An Automotive Application of the Organic Rankine Cycle for Power Generation Using Recovered Waste Heat

Glenn Michael Clapp

A Thesis Submitted to the Graduate Faculty of

GRAND VALLEY STATE UNIVERSITY

In

Partial Fulfillment of the Requirements

For the Degree of

Master of Science in Engineering

Padnos College of Engineering and Computing

April 2019

Contents

[1. Abstract 4](#_Toc10044650)

[2. Literature review 5](#_Toc10044651)

[2.1. Theory 5](#_Toc10044652)

[2.2. Cycle selection 10](#_Toc10044653)

[2.3. Working fluid selection 12](#_Toc10044654)

[2.4. Heat exchange 14](#_Toc10044655)

[2.5. Cycle evaluation 15](#_Toc10044656)

[3. Plan of study 17](#_Toc10044657)

[3.1. Objectives and outcomes 17](#_Toc10044658)

[3.1.1. Develop a mathematical model for iterative design of an organic Rankine cycle 17](#_Toc10044659)

[3.1.2. Design an organic Rankine cycle to recover heat from an automotive cooling system 18](#_Toc10044660)

[4. Method 19](#_Toc10044661)

[4.1. Models 19](#_Toc10044662)

[4.2. Test case 22](#_Toc10044663)

[5. Preliminary results 27](#_Toc10044664)

[5.1. Working fluid 27](#_Toc10044665)

[5.2. Working pressures and temperatures 27](#_Toc10044666)

[5.3. Increased energy efficiency 29](#_Toc10044667)

[6. Experimental design 30](#_Toc10044668)

[6.1. Boiler factors 30](#_Toc10044669)

[6.2. Turbine factors 31](#_Toc10044670)

[6.3. Condenser factors 31](#_Toc10044671)

[6.4. Pump factors 31](#_Toc10044672)

[7. References 33](#_Toc10044673)

[8. Appendix A – Acronyms 35](#_Toc10044674)

[9. Glossary 36](#_Toc10044675)

[10. Appendix C – Source Code 37](#_Toc10044676)

# Abstract

Keywords: Rankine cycle; Working pressure; Working temperature; Boiler; Condenser; Power-vapor cycle; Organic Rankine cycle; Low quality heat; Automotive; Waste heat recovery

The goal of this project is to develop a parametric model of an organic Rankine cycle for the purpose of generating electrical power using waste heat from the coolant system of an automobile. This application requires a small package size, and the utilization of low temperature, low quality waste heat.

Some of the barriers presented to the development of low-temperature waste heat recovery are [1]:

1. Long payback periods
2. Material constraints and costs
3. Economies of scale (WHR does not lend itself well to a general solution)
4. Operation and maintenance costs – Corrosion, scaling, and fouling of heat exchange materials lead to higher maintenance costs and lost productivity.

A successful solution to this problem will benefit from mass production and therefore mitigate the capital costs of development, utilize materials common to automotive parts in existing electric motors and pumps as well as refrigerants that are used in climate control applications. This project will develop industry knowledge in the area of waste heat recovery and increase the fuel efficiency of all vehicles in which it is implemented.

This project will have tertiary benefits as well, the model will develop methodology for harnessing low quality waste heat in other applications where development was not previously practical due to prohibitive upstart capital requirements and return on investment timing.

The parametric model developed for this project will be verified where applicable with physical prototypes and measurements. Previous work in mathematical modeling and literature on the subjects of waste heat recovery and vapor-power cycles will be reviewed to inform an efficient starting point for this project.

# Literature review

The following table contains the usage of the symbols throughout this document unless specified otherwise.

Table : List of symbols and constants

|  |  |  |
| --- | --- | --- |
| Symbol | Description | Value |
| σ | Stefan-Boltzmann constant | 5.67e-8 W/(m2K4) |
| m | Meters - SI unit of length |  |
| K | Kelvin - SI unit of absolute temperature |  |
| ε | Emissivity | 0 < ε < 1 |
| κ | Thermal conductivity (kJ/(kgK) |  |
| g | Gram - SI unit of mass |  |
| J | Joule - SI unit of energy or work |  |
| h | Heat transfer coefficient (W/(m2K) |  |
| W | Watt - SI unit of power |  |
| p | Pressure (Pa) |  |
| V | Volume (m3) |  |
| U | Internal energy (J) |  |
| H | Enthalpy | U + pV |

## Theory

**Conductive heat transfer**

Heat transfer by conduction is the process by which energy is transferred from energetic molecules to adjacent, less energetic molecules due to interactions between particles. In this form of heat transfer, there is no bulk transfer of material.

Equation : Conductive heat transfer - Fourier's law

Where is the rate of heat transfer across a plane normal to a direction x with area A. κ is a property of the material through which the energy is transferred referred to as the thermal conductivity of that material.

**Radiative heat transfer**

Radiative heat transfer is associated with the fourth power of the absolute temperature of a given surface, Tb. Note that this equation considers only a single surface.

Equation : Radiative heat transfer - Stefan-Botzmann law

Radiative heat transfer between a large surrounding surface of temperature Ts and a smaller surface at temperature Tb can be found by the following equation.

Where is the rate at which energy is emitted from a surface, ε is the property of a surface that indicates how effectively the surface radiates. It is a number between 0 and 1. σ is the Stefan-Boltzmann constant shown in Table 1.

**Convective heat transfer**

Heat transfer from a solid to an adjacent fluid (liquid or gas) is referred to as convection.

Where is the heat transfer rate from the solid to the fluid, Tb is the temperature of the surface, Tf is the temperature of the fluid, and h is the heat transfer coefficient.

**The first law of thermodynamics**

The first law of thermodynamics states that the energy in a closed system is conserved. Mathematically speaking this means that

Equation : The first law of thermodynamics

Where KE is the kinetic energy of the system, PE is the potential energy of the system and U is the internal energy of the system; Q is the heat transfer into the system and W is the work performed by the system.

In a power cycle such as the ones that will be analyzed by this study, the system returns to its initial state after every cycle. Because of this, the change in energy is considered to be 0.

**Power cycles**

Because the change in energy in a power cycle is considered zero from one iteration of the cycle to the next the following expression can be derived from Equation 1:

Equation : Power cycle

And the efficiency of the power cycle can be found by the ratio of the heat transfer into the system to the work produced by the system

Equation : Power cycle efficiency

In the development of a power cycle a tool called a p-v diagram is commonly utilized. This diagram is unique for any given working fluid and it shows the phase of the working fluid at a given pressure and specific volume. On the diagram, there is a zone in which the vapor and liquid phases simultaneously exist. On the low specific volume boundary of this zone is the saturated liquid line. Departing from this line away from the dome results in a working fluid that is in the liquid state only.

On the high specific volume boundary of this zone is the saturated vapor line. Departing from this line away from the dome results in a working fluid that is exclusively in the vapor state.

Traversing across the vapor dome along the specific volume axis represents a phase change from saturated liquid to saturated vapor or vice-versa at a constant temperature and constant pressure. In other words, as heat is added to a saturated liquid at a constant pressure, the liquid will transition into a vapor without any increase in temperature, but with a rapid increase of specific volume. Between the saturated liquid and saturated vapor states there exists a liquid-vapor mixture. The ration of vapor mass to the total mass of the mixture is referred to as the mixtures "quality" and is a quantity describing the progress of the phase transition.

Equation : Mixture quality

Where x is the mixture quality and m is the mass of the specified phase. x ranges between 0 and 1 where 0 is a saturated liquid and 1 is a saturated vapor.

The specific volume of a saturated liquid is often several orders of magnitude lower than that of a saturated vapor. This fact allows for two useful processes. The first is that the rapidly expanding mass of working fluid can be used to drive a turbine, performing useful work. The second is that the much denser liquid working fluid can be very efficiently pumped back to a boiler due to its much lower volume.

Take the following case for example:

There is a 12mm diameter cylinder filled with 3mm of saturated liquid water with a piston on top. The piston has a mass of 0.56kg and the air on the other side of the piston is standard air at sea level.



Figure : Piston with saturated liquid water

Heat is added to the water until all of the water has evaporated and what remains in the cylinder below the piston is a saturated vapor. The height of the piston can then be calculated by the following means.

First, a free body diagram of the piston yields the following equation which provides the initial pressure of the saturated liquid below the piston.

Equation : Newtons second law applied to the free body diagram of the piston

The pressure of the atmosphere at sea level is 101325Pa, therefore the pressure of the saturated liquid is 150000Pa. Table A-3 from *Fundamentals of Engineering Thermodynamics* [2] yields the specific volume of saturated liquid and of saturated vapor at that pressure.

Equation : Specific volumes of saturated water at liquid and vapor states

Because the mass of the water in the system remains constant, the following relationship holds:

Equation : Volume to specific volume ratio at saturated liquid and saturated vapor states

The volume of the saturated liquid can be found using the dimensions on Figure 1.

Equation : Vapor volume

And finally the height of the piston can be found by dividing that volume by the area of the piston yielding 2314mm.



Figure : Piston with saturated vapor water

This is a dramatic increase in volume; much more than what is achieved by adding heat to a fluid without its changing phase. This fact is what the Rankine cycle seeks to take advantage of for the generation of power by using this expansion to do useful work on a turbine.

