpy of the fluid that comes out from the compressor (labelled as state point 2) is thus calculated by:

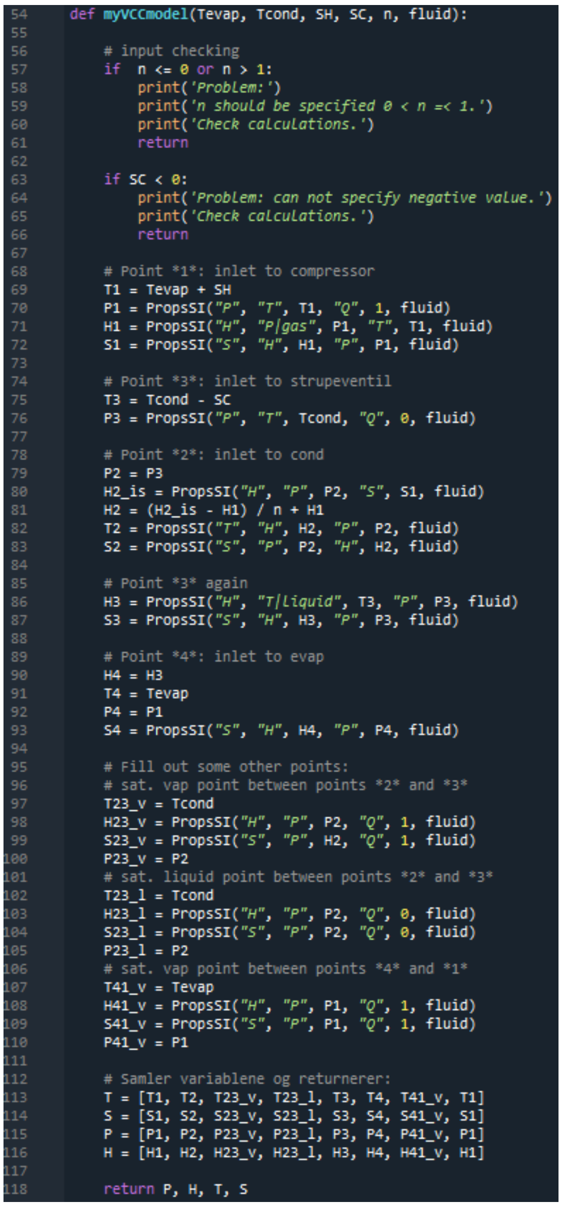
$$h_2 = h_1 - \frac{h_{2s} - h_1}{n_s} \tag{6}$$

which comes from the definition of isentropic efficiency, n_s, and where h_{2s} is the specific enthalpy at state point 2 as if the process was reversible (i.e., $s_1 = s_2$). In this work, the mechanical and electrical efficiencies of the compressor are both assumed to be 1, thus the overall work of the compressor is $\dot{W}_{comp} = \dot{m}\Delta h_{comp}$. However, if one wanted to include mechanical and electrical efficiencies, n_m and n_e, respectively, they can implement $\dot{W}_{comp} = \dot{m}\Delta h_{comp}/(n_m n_e)$ in the code.

Figure 1 shows the *myVCCmodel* definition. This is the central part of this work's VCC-repository. That is, the main state points of the VCC is obtained by this definition based on values for Tcond, Tevap, SH, SC, compressor efficiency, and fluid, which are passed in as known input parameters. This definition is based on thermodynamic principals, assumptions about the VCC already listed, and state point properties obtained using the CoolProp thermophysical property library (Bell et al., 2014). Since the CoolProp library is used in this work's code, users must first download and install CoolProp before running any example that uses the *myVCCmodel* definition. Download and installation instructions can be found online at coolprop.org.

The values returned by *myVCCmodel* are pressure, specific enthalpy, temperature, and specific entropy. These values can be used to visualize the thermodynamic cycle in a pressure versus specific enthalpy (hereafter called P-h) plot and a temperature versus specific entropy (hereafter called T-s) plot. The code developed in this work to create these plots make use of CoolProp's *Plots* module and its *PropertyPlot* definition in order to add the thermophysical properties of the refrigerant to the plot. The definitions *myPhPlot* and *myTsPlot* developed in this current work are not shown here for brevity, however they can be found in the online repository in the utilities folder.

