AUTOMATIC PERFORMANCE CALCULATION OF IDEAL VAPOR COMPRESSION REFRIGERATION CYCLE

BY

Mr. CHANATIP CHAIPINIT 63011122

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF ENGINEERING IN FOURTH YEAR KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG ACADEMIC YEAR 2023

SCHOOL OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG PROJECT
CERTIFICATE

Project Title	AUTOMATIC PERFORMANCE CALCULATION OF IDEAL VAPOR COMPRESSION REFRIGERATION CYCLE
Student Name	Mr. Chanatip Chaipinit Student ID 63011122
Degree	Bachelor of Engineering
Major	Energy Engineering
Project Advisor	Dr. kittipass Wasinarom

Signed: _____

(Dr. kittipass Wasinarom)

Project Title	Automatic performance calculation of ideal vapor compression refrigeration cycle
Student Name	Mr. Chanatip Chaipinit Student ID 63011122
Degree	Bachelor of Engineering
Major	Energy Engineering
Project Advisor Project	Dr. kittipass Wasinarom
Co-Advisor	Assoc.Prof. International Program, D. Eng
Academic Years	2023

Abstract

This project introduces an automated method for evaluating the performance of ideal vapor compression refrigeration cycles, focusing primarily on air conditioning systems. Traditional performance metrics in this field typically center on energy consumption, often neglecting critical thermal and pressure dynamics within the system. To address this limitation, our approach integrates direct pressure measurements from both the high side (condenser) and low side (evaporator) of the refrigeration cycle. These measurements are taken using advanced pressure gauges, strategically positioned to capture data immediately upon activation of the air conditioning unit.

Once collected, these pressure values are fed into a python software tool developed specifically for this project. The software is designed to compute essential performance indicators automatically, including the heat absorbed by the refrigerant (QL), the heat rejected (QH), the work input to the compressor (WIN), and the overall Coefficient of Performance (COP). These calculations allow for a nuanced understanding of the system's efficiency and operational characteristics.

To ensure the robustness and applicability of our method, the system is tested under two distinct conditions: operating the air conditioning unit within a controlled indoor environment and in an uncontrolled outdoor setting. This dual testing protocol is devised to evaluate the system's adaptability to environmental variances, thus providing a thorough analysis of performance fluctuations and their implications under different climatic conditions.

The anticipated outcomes of this project include pioneering contributions towards the integration of real-time data interfaces between pressure sensors and analytical software, specifically Python programs. This study aims to develop and refine a Python-based program capable of calculating

crucial thermodynamic parameters from live pressure data inputs. Additionally, we plan to construct a small-scale test rig that will allow us to simulate and observe the behavior of refrigeration cycles under controlled conditions. This initiative not only focuses on the technological integration of sensor data into programmable platforms but also enhances the practical understanding of vapor compression cycle dynamics. Through these developments, the project seeks to improve the precision of data analysis and foster more informed decisions regarding system design, maintenance, and operation. Ultimately, this could lead to significant advancements in energy efficiency and cost reductions in air conditioning and refrigeration systems across various industries.

Acknowledgement

I am deeply grateful to a number of individuals whose expertise, assistance, and support were instrumental in the completion of this project.

Foremost, I extend my heartfelt gratitude to my advisor, Kittipass Wasinarom, whose insights and guidance were invaluable throughout this research. Their expertise in the field of thermodynamics and dedication to academic excellence significantly shaped both the direction and success of this study. Their patience and commitment to nurturing my scholarly skills have left a profound impact on my personal and professional growth.

I would also like to thank the technical staff and my fellow researchers at the laboratory whose assistance was crucial in setting up the experimental apparatus and collecting data. Their readiness to help and their attention to detail ensured the precision of our results.

My appreciation goes out as well to my peers who provided both support and constructive criticism, pushing me to refine my analysis and broaden my understanding of the subject matter.

Finally, I am indebted to my family and friends for their unwavering support and encouragement throughout the duration of my studies. Their belief in my capabilities has been a constant source of motivation. This research benefitted from the contributions of each of these individuals, and I am truly thankful for their roles in bringing this project to fruition.

TABLE OF CONTENTS

page

ABSTRACT (i)

ACKNOWLEDGEMENTS (ii)

LIST OF FIGURES (iii)

CHAPTER 1 INTRODUCTION

1.1 Background and significance 1

1.2 Research objective 1

1.3 Scope of Research 2

1.4 Method of Conducting Research 2

1.5 Expected Results 2

CHAPTER 2 Concepts, Theories and Related Research

2.1 Cycle Analysis of Vapor Compression Refrigeration 4

2.1.1 Isentropic Compression (Process 1-2) 5

2.1.2 Constant Pressure Heat Rejection (Process 2-3) 5

2.1.3 Isenthalpic Expansion (Process 3-4) 5

2.1.4 Constant Pressure Heat Absorption (Process 4-1) 6

2.1.5 T – S diagram 6

2.1.5.1 Basics of T-s Diagram 7

2.1.5.2 Key Features on a T-s Diagram 7

2.1.5.3 Using T-s Diagram for Thermodynamic Cycles 7

2.1.5.4 Practical Importance 8

2.2 Component in air conditioning 9

2.2.1 Compressor 9

2.2.2 Condenser 10

2.2.3 Expansion Device (Throttling Valve) 11

2.2.4 Evaporator 11

2.3 Saturation and Superheat in air conditioning 12

2.3.1 Saturation 12

2.3.2 Superheat 13

2.4 Parameter in air conditioning 14

2.4.1 Heat absorbed by the refrigerant 14

2.4.2 Heat rejection 14

2.4.3 Work output 15

2.4.4 Coefficient of performance 15

CHAPTER 3 Methodology

3.1 Flowchart Creation 17

3.2 Pseudocode Development 18

3.3 Write actual code from pseudocode 18

3.3.1 Import Required Libraries 18

3.3.2 Create Interpolate Functions 20

3.3.3 Calculate Evaporator Enthalpy (H1) and Entropy (S1) using Interpolated Function 20

3.3.4 Calculate Condenser Enthalpy (H3) and Entropy (S3) using Interpolated Function 21

3.3.4.1 User Input for Pressure (P3) 22

3.3.4.2 Interpolation Calculations (H3 and S3) 22

3.3.4.3 Variable Assignment (P2) 23

3.3.4.4 Outputting and Formatting Interpolated Values in Console Applications 23

3.3.5 Create R22 Table (Superheated) 24

3.3.6 Superheat vapor Calculate Compressor Enthalpy (H2)

and Entropy (S2) by Compressor Pressure (P2) 24

3.3.6.1 Conditional Check for P2 Equal to 160 24

3.3.6.2 Conditional Check for P2 Between 160 and 170 25

3.3.7 Calculate parameter 26

3.3.7.1 Heat Removed (QL) 26

3.3.7.2 Work Input (Win) 26

3.3.7.3 Heat Added (QH) 27

3.3.7.4 Coefficient of Performance (COP) 28

3.4 Experimental Setup 28

3.5 Methodology for Varying Evaporator Temperature 29

3.6 Methodology for Varying Condenser Temperature 30

CHAPTER 4 EXPERIMENTAL RESULT

4.1 Analysis for Outside Environment Temperature at 29 degrees Celsius 32

4.2 Analysis for Inside Room Temperature at 23 degrees Celsius 33

4.3 Comparative Analysis and Thermodynamic Implications 34

CHAPTER 5 CONCLUSION 35

APPENDIX 46

LIST OF FIGURES

Figures

2.1 Air conditioning cycle 8

2.2 Compressor air 9

2.3 Condenser air 10

2.4 Throttling valve 11

2.5 Evaporator air 11

2.6 R-22 Saturation table 12

2.7 R-22 Superheat table 13

3.1 Flowchart coding 17

3.2 Import library 18

3.3 Interpolate function 19

3.4 Input evaporator pressure (P1) in order to calculate H1 and S1 20

3.5 Input evaporator pressure (P3) in order to calculate H3 and S3 21

3.6 Input pressure condenser 21

3.7 Set interpolate P3 22

3.8 From T-S diagram P3 equal to P2 22

3.9 Output interpolate 23

3.10 Period at 160 to 2100 kpa superheat value 23

3.11 Function interpolate in superheat 24

3.12 Set the first range value 24

3.13 Write the condition interpolate 25

3.14 Step to calculate each parameter 26

3.15 Q_L formula 26

3.16 Q_H formula 26

3.17 W_{in} formula 27

3.18 COP formula 27

3.19 COP - Temperature evaporator graph 29

3.20 COP - Temperature condenser graph 30

4.1 Pressure gauge evaporator and condenser at outside 32

4.1 Pressure gauge evaporator and condenser at inside 33

Chapter 1
Introduction

1.1 Background and significance

Vapor compression refrigeration cycles are essential in various applications ranging from domestic cooling to industrial processes. Traditional assessments of these systems have predominantly focused on energy consumption metrics, often neglecting the crucial role of real-time thermodynamic parameters such as pressure. This oversight limits the accuracy of performance evaluations and the optimization potential of these systems, thereby affecting their operational efficiency and economic viability. Recognizing these gaps, our project seeks to provide a comprehensive and accurate analysis by integrating real-time pressure data into performance evaluations.

1.2 Research objective

- Development of a Python-based software: This software will automatically compute key performance indicators (KPIs) like heat absorption, heat rejection, work input, and Coefficient of Performance (COP) using real-time pressure data from sensors installed on both the high and low pressure sides of the refrigeration cycle.

- Construction and testing of a small-scale test rig: To rigorously evaluate the reliability and accuracy of our software under controlled experimental conditions.

1.3 Scope of Research

The scope of this research encompasses the design and implementation of a measurement and analysis system that integrates pressure sensor data into a Python program for real-time performance calculation. This system will be tested using a custom-built test rig that represents typical conditions found in vapor compression refrigeration systems.

1.4 Method of Conducting Research

- Installation of Pressure Sensors: Precision sensors will be installed at critical points to capture accurate pressure readings continuously.

- Software Development and Integration: Develop a user-friendly Python program that can process inputs from the sensors and calculate essential thermodynamic variables in real-time.

- Test Rig Construction and Experimental Validation: Construct a versatile test rig that can replicate various environmental conditions. This rig will be used to assess the software's calculations and the sensors' performance in both indoor and outdoor settings.

1.5 Expected Results

Enhanced Accuracy and Insight: By incorporating real-time pressure data, the project will provide more accurate and comprehensive performance assessments than current methodologies. Improved System Optimization: The detailed operational insights gained will facilitate better-informed decisions in system design, maintenance, and operational strategies. Economic and Environmental Impact: More efficient system operation will likely result in lower energy consumption and reduced operational costs, promoting economic benefits and environmental sustainability.

Chapter 2
Concepts, Theories and Related Research

In this chapter, the principles, methodologies, and relevant studies concerning the cycle analysis of vapor compression refrigeration systems are explored, as detailed in the subsequent section.

2.1 Cycle Analysis of Vapor Compression Refrigeration

2.1.1 Isentropic Compression (Process 1-2)

2.1.2 Constant Pressure Heat Rejection (Process 2-3)

2.1.3 Isenthalpic Expansion (Process 3-4)

2.1.4 Constant Pressure Heat Absorption (Process 4-1)

2.1.5 T – S diagram

2.1.5.1 Basics of T-s Diagram

2.1.5.2 Key Features on a T-s Diagram

2.1.5.3 Using T-s Diagram for Thermodynamic Cycles

2.1.5.4 Practical Importance

2.2 Component in air conditioning

2.2.1 Compressor

2.2.2 Condenser

2.2.3 Expansion Device (Throttling Valve)

2.2.4 Evaporator

2.3 Saturation and Superheat in air conditioning

2.3.1 Saturation

2.3.2 Superheat

2.4 Parameter in air conditioning

2.4.1 Heat Absorbed by the Refrigerant (Q_L)

2.4.2 Heat Rejected (Q_H)

2.4.3 Work Input (W_{in})

2.4.4 Coefficient of Performance (COP)

2.1 Cycle Analysis of Vapor Compression Refrigeration

The vapor compression refrigeration cycle is an intricate process that involves four essential stages: Isentropic Compression (Process 1-2), Constant Pressure Heat Rejection (Process 2-3), Isenthalpic Expansion (Process 3-4), and Constant Pressure Heat Absorption (Process 4-1). Each stage plays a pivotal role in the functionality and efficiency of refrigeration and air conditioning systems.