**Ideal Rankine cycle**

The Rankine cycle is one of the practical ways that the preceding facts can be utilized to generate power; in the case of this project, for the generation of electric power.

An ideal Rankine cycle contains the following 4 processes:

1. Isentropic expansion of the working fluid through the turbine passing the working fluid to the condenser shown as DEF in the figure below. Note that because DE departs the saturated vapor line, E represents a superheated vapor rather than a saturated vapor.

2. Isobaric heat transfer from the working fluid to the surroundings from the condenser resulting in a saturated liquid represented below by the line FA.

3. Isentropic compression in the pump to a compressed liquid passed to the boiler. Represented by the line AB.

4. Isobaric heat transfer to the working fluid from the heat source resulting in a saturated vapor ready to begin the cycle again. This is represented by the line BCD



Figure : T-s diagram for an ideal Rankine cycle

**Boiler**

**Turbine:**

The assumption of an isentropic expansion means that the entropy prior to and after expansion through the turbine is equal. This assumption is mitigated by an efficiency that is input from the user which is then used to pad the power output and thermal efficiency numbers in such a way as to more closely model reality.

**Condenser**

**Pump**

## Cycle selection

Because of the low temperature and quality of waste heat in automotive applications, the literature on the subject indicates that the optimal cycle for recovering that energy is the Rankine cycle. Specifically, two variations of the Rankine cycle: the Organic Rankine cycle and the Kalina cycle.

There are other options for low quality waste heat recovery such as thermoelectric and piezoelectric generation which both show promise because of their low technical complexity and long service life due to the lack of moving components. Unfortunately, both of those strategies promise very low efficiency and power yield for a small automotive application and require expensive materials for fabrication.

The organic Rankine cycle (ORC), is so named for the hydrocarbons and refrigerants that are typically used in those cycles. ORCs have the same configuration as traditional steam Rankine cycles but make use of hydrochlorofluorocarbons (HCFSs), fossil fuels such as propane and cyclopentane, benzene, cyclohexane and isobutene, and refrigerants such as R134a, R22, R245fa, R236ea, R152a, R123, R600a, R600, R290 and R227ea as working fluids. [3] [4] [5] [4] These working fluids make the ORC particularly well suited to low heat quality applications due to their high molecular weight and low phase transition temperatures.

A single stage turbine is typically used for an ORC [3] which greatly reduces the technical complexity and size of the turbine, both of which are positive features for an automotive application where physical space is crucial.

The Kalina cycle (KC) is a modified form of the Rankine cycle and has a better operating efficiency in certain applications. The use of non-azeotropic[[1]](#footnote-1) working fluid mixtures in an ORC has often been proposed with the aim of reducing thermal irreversibilities; particularly those occurring between the heat source and the evaporating working fluid. The KC is typically implemented with a water/ammonia mixture for a working fluid. The ratio of that mixture is dynamically varied depending on the temperature of the heat source. [6]

Within the temperature range 200oC - 400oC, the Kalina cycle is 20% - 40% more efficient than a standard Rankine cycle. Some studies indicated that the Kalina cycle had better thermodynamic performance than the organic Rankine cycle as well. Because of the ammonia-water mixture used for a working fluid and the industry's long experience with both substances the Kalina cycle is also considered safe and environmentally friendly. This is not the case with some of the fossil fuels and refrigerants used in some organic Rankine cycles. [3] “For low temperature sources, the Kalina is often mentioned as an alternative for the ORC. Although the Kalina cycle is often called to be [sic] superior to the ORC, Dippo (See source in reference material) has shown that an existing Kalina cycle has about the same performance as existing ORCs.” [5] “Although the obtained useful powers are actually equal in value, the Kalina cycle requires a very high maximum pressure in order to obtain high thermodynamic performances. So, the adoption of Kalina cycle, at least for low power level and medium-high temperature thermal sources, seems not to be justified because the gain in performance with respect to a properly optimized ORC is very small and must be obtained with a complicated plant scheme, large surface heat exchangers and particular high pressure resistant and no-corrosion materials.” [6] In the quoted study, the recovered power calculated from two identical diesel engines was 1615 kW and 1603 kW for the Kalina and ORC cycles respectively.

The efficiency of a KC can be increased due to a close temperature match with heat transfer fluids in the evaporator and condenser. “For instance, a KC system using an ammonia-water mixture as the working fluid to generate power from waste heat of a gas turbine achieved a thermal efficiency of 32.8%.” [7] “Some studies showed that a KC can achieve a better thermal efficiency than ORC systems.” [7] “In practice, the expansion ratio of the turbine for KCS-34 is relatively high and a multi-stage turbine is required.” [7] This may be the drawback of the KC that makes ORC somewhat more attractive for an automotive application.

The KC also requires a substantial amount of control in order to adjust the mass-fraction of ammonia including density sensors and real-time monitoring of working fluid composition in some implementations making this cycle more complicated to implement. KC is generally used as a method of improving a conventional (steam) RC. In some of these cases, the KC is the only real choice for waste heat recovery. It also appears that the working temperatures are slightly higher than some refrigerant based ORCs. This makes the KC less appropriate for an automotive application where the heat source is less than 100oC.

For the reasons outlined above, the ORC and the KC are the two leading competitors, and from those two, the ORC was selected for further study due to its comparable power output and efficiency that come without the draw backs of system complexity and corrosion that are common in implementations of the Kalina cycle.

The goal of this project is to design a parameterized mathematical model of an ORC such that an automotive scale generator can be designed and built. This model will use one of several hydrocarbons or refrigerants as a working fluid because the low quality of waste heat in the cooling system requires a working fluid with a low phase transition temperature. This model must also take as arguments the space available for the simulated system, though the project will not strictly limit the size of the application.

## Working fluid selection

There are several viable working fluids for the organic Rankine cycle. The usage of each depends on a number of factors. Foremost among those factors are the working temperatures and pressures of the desired system.

In a traditional Rankine cycle, the working fluid used is water. That is not an appropriate selection for an automotive application due to the comparatively low temperature of the waste heat utilized. The Organic Rankine cycle uses organic compounds as a working fluid. These include hydrocarbons and refrigerants.

R134a appeared to be the most suitable for small scale solar applications, though R152a, R600a, R600 and R290 were promising though they required handling precautions due to their flammability. Isobutene also showed improved system performance when compared to R123 and R245fa. [8] Reading through the literature review in M.A. Khatita et al. did not show consensus on the best working fluid for ORC, however. With most fluids the use of a regenerative ORC instead of the basic cycle reduced the irreversibility of a solar ORC. Additionally, at the two temperature ranges studied fluids with higher molecular complexity resulted in more effective regenerative cycles with the exception of cyclo-hydrocarbons. [8] This was primarily due to higher turbine efficiency and increased mass flow rates.

“Unlike water, most organic fluids suffer chemical decomposition and deterioration at high temperatures and pressures.” [9] These working fluids eventually foul the heat exchanger passages through which they pass, inhibiting heat transfer from the that source to the working fluid and from the working fluid to the heat sink. This is often referred to as a fouling factor and is an inefficiency that will be considered in the proposed model. ORC systems showed efficiency gains with higher turbine inlet pressures, and efficiency losses for higher condenser outlet temperatures. This suggests that operating conditions, primarily temperature, could have a significant effect on the subsequent fouling of the heat exchangers and the efficiency of a given system. [9] [1]

In addition to the fouling on the working fluid side of the heat exchangers, the heat source side of the heat exchangers can also experience detrimental fouling if the exhaust stream from the vehicle is used as a heat source. This is the reason why, when considering which heat source to model, the cooling loop that exists in the engine compartment was chosen. This is another area in which the proposed project can offer an improvement on current technology. Since the waste heat is harvested from a closed cooling loop, the only fouling potential that exists is from the decomposition of the coolant itself. This is much less than what might be present in an open system where compounds could precipitate from the waste heat stream as it is cooled. The system architecture being considered for this project assumes that there will be a heat exchanger through which the heat source (engine coolant) and the working fluid of the simulated cycle will pass without directly interacting with one another. There is another benefit to this type of heat exchange. Because of the flammable nature of many organic compounds, separating the working fluid from the heat source also reduces fire hazards that might otherwise arise in a system of this kind. Because of the lack of oxygen in a closed loop of working fluid there is no possibility of fire without some sort of leak. There is further advantage to using the cooling loop of the vehicle as a heat source with respect to this design consideration in that there is no oxygen present in either system, and therefore reduced risk even in the case of a leak between the coolant system and the waste heat recovery system.

An analysis of a much more exhaustive list of working fluids used in an ORC is contained in the paper Exergetic and economic comparison of ORC and KC for low temperature enhanced geothermal system in Brasil. [10] This analysis includes several fluids that are not commonly considered in the literature and may be worth consideration.

## Heat exchange

According to Singh and Pedersen’s work, a heat balance for a typical maritime application might look like the following diagram.



Figure : Heat balance diagram for MAN 12K98ME/MC marine diesel engine operating at 100 SMCR under ISO conditions [3]

While this application is not the subject of the proposed study, the breakdown of energy availability is likely to be very similar, though the quantities, temperature, and quality of those heat sources diverge from the diagram above.