It is interesting to visualize the cycle in the P-h and T-s diagrams, particularly in terms of the cycle's position (or proximity to the critical point at the top of the saturation envelope), the degree of temperature lift between the process lines through the evaporator and condenser, and the degree of entropy change in the process line through the compressor (i.e., the slope). Besides these notable cycle characteristics, it is important to quantify the performance of the cycle according to the COP (coefficient of performance). The COP is defined as the ratio between useful and spent energy, or $\dot{Q}_h/\dot{W}_{comp}$ in the case of heating mode, which is reduced to a formula based on only specific enthalpy differences:



```python
def myVCCmodel(Tevap, Tcond, SH, SC, n, fluid):

    # input checking
    if  n <= 0 or n > 1:
        print('Problem:')
        print('n should be specified 0 < n =< 1.')
        print('Check calculations.')
        return

    if SC < 0:
        print('Problem: can not specify negative value.')
        print('Check calculations.')
        return

    # Point *1*: inlet to compressor
    T1 = Tevap + SH
    P1 = PropsSI("P", "T", T1, "Q", 1, fluid)
    H1 = PropsSI("H", "P|gas", P1, "T", T1, fluid)
    S1 = PropsSI("S", "H", H1, "P", P1, fluid)

    # Point *3*: inlet to strupeventil
    T3 = Tcond - SC
    P3 = PropsSI("P", "T", Tcond, "Q", 0, fluid)

    # Point *2*: inlet to cond
    P2 = P3
    H2_is = PropsSI("H", "P", P2, "S", S1, fluid)
    H2 = (H2_is - H1) / n + H1
    T2 = PropsSI("T", "H", H2, "P", P2, fluid)
    S2 = PropsSI("S", "P", P2, "H", H2, fluid)

    # Point *3* again
    H3 = PropsSI("H", "T|liquid", T3, "P", P3, fluid)
    S3 = PropsSI("S", "H", H3, "P", P3, fluid)

    # Point *4*: inlet to evap
    H4 = H3
    T4 = Tevap
    P4 = P1
    S4 = PropsSI("S", "H", H4, "P", P4, fluid)

    # Fill out some other points:
    # sat. vap point between points *2* and *3*
    T23_v = Tcond
    H23_v = PropsSI("H", "P", P2, "Q", 1, fluid)
    S23_v = PropsSI("S", "P", H2, "Q", 1, fluid)
    P23_v = P2
    # sat. liquid point between points *2* and *3*
    T23_l = Tcond
    H23_l = PropsSI("H", "P", P2, "Q", 0, fluid)
    S23_l = PropsSI("S", "P", P2, "Q", 0, fluid)
    P23_l = P2
    # sat. vap point between points *4* and *1*
    T41_v = Tevap
    H41_v = PropsSI("H", "P", P1, "Q", 1, fluid)
    S41_v = PropsSI("S", "P", P1, "Q", 1, fluid)
    P41_v = P1

    # Samler variablene og returnerer:
    T = [T1, T2, T23_v, T23_l, T3, T4, T41_v, T1]
    S = [S1, S2, S23_v, S23_l, S3, S4, S41_v, S1]
    P = [P1, P2, P23_v, P23_l, P3, P4, P41_v, P1]
    H = [H1, H2, H23_v, H23_l, H3, H4, H41_v, H1]

    return P, H, T, S
```

Figure 1: Entire code used in the *myVCCmodel* definition.

$$COP = \frac{\Delta h_{cond}}{\Delta h_{comp}} = \frac{h_2 - h_3}{h_2 - h_1} \tag{7}$$

This is because $\dot{Q}_{cond} = \dot{m}\Delta h_{cond}$ and $\dot{W}_{comp} = \dot{m}\Delta h_{comp}$ as previously presented.

The amount of heat transfer from the hot-fluid stream (i.e., the fluid releasing heat energy) to the cold-fluid stream (i.e., the fluid receiving heat energy) can be expressed according to the log-mean temperature difference (LMTD) across the inlets and outlets of the heat exchanger:

$$\dot{Q}_{cond} = U_{cond}A_{cond}LMTD_{cond} \tag{8}$$

and

$$\dot{Q}_{evap} = U_{evap}A_{evap}LMTD_{evap} \quad (9)$$

where U and A are the overall convective heat transfer coefficient and the surface area between fluids, respectively. When calculating the LMTD, a double-pipe heat exchanger is assumed. As such, the equation implemented in the *getMyLMTD* definition is:

$$LMTD = \frac{\Delta T_1 - \Delta T_2}{ln\ (\Delta T_1/\Delta T_2)} \quad (10)$$

where ΔT_1 and ΔT_2 are the differences between the hot and cold fluid streams between the inlet and outlet of the heat exchanger, depending on whether the fluid streams are parallel or counter-flow. The results in Example 3 are based on a counter-flow configuration, but the code is set up to handle parallel-flow as well. While a double-pipe heat exchanger is assumed in this work, there is of course a possibility to extend the code to include other heat exchanger configurations. For example, a shell-and-tube or a plate heat exchanger can be implemented, as was done and presented in Svortevik (2023).