2.1.1 Isentropic Compression (Process 1-2)

The refrigerant enters the compressor as a saturated or slightly superheated vapor. The isentropic (constant entropy) compression signifies no heat transfer with the surroundings, though internal energy and enthalpy increase due to work done on the refrigerant. This results in a significant increase in both the temperature and pressure of the refrigerant. The design and efficiency of the compressor are critical. Factors like the type of compressor (e.g., screw, scroll,

centrifugal) and its operating conditions directly affect the efficiency and operational cost. Efficiency improvements, such as using variable speed drives, can align compressor output with system demand, reducing energy usage.

2.1.2 Constant Pressure Heat Rejection (Process 2-3)

After compression, the refrigerant is at its highest temperature and pressure. It travels through the condenser coils, where it rejects heat to the environment through a constant pressure process. This heat rejection includes both the heat absorbed from the refrigerated space and the heat added by the compressor. As heat is rejected, the refrigerant condenses into a high-pressure liquid. The effectiveness of the condenser depends heavily on its ability to dissipate heat. Factors affecting performance include ambient air temperature, airflow over the condenser, and the cleanliness of the coils. In designs like water-cooled systems, the water temperature and flow rate also play significant roles.

2.1.3 Isenthalpic Expansion (Process 3-4)

The expansion device (throttle valve or capillary tube) facilitates a rapid drop in pressure and temperature of the refrigerant but without a change in enthalpy. This drop leads to partial vaporization and cooling of the refrigerant, preparing it for the heat absorption process in the evaporator. The precision of the expansion device is crucial. Over or under-expansion can lead to inefficiencies or system damage. Adjustable expansion valves can optimize the refrigerant flow based on current load conditions, improving system responsiveness and efficiency.

2.1.4 Constant Pressure Heat Absorption (Process 4-1)

In the evaporator, the now low-pressure and cooler refrigerant mixture absorbs heat from the environment. This heat absorption occurs at constant pressure and results in the evaporation of the refrigerant, significantly increasing its volume and decreasing its density. The process ends

when the refrigerant exits the evaporator as a saturated vapor, ready to be compressed again. The design of the evaporator impacts its ability to effectively absorb heat. Factors like the arrangement and surface area of the coils, as well as the airflow across them, are vital. Ensuring that the evaporator does not freeze (which can happen due to excessive moisture condensation and freezing) is crucial for maintaining efficient heat transfer.

2.1.5 T – S diagram

A T-s diagram, or temperature-entropy diagram, is a graphical representation that shows the temperature (T) and entropy (S) of a thermodynamic system. It is particularly useful in visualizing the heat transfer and work interactions in thermodynamic processes and cycles, such as those found in refrigeration, air conditioning systems, and various engines. Here's a theoretical explanation of how a T-s diagram is used and what it represents:

2.1.5.1 Basics of T-s Diagram

- Temperature and Entropy Axes:
  The vertical axis represents the absolute temperature (T) of the system.
  The horizontal axis represents the entropy (S) of the system.

- Understanding Entropy:
  Entropy is a measure of the disorder or randomness in a system. In practical terms, it also relates to the amount of energy in a system that is unavailable to do work.

2.1.5.2 Key Features on a T-s Diagram

- Isotherms:

Lines of constant temperature (isotherms) run horizontally across the diagram. This reflects the definition of entropy, as changes at constant temperature (isothermal processes) do not affect the temperature but can change the entropy.

- Adiabatic Processes:

  Adiabatic processes, where no heat is transferred into or out of the system (Q=0), are represented by vertical lines because the temperature can change while the entropy remains constant, assuming no internal irreversibility.

- Constant Pressure and Volume Lines:

  Lines of constant pressure and constant volume can also be plotted on the T-s diagram. These lines help in understanding the behavior of the system under compression or expansion at constant properties.

2.1.5.3 Using T-s Diagram for Thermodynamic Cycles

- Heat Engine Cycles:

  In a typical heat engine cycle (e.g., Carnot, Otto, Diesel), the T-s diagram can be used to illustrate the compression, heat addition, expansion, and heat rejection stages of the cycle. The area within the cycle loop on the T-s diagram represents the net work output of the cycle, which is also the net area enclosed by the cycle path.

- Refrigeration Cycles:

  For refrigeration or heat pump cycles (e.g., vapor-compression cycle), the T-s diagram helps in visualizing the refrigeration process, including evaporation (heat absorption at constant pressure), compression, condensation (heat rejection at constant pressure), and expansion.

- Efficiency Analysis:

  The T-s diagram is particularly useful for analyzing the efficiency of cycles. The larger the area enclosed by the cycle on a T-s diagram, the greater the work output or heat transfer involved, relative to the entropy changes.

2.1.5.4 Practical Importance

- Design and Analysis: Engineers use T-s diagrams to design more efficient engines and refrigeration systems by analyzing the thermodynamic properties and optimizing the processes to reduce entropy generation and energy losses.

- Educational Tool: T-s diagrams serve as an excellent educational tool for students and professionals to understand and apply the principles of thermodynamics in practical scenarios.

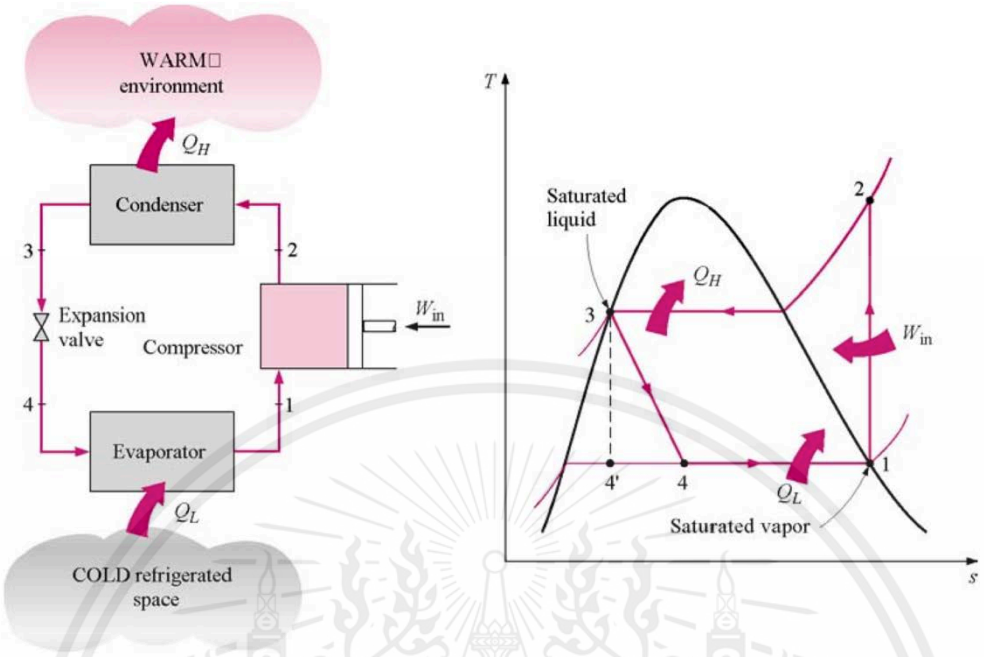

Figure 2.1 Air conditioning cycle

2.2 Component in air conditioning

Air conditioning systems are pivotal in providing indoor comfort by regulating air temperature and humidity. One of the most critical components of modern air conditioning systems is the vapor compression refrigeration cycle. This process is at the heart of most air conditioning units, playing a vital role in cooling and dehumidifying the indoor air efficiently. In this section, we will delve into the mechanics of the vapor compression refrigeration cycle, examining the roles and functions of its key components—compressors, condensers, expansion devices, and evaporators.

2.2.1 Compressor

Compressors in refrigeration systems can vary widely in type, including reciprocating, rotary, and screw compressors. Each type has its specific operational dynamics, but all serve the same purpose: to compress the refrigerant vapor, thereby increasing its pressure and temperature. The efficiency of these compressors is influenced by factors such as clearance volume, mechanical losses, and the thermodynamic properties of the refrigerant. As the compressor does work on the refrigerant vapor, its entropy increases, leading to a higher temperature and pressure. The energy efficiency of the compressor is paramount because it determines the bulk of the system's energy consumption. Techniques such as multi-stage compression, where the refrigerant is compressed in stages, and intercooling, which cools the refrigerant between stages to reduce work input, are used to enhance efficiency.

Figure 2.2 Compressor air

2.2.2 Condenser

The condenser must efficiently reject the heat absorbed from the indoor space and the heat added by the compressor. This is typically achieved through air-cooled or water-cooled mechanisms. Air-cooled condensers use fans to blow air across the condenser coils, while water-cooled types use water flowing through coils or panels that house the refrigerant. The

efficiency of the condenser is heavily dependent on the ambient temperature; higher temperatures can significantly reduce its ability to reject heat. This is a critical design consideration in hot climates where the condenser load is considerably higher.

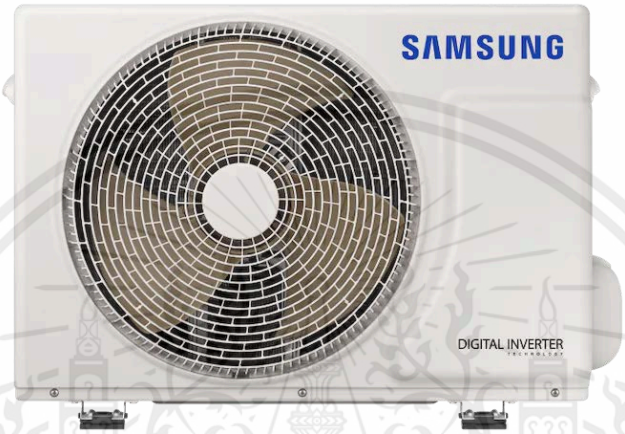

Figure 2.3 Condenser air

2.2.3 Expansion Device (Throttling Valve)

The expansion device regulates the refrigerant flow into the evaporator while maintaining the high pressure in the condenser. Common types of expansion devices include capillary tubes, which are fixed orifice devices, and thermostatic expansion valves, which adjust the flow based on the evaporator temperature and load. While the expansion device itself does not consume energy, its precision in maintaining the optimal refrigerant flow rate is crucial for the overall efficiency of the cycle. Incorrect sizing or malfunctioning can lead to either excessive or insufficient refrigerant flow, both of which can degrade system performance and efficiency.

Figure 2.4 Throttling Valve

2.2.4 Evaporator

In the evaporator, the refrigerant absorbs heat through phase change. The design of the evaporator must ensure that the refrigerant completely vaporizes before leaving the evaporator to avoid compressor damage and maintain efficiency. The performance of the evaporator is enhanced by increasing the surface area in contact with the air or by optimizing the airflow pattern over the coils. Modern evaporators often use fins attached to the tubes to increase the surface area and turbulent flow designs that enhance the heat transfer coefficient.

Figure 2.5 Evaporator air

2.3 Saturation and Superheat in air conditioning

In the field of air conditioning, the concepts of saturation and superheat are fundamental to understanding how refrigeration cycles achieve effective cooling. This chapter focuses on these two critical thermal states, which play essential roles in the operation and efficiency of air conditioning systems. Saturation occurs when a refrigerant is at a temperature and pressure where it exists simultaneously as both a liquid and a vapor, which is crucial during the heat absorption and rejection phases of the cycle. Superheat, on the other hand, refers to the condition where the refrigerant vapor's temperature is raised above its saturation temperature, ensuring that no liquid enters the compressor, thus preventing potential damage and efficiency losses.

2.3.1 Saturation

In refrigeration cycles, the saturation points define the conditions at which refrigerants change phase in the evaporator and condenser. Understanding and accurately predicting these points is crucial for efficient cycle design. Saturation tables or diagrams are used to find the properties of the refrigerant at these points.

Saturation Tables: List the properties of a refrigerant, including temperature and pressure, at its saturation point for various pressures or temperatures. These tables are essential for determining the necessary conditions for phase changes in the evaporator and condenser.