“Engine cooling water temperatures of 80-90 oC are fairly standard for most engines.” [3] While exhaust gas appears to be the most lucrative source of waste heat energy, there are several complications which, though not as important for the Pederson and Singh study, would prevent it’s being as lucrative in an automotive application. Some of these factors are the increase in exhaust back pressure on the engine, the cooling of exhaust gases below the dew point of steam which could result in liquid water in the exhaust system causing corrosion, and reduced efficacy of reactions in the catalytic converter due to sub-optimal temperatures and high pressures caused by the heat harvesting system.

“Many gaseous waste heat streams are discharged at near-atmospheric pressure (limiting the ability to transport them to and through equipment without additional energy input).” [1] Here, an automotive application really shines. The coolant from which the waste heat will be recovered is already being circulated through the engine and through the radiator for the purpose of cooling the engine. It is plausible that WHR system need not impose an additional requirement for energy to circulate this heat source. Because of the increased density, heat exchange occurs more efficiently if the two working fluids passing through a heat exchanger are each in a liquid state. Therefore, it should be a design consideration that the heat echanger pipes are always submerged in the liquid working fluid.

There is further, related upside to using organic compounds as a working fluid. One of the features of organic molecules is their relatively high molecular mass. In comparison with water vapor, the fluids used in ORCs have a higher molecular mass, enabling more compact designs, a higher mass flow rate, and higher turbine efficiencies. However, since the cycle functions at lower working temperatures, the overall efficiency is only around 10%-20%, depending on the precise temperature of the condenser and of the evaporator.

## Cycle evaluation

There are many ways to evaluate the efficacy of an Organic Rankine cycle and across the literature many of these are employed. It is also not uncommon for the authors of a given paper to develop their own fitness criteria and use that to evaluate competing system designs. One common way in which waste heat recovery systems are compared is by comparing efficiency. Maximum efficiency is given by the following equation.

Equation 11: Maximum work potential

Where η is the efficiency of the given power cycle, Tc is the temperature of the cold reservoir and T­H is the temperature of the hot reservoir. Other methods of comparison include parameters such as system cost, research and development time and cost, system complexity and maintenance, and other recurring factors that may offset, in some cases entirely, the benefit of adding a waste heat recovery system to the process under consideration. This study will define an evaluation algorithm. The purpose of this is to iteratively evaluate parameters which are continuous over a user-defined range for the purpose of optimization.



# Plan of study

## Objectives and outcomes

There are professional, academic, and personal objectives for this course of study. They are summarized in this section and expounded upon in the following sections. The industry objectives for this study are to design a tool with which to develop design specifications for an automotive waste heat recovery system that can be utilized on a light truck or similar platform. The tool will allow for the manipulation of various parameters of a Rankine cycle and estimate the power output of a system conforming to those parameters. The parameters chosen for manipulation are justified in the following sections and supported by the literature review. The parameters which will be studied for this application are shown in the table below.

Table 2: Study parameters

|  |  |
| --- | --- |
| **Parameter** | **Cases** |
| Working fluid | R245fa |
| Boiler working pressure | 0.5 MPa - 3.5 MPa |
| Condenser working pressure | 0.0 MPa - 0.5 MPa |
| Pump mass flow rate | 350 L/min – 570 L/min |

The academic objectives for this course of study are to expand my knowledge of vapor power cycles to the end of developing a useful automotive application. Further to complete my degree at GVSU in order to open up avenues for further study on this and related subjects. It is also an objective of this work to contribute to the body of knowledge in this field for the benefit of the engineering profession at large and for the environmental and commercial benefits of improving energy utilization.

The personal objectives for this project include satisfying a personal interest I have in the field of alternative energy, and novel technical projects.

The ultimate research objective of this study is to produce a model that can be manipulated to evaluate arbitrary attributes of an Organic Rankine cycle. That model will be used in future work to develop prototypes and eventually commercially viable products for my employer.

### Develop a mathematical model for iterative design of an organic Rankine cycle

A model will be developed using Python to simulate a Rankine cycle with the parameters specified by the user. This model will be used to iteratively evaluate cycles with parameters over a specified range. The squares in the figure below represent the various subcomponents of that mathematical model as well as the key components of this project.



Figure : Proposed study components diagram

### Design an organic Rankine cycle to recover heat from an automotive cooling system

Using the model described under the previous heading, ideal parameters will be developed for a specific automotive application. The purpose of the model is to develop design specifications for an actual application and eventually as part of a future project to build a working prototype.

# Method

## Models

An ideal Rankine cycle can be modeled by the four principal components of the system. These models are all derived from the mass and energy rate balances for control volumes around the given component.

Figure : Mass and energy rate balance for a control volume

For all models, the potential energy of the working fluid is ignored as the model is not informed of the physical orientation of the components of a candidate system with respect to each other or with respect to gravity.



Figure : Rankine cycle with no superheat

The model for the Turbine is:

Equation : Turbine model

Where m is the mass flow rate of the working fluid, Wt is the work produced by the turbine, and h is the enthalpy at the respective states of the working fluid as denoted in Figure 1. Note that some work needs to be done to find the enthalpy at state 2 as that point in the cycle is neither a saturated vapor nor a saturated liquid and will therefore not be found in a lookup table. The enthalpy at that state is found by the following:

Equation : Enthalpy of a vapor-liquid mixture

Where hf is the enthalpy of a saturated liquid, hfg is the evaporation enthalpy and x2 is the quality of the vapor-liquid mixture at state 2 which is given by the following:

Equation : Vapor-liquid mixture quality

Where s2 is the entropy of state 2, sg is the entropy of a saturated vapor, and sf is the entropy of a saturated liquid.[[2]](#footnote-2)

The model for the condenser is:

Equation : Condenser model

Where Q is the heat exchange rate from the condenser to the surroundings.

Equation : Pump model

Where ν is the specific volume at state 3 and p is the pressure at the state indicated.

Equation : Boiler model

There are several assumptions that are made for an engineering model of an ideal Rankine cycle.

1. Each component of the cycle is analyzed as a control volume at steady state.
2. All processes of the working fluid are internally reversible.
3. The turbine and the pump operate adiabatically.
4. Kinetic and potential energy effects are negligible
5. Saturated vapor enters the turbine. Condensate exits the condenser as a saturated liquid.

A non-adiabatic turbine, that is one with irreversibilities, can be modelled most simply by making the following edit to the enthalpy calculation in Equation 13:

Equation : Enthalpy of an adiabatic turbine at state 2

Where h2 has simply been replaced with h2s indicating that it is an ideal state, and finding the actual enthalpy h2 by the following:

Equation : State 2 enthalpy adjusted for turbine irreversibilities

Where, ηturbine is the experimentally determined, or estimated efficiency of the turbine (a number between 0 and 1).

A non-adiabatic pump, that is one with irreversibilities, can be modelled by making the following modification to the pump model shown in Equation 16.

Equation : Pump model with irreversibilities

Where ηpump is the experimentally determined, or estimated efficiency of the pump (a number between 0 and 1).

In order to evaluate a cycle where the working pressures of the boiler and condenser are known, and the efficiencies of the turbine and of the pump are known, the enthalpy and entropy at states 1 and 3 can be interpolated from a table for the given working fluid as can the working temperatures at both states 1 and 3. These temperatures will be required for the design of the heat exchangers.

Next, the enthalpy in the adiabatic case, and subsequently the non-adiabatic case can be determined by looking up the saturated liquid and vapor enthalpy and entropy at state 2, determining the quality of the mixture using Equation 14, then using Equation 18 and Equation 19 to calculate the enthalpy at state 2.

The enthalpy at state 4 can be determined using equation 16 with the modification from equation 20 as shown below:

Now that the enthalpy at all four states is known, the net work per unit mass flow and thermal efficiency of the cycle can be determined as follows:

Equation : Rankine cycle net power production

Equation : Rankine cycle thermal efficiency

Note that these formulas are valid both for the ideal case, and the case with irreversibilities because the irreversibilities were considered when calculating the enthalpy at each of the states. The same is true for the overall thermal efficiency.

Python code was developed for the purpose of performing these calculations so that a range of working pressures for the boiler and condenser could be iterated over and optimized numerically. The following section will validate that code using an Organic Rankine cycle that was evaluated experimentally by L. Li, Y.T. Ge, & S.A. Tassou. [11]

The specified working pressures of the boiler and condenser will also fix the temperatures at those states which can be used in conjunction with the mass flow rate and specific heat capacity of the working fluid to determine the necessary area of the heat exchangers for that particular case. Because the heat is being exchanged between two fluid flows, a cross-flow or counter-flow heat exchanger model can be used to make these estimates. The following procedure is found in *Heat Transfer* [12].

Equation : Heat exchanger heat transfer equation

Where q = heat transfer, U = Overall heat-transfer coefficient, A = surface area for heat transfer consistent with the definition of U, ∆Tm = suitable mean temperature difference across the heat exchanger.

The mean temperature difference in the case of a heat exchanger of this type is found with the following equation known as the log-mean temperature difference.