The *myVCCmodel* requires a value for compressor efficiency. In Example 1, the value for $\mathrm{n_{comp}}$ is assumed. However, in Example 2, the value for $\mathrm{n_{comp}}$ is calculated in the compressor model based on the pressure ratio between the condenser and evaporator. The compressor model used comes from Corberan et al., (2000) and was also used in Ouadha et al., (2008). It is based on empirical data specific to a compressor type, and takes the polynomial form of:

$$n_{comp}(PR) = A + B * PR + C * PR^2 \quad (11)$$

where the coefficients A, B, and C are 0.66768, 0.0025, and -0.00303, respectively. While this particular compressor model is used here, other compressor models could be easily implemented in this work's online repository as explained in Example 2 in the Results section.

Example 3 was inspired by Camdali et al., (2015) where the cycle is found iteratively such that it yields a predefined heating capacity, $\dot{Q}_{cond}$, while fitting into predefined operating conditions given by Tcond,in, Tcond,out, Tevap,in, and Tevap,out. Similar work was presented in Svortevik (2023), and results presented here in this work is based on the online version of the code. The iterative approach taken in Camdali et al., (2015) and implemented in here in this work could be viewed as a brute-force optimization approach, and that other optimization methods could be applied to solve for the cycle that meets the required heating capacity while minimizing discrepancies between the modelling equations.

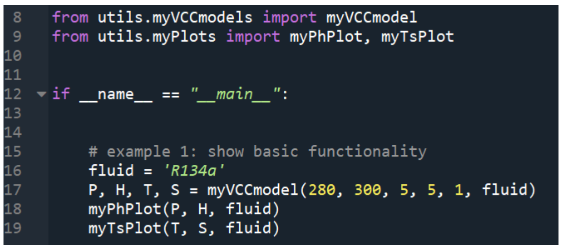
3. Results

3.1. Example 1

In the first example, values for Tcond, Tevap, and compressor efficiency are considered to be known input parameters, as well as refrigerant name, amount of super-heating, and amount of sub-cooling; see Table 1. The VCC-calculator (named *myVCCmodel* in Figure 2, code line 17) uses these input values to get P, h, T, and s at the 7 main state points. The cycle is then drawn in a P-h and T-s diagram (see code lines 18 and 19 which call *myPhPlot* and *myTsPlot* respectively). Figure 4 shows the P-h and T-s diagrams for the cases considered in this example. Cycle performance *can* be quantified by the COP, calculated via Eqn. 7 which is based solely on the specific enthalpies across the condenser and compressor, however is not presented here since this example is meant to be illustratively rather than for design or operation purposes. Mass flow rate *cannot* be calculated until either $\dot{Q}_{cond}$, $\dot{Q}_{evap}$, or $\dot{W}_{comp}$ are specified, and is thus not presented here.

Table 1: Case numbers and their input values considered in Example 1.

Case	**Tevap (K)**	**Tcond (K)**	n_{comp}	**SH (K)**	**SC (K)**	**fluid**
1-1	270	300	1	0	0	R134a
1-2	270	300	0.6	5	5	R134a
1-3	265	325	0.6	5	5	R134a



```
8  from utils.myVCCmodels import myVCCmodel
9  from utils.myPlots import myPhPlot, myTsPlot
10
11
12 if __name__ == "__main__":
13
14
15     # example 1: show basic functionality
16     fluid = 'R134a'
17     P, H, T, S = myVCCmodel(280, 300, 5, 5, 1, fluid)
18     myPhPlot(P, H, fluid)
19     myTsPlot(T, S, fluid)
```

Figure 2: Snippet of Python code used to run Example 1.

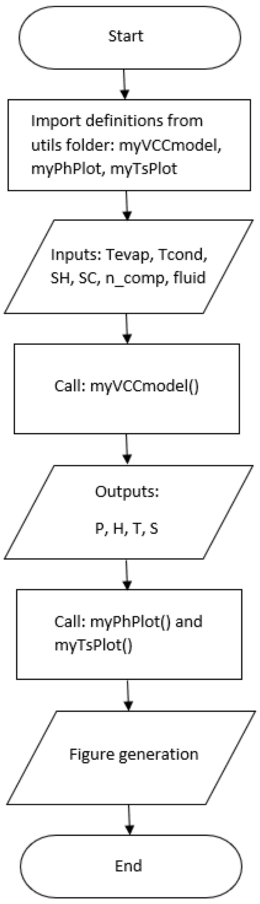

Start
Import definitions from utils folder: myVCCmodel, myPhPlot, myTsPlot
Inputs: Tevap, Tcond, SH, SC, n_comp, fluid
Call: myVCCmodel()
Outputs:
P, H, T, S
Call: myPhPlot() and myTsPlot()
Figure generation
End

Figure 3: Flowchart of algorithm used in Example 1.