Temp °C	Pressure [kPa]	Volume [m^3/kg]		Density [kg/m^3]		Enthalpy [kJ/kg]			Entropy [kJ/K-kg]		Temp °C
		Liquid v_f	Vapour v_g	Liquid d_f	Vapour d_g	Liquid H_f	Latent H_{fg}	Vapour H_g	Liquid S_f	Vapour S_g	
8	640.9	0.0008	0.0368	1254.0	27.150	209.5	198.4	407.9	1.034	1.739	8
9	660.7	0.0008	0.0358	1250.0	27.970	210.7	197.5	408.2	1.038	1.738	9
10	680.9	0.0008	0.0347	1247.0	28.820	211.9	196.7	408.6	1.042	1.737	10
11	701.7	0.0008	0.0337	1243.0	29.690	213.1	195.8	408.9	1.046	1.735	11
12	722.9	0.0008	0.0327	1239.0	30.570	214.3	194.9	409.2	1.051	1.734	12
13	744.5	0.0008	0.0318	1236.0	31.480	215.5	194.0	409.5	1.055	1.733	13
14	766.7	0.0008	0.0309	1232.0	32.410	216.7	193.2	409.9	1.059	1.732	14
15	789.3	0.0008	0.0300	1229.0	33.360	217.9	192.3	410.2	1.063	1.730	15
16	812.4	0.0008	0.0291	1225.0	34.340	219.1	191.4	410.5	1.067	1.729	16
17	836.1	0.0008	0.0283	1221.0	35.340	220.4	190.4	410.8	1.071	1.728	17
18	860.2	0.0008	0.0275	1217.0	36.360	221.6	189.5	411.1	1.076	1.726	18
19	884.8	0.0008	0.0267	1214.0	37.410	222.8	188.6	411.4	1.080	1.725	19
20	910.0	0.0008	0.0260	1210.0	38.480	224.1	187.6	411.7	1.084	1.724	20
21	935.7	0.0008	0.0253	1206.0	39.570	225.3	186.6	411.9	1.088	1.722	21
22	961.9	0.0008	0.0246	1202.0	40.700	226.5	185.7	412.2	1.092	1.721	22
23	988.7	0.0008	0.0239	1198.0	41.850	227.8	184.7	412.5	1.096	1.720	23
24	1016.0	0.0008	0.0232	1195.0	43.030	229.0	183.8	412.8	1.100	1.719	24
25	1044.0	0.0008	0.0226	1191.0	44.230	230.3	182.7	413.0	1.105	1.717	25
26	1072.0	0.0008	0.0220	1187.0	45.470	231.5	181.8	413.3	1.109	1.716	26
27	1101.0	0.0009	0.0214	1183.0	46.730	232.8	180.7	413.5	1.113	1.715	27
28	1131.0	0.0009	0.0208	1179.0	48.020	234.1	179.7	413.8	1.117	1.714	28
29	1161.0	0.0009	0.0203	1175.0	49.350	235.3	178.7	414.0	1.121	1.712	29
30	1192.0	0.0009	0.0197	1171.0	50.700	236.6	177.7	414.3	1.125	1.711	30
31	1223.0	0.0009	0.0192	1167.0	52.090	237.9	176.6	414.5	1.129	1.710	31
32	1255.0	0.0009	0.0187	1163.0	53.520	239.2	175.5	414.7	1.133	1.709	32
33	1288.0	0.0009	0.0182	1158.0	54.970	240.5	174.4	414.9	1.138	1.707	33
34	1321.0	0.0009	0.0177	1154.0	56.460	241.8	173.3	415.1	1.142	1.706	34
35	1355.0	0.0009	0.0172	1150.0	57.990	243.1	172.2	415.3	1.146	1.705	35
36	1389.0	0.0009	0.0168	1146.0	59.550	244.4	171.1	415.5	1.150	1.704	36
37	1424.0	0.0009	0.0164	1142.0	61.150	245.7	170.0	415.7	1.154	1.702	37

Figure 2.6 R-22 Saturation table

2.3.2 Superheat

Superheating ensures that all the refrigerant is vaporized before it returns to the compressor, preventing any liquid from entering and potentially damaging the compressor. The degree of superheat is a key performance indicator in refrigeration systems, influencing both efficiency and safety.

Superheat Tables: Provide information on the properties of the refrigerant vapor well above its boiling point at various pressures. These tables are crucial for ensuring that the refrigerant entering the compressor is in the superheated state, which is necessary to prevent compressor damage.

Temp °C	1400 (36.31°C)			1500 (39.10°C)		
	V	H	S	V	H	S
	(0.0167)	(415.6)	(1.703)	(0.0155)	(416.1)	(1.700)
40	0.0171	419.1	1.714	0.0156	417.0	1.702
45	0.0177	423.7	1.729	0.0162	421.8	1.718
50	0.0183	428.2	1.743	0.0167	426.4	1.732
55	0.0188	432.5	1.756	0.0172	430.9	1.746
60	0.0193	436.9	1.769	0.0177	435.4	1.759
65	0.0198	441.1	1.782	0.0182	439.7	1.772
70	0.0203	445.3	1.795	0.0187	444.0	1.785
75	0.0208	449.5	1.807	0.0192	448.3	1.797
80	0.0213	453.7	1.819	0.0196	452.5	1.809
85	0.0217	457.8	1.830	0.0201	456.7	1.821
90	0.0222	462.0	1.842	0.0205	460.9	1.833
95	0.0226	466.1	1.853	0.0209	465.1	1.844
100	0.0231	470.2	1.864	0.0213	469.2	1.855
105	0.0235	474.3	1.875	0.0218	473.4	1.866
110	0.0239	478.4	1.886	0.0222	477.5	1.877
115	0.0244	482.6	1.896	0.0226	481.7	1.888
120	0.0248	486.7	1.907	0.0230	485.8	1.899
125	0.0252	490.8	1.917	0.0234	490.0	1.909
130	0.0256	494.9	1.928	0.0238	494.1	1.920
135	0.0260	499.1	1.938	0.0242	498.3	1.930
140	0.0265	503.2	1.948	0.0246	502.5	1.940
145	0.0269	507.4	1.958	0.0249	506.7	1.950
150	0.0273	511.5	1.968	0.0253	510.9	1.960
155	0.0277	515.7	1.978	0.0257	515.1	1.970
160	0.0281	519.9	1.987	0.0261	519.3	1.980
165	0.0285	524.1	1.997	0.0264	523.5	1.989
170	0.0289	528.3	2.007	0.0268	527.7	1.999
175	0.0292	532.6	2.016	0.0272	532.0	2.009
180	0.0296	536.8	2.026	0.0276	536.2	2.018
185	0.0300	541.1	2.035	0.0279	540.5	2.027
190	0.0304	545.3	2.044	0.0283	544.8	2.037

Figure 2.7 R-22 Superheat table

2.4 Parameter in air conditioning

The parameters of Q_L, Q_H, W_{in}, and COP are fundamental to understanding the efficiency and operation of refrigeration cycles, such as the vapor compression cycle used in air conditioning systems. Let's break down these concepts and explain how they are calculated and applied in the context of refrigeration cycle.

2.4.1 Heat Absorbed by the Refrigerant (Q_L)

Q_L is the heat absorbed by the refrigerant in the evaporator. It can be calculated by the formula:

$$Q_L = H1 - H4$$

Where:

- m is the mass flow rate of the refrigerant.
- h1 is the enthalpy of the refrigerant at the exit of the evaporator (after absorbing heat).
- h4 is the enthalpy of the refrigerant at the exit of the expansion device (before it enters the evaporator).

2.4.2 Heat Rejected (Q_H)

Q_H is the heat rejected by the refrigerant in the condenser. It can be calculated by the formula:

$$Q_H = H2 - H3$$

Where:

- h2 is the enthalpy of the refrigerant at the exit of the compressor (before it enters the condenser).
- h3 is the enthalpy of the refrigerant at the exit of the condenser (after releasing heat).

2.4.3 Work Input (W_{in})

W_{in} is the work input to the compressor, which is the energy required to compress the refrigerant from its evaporator pressure to its condenser pressure. It can be calculated by the formula:

$$W_{in} = H2 - H1$$

Where:

- h1 is the enthalpy of the refrigerant at the entrance of the compressor (after absorbing heat in the evaporator).
- h2 is the enthalpy of the refrigerant at the exit of the compressor (before it releases heat in the condenser).

2.4.4 Coefficient of Performance (COP)

The COP is defined as the ratio of the heat removed from the cold reservoir (or the cooling effect, Q_L) to the work input to the system (W_{in}). The formula is:

$$COP = \frac{Q_L}{W_{in}}$$

Where:

- Q_L is the amount of heat absorbed by the refrigerant in the evaporator (as explained in my previous response).
- W_{in} is the work done by the compressor to increase the pressure of the refrigerant, thereby moving it through the cycle.

Chapter 3
Methodology

The methodology is structured into several key phases: data collection, analysis, and validation. Initially, we engage in the meticulous compilation of thermodynamic data, which includes the specific enthalpy and entropy of the refrigerant at various stages of the refrigeration cycle. This data serves as the foundational bedrock upon which the subsequent analysis is constructed.

3.1 Flowchart Creation
3.2 Pseudocode Development
3.3 Write actual code from pseudocode
3.3.1 Import Required Libraries
3.3.2 Create Interpolate Functions
3.3.3 Calculate Evaporator Enthalpy (H1) and Entropy (S1) using Interpolated Function
3.3.4 Calculate Condenser Enthalpy (H3) and Entropy (S3) using Interpolated Function
3.3.4.1 User Input for Pressure (P3)
3.3.4.2 Interpolation Calculations (H3 and S3):
3.3.4.3 Variable Assignment (P2):
3.3.4.4 Outputting and Formatting Interpolated Values in Console Applications

3.3.5 Create R22 Table (Superheated)

3.3.6 Superheat vapor Calculate Compressor Enthalpy (H2) and Entropy (S2) by Compressor Pressure (P2)

3.3.6.1 Conditional Check for P2 Equal to 160

3.3.6.2 Conditional Check for P2 Between 160 and 170

3.3.7 Calculate parameter

3.3.7.1 Heat Removed (QL)

3.3.7.2 Work Input (Win)

3.3.7.3 Heat Added (QH)

3.3.7.4 Coefficient of Performance (COP)

3.4 Experimental Setup

3.5 Methodology for Varying Evaporator Temperature

3.6 Methodology for Varying Condenser Temperature

3.1 Flowchart Creation

A flowchart is a visual representation of the sequential steps involved in your project, starting from data collection to final analysis. It serves as a blueprint that outlines the entire process in a clear and concise manner. Creating a flowchart involves identifying all the key processes and decisions involved in your project, such as initializing the sensors, collecting data, converting the data from analog to digital, transmitting the data to a computer, and analyzing the data using Python. Each of these steps is represented by specific symbols (e.g., rectangles for processes, diamonds for decision points) and connected by arrows that indicate the flow of operations.

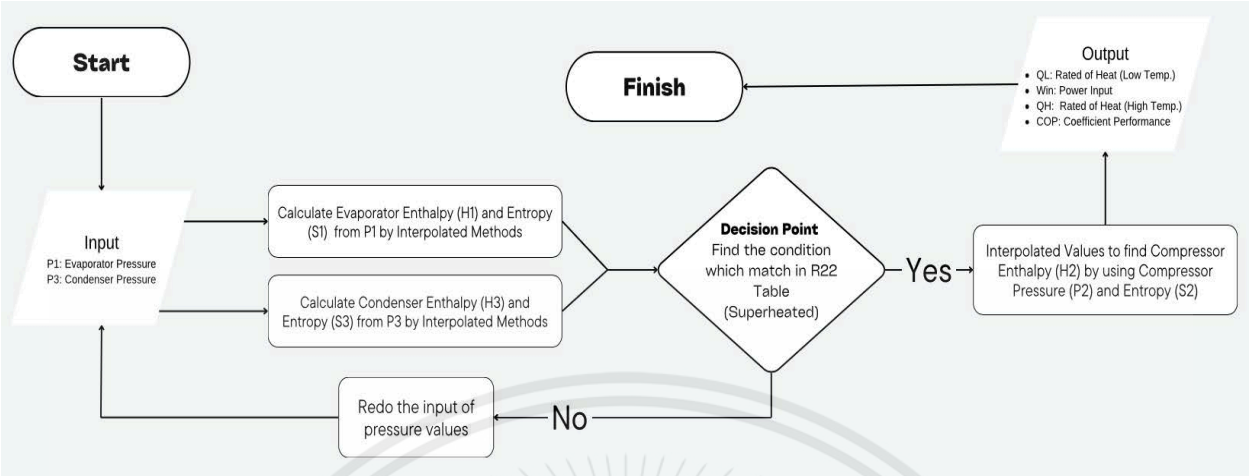

Figure 3.1 Flowchart coding

3.2 Pseudocode Development

Pseudocode is a simplified, informal way of describing your program's logic without using the syntax of a specific programming language. It helps you to plan the structure of your code, making it easier to translate into actual programming languages later. For your project, the pseudocode would include steps for initializing the pressure sensors and Arduino, reading pressure values, converting these values from analog to digital, sending the data to the Python environment, and then using Python to process this data and calculate the AC performance metrics (Q_L, W_{in}, Q_H), **COP**).