Equation : Log mean temperature difference

Where Th1 is the temperature of the hot fluid at its inlet and Th2 is the temperature of the hot fluid at its outlet, and Tc1 is the temperature of the cool fluid at its inlet and Tc2 is the temperature of the cool fluid at its outlet.

The heat transfer to or from a working fluid through a heat exchanger can be determined with the following:

Equation : Heat transfer in a heat exchanger

In order to determine the heat exchange area, the overall heat-transfer coefficient must be determined, table 10-1 from *Heat Transfer* gives an approximate value for the overall heat transfer for various situations including a water-to-oil exchanger which approximates a coolant to R245fa situation like the one in this paper to be 110-350 W/m2 oC. In order to obtain a conservative estimate for heat exchanger size, the lowest number in this range is used to obtain a maximum heat exchanger size; 110 W/m2 oC. [12]

## Test case

In an experimental study performed by Seok Hun Kang, the following cycle parameters were used to construct an organic Rankine cycle:

Table : Thermodynamic properties of the working fluid at the cycle design points [13]



Where the state labels are derived from the following diagram:



Figure : T-s diagram of the experimental cycle [13]

In the actual study, Kang observed different operating pressures however, and recorded them in the table below:

Table : Observed operating conditions of an ORC [13]



Using the boiler pressure 8.65 bar (0.865 MPa) and the condenser pressure 2.54 bar (0.254 MPa) From the 83oC evaporator input temperature test case – which is closest to the likely heat source temperature for the automotive applications we desire to study – and a turbine efficiency of 0.787 which was published by Kang as the maximum achieved turbine efficiency; the following cycle parameters are found using the model developed in this study assuming a pump efficiency of 0.9:

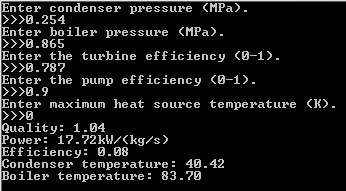


Figure : Model verification output

Using an assumed efficiency of 90% for both the pump and the turbine, the power output of the cycle was predicted to be 17.72kW/(kg/s) and the thermal efficiency of the cycle was predicted to be 0.08.

The following figures show the published results obtained by Kang:

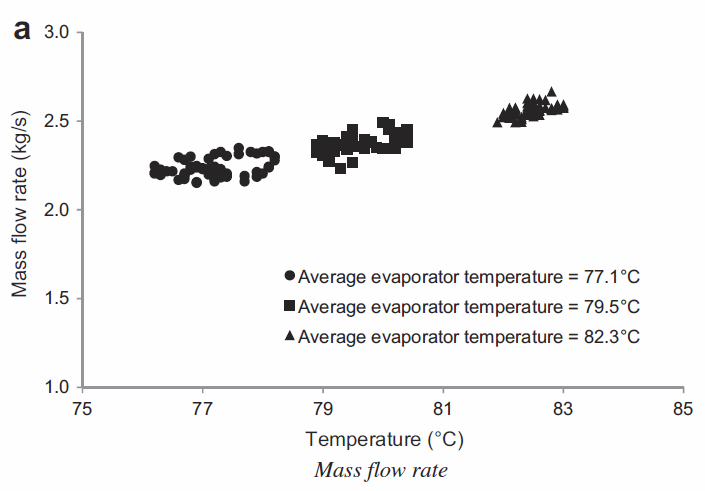


Figure : Mass flow rate observations for each evaporator input temperature test case [13]

Using this figure, the mass flow rate for the test case we chose to evaluate would be approximately 2.5 kg/s Which means the power output predicted would be 44.3kW.



Figure : Electric power output for the experimental ORC [13]

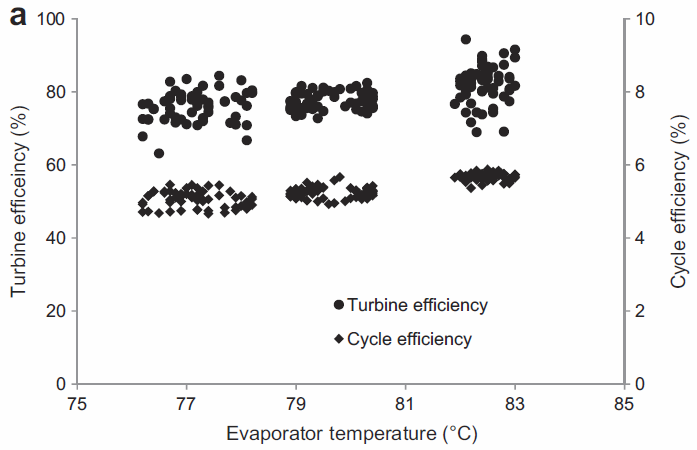


Figure : Turbine and cycle efficiencies [13]

So the model developed in this study slightly over-estimated the power output of the ORC and correctly predicted the cycle efficiency within the margin of the published study. This could be a result of a bad assumption of pump efficiency or efficiency losses elsewhere in the system that were not modeled such as fouling in the heat exchangers which is common in systems that use refrigerants as a working fluid. There are also inefficiencies associated with the way the working fluid condenses in the condenser which where the liquid can insulate the gas from further heat transfer.

# Results

## Working fluid

The most common working fluids used in ORC are R134a, R245fa, R22, isobutene, pentane, propane and PFCs. In the past, CFCs and HCFC were commonly used but are being phased out of current applications, and avoided for new applications due to environmental and safety concerns. Any leaks that may occur in the system additionally pose a fire safety hazard if the working fluid selected is flammable. If a hydro-carbon were selected, heat exchangers in which the working fluid does not come into direct contact with the heat source become necessary and the flash point of the working fluid must be considered as a boiler maximum temperature.

The working fluid used for the preliminary study was R245fa, a popular choice for similar applications with medium to low grade waste heat. The phase transition diagram is shown below. A table from the same source was used in the Python model used to produce the results shown in the following section.



Figure 13: R245fa Pressure/Enthalpy diagram [14]

## Boiler working pressure

First, the planned range for boiler working pressure was tested. Because the temperature of the heat source is limited, the working pressure will also be limited. The model that was developed was used to iteratively increase the working pressure of the boiler until the corresponding temperature was equal to the max source temperature. Those results follow.

Table 5: Max boiler working pressure and temperature

|  |  |
| --- | --- |
| Boiler working pressure (MPa) | Boiler working temperature (Celsius) |
| 1.26 | 100 |

This working pressure was then used as an upper bound in the numerical analysis which used the model to determine the max power and max efficiency of the cycle and report back the working pressures that achieved each. Those results are shown below.

Table : Working pressures in a cycle optimized for power and efficiency respectively

|  |  |
| --- | --- |
| Maximum power output (kW/(kg/s)) | 38.93 |
| Boiler pressure at max power (MPa) | 1.26 |
| Condenser pressure max power (MPa) | 0.1225 |
| Maximum efficiency | 0.137 |
| Boiler pressure at max efficiency (MPa) | 1.26 |
| Condenser pressure at max efficiency (MPa) | 0.1225 |

# Experimental design

Some of the parameters of the system cannot be independent for each of the subsystems. The parameter that is necessarily shared by all subsystems is the working fluid and the mass flow rate of the working fluid.

A preliminary investigation of working fluids was performed and the working fluid used to obtain the preliminary results was R245fa. However, this investigation was far from exhaustive and the literature review yielded several more working fluids of interest that this study will investigate. Some of the working fluids of particular interest are isopentane and R236ea which at least one study cited as the best working fluid for a heat source of 145oC. [5] Another source used R227ea as a working fluid which was observed to have an electric efficiency of 4.88% [4]

There are also efficiency improving features of a Rankine cycle which could be explored. Boiler temperatures super, trans, and sub critical are all discussed in the literature; recuperators, secondary turbine stages, turbine bleeds, heated feed-water and many others are discussed. Most of these will be beyond the scope of this study, but will likely be included among the recommendations for further study as the literature seems to indicate that marginal gains in efficiency and power output can be achieved by the integration of some or all of these features.

## Boiler factors

In addition to the working pressure and temperature of the boiler, mass flow rates of the heat source, the working fluid from the automobile’s cooling system, and of the vapor power system can be manipulated to increase the rate of heat transfer. The interaction surface area of the heat exchanger can also be manipulated. Some of these factors are much easier to manipulate than others. So, while the model will be developed such that any relevant parameter can be manipulated, this study will focus on the results of changing those that make the most practical and economic sense.

For the boiler, because this application requires a very small size with respect to traditional power generation layouts, the surface area of the heat exchanger is not a parameter that can be manipulated easily in practice. The temperature is also difficult to manipulate as the application is going to be integrated into an existing vehicle system which will have waste heat at a given temperature over which designers have very little control.

The factor that does make sense to manipulate is the operating pressure. And to that end, preliminary results have been included to demonstrate the design space that exists for this application.

## Turbine factors

The number of stages, and turbine blade size, shape, and angle can all be manipulated to affect the efficiency of the turbine. The type of cycle in this case, the organic Rankine cycle, also has a large effect on the efficiency of this component of the system.

The literature review indicates that the turbine design selected for an organic Rankine cycle is usually a single stage turbine. This greatly simplifies the design space for the turbine. Blade size, shape and angle can be optimized for a given application and therefore are not parameters that are transparent to the user of the mathematical model as these parameters will be determined by the selection of other parameters.