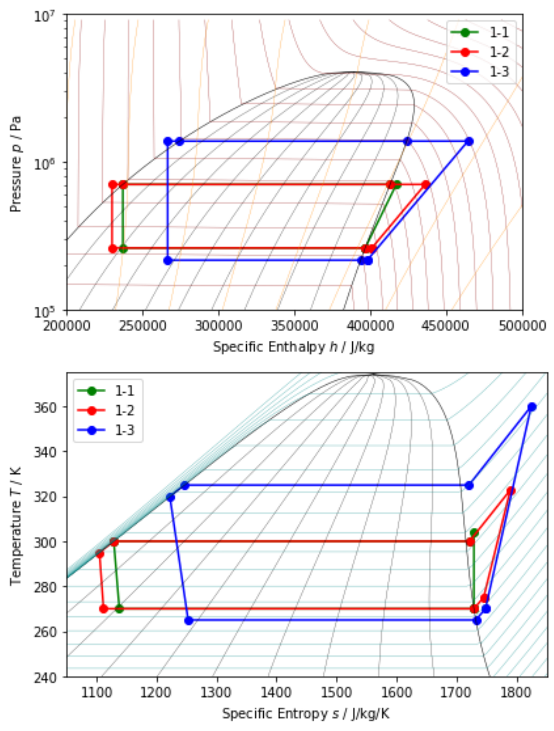
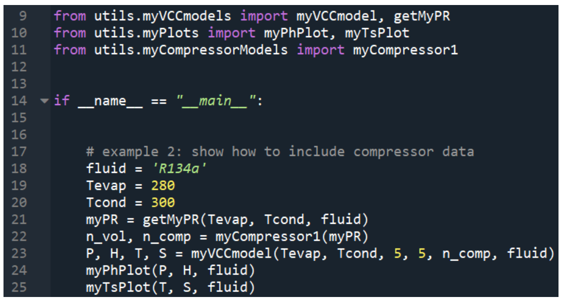


Figure 4: Cycle results for Example 1 obtained using this work's VCC-calculator, illustrated in P-h (top) and T-s (bottom) diagrams.

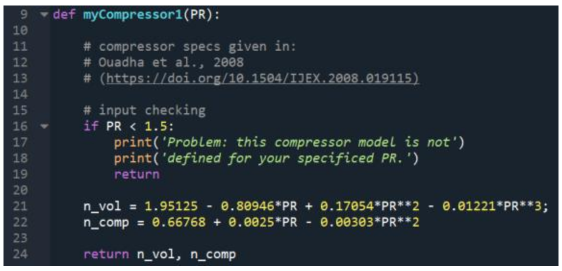
3.2. Example 2

The second example builds off the first in that Tcond and Tevap are considered to be known (as well as SH, SC, and fluid), however compressor efficiency is given as a function of the pressure ratio. This is more realistic than treating compressor efficiency as a fixed value. Figure 5 shows the code for this example, where code line 22 calls *myCompressor1* which is a definition contained in the *myCompressorModels* module. The compressor efficiency model used here comes from Ouadha et al., (2008), however other compressor models could be easily implemented in the online repository by first creating a copy of *myCompressor1* definition, renaming it *myCompressor2* or something else suitable, and updating the empirically-based expressions for n_vol and n_comp. Table 2 summarizes the cases studied for Example 2. Table 3 presents the pressure ratios that correspond to the Tcond and Tevap input values, and then the compressor efficiency values. Once the compressor efficiency is determined, all of the 7 state points are determined by the VCC-calculator (code line 23 in Figure 5), and P-h and T-s diagrams of the cycle can be made (code lines 24 and 25, respectively). Again, mass flow rate is undefined until $\dot{Q}_{cond}$, $\dot{Q}_{evap}$, or $\dot{W}_{comp}$ are specified. Cycle results are shown in Figure 8, and once again the focus is illustratively rather than conclusive.



```
 9  from utils.myVCCmodels import myVCCmodel, getMyPR
10  from utils.myPlots import myPhPlot, myTsPlot
11  from utils.myCompressorModels import myCompressor1
12
13
14  if __name__ == "__main__":
15
16
17      # example 2: show how to include compressor data
18      fluid = 'R134a'
19      Tevap = 280
20      Tcond = 300
21      myPR = getMyPR(Tevap, Tcond, fluid)
22      n_vol, n_comp = myCompressor1(myPR)
23      P, H, T, S = myVCCmodel(Tevap, Tcond, 5, 5, n_comp, fluid)
24      myPhPlot(P, H, fluid)
25      myTsPlot(T, S, fluid)
```