3.3 Write actual code from pseudocode

3.3.1 Import Required Libraries

```
import numpy as np
```

Figure 3.2 Import library

1. import numpy: This tells Python to import the NumPy library. NumPy is a popular library for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
2. as np: This part of the statement creates an alias for numpy so that you can use np instead of numpy when referring to the library in your code. This is done simply for convenience and to save typing; it's a widely adopted convention when using NumPy.

3.3.2 Create Interpolate Functions

```
def interpolate(x, x_points, y_points):
    for i in range(len(x_points)):
        if x_points[i] > x:
            if x_points[i - 1] == x:
                return y_points[i - 1]
            else:
                    return y_points[i - 1] + (x - x_points[i - 1]) *
((y_points[i] - y_points[i - 1])/(x_points[i] - x_points[i - 1]))
    return None
```

Figure 3.3 Interpolate function

The screenshot shows a Python function called interpolate. This function is designed to find an interpolated value for a given x within a set of x and y points (assuming x_points and y_points are lists of numbers that represent a discrete set of points on a curve). Here's a step-by-step explanation:

1. The function takes three arguments:
    - x: the x-coordinate for which you want to interpolate a y-value.
    - x_points: a list of x-coordinates from a dataset.
    - y_points: a list of y-coordinates corresponding to x_points.

2. It loops through each element in x_points and checks if the current element is greater than x. This is used to find the interval that x falls into.

3. Once the correct interval is found, the function checks if x is exactly equal to one of the x-coordinates in x_points. If it is, the corresponding y-coordinate from y_points is returned.

4. If x is not exactly equal to any of the x-coordinates, the function performs linear interpolation. It calculates the slope between the two points that bracket x and uses this slope to estimate the y-value at x.
5. If no suitable interval is found, the function returns None, indicating that x is out of the range provided by x_points.

3.3.3 Calculate Evaporator Enthalpy (H1) and Entropy (S1) using Interpolated Function

```
P1 = float(input("Enter Pressure I: "))
H1 = interpolate(P1, P, HG)
S1 = interpolate(P1, P, SG)
S2 = S1
print("H1 values = ",round(H1,3))
print("S1 values = ",round(S1,3),"\n")
```

Figure 3.4 Input Evaporator Pressure (P1) in order to calculate H1 and S1

The screenshot shows a script where the interpolate function is used. Here's what the script does:

1. It prompts the user to input a value for Pressure I which it stores in P1 as a floating-point number.

2. It calls the interpolate function twice, using P1 as the x value to interpolate. The P, HG, and SG are presumably lists that are not shown in the screenshot but represent x and y points for two different datasets. The results are stored in H1 and S1.

3. The value of S1 is copied to S2.

4. It prints the interpolated values H1 and S1, rounding them to three decimal places.

3.3.4 Calculate Condenser Enthalpy (H3) and Entropy (S3) using Interpolated Function

```
P3 = float(input("Enter Pressure III: "))
H3 = interpolate(P3, P, HF)
S3 = interpolate(P3, P, SG)
P2 = P3
H4 = H3 #* Throtting Valves
print("H3 values = ",round(H3,3))
print("S3 values = ",round(S3,3),"\n")
```

Figure 3.5 Input Evaporator Pressure (P3) in order to calculate H3 and S3

The screenshot shows a script where the interpolate function is used. Here's what the script does:

1. It prompts the user to input a value for Pressure III which it stores in P3 as a floating-point number.

2. It calls the interpolate function twice, using P3 as the x value to interpolate. The P, HG, and SG are presumably lists that are not shown in the screenshot but represent x and y points for two different datasets. The results are stored in H3 and S3.

3. It prints the interpolated values H3 and S3, rounding them to three decimal places.

3.3.4.1 User Input for Pressure (P3)

```
P3 = float(input("Enter Pressure III: "))
```

Figure 3.6 Input pressure condenser

This line creates an interactive prompt for the user, asking them to input a value that represents a pressure (Pressure III). Once the user inputs this value, it is converted from a string to a floating-point number (which can include decimal values) and is stored in the variable P3.

3.3.4.2 Interpolation Calculations (H3 and S3):

```
H3 = interpolate(P3, P, HF)
S3 = interpolate(P3, P, SG)
```

Figure 3.7 Set interpolate P3

Here, two separate interpolations are calculated using the value entered by the user (P3):

- H3 is obtained by interpolating between known data points in P and HF. P could be a list of pressure values, and HF a list of corresponding values for a certain property (e.g., enthalpy, height, etc.).

- S3 is calculated in the same way but uses the SG dataset instead. The SG dataset could represent another property at the given pressures (e.g., specific gravity).

The interpolate function must be previously defined and should follow the logic of finding the position of P3 within the P list, then using that position to interpolate the corresponding value from HF or SG.

3.3.4.3 Variable Assignment (P2):

```
P2 = P3
```

Figure 3.8 From T-S diagram P3 equal to P2

This is a straightforward assignment operation where the value of P3 is assigned to another variable P2 form principal AC cycle.

3.3.4.4 Outputting and Formatting Interpolated Values in Console Applications

```
print("H3 values = ",round(H3,3))
print("S3 values = ",round(S3,3),"\n")
```

Figure 3.9 Output interpolate

These lines of code output the interpolated values (H3 and S3) to the console for the user to see. The round function is used to limit the number of decimal places to three, likely for readability or to match significant figures based on the precision of the input data. The "\n" is a newline character that adds an empty line after the output of S3, separating the output visually for clarity.

3.3.5 Create R22 Table (Superheated)

```
Hsup160 = [392.8, 396.0, 399.1, 402.3, 405.5, 408.8, 412.0, 415.3,
418.5, 421.8, 425.1, 428.5, 431.8, 435.2, 438.6, 442.0, 445.5]
Ssup160 = [1.804, 1.817, 1.830, 1.842, 1.855, 1.867, 1.879, 1.890,
1.902, 1.914, 1.925, 1.936, 1.947, 1.959, 1.970, 1.980, 1.991]
Hsup170 = [0, 395.7, 398.9, 402.1, 405.3, 408.6, 411.8, 415.1, 418.4,
421.7, 425.0, 428.3, 431.7, 435.1, 438.5, 441.9, 445.4]
Ssup170 = [0, 1.810, 1.823, 1.836, 1.848, 1.860, 1.872, 1.884, 1.896,
1.907, 1.919, 1.930, 1.941, 1.952, 1.963, 1.974, 1.985]
Tsup160 = [-30, -25, -20, -15, -10, -5, 0, 5, 10, 15, 20, 25, 30, 35,
40, 45, 50]
Tsup170 = Tsup160
```

Figure 3.10 Period at 400 to 2100 kpa superheat value

Creating a table for R22 refrigerant in a superheated state with Enthalpy (H) and entropy (S) are properties used in thermodynamics to describe the state of a system, often used in the study of refrigeration cycles. The numbers are likely values of these properties at various temperatures, given in degrees Celsius for temperature, kJ/kg for enthalpy, and kJ/kg·K for entropy.

3.3.6 Superheat vapor Calculate Compressor Enthalpy (H2) and Entropy (S2) by Compressor Pressure (P2)

```
if P2 == 160:
    H2 = interpolate(S2,Ssup160,Hsup160)
    print(round(H2,3))
elif 160<P2<170:
    H2Down = interpolate(S2,Ssup160,Hsup160)
    H2Up = interpolate(S2,Ssup170,Hsup170)
    H2 = H2Down + (((H2Up-H2Down)*(P2-160))/(170-160))
    print("H2 Values:",round(H2,3))
```

Figure 3.11 Function interpolate in superheat

3.3.6.1 Conditional Check for P2 Equal to 160

```
if P2 == 160:
    H2 = interpolate(S2, Ssup160, Hsup160)
    print(round(H2,3))
```

Figure 3.12 Set the first range value

If P2 is exactly equal to 160, the code uses the interpolate function to calculate H2 by interpolating at the specific value of 160. It's assumed that Ssup160 and Hsup160 are data sets or lists that correspond to values at pressure 160. The resulting H2 value is then printed to the console, rounded to three decimal places.

3.3.6.2 Conditional Check for P2 Between 160 and 170

```
elif 160 < P2 < 170:
    H2Down = interpolate(S2, Ssup160, Hsup160)
    H2Up = interpolate(S2, Ssup170, Hsup170)
    H2 = H2Down + ((H2Up-H2Down) * (P2-160)) / (170-160))
    print("H2 Values:", round(H2,3))
```

Figure 3.13 Write the condition interpolate

If P2 falls between 160 and 170 (exclusive), the code performs two interpolations:

- H2Down is calculated by interpolating the value at the lower boundary of the interval (160).
- H2Up is calculated by interpolating the value at the upper boundary of the interval (170).

Then, H2 is calculated by taking the value at the lower boundary and adding the fraction of the interval (P2 - 160) to it. The fraction is scaled by the difference between the interpolated values at the upper and lower boundaries (H2Up - H2Down), divided by the total interval (170 - 160). This is essentially performing a linear interpolation for a value of P2 that lies between two known points. Finally, the interpolated value H2 is printed, rounded to three decimal places.

3.3.7 Calculate parameter

```
QL = (H1-H4)
print("QL Values:",round(QL,3))
Win = (H2-H1)
print("Win Values:",round(Win,3))
QH = (H2-H3)
print("QH Values:",round(QH,3))
COP = (QL/Win)
print("COP Values:",round(COP,3))
```

figure 3.14 Step to calculate each parameter

Calculating various thermodynamic parameters in a refrigeration cycle. Each of these parameters represents a specific energy value or efficiency metric in the system.

3.3.7.1 Heat Removed (QL)

```
QL = (H1-H4)
print("QL Values:", round(QL, 3))
```

Figure 3.15 Q_L formula

In thermodynamics, H1 and H4 represent the enthalpy values at different points of a cycle. The calculation H1-H4 suggests this is a measure of the heat extracted from a colder environment or substance within the cycle (e.g., the heat extracted from the inside of a refrigerator). The value QL is printed with three decimal places, indicating precision in measurement or calculation.

3.3.7.2 Work Input (Win)

```
Win = (H2-H1)
print("Win Values:", round(Win, 3))
```

Figure 3.16 W_{in} formula

This calculates the work input (Win) required for the process. It is the difference in enthalpy between H2 and H1. This is typical in cycles where work is done on the system (like compression in refrigeration cycles). The result is printed, rounded to three decimal places.

3.3.7.3 Heat Added (QH)

```
QH = (H2-H3)
print("QH Values:", round(QH, 3))
```

Figure 3.17 Q_H formula

The term QH is typically associated with the heat discharged at the higher temperature side of a cycle, for example, the heat expelled to the outside environment by a refrigerator or the heat released by a heat pump into a building. The difference H2-H3 implies this is the part of the cycle where the system releases heat as the enthalpy decreases from H2 to H3.

3.3.7.4 Coefficient of Performance (COP)

```
COP = (QL/Win)
print("COP Values:", round(COP, 3))
```

Figure 3.18 COP formula

The coefficient of performance (COP) is a dimensionless ratio used to express the efficiency of a heating or cooling system. It is calculated by dividing the amount of heat removed (QL) by the work input (Win). In the context of a refrigerator or air conditioner, a higher COP means for each unit of energy put into the system, more cooling is achieved. The reverse is true for a heat pump

where a higher COP means more heating is achieved for each unit of energy. The COP gives an immediate sense of the effectiveness of the system in energy terms: the higher the COP, the less energy is consumed to achieve the same amount of heating or cooling.