These reasons make the turbine design an area in which there is not a lot of interesting work to be done in this project. Because of this, it will not be a subject that receives much specific attention in this study.

## Condenser factors

In addition to the working pressure and temperature of the condenser, the mass flow rate of the heat sink can be manipulated; though the mass flow rate of the heat sink, in this case the ambient air, is harder to manipulate. The interaction surface area of the heat exchanger can also be manipulated.

The condenser also represents a potential design challenge in an automotive application because the environment in which the system will be placed, the engine compartment, can be expected to have a highly variable temperature. It will also not be practically feasible to manipulate that temperature in any meaningful way.

The condenser factors face many of the same limitations as those of the boiler. Primarily that space in this application is a premium and therefore will likely be fixed. The condenser temperature is fixed to the ambient temperature which, in addition to being largely out of the control of this device, is somewhat variable depending on other heat sources and sinks in the engine compartment as well as operating conditions of the vehicle and the weather in which the vehicle is operating.

As a result the parameter that is of interest to this study is primarily that of working pressure of the condenser and to that end the preliminary results contain the likely design space for this application.

## Pump factors

The pump working pressure can be manipulated, and it may be possible to drive the pump directly with mechanical energy from the turbine shaft rather than electrically which could represent some efficiency gains. This area is also of particular interest to GHSP as they design and manufacture pumps currently and can be considered experts in this area.

Mass flow rate of the working fluid and the pressure of the boiler, which is the output pressure of the pump, are parameters of interest to this study. The pressure is explored in the preliminary results, but the mass flow rate is represented in the results as a variable quantity and will be one of the primary foci of the proposed study.

# References

|  |  |
| --- | --- |
| [1] | BCS, Incorporated, "Waste Heat Recovery: Technology and Opportunities in U.S. Industry," *U.S. Department of Energy Industrial Technologies Program,* p. 112, 2008. |
| [2] | H. N. S. Michael J. Moran, Fundamentals of Engineering Thermodynamics, John Wiley & Sons, 2008. |
| [3] | E. P. Dig Vijay Singh, "A review of waste heat recovery technologies for maritime applications," *Energy Conversion and Management,* no. 111, pp. 315-328, 2016. |
| [4] | S. H. Kang, "Design and Preliminary Tests of ORC (Organic Rankine Cycle) with Two-Stage Radial Turbine," *Energy,* no. 96, pp. 142-154, 2016. |
| [5] | B. L. W. D. Daniel Walraven, "Comparison of Thermodynamic cycles for Power Production from Low-temperature Geothermal Heat Sources," *Energy Conversion and Management,* no. 66, p. 14, 2012. |
| [6] | C. M. I. C. P. Paola Bombarda, "Heat Recovery from Diesel Engines: A Thermodynamic Comparison Between Kalina and ORC cycles," *Applied Thermal Engineering,* no. 30, pp. 212-219, 2010. |
| [7] | Z. Y. Enhua Wang, "A numerical analysis of a composition-adjustable Kalina cycle power plant for power generation from low-temperature geothermal sources," *Applied Energy,* vol. 180, pp. 834-848, 2016. |
| [8] | T. S. A. F. H. A. I. M. I. Mohammed A. Khatita, "Power generation using waste heat recovery by organic Rankine cycle in oil and gas sector in Egypt: A case study," *Energy,* vol. 64, pp. 462-472, 2013. |
| [9] | T. C. Hung, T. Shai and S. K. Wang, "A Review of Organic Rankine Cycles (ORCs) for the Recovery of Low-Grade Waste Heat," *Energy,* vol. 22, no. 7, pp. 661-667, 1997. |
| [10] | J. C. E. P. O. J. V. E. E. S. L. V. M. C. D. M. d. S. F. R. L. D. V. G. Carlos Eymel Campos Rodriguez, "Exergetic and economic comparison of ORC and KC for low temperature enhanced geothermal system in Brazil," *Applied Thermal Engineering,* no. 52, pp. 109-119, 2013. |
| [11] | Y. G. S. T. L. Li, "Experimental study on a small-scale R245fa organic Rankine cycle system for low-grade thermal energy recovery," *Energy Procedia,* no. 105, pp. 1827-1832, 2017. |
| [12] | J. P. Holman, "Heat Transfer," in *Heat Transfer*, New York, McGraw-Hill, 2010, p. 713. |
| [13] | S. H. Kang, "Design and Experimental Study of ORC (Organic Rankine Cycle) and a Radial Turbine Using R245fa Working Fluid," *Energy,* no. 41, pp. 514-524, 2011. |
| [14] | American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc., "Chapter 30 - Thermophysical Properties of Refrigerants," in *2009 ASHRAE Handbook*, Atlanta, https://app.knovel.com/hotlink/toc/id:kpASHRAE37/ashrae-handbook-fundamentals/ashrae-handbook-fundamentals, 2009, p. 75. |
| [15] | C. B. Arvind Thekdi, "Waste Heat Reduction and Recovery Options for Metals Industry," *Energy Technology. John Wiley & Sons, Inc.,* pp. 17-24, 2011. |
| [16] | BCS, Incorporated, "Waste Heat Recovery: - Technology and Opportunities in U.S. Industry -," 2008. |
| [17] | 15 December 2017. [Online]. Available: https://en.wikipedia.org/wiki/Carnot\_cycle. |
| [18] | 28 December 2017. [Online]. Available: https://hellafunctional.com/?p=629. |

# Appendix A – Acronyms

C – Celsius

CHP – Combined cooling heating and power

F – Fahrenheit

LNG – Liquefied natural gas

ORC – Organic Rankine cycle

RC – Rankine cycle

SCRC – Super-critical Rankine cycle

SRC – Steam Rankine cycle

WHR – Waste heat recovery

WHRS – Waste heat recovery system

# Glossary

Adiabatic: Without heat transfer

Enthalpy: The sum of the internal energy of a system and the product of its pressure and volume. H = U +pV

# Appendix C – Source Code

In addition to the custom libraries written for this project, several open source libraries were also used. The source code for those libraries is not included.

## Shell.py

import ORC\_Model as orc

import os

R245fa\_db = '\\R245fa Saturated properties temperature table.csv'

os.chdir("..")# Navigate up a directory

db\_path=os.path.abspath(os.curdir)+r"\GHSP study\Additional references"+R245fa\_db

# Navigate to the directory containing the working fluid database.

condenser\_pressure = 0.3

boiler\_pressure = 0.7

turbine\_efficiency = 0.9

pump\_efficiency = 0.9

max\_source\_temp = 100

test\_temp = 0

while test\_temp < max\_source\_temp:

(Wm,eff,b\_temp,c\_temp,Qin\_m,Qout\_m) = orc.ORC\_model(condenser\_pressure,

boiler\_pressure,

turbine\_efficiency,

pump\_efficiency,

db\_path)

test\_temp = b\_temp

boiler\_pressure += 0.01

## ORC\_Model.py

import csv

import math

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import misc\_functions as mf

import sanitize\_inputs as si

import time

import os

def LMTD(T\_hot\_in, T\_hot\_out, T\_cold\_in, T\_cold\_out):

'''This function determines the log mean temperature difference of two

working fluids where the temperature at the inlet and outlet of both

fluids is known. This function is valid for a counter flow heat

exchanger.'''

LMTD = ((T\_hot\_out-T\_cold\_out) - (T\_hot\_in-T\_cold\_in)) / math.log((T\_hot\_out-T\_cold\_out)/(T\_hot\_in-T\_cold\_in))

return(LMTD)

def boiler\_model():

pass

def condenser\_model():

pass

def turbine\_model():

pass

def pump\_model():

pass

def ORC\_model(cond\_pres, boil\_pres, eff\_t, eff\_p,working\_fluid\_db):

'''This function takes a working pressure for a condenser and boiler, an

efficiency for a pump and turbine, and outputs the work per unit mass flow

and thermal efficiency for a generic ORC, as well as the working temperature

and enthalpy at each of the four fixed states which can then be used in the

models for the individual major components.'''

temp\_col = 0 # Degrees Celsius

press\_col = 1 # MPa

v\_col = 3 # Specific volume of vapor m3/kg

hl\_col = 4 # Enthalpy of saturated liquid kJ/kg

hv\_col = 5 # Enthalpy of saturated vapor kJ/kg

sl\_col = 6 # Entropy of saturated liquid kJ/(kgK)

sv\_col = 7 # Entropy of saturated vapor kJ/(kgK)

x1,y1,x2,y2 = mf.vlookup(working\_fluid\_db,

boil\_pres, press\_col, hv\_col)

h1 = mf.interpolate(x1,y1,x2,y2,boil\_pres)

x1,y1,x2,y2 = mf.vlookup(working\_fluid\_db,

boil\_pres, press\_col, sv\_col)

s1 = mf.interpolate(x1,y1,x2,y2,boil\_pres)

s2 = s1 # This is the adiabatic volumetric assumption in the turbine

x1,y1,x2,y2 = mf.vlookup(working\_fluid\_db,

cond\_pres, press\_col, hl\_col)

h3 = mf.interpolate(x1,y1,x2,y2,cond\_pres)

x1,y1,x2,y2 = mf.vlookup(working\_fluid\_db,

cond\_pres, press\_col, sl\_col)

s3 = mf.interpolate(x1,y1,x2,y2,cond\_pres)