Figure 5: Snippet of Python code used to run Example 2.



```
def myCompressor1(PR):

    # compressor specs given in:
    # Ouadha et al., 2008
    # (https://doi.org/10.1504/IJEX.2008.019115)

    # input checking
    if PR < 1.5:
        print('Problem: this compressor model is not')
        print('defined for your specificed PR.')
        return

    n_vol = 1.95125 - 0.80946*PR + 0.17054*PR**2 - 0.01221*PR**3;
    n_comp = 0.66768 + 0.0025*PR - 0.00303*PR**2

    return n_vol, n_comp
```

Figure 6: Python-definition of *myCompressor1* used to calculate compressor efficiency in Example 2.

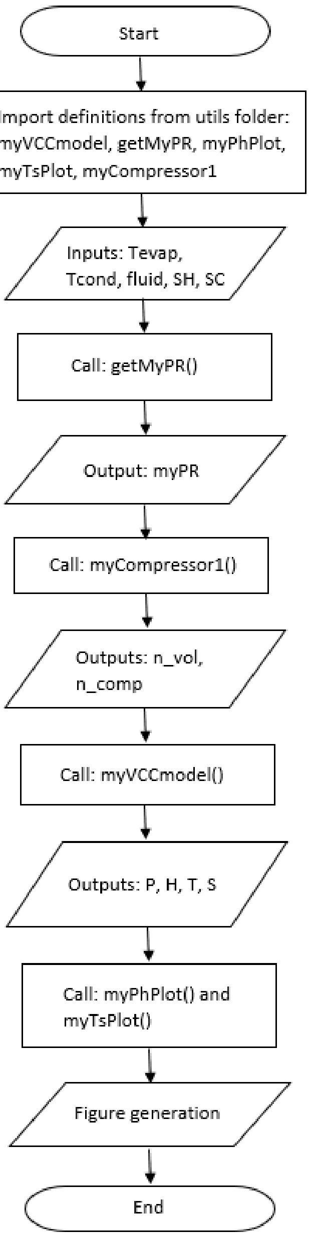

Start
Import definitions from utils folder: myVCCmodel, getMyPR, myPhPlot, myTsPlot, myCompressor1
Inputs: Tevap, Tcond, fluid, SH, SC
Call: getMyPR()
Output: myPR
Call: myCompressor1()
Outputs: n_vol, n_comp
Call: myVCCmodel()
Outputs: P, H, T, S
Call: myPhPlot() and myTsPlot()
Figure generation
End

Figure 7: Flowchart of algorithm used in Example 2.

Table 2: Case numbers and their input values considered in Example 2.

Case	Tevap (K)	Tcond (K)	SH (K)	SC (K)	fluid
2-1	270	300	0	0	R134a
2-2	270	300	5	5	R134a
2-3	265	325	5	5	R134a

Table 3: Corresponding pressure ratios and calculated compressor efficiencies.

Case	Pcond (kPa)	Pevap (kPa)	PR	n_{comp}
2-1	702.8	260.8	2.69	0.6524
2-2	702.8	260.8	2.69	0.6524
2-3	1380.3	215.7	6.40	0.5596

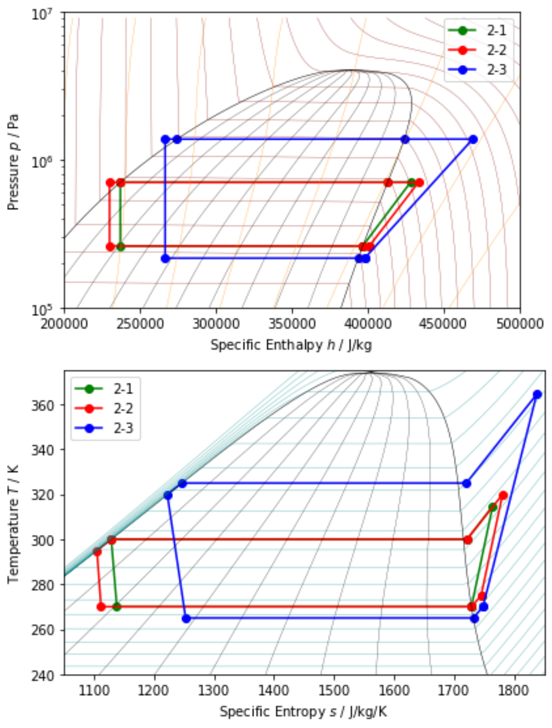


Figure 8: Cycle results for example 2 obtained using this work's VCC-calculator, illustrated in P-h (top) and T-s (bottom) diagrams.