3.4 Experimental Setup

The experiment was designed to assess the impact of evaporator and condenser temperatures on the performance of a refrigeration cycle. The setup included:

- Controlled Temperature Variation: The ability to precisely control and vary the temperature at the evaporator and condenser.
- Instrumentation: Use of calibrated sensors to measure temperatures and pressures at key points in the refrigeration cycle.
- Data Acquisition: Implementation of a system to record the sensor data, which provided inputs for subsequent COP and Win calculations.

3.5 Methodology for Varying Evaporator Temperature

- Fixed Condenser Temperature: The temperature at the condenser was maintained at a constant 40°C to isolate the effect of the evaporator temperature on system performance.
- Variable Evaporator Temperature: The temperature at the evaporator was incrementally increased from 0°C to 10°C.
- Data Recording: At each increment, the system's COP and Win were measured and recorded after reaching steady state.
- Analysis: These data points allowed for the observation of a trend where the COP improved with an increase in evaporator temperature, which signifies better system efficiency at higher evaporator temperatures.

T (°C)	COP	Win
0	5.521	28.146
1	5.758	27.06
2	5.935	26.319
3	6.176	25.339
4	6.374	24.615
5	6.552	23.991
6	6.869	22.944
7	7.074	22.32
8	7.442	21.272
9	7.681	20.648
10	7.98	19.924

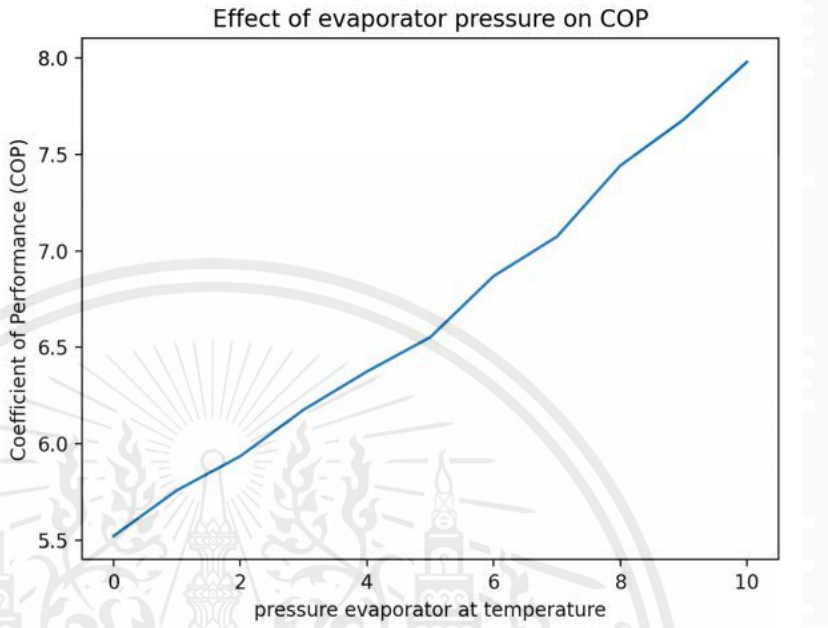

Figure 3.19 COP - Temperature (Evaporator) Graph

3.6 Methodology for Varying Condenser Temperature

- Fixed Evaporator Temperature: The evaporator temperature was held steady at 0°C to understand the effect of condenser temperature on the cycle.
- Variable Condenser Temperature: The temperature at the condenser was increased from 40°C to 50°C.
- Measurement: For each temperature setting, the COP and Win values were noted once the system stabilized.
- Interpretation: The recorded data illustrated a decrease in COP as the condenser temperature was raised, suggesting lower efficiency due to the increased energy required to reject heat at higher temperatures.

T (°C)	COP	Win
40	5.521	28.146
41	5.365	28.707
42	5.209	29.313
43	5.049	29.964
44	4.897	30.631
45	4.755	31.252
46	4.622	31.87
47	4.492	32.483
48	4.372	33.055
49	4.254	33.641
50	4.134	34.3

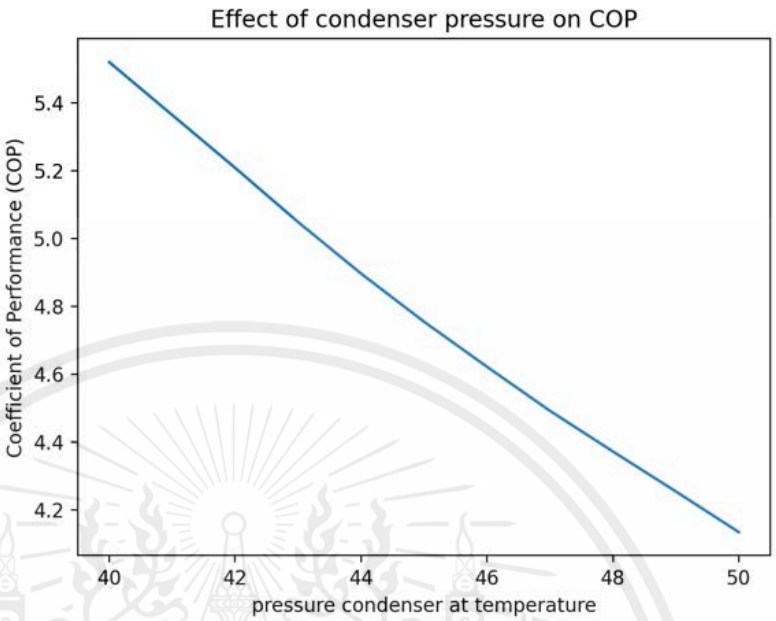

Figure 3.20 COP – Temperature (Condenser) Graph

Chapter 4
Experimental Result

This section presents an in-depth analysis of the experimental results concerning the Coefficient of Performance (COP) of a cooling system when tested under varied inside and outside environmental conditions. The COP, a critical metric in evaluating the efficiency of air conditioning and refrigeration systems, indicates the ratio of cooling or heating provided to the energy consumed.

4.1 Analysis for Outside Environment Temperature at 29 degrees Celsius

4.2 Analysis for Inside Room Temperature at 23 degrees Celsius

4.3 Comparative Analysis and Thermodynamic Implications

4.1 Analysis for Outside Environment Temperature at 29 degrees celsius

- Operation Conditions: The system was tested in a steady-state operation at an outside ambient temperature and the same room temperature of 29 degrees Celsius. This similarity in temperature suggests that the system was probably not subjected to significant thermal gradients that could influence the heat transfer processes excessively.

- System Pressures: The condenser pressure was recorded at 180 psi (1241 kpa) when the condenser temperature was at 31 degrees Celsius. The evaporator was at a pressure of 50 psi (344 kpa) and a temperature of -10 degrees Celsius. These pressure readings are likely to correspond to the saturation pressures of R-22 at the given temperatures, indicating where the refrigerant is changing phases.

- COP Value: A COP of 5.07 is quite efficient, indicating that for every unit of work the system consumes, it is transferring over five units of heat from the cooler area. This efficiency in heat transfer performance is likely due to effective system design and component operation.

Figure 4.1 Pressure gauge evaporator and condenser at outside

4.2 Analysis for Inside Room Temperature at 23 degrees celsius

- Operation Conditions: Under these conditions, the system was operated at a uniform ambient and room temperature of 23 degrees Celsius, which is cooler than the outside temperature scenario.

- System Pressures: The condenser pressure here was 150 psi (1034 kpa) at 26 degrees Celsius, while the evaporator pressure was slightly higher at 60 psi (413 kpa) at -5

degrees Celsius compared to the outside temperature scenario. This suggests a lower temperature differential between the condenser and the ambient, potentially allowing for more efficient heat rejection.

- COP Value: The higher COP of 7.689 indicates a very high level of efficiency, which suggests that the refrigeration cycle is operating closer to its optimal performance conditions. The reduced ambient temperature leads to less energy expenditure for the same amount of refrigeration, as the lower ambient temperature assists in more efficient condenser operation.

Figure 4.2 Pressure gauge evaporator and condenser at inside

4.3 Comparative Analysis and Thermodynamic Implications

- Impact of Ambient Conditions: The increased COP at the lower ambient room temperature strongly implies that refrigeration systems perform more efficiently in cooler environments. This is because the condenser can reject heat more readily when the

outside temperature is lower, requiring less work input to achieve the desired refrigeration effect.

- System Design Considerations: These results underscore the importance of considering ambient temperature in the design and placement of refrigeration systems. Systems in hotter climates may need to be designed with greater capacity or enhanced heat rejection capabilities to achieve similar efficiencies.

- Refrigerant Behavior: R-22 is known to be an effective refrigerant; however, its performance will vary based on the operating pressures and temperatures. The experimental data provides insight into the behavior of R-22 under the specific tested conditions, which could be valuable for designing systems that use this or similar refrigerants.

Chapter 5
Conclusion

The objective of our study was to analyze the performance of a vapor compression refrigeration cycle with respect to varying evaporator and condenser temperatures, using R-22 as the refrigerant. The experimental data were gathered through a series of controlled tests, where temperatures at key components were systematically varied, and the corresponding Coefficient of Performance (COP) and Work Input (Win) to the compressor were calculated.

Our findings reveal a clear dependency of the refrigeration cycle's efficiency on the thermal conditions of the evaporator and condenser. Specifically, when the evaporator temperature increased from 0°C to 10°C with the condenser temperature held steady at 40°C, the COP of the system improved significantly from 5.521 to 7.98. Concurrently, the work input to the compressor decreased, suggesting that the refrigeration system required less energy to achieve the desired cooling effect at higher evaporator temperatures.

Conversely, increasing the condenser temperature from 40°C to 50°C while maintaining the evaporator temperature at 0°C led to a decrease in COP from 5.521 to 4.134. This trend was indicative of reduced system efficiency, as higher condenser temperatures necessitated greater compressor work input, thus escalating energy consumption and operating costs.

These results underscore the critical role of ambient conditions in the operational efficiency of refrigeration systems. They confirm the thermodynamic principle that lower condenser

temperatures facilitate heat rejection, thereby enhancing the cycle's efficiency, while higher temperatures impede it.

In conclusion, our research contributes valuable insights into the efficient design and operation of vapor compression refrigeration systems. It highlights the necessity for strategic temperature management and offers a basis for further innovation.