# get h3 enthalpy of a saturated liquid at condenser pressure

# get s3 entropy of a saturated vapor at condenser pressure

# determine if the working fluid is wet, dry, or adiabatic.

x1,y1,x2,y2 = mf.vlookup(working\_fluid\_db,

cond\_pres, press\_col, sl\_col)

s2f = mf.interpolate(x1,y1,x2,y2,cond\_pres)

x1,y1,x2,y2 = mf.vlookup(working\_fluid\_db,

cond\_pres, press\_col, sv\_col)

s2g = mf.interpolate(x1,y1,x2,y2,cond\_pres)

try:

quality = (s2 - s2f)/(s2g - s2f)

except ZeroDivisionError:

quality = 0

h2f = h3

# h2g = enthalpy at state 2 for a saturated liquid

x1,y1,x2,y2 = mf.vlookup(working\_fluid\_db, cond\_pres, press\_col, hv\_col)

h2g = mf.interpolate(x1,y1,x2,y2, cond\_pres)

h2fg = h2g-h2f

h2s = h2f + quality\* h2fg

h2 = h1 - eff\_t\*(h1 - h2s)

x1,y1,x2,y2 = mf.vlookup(working\_fluid\_db, boil\_pres, press\_col, v\_col)

specific\_vol\_3 = mf.interpolate(x1,y1,x2,y2, boil\_pres)

h4 = h3 + (specific\_vol\_3\*(boil\_pres-cond\_pres))/eff\_p

W\_m = h1-h2-h4+h3 # kilowatts of power per kg/s of mass flow rate

Qin\_m = h1-h4 # kilowatts of heat trasfer in per kg/s of mass flow rate

Qout\_m = h2-h3 # kilowatts of heat transfer out per kg/s of mass flow rate

efficiency = ((h1-h2) - (h4-h3))/(h1-h4)

print("Quality: {:4.2f}\nPower: {:4.2f}kW/(kg/s)\nEfficiency: {:4.2f}" \

.format(quality, W\_m, efficiency))

print("Heat in: {:4.2f}kW/(kg/s)\nHeat out: {:4.2f} kW/(kg/s)"\

.format(Qin\_m,Qout\_m))

x1, y1, x2, y2 = mf.vlookup(working\_fluid\_db,

cond\_pres, press\_col, temp\_col)

cond\_temp = mf.interpolate(x1,y1,x2,y2,cond\_pres)

x1,y1,x2,y2 = mf.vlookup(working\_fluid\_db,

boil\_pres, press\_col, temp\_col)

boil\_temp = mf.interpolate(x1,y1,x2,y2,boil\_pres)

print("Condenser temperature: {:4.2f} deg Celsius\nBoiler temperature: {:4.2f} deg Celsius" \

.format(cond\_temp,boil\_temp))

return(W\_m,efficiency,boil\_temp,cond\_temp,Qin\_m,Qout\_m)

#------Main------#

if \_\_name\_\_ == '\_\_main\_\_':

'''Manual entry begins here.'''

condenser\_pressure = si.get\_real\_number("Enter condenser pressure (MPa).\n>>>",lower = 0)

boiler\_pressure = si.get\_real\_number("Enter boiler pressure (MPa).\n>>>", lower = condenser\_pressure)

turbine\_efficiency = si.get\_real\_number("Enter the turbine efficiency (0-1).\n>>>",

upper=1.0,lower=0)

pump\_efficiency = si.get\_real\_number("Enter the pump efficiency (0-1).\n>>>",

upper=1.0, lower=0)

max\_heat = si.get\_real\_number("Enter maximum heat source temperature (C).\n>>>", lower = -273)

# Run cycle model

boil\_temp = float("Inf")

while boil\_temp > max\_heat:

(Wm,efficiency,boil\_temp,cond\_temp,Qin\_m,Qout\_m) = ORC\_model(condenser\_pressure,boiler\_pressure,turbine\_efficiency,pump\_efficiency)

boiler\_pressure -= 0.01

print("Boiler pressure: {}".format(boiler\_pressure))

# Determine boiler heat exchanger size

lmtd = LMTD(T\_hot\_in, T\_hot\_out, T\_cold\_in, T\_cold\_out)

time.sleep(30)

else:

pass

## Sanitize\_inputs.py

'''This package allows the user to request input from the user and handles

most error checking and input rules.'''

\_\_version\_\_ = "0.3.0"

import numpy as np

import readchar

from colorama import init

init()

# select function built by Kamik423 in cutie library

def select(

options,

deselected\_prefix: str = '\033[1m[ ]\033[0m ',

selected\_prefix: str = '\033[1m[\033[32;1mx\033[0;1m]\033[0m ',

selected\_index: int = 0) -> int:

"""Select an option from a list.

Args:

options (List[str]): The options to select from.

deselected\_prefix (str, optional): Prefix for deselected option ([ ]).

selected\_prefix (str, optional): Prefix for selected option ([x]).

selected\_index (int, optional): The index to be selected at first.

Returns:

int: The index that has been selected.

"""

print('\n' \* (len(options) - 1))

while 1:

print(f'\033[{len(options) + 1}A')

for i, option in enumerate(options):

print('\033[K{}{}'.format(

selected\_prefix if i == selected\_index else deselected\_prefix,

option))

keypress = readchar.readkey()

if keypress == readchar.key.UP:

if selected\_index == 0:

selected\_index = len(options) - 1

else:

selected\_index -= 1

elif keypress == readchar.key.DOWN:

if selected\_index == len(options) - 1:

selected\_index = 0

else:

selected\_index += 1

else:

break

return selected\_index

class col\_vec():

'''Retrieves a list of real number for x, y, and z from the user,

and constructs a numpy column vector.'''

def \_\_init\_\_(self, coords):

self.x = coords[0]

self.y = coords[1]

self.z = coords[2]

self.vec = np.array([[self.x],[self.y],[self.z]])

def get\_real\_number(prompt=None, upper=float('Inf'), lower=float('-Inf')):

'''Gets a real number from the user with an optional prompt. Positive and

negative limits can be set. If not set, the default values are 'Inf' and

'-Inf' respectively.'''

num\_flag = False

while(not num\_flag):

try:

number = float(input(prompt))

if lower < number < upper:

num\_flag = True

else:

print("value must be between {} and {} exclusive.".format(lower, upper))

print("\033[2A\033[K\033[1A\033[K\r", end='')

except ValueError:

print("\033[1A\033[K\033[1A\033[K\r", end='')

num\_flag = False

print("\033[K", end='')

return(number)

def get\_integer(prompt=None, upper=float('Inf'), lower=float('-Inf')):

'''Gets an integer from the user with an optional prompt. Positive and

negative limits can be set. If not set, the default values are 'Inf' and

'-Inf' respectively.'''

num\_flag = False

while(not num\_flag):

try:

number = int(input(prompt))

number += 0

# This will throw an exception if number is not an integer.

if lower < number < upper: # excludes endpoints

num\_flag = True

else:

print("value must be between {} and {} exclusive.".format(lower, upper))

print("\033[2A\033[K\033[1A\033[K\r", end='')

except ValueError:

print("\033[1A\033[K\033[1A\033[K\r", end='')

# \033[K = Erase to the end of line

# \033[1A = moves the cursor up 1 line.

# \r = return

num\_flag = False

print("\033[K", end='')

return(number)

def get\_letter(prompt=None, accept=None):

'''Gets a single alpha character that is included in the list 'accept'

Optionally include a prompt to the user

omitting the accept list allows all alpha characters.'''

flag = False

while(not flag):

letter = str(input(prompt))

if(letter.isalpha() and len(letter) == 1):

if accept != None:

for i in accept:

if letter == i or accept==None:

flag = True

break

else:

pass

else:

flag = True

else:

pass

return(letter)

def get\_coords(rows=3):

'''This function gets the coordinates for a point in 3D space from the user.

It includes the error checking logic required to ensure the point's

useability in subsequent functions.'''

P\_x = get\_real\_number("X >>> ")

P\_y = get\_real\_number("Y >>> ")

P\_z = get\_real\_number("Z >>> ")

point = col\_vec([P\_x,P\_y,P\_z])

if rows == 3:

return(point)

elif rows == 4:

point.vec = np.row\_stack([point.vec,[1]])

return(point)

else:

print("Invalid argument.")

return(None)

## Misc\_functions.py

'''It is recommended to use this package with the sanitize\_inputs package.\n

The functions contained herein do not check for erroneous inputs.'''