3.3. Example 3

The last example aims to be a bit more practical than Examples 1 and 2. That is, it is unlikely that Tcond and Tevap are known, but rather, an amount of heat output from the heat pump cycle, $\dot{Q}_{cond}$, is more likely to be specified, along with values for Tcond,in, Tcond,out, Tevap,in, and Tevap,out. The required $\dot{Q}_{cond}$ can be used in combination with Tcond,in and Tcond,out values to calculate a Tcond value. But the rest of the cycle must be such that it agrees with this Tcond. In other words, the question is what values do Tcond and Tevap take on such that the cycle yields the required until $\dot{Q}_{cond}$? This question can be answered using an iterative approach that is similar to what was presented in Camdali et al., 2015.

Cases are presented in Table 4, where SH, SC, and fluid are 5, 5, and R134a, respectively, and compressor efficiency is given as a function of pressure ratio by Eqn. 11. Also, U and A values for both the condenser and evaporator are set to 1000 W/m^{2}K and 2 m^2, respectively. Results are shown in Table 5 and Figure 11. This time, mass flow rate of the refrigerant can be determined since $\dot{Q}_{cond}$ is specified. Figure 12 shows the convergence behavior of two main quantities or residuals, namely how close the cycle's Qcond value (black x's) is from the wanted heat capacity (red dashes), and how close Qevap1 and Qevap2 are from each other (blue dots) since their difference should be as close to zero as possible. The cycle was found after about 2300 iterations. Other methods could be used to obtain this cycle solution faster, for example, using an optimization method where the objective is to minimize the residuals.

Table 4: Case numbers and their input values considered in Example 3.

Case	**Capacity** $\dot{Q}_{cond}$ **(kW)**	**Tcond,in / Tcond,out (°C)**	**Tevap,in / Tevap,out (°C)**
3-1	30	30/40	8/0

Table 5: Results for cases considered in Example 3.

Case	**#its. needed**	**Tcond**	**Tevap**	n_{comp}
3-1	2349	323.55	255.50	0.4411

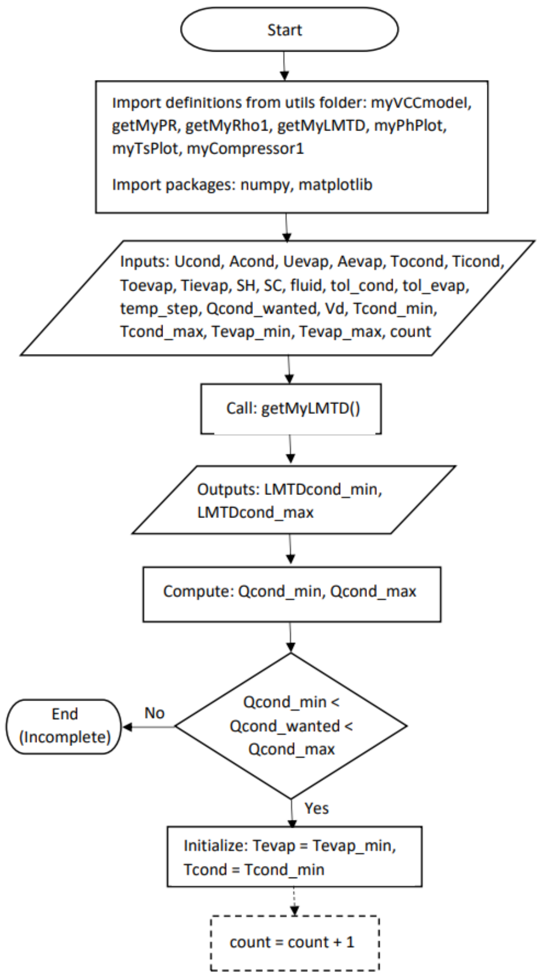

Start
Import definitions from utils folder: myVCCmodel, getMyPR, getMyRho1, getMyLMTD, myPhPlot, myTsPlot, myCompressor1
Import packages: numpy, matplotlib
Inputs: Ucond, Acond, Uevap, Aevap, Tocond, Ticond, Toevap, Tievap, SH, SC, fluid, tol_cond, tol_evap, temp_step, Qcond_wanted, Vd, Tcond_min, Tcond_max, Tevap_min, Tevap_max, count
Call: getMyLMTD()
Outputs: LMTDcond_min, LMTDcond_max
Compute: Qcond_min, Qcond_max
Qcond_min < Qcond_wanted < Qcond_max
No
End (Incomplete)
Yes
Initialize: Tevap = Tevap_min, Tcond = Tcond_min
count = count + 1

Figure 9: First half of flowchart for Example 3.