APPENDIX

```
import numpy as np
import matplotlib.pyplot as plt

#TODO: Create Interpolate Functions
def interpolate(x, x_points, y_points):
    for i in range(len(x_points)):
        if x_points[i] > x:
            if x_points[i - 1] == x:
                return y_points[i - 1]
            else:
                return y_points[i - 1] + (x - x_points[i - 1]) * ((y_points[i] - y_points[i - 1]) / (x_points[i] - x_points[i - 1]))
    return None

#! Table Data Saturated R22
T1 = [-100, -99, -98, -97, -96, -95, -94, -93, -92, -91, -90, -89, -88, -87, -86, -85, -84, -83, -82, -81, -80, -79, -78, -77,
-76, -75, -74, -73, -72, -71, -70, -69, -68, -67, -66, -65, -64, -63, -62, -61, -60, -59, -58, -57, -56, -55, -54, -53, -52, -51,
-50, -49, -48, -47, -46, -45, -44, -43, -42, -41, -40, -39, -38, -37, -36, -35, -34, -33, -32, -31, -30, -29, -28, -27, -26, -25,
-24, -23, -22, -21, -20, -19, -18, -17, -16, -15, -14, -13, -12, -11, -10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7,
8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70,
71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95]
HG = [359.0, 359.5, 359.9, 360.4, 360.9, 361.4, 361.9, 362.4, 362.9, 363.4, 363.9, 364.3, 364.8, 365.3, 365.8, 366.3,
366.8, 367.3, 367.8, 368.3, 368.8, 369.3, 369.8, 370.3, 370.7, 371.2, 371.7, 372.2, 372.7, 373.2, 373.7, 374.2, 374.7,
375.2 ,375.7, 376.2, 376.6, 377.1, 377.6, 378.1, 378.6, 379.1, 379.6, 380.0, 380.5, 381.0, 381.5, 382.0, 382.5, 382.9,
383.4, 383.9, 384.4, 384.8,385.3, 385.8, 386.3, 386.7, 387.2, 387.7, 388.1, 388.6, 389.1, 389.5, 390.0, 390.4, 390.9,
391.3, 391.8, 392.2, 392.7, 393.1, 393.6, 394.0, 394.5, 394.9, 395.3, 395.8, 396.2, 396.6, 397.1, 397.5, 397.9, 398.3,
398.7, 399.2, 399.6, 400.0, 400.4, 400.8, 401.2, 401.6, 402.0, 402.4, 402.8, 403.2, 403.5, 403.9, 404.3, 404.7, 405.0,
405.4, 405.8, 406.1, 406.5, 406.8, 407.2, 407.5, 407.9, 408.2, 408.6, 408.9, 409.2, 409.5, 409.9, 410.2, 410.5, 410.8,
411.1, 411.4, 411.7, 411.9, 412.2, 412.5, 412.8, 413.0, 413.3, 413.5, 413.8, 414.0, 414.3, 414.5, 414.7, 414.9, 415.1,
415.3, 415.5, 415.7, 415.9, 416.1, 416.2, 416.4, 416.6, 416.7, 416.8, 417.0, 417.1, 417.2, 417.3, 417.4, 417.4, 417.5,
417.6, 417.6, 417.6, 417.7, 417.7, 417.7, 417.6, 417.6, 417.5, 417.5,417.4, 417.3, 417.2, 417.1, 416.9, 416.7, 416.5,
416.3, 416.1, 415.8, 415.5, 415.2, 414.9, 414.5, 414.1, 413.6, 413.1, 412.6, 412.0, 411.4, 410.7, 409.9, 409.1, 408.2,
407.2, 406.1, 404.8, 403.4, 401.9, 400.1, 397.9, 395.3, 392.0, 387.3]
```

```
SG = [2.054, 2.048, 2.042, 2.036, 2.031, 2.025, 2.019, 2.014, 2.009, 2.003, 1.998, 1.993, 1.988, 1.983, 1.978, 1.973,
1.969, 1.964, 1.960, 1.955, 1.951, 1.946, 1.942, 1.938, 1.934, 1.930, 1.926, 1.922, 1.918, 1.915, 1.911, 1.907, 1.904,
1.900, 1.897, 1.893, 1.890, 1.887, 1.883, 1.880, 1.877, 1.874, 1.871, 1.868, 1.865, 1.862, 1.859, 1.856, 1.853, 1.851,
1.848, 1.845, 1.843, 1.840, 1.838, 1.835, 1.833, 1.830, 1.828, 1.825, 1.823, 1.821, 1.819, 1.816, 1.814, 1.812, 1.810,
1.808, 1.806, 1.804, 1.802, 1.800, 1.798, 1.796, 1.794, 1.792, 1.790, 1.788, 1.786, 1.784, 1.783, 1.781, 1.779, 1.777,
1.776, 1.774, 1.772, 1.771, 1.769, 1.767, 1.766, 1.764, 1.763, 1.761, 1.760, 1.758, 1.757, 1.755, 1.754, 1.752, 1.751,
1.749, 1.748, 1.746, 1.745, 1.744, 1.742, 1.741, 1.739, 1.738, 1.737, 1.735, 1.734, 1.733, 1.732, 1.730, 1.729, 1.728,
1.726, 1.725, 1.724, 1.722, 1.721, 1.720, 1.719, 1.717, 1.716, 1.715, 1.714, 1.712, 1.711, 1.710, 1.709, 1.707, 1.706,
1.705, 1.704, 1.702, 1.701, 1.700, 1.698, 1.697, 1.696, 1.695, 1.693, 1.692, 1.691, 1.689, 1.688, 1.687, 1.685, 1.684,
1.682, 1.681, 1.680, 1.678, 1.677, 1.675, 1.674, 1.672, 1.670, 1.669, 1.667, 1.666, 1.664, 1.662, 1.660, 1.659, 1.657,
1.655, 1.653, 1.651, 1.649, 1.647, 1.645, 1.642, 1.640, 1.638, 1.635, 1.633, 1.630, 1.627, 1.624, 1.621, 1.618, 1.614,
1.610, 1.606, 1.602, 1.597, 1.592, 1.586, 1.580, 1.572, 1.562, 1.549]
P  = [2, 2.2, 2.4, 2.6, 2.9, 3.2, 3.4, 3.7, 4.1, 4.4, 4.8, 5.2, 5.7, 6.1, 6.6, 7.2, 7.7, 8.3, 9, 9.6, 10.4, 11.1, 12, 12.8, 13.8,
14.7, 15.8, 16.8, 18, 19.2, 20.5, 21.8, 23.2, 24.7, 26.3, 27.9, 29.7, 31.5, 33.4, 35.4, 37.5, 39.7, 42, 44.4, 46.9, 49.6,
52.3, 55.2, 58.2, 61.3, 64.5, 67.9, 71.5, 75.1, 78.9, 82.9, 87.1, 91.3, 95.8, 100.4, 105.2, 110.2, 115.4, 120.7, 126.3,
132.0, 138.0, 144.1, 150.5, 157.1, 163.9, 170.9, 178.2, 185.7, 193.4, 201.4, 209.7, 218.2, 227.0, 236.0, 245.3, 254.9,
264.8, 275.0, 285.4, 296.2, 307.3, 318.7, 330.4, 342.4, 354.8, 367.5, 380.5, 393.9, 407.7, 421.8, 436.3, 451.1, 466.4,
482.0, 498.0, 514.4, 531.2, 548.4, 566.1, 584.1, 602.6, 621.5, 640.9, 660.7, 680.9, 701.7, 722.9, 744.5, 766.7, 789.3,
812.4, 836.1, 860.2, 884.8, 910.0, 935.7, 961.9, 988.7, 1016.0, 1044.0, 1072.0, 1101.0, 1131.0, 1161.0, 1192.0,
1223.0, 1255.0, 1288.0, 1321.0, 1355.0, 1389.0, 1424.0, 1460.0, 1497.0, 1534.0, 1571.0, 1610.0, 1649.0, 1689.0,
1729.0, 1770.0, 1812.0, 1855.0, 1899.0, 1943.0, 1988.0, 2033.0, 2080.0, 2127.0, 2175.0, 2224.0, 2274.0, 2324.0,
2375.0, 2427.0, 2480.0, 2534.0, 2589.0, 2645.0, 2701.0, 2759.0, 2817.0, 2876.0, 2936.0, 2997.0, 3059.0, 3123.0,
3187.0, 3252.0, 3318.0, 3385.0, 3453.0, 3522.0, 3592.0, 3664.0, 3736.0, 3810.0, 3885.0, 3961.0, 4038.0, 4116.0,
4196.0, 4277.0, 4359.0, 4442.0, 4527.0, 4614.0, 4702.0, 4791.0, 4882.0, ]
HF = [90.7, 91.8, 92.8, 93.9, 95.0, 96.0, 97.1, 98.1, 99.2, 100.3, 101.3, 102.4, 103.4, 104.5, 105.6, 106.6, 107.7,
108.7, 109.8, 110.9, 111.9, 113.0, 114.1, 115.1, 116.2, 117.3, 118.3, 119.4, 120.4, 121.5, 122.6, 123.6, 124.7, 125.8,
126.8, 127.9, 129.0, 130.1, 131.1, 132.2, 133.3, 134.3, 135.4, 136.5, 137.6, 138.6, 139.7, 140.8, 141.9, 142.9, 144.0,
145.1, 146.2, 147.3, 148.4, 149.4, 150.5, 151.6, 152.7, 153.8, 154.9, 156.0, 157.1, 158.2, 159.3, 160.4, 161.5, 162.6,
163.7, 164.8, 165.9, 167.0, 168.1, 169.2, 170.3, 171.4, 172.6, 173.7, 174.8, 175.9, 177.0, 178.2, 179.3, 180.4, 181.6,
182.7, 183.8, 185.0, 186.1, 187.3, 188.4, 189.6, 190.7, 191.9, 193.0, 194.2, 195.3, 196.5, 197.7, 198.8, 200.0, 201.2,
202.4, 203.5, 204.7, 205.9, 207.1, 208.3, 209.5, 210.7, 211.9, 213.1, 214.3, 215.5, 216.7, 217.9, 219.1, 220.4, 221.6,
222.8, 224.1, 225.3, 226.5, 227.8, 229.0, 230.3, 231.5, 232.8, 234.1, 235.3, 236.6, 237.9, 239.2, 240.5, 241.8, 243.1,
244.4, 245.7, 247.0, 248.3, 249.6, 251.0, 252.3, 253.7, 255.0, 256.4, 257.7, 259.1, 260.5, 261.9, 263.2, 264.6, 266.0,
267.5, 268.9, 270.3, 271.8, 273.2, 274.7, 276.1, 277.6, 279.1, 280.6, 282.1, 283.6, 285.2, 286.7, 288.3, 289.9, 291.5,
293.1, 294.7, 296.4, 298.0, 299.7, 301.5, 303.2, 305.0, 306.8, 308.6, 310.4, 312.3, 314.3, 316.3, 318.3, 320.4, 322.5,
324.8, 327.1, 329.5, 332.1, 334.8, 337.8, 341.0, 344.8, 349.6]
print(len(T1),len(SG),len(HG),len(P),len(HF))
#TODO: Calculate H1 and S1 from P1
P1 = float(input("Enter Pressure I: "))
H1 = interpolate(P1, P, HG)
S1 = interpolate(P1, P, SG)
S2 = S1
print("H1 values = ",round(H1,3))
print("S1 values = ",round(S1,3),"\n")

#TODO: Calculate H2, H3 and S3 from P3
P3 = float(input("Enter Pressure III: "))
H3 = interpolate(P3, P, HF)
#S3 = interpolate(P3, P, SG)
P2 = P3
H4 = H3 #* Throtting Valves
print("H3 values = ",round(H3,3))
```

```
#print("S3 values = ",round(S3,3),"\n")

#! Create Table Superheated R22
Hsup160 = [392.8, 396.0, 399.1, 402.3, 405.5, 408.8, 412.0, 415.3, 418.5, 421.8, 425.1, 428.5, 431.8, 435.2, 438.6,
442.0, 445.5]
Ssup160 = [1.804, 1.817, 1.830, 1.842, 1.855, 1.867, 1.879, 1.890, 1.902, 1.914, 1.925, 1.936, 1.947, 1.959, 1.970,
1.980, 1.991]
Hsup170 = [0, 395.7, 398.9, 402.1, 405.3, 408.6, 411.8, 415.1, 418.4, 421.7, 425.0, 428.3, 431.7, 435.1, 438.5,
441.9, 445.4]
Ssup170 = [0, 1.810, 1.823, 1.836, 1.848, 1.860, 1.872, 1.884, 1.896, 1.907, 1.919, 1.930, 1.941, 1.952, 1.963,
1.974, 1.985]
Tsup160 = [-30, -25, -20, -15, -10, -5, 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
Tsup170 = Tsup160

Hsup400 = [403.7, 407.2, 410.7, 414.3, 417.8, 421.3, 424.8, 428.4, 431.9, 435.4, 439.0, 442.6, 446.2, 449.8, 453.4,
457.1, 460.7]
Ssup400 = [1.765, 1.778, 1.790, 1.803, 1.815, 1.827, 1.839, 1.851, 1.863, 1.874, 1.885, 1.897, 1.908, 1.919, 1.929,
1.940, 1.951]
Hsup425 = [0, 406.7, 410.2, 413.8, 417.3, 420.9, 424.4, 428.0, 431.5, 435.1, 438.7, 442.3, 445.9, 449.5, 453.1,