\_\_version\_\_ = "0.2.6"

import math

import csv

import pandas as pd

import pdb

import os

from difflib import SequenceMatcher

import datetime as dt

import time

def dxdy(df, y\_col, x\_col):

'''Given a dataframe and labels of an x and y column, This function

returns the data frame with an additional column dx/dy that is the first

derivative of x with respect to y.'''

df\_shifted = df.shift(1)

df["dx/dy"] = (df[y\_col]-df\_shifted[y\_col])/(df[x\_col]-df\_shifted[x\_col])

return(df)

def activate(x, k, epsilon=1):

'''This function takes a value x, a trigger value k, and a smoothing factor

epsilon and returns a number between 0 and 1 with an epsilon smoothed

transition at x=k.'''

y=math.tanh((x-k)/epsilon)/2+0.5

return(y)

def deactivate(x, k, epsilon=1):

'''This function takes a value x, a trigger value k, and a smoothing factor

epsilon and returns a number between 0 and 1 with an epsilon smoothed

transition at x=k.'''

y=math.tanh((x-k)/epsilon)/-2+0.5

return(y)

def timestamp():

now = dt.datetime.fromtimestamp(time.time())

sep=":"

stamp = sep.join([str(now.year),

str(now.month),

str(now.day),

str(now.hour),

str(now.minute),

str(now.second)])

return("["+stamp+"]")

def similar(a,b):

return(SequenceMatcher(None,a,b).ratio())

def similar\_dir(directory, desired, threshold=0.75):

'''Takes the name of a desired subdirectory, and returns a directory that

best matches the desired one with certainty above given threshold.'''

candidates = []

for f in os.listdir(directory):

candidates.append((f, similar(f,desired)))

candidates.sort(key=lambda x: x[1],reverse=True)

best\_match = candidates[0]

if best\_match[1] >= threshold:

return(best\_match[0])

else:

return(None)

def interpolate(x1,y1,x2,y2,x):

'''This function returns a value, y, linearly interpolated using two x,y

pairs of data and a given x between those pairs.'''

try:

y = ((y2-y1)/(x2-x1))\*(x-x1) + y1

except TypeError:

y = y1

return(y)

def interpolate\_y(x1,y1,x2,y2,y):

'''This function returns a value x, linearly interpolated using two x,y

pairs of data and a given y between those pairs.'''

try:

m = (y2-y1)/(x2-x1)

b = y1

x = (y-b)/m

except TypeError:

x = x1

return(x)

def tab\_dict(rfile):

'''This is a function that opens an excel file and returns a dictionary

where the keys of the dictionary are the sheet names and the values are

dataframes containing the data from the sheet. rfile must include the path

if the file is not in the current working directory.'''

try:

xlsx = pd.ExcelFile(rfile)

Sheet\_frames = {sh:xlsx.parse(sh) for sh in xlsx.sheet\_names}

# This line creates a dictionary where the keys are the tab names,

# and the values are the data from that tab.

return(Sheet\_frames)

except FileNotFoundError:

print(rfile,"Does not exist.")

return(None)

def list\_headers(rfile, r\_c='r'):

'''rfile is the csv file in which the data are stored. pass 'r' or 'c' for

the second argument to indicate whether the headers are in the first row or

the first column.'''

headers = []

RDR = csv.reader(open(rfile))

if r\_c.lower() == 'c':

for row in RDR:

print(row[0])

headers.append(row[0])

elif r\_c.lower() == 'r':

headers = next(RDR)

return(headers)

def vlookup(rfile, index, search\_col, result\_col,skip\_headers=False):

'''rfile is the name of file in which data are stored. index is the value

to search database rows for. search\_col is the column in which the

index can be found. result\_col should be the column from which the result

should be extracted. This function is made to work smoothly with

interpolate() Skip headers allows the user to skip searching the first row

which will not happen automatically if the column labels are numbers.'''

index = float(index)

search\_col = int(search\_col)

result\_col = int(result\_col)

RDR = csv.reader(open(rfile,'r'), dialect = 'excel')

pos\_diff = math.inf

neg\_diff = math.inf\*-1

x1 = None

y1 = None

x2 = None

y2 = None

for i, row in enumerate(RDR):

# Search for the rows just smaller and just larger than the search

# term. Calculate the difference between the x value in a given row

# and the search term. Keep the rows that result in the smallest

# positive difference and the smallest negative difference.

if i == 0 and skip\_headers:

#next(RDR)

print("Advanced a row")

continue

try:

diff = index - float(row[search\_col])

except ValueError:

if row[search\_col] == "Inf":

diff = math.inf

continue

if diff < pos\_diff and diff > 0:

x1 = float(row[search\_col])

y1 = float(row[result\_col])

pos\_diff = diff

elif diff > neg\_diff and diff < 0:

x2 = float(row[search\_col])

y2 = float(row[result\_col])

neg\_diff = diff

elif diff == 0:

x1 = float(row[search\_col])

y1 = float(row[result\_col])

x2 = None

y2 = None

return (x1, y1, x2, y2)

# Return the x,y pairs of the search column and result column just

# above and below the desired x value.

def bernoulli\_trial(n, k, p):

'''Returns the probability between 0 and 1 of exactly k successes given

n trials where the probability of success is p. k and n must be integers

and p is a float between 0 and 1.'''

q = 1-p

binomial\_coeff = math.factorial(n)/(math.factorial(k)\*math.factorial(n-k))

P = binomial\_coeff\*(p\*\*k)\*(q\*\*(n-k))

return(P)

def bernoulli\_trial\_n(k, p, P=0.95):

'''Returns the sample size required to observe at least k successes if

the probability of success in each trial is p to a confidence level of P.'''

n = k # Sample size must be at least k in order to observe k successes.

test\_P = 0

while test\_P < P:

test\_P = 0

for test\_k in range(k, n-1):

test\_P += bernoulli\_trial(n, test\_k, p)

n+= 1

print("One can be {0:4.2f} confident that at least {1} successes will occur if {2} parts are tested.".format(test\_P, k, n))

return(n)

def favstats(rfile, column):

'''This function calculates common statistical values for a given column

of data found in the specified file.'''

df = pd.read\_csv(rfile)

xbar = df[column].mean()

sd = df[column].std()

minimum = df[column].min()

first = df[column].quantile(0.25) # first quartile

median = df[column].median()

third = df[column].quantile(0.75) # third quartile

maximum = df[column].max()

IQR = third - first

print("Minimum: ", minimum,

"\nFirst quartile: ", first,

"\nMedian: ", median,

"\nMean: ", xbar,

"\nThird quartile: ",third,

"\nMaximum: ", maximum,

"\nStandard deviation: ",sd,

"\nInter-quartile range: ",IQR,sep='')

def t\_test\_file(rfile, col, xbar=0, alpha=0.05, twotail=True, lower=True):

'''One sample t-test. Arguments are the csv file in which the data are

located and the column in which the data are found along with an alpha

value. var is the column name in which the category of interest is stored.

col is the column in which the response variable is stored. xbar is the

variable to which the mean will be compared. twotail tells the function

whether it should do a two tail test as opposed to a one tail test. lower

is ignored for two tail, but determines which tail is considered in the one

tail variant.'''

df = pd.read\_csv(rfile)

# This line pulls data out of the data frame creating two new data frames

# one for each label.

# The resulting data structure is a tuple where element 0 is the group name

# and element 1 is the actual sub-dataframe.

xbar\_test = df[col].mean()

sd = df[col].std()

n = len(df[col])

DOF = n-1

# Look up the appropriate t statistic - a 2 parameter interpolation function

# would be nice here for an arbitrary value of alpha.

if twotail:

lookupfile = "twotail tstat.csv"

else:

lookupfile = "onetail tstat.csv"

headers = list\_headers(lookupfile,'r')

for i, h in enumerate(headers):

try:

if float(h) == float(alpha):

print("Alpha level is: ",float(alpha))

else:

pass

except ValueError:

continue

x1,y1,x2,y2 = vlookup(lookupfile, DOF, 0, i,skip\_headers=False)

tsalpha = interpolate(x1,y1,x2,y2,DOF)

std\_err = sd/n\*\*0.5

# calculate the confidence interval

diff = (xbar\_test - xbar)

upper = (diff) + tsalpha\*std\_err

lower = (diff) - tsalpha\*std\_err

print((1-float(alpha))\*100,"% Confidence interval: ",lower," - ",upper,sep='')

# calculate p-value

ts = abs(diff/std\_err)

if twotail:

#find p for given ts in twotail tstat.csv

lookupfileT = ("twotail tstat Transpose.csv")

else:

#find p for given ts in onetail tstat.csv

lookupfileT = ("twotail tstat Transpose.csv")

if lower:

pass

else:

pass

headersT = list\_headers(lookupfileT,'r')

for i, h in enumerate(headersT):

try:

if float(h) == float(DOF):

break

else:

pass

except ValueError:

continue

x1,y1,x2,y2 = vlookup(lookupfileT, ts, i, 0,skip\_headers=True)

print("({0},{1}) - ({2},{3})".format(x1,y1,x2,y2))

print("avg: {0}\nsd: {1}\nn: {2}\ndiff: {3}\nstd\_err: {4}"\

.format(xbar\_test,sd,n,diff,std\_err))

print("ts = {0}".format(ts))

p = interpolate(x1,y1,x2,y2,ts)

print("p = ",p)

# formulate conclusion

def t\_test2\_file(rfile, var, c1, c2, treat, alpha=0.05, twotail=True, lower=True):

'''Two sample t-test. Arguments are the csv file in which the data are

located and the two columns to be compared along with an alpha value.