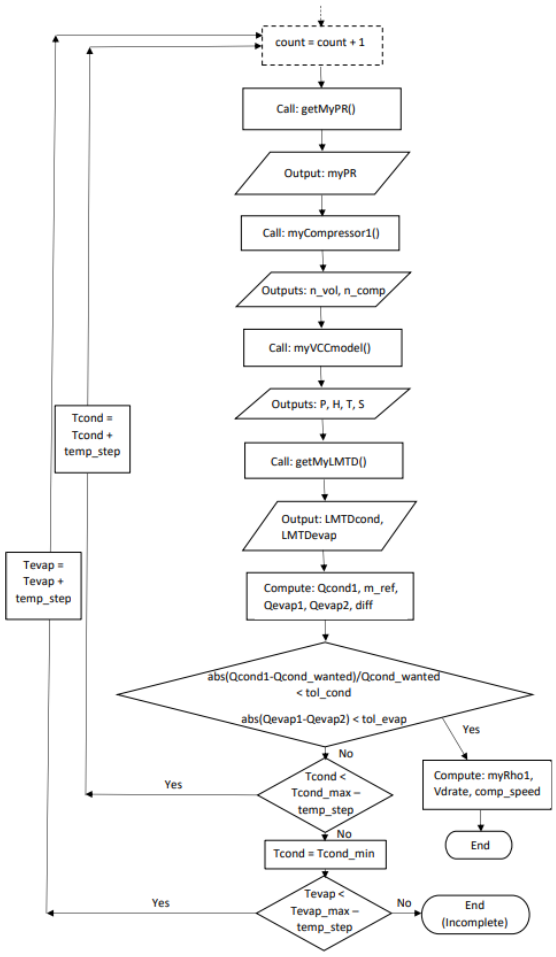

count = count + 1
Call: getMyPR()
Output: myPR
Call: myCompressor1()
Outputs: n_vol, n_comp
Call: myVCCmodel()
Outputs: P, H, T, S
Call: getMyLMTD()
Output: LMTDcond, LMTDevap
Compute: Qcond1, m_ref, Qevap1, Qevap2, diff
abs(Qcond1-Qcond_wanted)/Qcond_wanted < tol_cond
abs(Qevap1-Qevap2) < tol_evap
Yes
Compute: myRho1, Vdrate, comp_speed
End
No
Tcond < Tcond_max – temp_step
Yes
Tcond = Tcond + temp_step
No
Tcond = Tcond_min
Tevap < Tevap_max – temp_step
Yes
Tevap = Tevap + temp_step
No
End (Incomplete)

Figure 10: Second half of flowchart for Example 3.

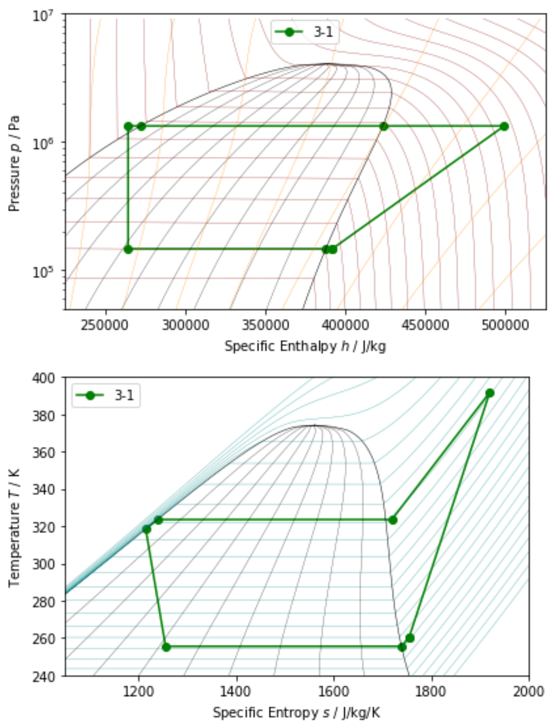


Figure 11: Cycle results for example 3 obtained using this work's VCC-calculator, illustrated in P-h (top) and T-s (bottom) diagrams.

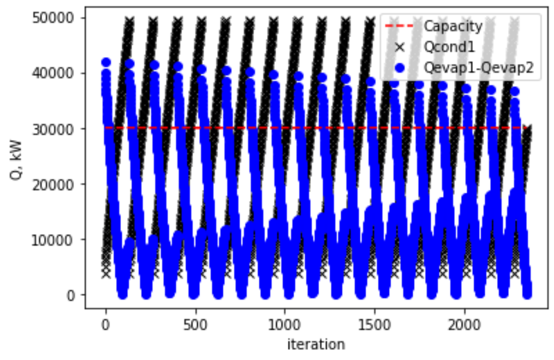


Figure 12: Convergence of solution towards a cycle that yields the wanted heat capacity with a minimal discrepancy between Qevap1 and Qevap2.