456.8, 460.5]
Ssup425 = [0, 1.770, 1.783, 1.796, 1.808, 1.821, 1.833, 1.844, 1.856, 1.868, 1.879, 1.890, 1.901, 1.912, 1.923,
1.934, 1.944]
Tsup425 = [-5, 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75]
Tsup400 = Tsup425

Hsup450 = [0, 406.1, 409.7, 413.3, 416.9, 420.5, 424.0, 427.6, 431.2, 434.8, 438.4, 442.0, 445.6, 449.2, 452.9,
456.5, 460.2]
Ssup450 = [0, 1.763, 1.776, 1.789, 1.802, 1.814, 1.826, 1.838, 1.850, 1.861, 1.873, 1.884, 1.895, 1.906, 1.917,
1.928, 1.938]
Hsup475 = [0, 405.6, 409.2, 412.8, 416.4, 420.0, 423.6, 427.2, 430.8, 434.4, 438.0, 441.6, 445.3, 448.9, 452.6,
456.3, 460.0]
Ssup475 = [0, 1.757, 1.770, 1.783, 1.795, 1.808, 1.820, 1.832, 1.844, 1.855, 1.867, 1.878, 1.889, 1.900, 1.911, 1.922,
1.933]
Tsup475 = [-5, 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75]
Tsup450 = Tsup475

Hsup500 = [408.7, 412.3, 416.0, 419.6, 423.2, 426.8, 430.4, 434.1, 437.7, 441.3, 445.0, 448.6, 452.3, 456.0, 459.7,
463.4, 467.2]
Ssup500 = [1.764, 1.777, 1.789, 1.802, 1.814, 1.826, 1.838, 1.849, 1.861, 1.872, 1.884, 1.895, 1.906, 1.916, 1.927,
1.938, 1.948]
Hsup525 = [408.1, 411.8, 415.5, 419.2, 422.8, 426.4, 430.1, 433.7, 437.4, 441.0, 444.7, 448.3, 452.0, 455.7, 459.4,
463.2, 466.9]
Ssup525 = [1.757, 1.771, 1.783, 1.796, 1.808, 1.820, 1.832, 1.844, 1.855, 1.867, 1.878, 1.889, 1.900, 1.911, 1.922,
1.932, 1.943]
Tsup525 = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85]
Tsup500 = Tsup525

Hsup550 = [407.6, 411.3, 415.0, 418.7, 422.4, 426.0, 429.7, 433.4, 437.0, 440.7, 444.4, 448.0, 451.7, 455.5, 459.2,
462.9, 466.7]
Ssup550 = [1.751, 1.765, 1.778, 1.790, 1.803, 1.815, 1.827, 1.839, 1.850, 1.862, 1.873, 1.884, 1.895, 1.906, 1.917,
1.927, 1.938]
```

```
Hsup575 = [407.1, 410.8, 414.6, 418.3, 422.0, 425.6, 429.3, 433.0, 436.7, 440.4, 444.0, 447.7, 451.5, 455.2, 458.9, 462.7, 466.4]
Ssup575 = [1.746, 1.759, 1.772, 1.785, 1.797, 1.810, 1.822, 1.834, 1.845, 1.857, 1.868, 1.879, 1.890, 1.901, 1.912, 1.923, 1.933]
Tsup575 = [5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85]
Tsup550 = Tsup575

Hsup600 = [410.3, 414.1, 417.8, 421.5, 425.2, 428.9, 432.6, 436.3, 440.0, 443.7, 447.5, 451.2, 454.9, 458.7, 462.4, 466.2, 470.0]
Ssup600 = [1.754, 1.767, 1.780, 1.792, 1.805, 1.817, 1.829, 1.840, 1.852, 1.863, 1.874, 1.886, 1.897, 1.907, 1.918, 1.929, 1.939]
Hsup625 = [409.8, 413.6, 417.4, 421.1, 424.8, 428.6, 432.3, 436.0, 439.7, 443.4, 447.2, 450.9, 454.6, 458.4, 462.2, 465.9, 469.7]
Ssup625 = [1.748, 1.762, 1.775, 1.787, 1.800, 1.812, 1.824, 1.836, 1.847, 1.859, 1.870, 1.881, 1.892, 1.903, 1.914, 1.924, 1.935]
Tsup625 = [10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90]
Tsup600 = Tsup625

Hsup650 = [409.2, 413.1, 416.9, 420.7, 424.4, 428.2, 431.9, 435.6, 439.4, 443.1, 446.8, 450.6, 454.4, 458.1, 461.9, 465.7, 469.5]
Ssup650 = [1.743, 1.757, 1.770, 1.782, 1.795, 1.807, 1.819, 1.831, 1.843, 1.854, 1.865, 1.877, 1.888, 1.899, 1.909, 1.920, 1.931]
Hsup675 = [408.7, 412.6, 416.4, 420.2, 424.0, 427.8, 431.5, 435.3, 439.0, 442.8, 446.5, 450.3, 454.1, 457.9, 461.7, 465.5, 469.3]
Ssup675 = [1.738, 1.752, 1.765, 1.778, 1.790, 1.803, 1.815, 1.827, 1.838, 1.850, 1.861, 1.872, 1.883, 1.894, 1.905, 1.916, 1.926]
Tsup675 = [10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90]
Tsup650 = Tsup675

Hsup700 = [412.1, 415.9, 419.8, 423.6, 427.4, 431.2, 434.9, 438.7, 442.5, 446.2, 450.0, 453.8, 457.6, 461.4, 465.2, 469.0, 472.9]
Ssup700 = [1.747, 1.760, 1.773, 1.786, 1.798, 1.810, 1.822, 1.834, 1.846, 1.857, 1.868, 1.879, 1.890, 1.901, 1.912, 1.923, 1.933]
Hsup725 = [411.5, 415.3, 419.0, 423.2, 427.0, 430.8, 434.6, 438.4, 442.1, 445.9, 449.7, 453.5, 457.3, 461.1, 465.0, 468.8, 472.6]
Ssup725 = [1.742, 1.755, 1.769, 1.781, 1.794, 1.806, 1.818, 1.830, 1.842, 1.853, 1.864, 1.875, 1.886, 1.897, 1.908, 1.919, 1.929]
Tsup725 = [15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95]
Tsup700 = Tsup725

Hsup750 = [411, 415, 418.9, 422.7, 426.6, 430.4, 434.2, 438.0, 441.8, 445.6, 449.4, 453.2, 457.1, 460.9, 464.7, 468.6, 472.4]
Ssup750 = [1.737, 1.751, 1.764, 1.777, 1.790, 1.802, 1.814, 1.826, 1.838, 1.849, 1.860, 1.872, 1.883, 1.893, 1.904, 1.915, 1.925]
Hsup775 = [410.5, 414.5, 418.4, 422.3, 426.2, 430.0, 433.9, 437.7, 441.5, 445.3, 449.1, 453.0, 456.8, 460.6, 464.5, 468.3, 472.2]
Ssup775 = [1.733, 1.747, 1.760, 1.773, 1.785, 1.798, 1.810, 1.822, 1.834, 1.845, 1.857, 1.868, 1.879, 1.890, 1.901, 1.911, 1.922]
Tsup775 = [15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95]
Tsup750 = Tsup775
```

```
Hsup800 = [414.0, 417.9, 421.9, 425.8, 429.6, 433.5, 437.3, 441.2, 445.0, 448.8, 452.7, 456.5, 460.4, 464.2, 468.1, 471.9, 475.8]
Ssup800 = [1.742, 1.756, 1.769, 1.781, 1.794, 1.806, 1.818, 1.830, 1.841, 1.853, 1.864, 1.875, 1.886, 1.897, 1.908, 1.918, 1.929]
Hsup850 = [412.9, 417.0, 421.0, 424.9, 428.9, 432.8, 436.6, 440.5, 444.4, 448.2, 452.1, 456.0, 459.8, 463.7, 467.6, 471.5, 475.4]
Ssup850 = [1.734, 1.747, 1.761, 1.774, 1.786, 1.799, 1.811, 1.823, 1.834, 1.846, 1.857, 1.868, 1.879, 1.890, 1.901, 1.912, 1.922]
Tsup850 = [20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100]
Tsup800 = Tsup850

Hsup900 = [411.9, 416.0, 420.1, 424.1, 428.1, 432.0, 435.9, 439.8, 443.7, 447.6, 451.5, 455.4, 459.3, 463.2, 467.1, 471.0, 474.9]
Ssup900 = [1.725, 1.739, 1.753, 1.766, 1.779, 1.791, 1.804, 1.816, 1.827, 1.839, 1.850, 1.862, 1.873, 1.884, 1.894, 1.905, 1.916]
Hsup950 = [0, 415.0, 419.1, 423.2, 427.3, 431.2, 435.2, 439.2, 443.1, 447.0, 450.9, 454.8, 458.8, 462.7, 466.6, 470.5, 474.5]
Ssup950 = [0, 1.732, 1.745, 1.759, 1.772, 1.784, 1.797, 1.809, 1.821, 1.832, 1.844, 1.855, 1.866, 1.877, 1.888, 1.899, 1.910]
Tsup950 = [20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100]
Tsup900 = Tsup950

Hsup1000 = [414.0, 418.2, 422.3, 426.4, 430.5, 434.5, 438.5, 442.4, 446.4, 450.3, 454.3, 458.2, 462.2, 466.1, 470.1, 474.0, 478.0]
Ssup1000 = [1.724, 1.738, 1.752, 1.765, 1.778, 1.790, 1.802, 1.814, 1.826, 1.838, 1.849, 1.860, 1.871, 1.882, 1.893, 1.904, 1.914]
Hsup1100 = [0, 416.2, 420.5, 424.7, 428.9, 433.0, 437.1, 441.1, 445.1, 449.1, 453.1, 457.1, 461.1, 465.1, 469.1, 473.1, 477.1]
Ssup1100 = [0, 1.724, 1.738, 1.751, 1.765, 1.777, 1.790, 1.802, 1.814, 1.826, 1.838, 1.849, 1.860, 1.871, 1.882, 1.893, 1.904]
Tsup1100 = [25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105]
Tsup1000 = Tsup1100

Hsup1200 = [0, 0, 418.6, 422.9, 427.2, 431.4, 435.6, 439.7, 443.8, 447.9, 452.0, 456.0, 460.0, 464.1, 468.1, 472.1, 476.2]
Ssup1200 = [0, 0, 1.725, 1.739, 1.752, 1.765, 1.778, 1.791, 1.803, 1.815, 1.827, 1.838, 1.850, 1.861, 1.872, 1.883, 1.893]
Hsup1300 = [0, 0, 416.5, 421.1, 425.5, 429.8, 434.1, 438.3, 442.5, 446.6, 450.8, 454.9, 459.0, 463.0, 467.1, 471.2, 475.3]
Ssup1300 = [0, 0, 1.712, 1.726, 1.740, 1.754, 1.767, 1.780, 1.792, 1.804, 1.816, 1.828, 1.840, 1.851, 1.862, 1.873, 1.884]
Tsup1300 = [25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105]
Tsup1200 = Tsup1300

Hsup1400 = [419.1, 423.7, 428.2, 432.5, 436.9, 441.1, 445.3, 449.5, 453.7, 457.8, 462.0, 466.1, 470.2, 474.3, 478.4, 482.6, 486.7]
Ssup1400 = [1.714, 1.729, 1.743, 1.756, 1.769, 1.782, 1.795, 1.807, 1.819, 1.830, 1.842, 1.853, 1.864, 1.875, 1.886, 1.896, 1.907]
Hsup1500 = [417.0, 421.8, 426.4, 430.9, 435.4, 439.7, 444.0, 448.3, 452.5, 456.7, 460.9, 465.1, 469.2, 473.4, 477.5, 481.7, 485.8]
Ssup1500 = [1.702, 1.718, 1.732, 1.746, 1.759, 1.772, 1.785, 1.797, 1.809, 1.821, 1.833, 1.844, 1.855, 1.866, 1.877, 1.888, 1.899]
```

```
Tsup1500 = [40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105, 110, 115, 120]
Tsup1400 = Tsup1500

Hsup1600 = [0, 419.8, 424.6, 429.2, 433.8, 438.3, 442.7, 447.0, 451.3, 455.6, 459.8, 464.0, 468.2, 472.4, 476.6,
480.8, 485.0]
Ssup1600 = [0, 1.706, 1.721, 1.736, 1.750, 1.763, 1.776, 1.788, 1.801, 1.813, 1.824, 1.836, 1.847, 1.858, 1.869,
1.880, 1.891]
Hsup1700 = [0, 417.6, 422.6, 427.5, 432.2, 436.8, 441.2, 445.7, 450.0, 454.4, 458.7, 463.0, 467.2, 471.5, 475.7,
479.9, 484.1]
Ssup1700 = [0, 1.695, 1.711, 1.726, 1.740, 1.754, 1.767, 1.780, 1.792, 1.804, 1.816, 1.828, 1.839, 1.851, 1.862,
1.873, 1.884]