var is the column name in which the categories are stored, c1 and c2 are

the two labels in that column to be compared. treat is the treatment

varible. ie the variable that will be used to compare the groups. twotail

tells the function whether it should do a two tail test as opposed to a one

tail test. lower is ignored for two tail, but determines which tail is

considered in the one tail variant.'''

df = pd.read\_csv(rfile)

groups = dict((x,y) for x,y in df.groupby(var))

# This line pulls data out of the data frame creating two new data frames

# one for each label.

# The resulting data structure is a tuple where element 0 is the group name

# and element 1 is the actual sub-dataframe.

xbar1 = groups[c1][treat].mean()

xbar2 = groups[c2][treat].mean()

# Pandas standard deviation function uses Bessel's correction by default.

s1 = groups[c1][treat].std()

s2 = groups[c2][treat].std()

n1 = len(groups[c1][treat])

n2 = len(groups[c2][treat])

n = min(n1,n2)

DOF = n-1

# n will be used to calculate the standard error. Choosing the smaller of

# the two sample sizes yields a conservative estimate.

# Calculate the pooled standard deviation

sp = (((n1-1)\*s1\*\*2+(n2-1)\*s2\*\*2)/(n1+n2-2))\*\*0.5

# Look up the appropriate t statistic - a 2 parameter interpolation function

# would be nice here for an arbitrary value of alpha.

if twotail:

lookupfile = "twotail tstat.csv"

else:

lookupfile = "onetail tstat.csv"

headers = list\_headers(lookupfile,'r')

for i, h in enumerate(headers):

try:

if float(h) == float(alpha):

break

else:

pass

except ValueError:

continue

x1,y1,x2,y2 = vlookup(lookupfile, DOF, 0, i,skip\_headers=True)

print("DOF: {}\ni: {}".format(DOF,i))

tsalpha = interpolate(x1,y1,x2,y2,DOF)

# There is an issue here where if a sample size n=2, DOF=1 and the lookup

# table returns a nonetype

std\_err = sp/n\*\*0.5

# calculate the confidence interval

diff = (xbar1 - xbar2)

upper = (diff) + tsalpha\*std\_err

lower = (diff) - tsalpha\*std\_err

# calculate p-value

ts = abs(diff/std\_err)

if twotail:

#find p for given ts in twotail tstat.csv

lookupfileT = ("twotail tstat Transpose.csv")

else:

#find p for given ts in onetail tstat.csv

lookupfileT = ("twotail tstat Transpose.csv")

if lower:

pass

else:

pass

headersT = list\_headers(lookupfileT,'r')

for i, h in enumerate(headersT):

try:

if float(h) == float(DOF):

break

else:

pass

except ValueError:

continue

x1,y1,x2,y2 = vlookup(lookupfileT, ts, i, 0,skip\_headers=True)

print("x1,y1,x2,y2",x1,y1,x2,y2)

p = interpolate(x1,y1,x2,y2,ts)

# formulate conclusion

print("xbar1 = ",xbar1,

"\nxbar2 = ",xbar2,

"\ns1 = ",s1,

"\ns2 = ",s2,

"\nn1 = ",n1,

"\nn2 = ",n2,

"\nn = ",n,

"\nsp = ",sp,

"\ntsalpha = ",tsalpha,

"\nts = ",ts,

"\ndiff = ",diff,

"\nupper = ",upper,

"\nlower = ",lower,

"\nstd\_err = ",std\_err,

"\np = ",p,sep='')

return(df)

def paired\_t\_test(DF1, DF2, alpha=0.05, mu=0, twotail=True, lower=True):

'''This function takes two pandas dataframes which must have the same index

and find the change in each item and perform a t-test to compare the

change to a given number.

Index entries which don't appear in both dataframes will be removed

automatically.'''

diffDF = pd.DataFrame()

for col in DF1:

try:

diffDF[col] = DF1[col].sub(DF2[col])

except:

print("Paired data can only be calculated for numeric data.")

continue

diffDF = diffDF.dropna()

print(diffDF)

p = t\_test(diffDF, alpha, mu, twotail, lower)

return(p)

def t\_test(series, alpha=0.05, mu=0, twotail=True, lower=True):

'''This function performs a 1 sample t-test on a pandas data series rather

than a file.'''

xbar\_test = series.mean()

sd = series.std()

n = len(series)

DOF = n-1

# Look up the appropriate t statistic - a 2 parameter interpolation function

# would be nice here for an arbitrary value of alpha.

if twotail:

lookupfile = "twotail tstat.csv"

else:

lookupfile = "onetail tstat.csv"

headers = list\_headers(lookupfile,'r')

for i, h in enumerate(headers):

try:

if float(h) == float(alpha):

print("Alpha level is: ",float(alpha))

else:

pass

except ValueError:

continue

x1,y1,x2,y2 = vlookup(lookupfile, DOF, 0, i,skip\_headers=False)

tsalpha = interpolate(x1,y1,x2,y2,DOF)

std\_err = sd/n\*\*0.5

# calculate the confidence interval

diff = (xbar\_test - mu)

upper = (diff) + tsalpha\*std\_err

lower = (diff) - tsalpha\*std\_err

print((1-float(alpha))\*100,"% Confidence interval: ",lower," - ",upper,sep='')

# calculate p-value

ts = abs(diff/std\_err)

if twotail:

#find p for given ts in twotail tstat.csv

lookupfileT = ("twotail tstat Transpose.csv")

else:

#find p for given ts in onetail tstat.csv

lookupfileT = ("twotail tstat Transpose.csv")

if lower:

pass

else:

pass

headersT = list\_headers(lookupfileT,'r')

for i, h in enumerate(headersT):

try:

if float(h) == float(DOF):

break

else:

pass

except ValueError:

continue

x1,y1,x2,y2 = vlookup(lookupfileT, ts, i, 0,skip\_headers=True)

print("({0},{1}) - ({2},{3})".format(x1,y1,x2,y2))

print("avg: {0}\nsd: {1}\nn: {2}\ndiff: {3}\nstd\_err: {4}"\

.format(xbar\_test,sd,n,diff,std\_err))

print("ts = {0}".format(ts))

p = interpolate(x1,y1,x2,y2,ts)

print("p = ",p)

return(p)

def r\_ch\_arc(Arc, Chord, dr):

'''Find the radius of a circle given a chord length and an arc length. This

is a numerical solution. The argument dr is the desired level of

precision.'''

a = float(Arc)

c = float(Chord)

dr = float(dr)

radius = 0

error = 100

# numberical solution for radius. Iterates until error is less than

# specified dr.

while (error > dr):

radius += dr

tempA = math.sin(a/(2\*radius))

tempB = c/(2\*radius)

error = abs(tempA-tempB)

return(radius)

## Heat\_exchanger\_model.py

import csv

import math

from mpl\_toolkits.mplot3d import axes3d

import matplotlib.pyplot as plt

import numpy as np

import misc\_functions as mf

whr\_mass\_flow\_rate = 1 # kg/s

feedwater\_mass\_flow\_rate = 2.25 # kg/s

feedwater\_density = 1.08 # kg/L

F = 1.0 # LMTD correction factor for boiler

Th245 = 90 # Celsius

Tc245 = 60 # Celsius

C245 = 1.302 # Heat capacity KJ/(kg K) @atm press & 15.14 deg C

Cfeed = 3.3488 # Heat capacity KJ/(kg K) @atm press & 26.7 deg C

Thfeedwater = 97.5 # Temperature of automotive feedwater

whr\_efficiency = 0.1366 # From previous study

# Heat energy required to raise working fluid temperature.

q\_in = whr\_mass\_flow\_rate\*C245\*(Th245 - Tc245)

print("Heat transfer in: ", q\_in) # kW

W\_out = q\_in \* whr\_efficiency

print("Work out: ", W\_out)

# Assuming that the heat transferred into the working fluid is 100% of the heat

# transferred out of the feedwater, the temperature drop of the feedwater can

# be found.

dT = q\_in/(feedwater\_mass\_flow\_rate\*Cfeed)

print("dT: ", dT)

Tcfeedwater = Thfeedwater - dT

# LMTD Method

dTm = ((dT) - (Th245 - Tc245))/math.log(dT/(Th245 - Tc245))

print("Mean temperature difference: ", dTm)

# In the future, the heat transfer coefficient should be calculated. for now,

# table 10-1 form "Heat Transfer" by J.P. Holman was used Umin = 850 W/(m2 C),

# Umax = 1400 W/(m2 C).

U = 850 # W/(m2 C)

A = q\_in\*1000/(U\*dTm) # square meters

print("Boiler heat exchange area: ", A)

# Calculate the number of tubes in shell based on tube diameter and tube and

# and shell exchanger length.

L = 0.25 # meters

d = 0.003 # diameter of tubes in meters

circ = A/L # circumference of all tubes

n = circ/(math.pi\*d)

print("Number of tubes: ", n)

1. Azeotrope: n: A liquid mixture that is characterized by a constant minimum or maximum boiling point which is lower or higher than that of any of the components. [↑](#footnote-ref-1)
2. This equation is only valid for adiabatic, internally reversible processes. A different method of determining the enthalpy of a fluid at state 2 will be required for adiabatic and dry working fluids – Those working fluids with a positively sloped saturated vapor line. [↑](#footnote-ref-2)