4. Summary

This work has presented the typically used modelling equations to simulate the thermodynamics of the vapor compression cycle. Three examples were presented to demonstrate the structure and capability of the code. A main contribution is the ability to implement various compressor models that can come from empirically derived expressions. Another contribution is the implementation of a similar iteration scheme presented by Camdali et al., 2015, however in Python versus MATLAB, which means the code can be assessed by more researchers, engineers, or others in the heat pump community. The method taken in Example 3 is similar to a brute-force optimization approach and other optimization methods could be implemented instead to reduce computational intensity.

The purpose of the three examples was illustrative in nature rather than to produce values that represent an engineering design or operation point of a heat pump. Indeed, this work marks the start of a developing online code that has the potential for modification and extension, and thus the focus here was to show basic structure of the VCC-calculator and how it can be used. As such, this paper did not focus too much on interpreting results, as they are intended to be illustrative only.

Source code used in this work can be found online: https://github.com/AllenGitCode/VCCmodelling.

References

Aulicino, C., and S. Bakrania, S., (2022) ‘A Python-based lab module to conduct thermodynamic cycle analysis,’ *IEEE Frontiers in Education Conference (FIE), Uppsala, Sweden, 2022*, pp. 1–6, doi: 10.1109/FIE56618.2022.9962388

Bell, I.H., Wronski, J., Quoilin, S., and Lemort, V. (2014) ‘Pure and Pseudo-pure Fluid Thermophysical Property Evaluation and the Open-Source Thermophysical Property Library CoolProp,’ *Industrial and Engineering Chemistry Research*, 53, pp. 2498–2508. doi: 10.1021/ie4033999

Camdali, U., Bulut, M., and Sozbir, N., (2015) ‘Numerical modeling of a ground source heat pump: The Bolu case,’ *Renewable Energy*, Elsevier, 83(C), pp. 352–361. doi: 10.1016/j.renene.2015.04.030

Corberan, J.M., Urchueguia, J., Gonzalves, J. and Calas, A. (2000) ‘Performance of a reciprocating hermetic refrigerant compressor using propane as working fluid’, *Proceeding of the 4th IIR-Gustav Lorentzen Conference on Natural Working Fluids at Purdue*, IIF-IIR Commission B1, B2, E1, E2, pp.225–232.

IEA HPT. ‘HTP – Heat Pumping Technologies’. https://heatpumpingtechnologies.org/ Accessed on Aug 17, 2023.

Jensen, J. K., Ommen, T., Reinholdt, L., Markussen, W. B., & Elmegaard, B. (2018) ‘Heat Pump COP, part 2: Generalized COP estimation of heat pump processes.’ In Proceedings of the 13th IIR-Gustav Larentzen Conference on Natural Refrigerants (Vol 2, pp. 1136-1145). International Institute of Refrigeration. https://doi.org/10.18462/iir.gl.2018.1386

Madessa, H. B., Torger, B., Bye, P. F., Erlend, A., (2017) ‘Parametric study of a vertically configured ground source heat pump system.’ *Energy Procedia*, 111, pp. 1040-1049. doi: 10.1016/j.egypro.2017.03.267

Ouadha, A., En nacer, M., and Imine, O., (2008) ‘Thermodynamic modelling of a water-to-water heat pump using propane as refrigerant,’ *International Journal of Exergy*, 5(4), pp. 451–469. doi: 10.1504/IJEX.2008.019115

Svortevik, E., (2023) ‘Python-based modeling of the vapor compression cycle focusing on heat exchangers and user-friendliness with web page,’ Master's thesis, Oslo Metropolitan University.

Vering, C., Engelpracht, M., Göbel, S., Hoseinpoori, S., Wüllhorst, F., Schwenzer, C., Rademacher, M., Hinrichs, S., Chandra, F., Mehrfeld, P., and Müller, D., (2022) ‘Open-Source vapor compression library (VCLib): Heat pump modeling for education and research,’ *Comput. Appl. Eng. Educ*. 30, pp. 1498–1509. doi: 10.1002/cae.22540

Wang, J., Qv, D., Ni, L., Fan, J., Kong, W., (2022) ‘Matching-design for inverster air-source heat pump system based on heating load characteristics of civil buildings,’ *Energy and Buildings*, 260(111952). doi: 10.1016/j.enbuild.2022.111952