Tsup1700 = [40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105, 110, 115, 120]
Tsup1600 = Tsup1700

Hsup1800 = [420.6, 425.6, 430.5, 435.2, 439.8, 444.3, 448.8, 453.2, 457.5, 461.9, 466.2, 470.5, 474.7, 479.0, 483.3,
487.5, 491.8]
Ssup1800 = [1.700, 1.716, 1.731, 1.745, 1.758, 1.771, 1.784, 1.796, 1.808, 1.820, 1.832, 1.843, 1.854, 1.866, 1.876,
1.887, 1.898]
Hsup1900 = [418.4, 423.7, 428.7, 433.6, 438.3, 442.9, 447.5, 451.9, 456.4, 460.8, 465.1, 469.5, 473.8, 478.1, 482.4,
486.7, 490.9]
Ssup1900 = [1.690, 1.706, 1.721, 1.736, 1.750, 1.763, 1.776, 1.788, 1.801, 1.813, 1.825, 1.836, 1.847, 1.859, 1.870,
1.880, 1.891]
Tsup1900 = [50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105, 110, 115, 120, 125, 130]
Tsup1800 = Tsup1900

Hsup2000 = [0, 421.6, 426.9, 431.9, 436.8, 441.5, 446.1, 450.7, 455.2, 459.6, 464.1, 468.5, 472.8, 477.2, 481.5,
485.8, 490.1]
Ssup2000 = [0, 1.696, 1.712, 1.727, 1.741, 1.755, 1.768, 1.781, 1.793, 1.806, 1.817, 1.829, 1.841, 1.852, 1.863,
1.874, 1.885]
Hsup2100 = [0, 419.4, 424.9, 430.1, 435.1, 440.0, 444.7, 449.4, 454.0, 458.5, 463.0, 467.4, 471.8, 476.2, 480.6,
484.9, 489.3]
Ssup2100 = [0, 1.686, 1.702, 1.718, 1.733, 1.747, 1.760, 1.773, 1.786, 1.798, 1.811, 1.822, 1.834, 1.845, 1.857,
1.868, 1.878]
Tsup2100 = [50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 105, 110, 115, 120, 125, 130]
Tsup2000 = Tsup2100

#print(len(Hsup160),len(Ssup160),len(Ssup170),len(Hsup170),len(Tsup160))

#TODO: Calculate H2 From P2 and S2 Using Superheated R22 Table
if P2 == 160:
    H2 = interpolate(S2,Ssup160,Hsup160)
    print(round(H2,3))
elif 160<P2<170:
    H2Down = interpolate(S2,Ssup160,Hsup160)
    H2Up = interpolate(S2,Ssup170,Hsup170)
    H2 = H2Down + (((H2Up-H2Down)*(P2-160))/(170-160))
    print("H2 Values:",round(H2,3))
elif P2 == 400:
    H2 = interpolate(S2,Ssup400,Hsup400)
    print(round(H2,3))
elif 400<P2<425:
    H2Down = interpolate(S2,Ssup400,Hsup400)
```

```
    H2Up = interpolate(S2,Ssup425,Hsup425)
    H2 = H2Down + (((H2Up-H2Down)*(P2-400))/(425-400))
    print("H2 Values:",round(H2,3))
elif P2 == 425:
    H2 = interpolate(S2,Ssup425,Hsup425)
    print(round(H2,3))
elif 425<P2<450:
    H2Down = interpolate(S2,Ssup425,Hsup425)
    H2Up = interpolate(S2,Ssup450,Hsup450)
    H2 = H2Down + (((H2Up-H2Down)*(P2-425))/(450-425))
    print("H2 Values:",round(H2,3))
elif P2 == 450:
    H2 = interpolate(S2,Ssup450,Hsup450)
    print(round(H2,3))
elif 450<P2<475:
    H2Down = interpolate(S2,Ssup450,Hsup450)
    H2Up = interpolate(S2,Ssup475,Hsup475)
    H2 = H2Down + (((H2Up-H2Down)*(P2-450))/(475-450))
    print("H2 Values:",round(H2,3))
elif P2 == 475:
    H2 = interpolate(S2,Ssup475,Hsup475)
    print(round(H2,3))
elif 475<P2<500:
    H2Down = interpolate(S2,Ssup475,Hsup475)
    H2Up = interpolate(S2,Ssup500,Hsup500)
    H2 = H2Down + (((H2Up-H2Down)*(P2-475))/(500-475))
    print("H2 Values:",round(H2,3))
elif P2 == 500:
    H2 = interpolate(S2,Ssup500,Hsup500)
    print(round(H2,3))
elif 500<P2<525:
    H2Down = interpolate(S2,Ssup500,Hsup500)
    H2Up = interpolate(S2,Ssup525,Hsup525)
    H2 = H2Down + (((H2Up-H2Down)*(P2-500))/(525-500))
    print("H2 Values:",round(H2,3))
elif P2 == 525:
    H2 = interpolate(S2,Ssup525,Hsup525)
    print(round(H2,3))
elif 525<P2<550:
    H2Down = interpolate(S2,Ssup525,Hsup525)
    H2Up = interpolate(S2,Ssup550,Hsup550)
    H2 = H2Down + (((H2Up-H2Down)*(P2-525))/(550-525))
    print("H2 Values:",round(H2,3))
elif P2 == 550:
    H2 = interpolate(S2,Ssup550,Hsup550)
    print(round(H2,3))
elif 550<P2<575:
    H2Down = interpolate(S2,Ssup550,Hsup550)
    H2Up = interpolate(S2,Ssup575,Hsup575)
    H2 = H2Down + (((H2Up-H2Down)*(P2-550))/(575-550))
    print("H2 Values:",round(H2,3))
elif P2 == 575:
```

```
    H2 = interpolate(S2,Ssup575,Hsup575)
    print(round(H2,3))
elif 575<P2<600:
    H2Down = interpolate(S2,Ssup575,Hsup575)
    H2Up = interpolate(S2,Ssup600,Hsup600)
    H2 = H2Down + (((H2Up-H2Down)*(P2-575))/(600-575))
    print("H2 Values:",round(H2,3))
elif P2 == 600:
    H2 = interpolate(S2,Ssup500,Hsup500)
    print(round(H2,3))
elif 600<P2<625:
    H2Down = interpolate(S2,Ssup600,Hsup600)
    H2Up = interpolate(S2,Ssup625,Hsup625)
    H2 = H2Down + (((H2Up-H2Down)*(P2-600))/(625-600))
    print("H2 Values:",round(H2,3))
elif P2 == 625:
    H2 = interpolate(S2,Ssup625,Hsup625)
    print(round(H2,3))
elif 625<P2<650:
    H2Down = interpolate(S2,Ssup625,Hsup625)
    H2Up = interpolate(S2,Ssup650,Hsup650)
    H2 = H2Down + (((H2Up-H2Down)*(P2-625))/(650-625))
    print("H2 Values:",round(H2,3))
elif P2 == 650:
    H2 = interpolate(S2,Ssup650,Hsup650)
    print(round(H2,3))
elif 650<P2<675:
    H2Down = interpolate(S2,Ssup650,Hsup650)
    H2Up = interpolate(S2,Ssup675,Hsup675)
    H2 = H2Down + (((H2Up-H2Down)*(P2-650))/(675-650))
    print("H2 Values:",round(H2,3))
elif P2 == 675:
    H2 = interpolate(S2,Ssup675,Hsup675)
    print(round(H2,3))
elif 675<P2<700:
    H2Down = interpolate(S2,Ssup675,Hsup675)
    H2Up = interpolate(S2,Ssup700,Hsup700)
    H2 = H2Down + (((H2Up-H2Down)*(P2-675))/(700-675))
    print("H2 Values:",round(H2,3))
elif P2 == 700:
    H2 = interpolate(S2,Ssup700,Hsup700)
    print(round(H2,3))
elif 700<P2<725:
    H2Down = interpolate(S2,Ssup700,Hsup700)
    H2Up = interpolate(S2,Ssup725,Hsup725)
    H2 = H2Down + (((H2Up-H2Down)*(P2-700))/(725-700))
    print("H2 Values:",round(H2,3))
elif P2 == 725:
    H2 = interpolate(S2,Ssup725,Hsup725)
    print(round(H2,3))
elif 725<P2<750:
    H2Down = interpolate(S2,Ssup725,Hsup725)
```

```
    H2Up = interpolate(S2,Ssup750,Hsup750)
    H2 = H2Down + (((H2Up-H2Down)*(P2-725))/(750-725))
    print("H2 Values:",round(H2,3))
elif P2 == 750:
    H2 = interpolate(S2,Ssup750,Hsup750)
    print(round(H2,3))
elif 750<P2<775:
    H2Down = interpolate(S2,Ssup750,Hsup750)
    H2Up = interpolate(S2,Ssup775,Hsup775)
    H2 = H2Down + (((H2Up-H2Down)*(P2-750))/(775-750))
    print("H2 Values:",round(H2,3))
elif P2 == 775:
    H2 = interpolate(S2,Ssup775,Hsup775)
    print(round(H2,3))
elif 775<P2<800:
    H2Down = interpolate(S2,Ssup775,Hsup775)
    H2Up = interpolate(S2,Ssup800,Hsup800)
    H2 = H2Down + (((H2Up-H2Down)*(P2-775))/(800-775))
    print("H2 Values:",round(H2,3))
elif P2 == 800:
    H2 = interpolate(S2,Ssup800,Hsup800)
    print(round(H2,3))
elif 800<P2<850:
    H2Down = interpolate(S2,Ssup800,Hsup800)
    H2Up = interpolate(S2,Ssup850,Hsup850)
    H2 = H2Down + (((H2Up-H2Down)*(P2-800))/(850-800))
    print("H2 Values:",round(H2,3))
elif P2 == 850:
    H2 = interpolate(S2,Ssup850,Hsup850)
    print(round(H2,3))
elif 850<P2<900:
    H2Down = interpolate(S2,Ssup850,Hsup850)
    H2Up = interpolate(S2,Ssup900,Hsup900)
    H2 = H2Down + (((H2Up-H2Down)*(P2-850))/(900-850))
    print("H2 Values:",round(H2,3))
elif P2 == 900:
    H2 = interpolate(S2,Ssup900,Hsup900)
    print(round(H2,3))
elif 900<P2<950:
    H2Down = interpolate(S2,Ssup900,Hsup900)
    H2Up = interpolate(S2,Ssup950,Hsup950)
    H2 = H2Down + (((H2Up-H2Down)*(P2-900))/(950-900))
    print("H2 Values:",round(H2,3))
elif P2 == 950:
    H2 = interpolate(S2,Ssup950,Hsup950)
    print(round(H2,3))
elif 950<P2<1000:
    H2Down = interpolate(S2,Ssup950,Hsup950)
    H2Up = interpolate(S2,Ssup1000,Hsup1000)
    H2 = H2Down + (((H2Up-H2Down)*(P2-950))/(1000-950))
    print("H2 Values:",round(H2,3))
elif P2 == 1000:
```

```
    H2 = interpolate(S2,Ssup1000,Hsup1000)
    print(round(H2,3))
elif 1000<P2<1100:
    H2Down = interpolate(S2,Ssup1000,Hsup1000)
    H2Up = interpolate(S2,Ssup1100,Hsup1100)
    H2 = H2Down + (((H2Up-H2Down)*(P2-1000))/(1100-1000))
    print("H2 Values:",round(H2,3))
elif P2 == 1100:
    H2 = interpolate(S2,Ssup1100,Hsup1100)
    print(round(H2,3))
elif 1100<P2<1200:
    H2Down = interpolate(S2,Ssup1100,Hsup1100)
    H2Up = interpolate(S2,Ssup1200,Hsup1200)
    H2 = H2Down + (((H2Up-H2Down)*(P2-1100))/(1200-1100))
    print("H2 Values:",round(H2,3))
elif P2 == 1200:
    H2 = interpolate(S2,Ssup1200,Hsup1200)
    print(round(H2,3))
elif 1200<P2<1300:
    H2Down = interpolate(S2,Ssup1200,Hsup1200)
    H2Up = interpolate(S2,Ssup1300,Hsup1300)
    H2 = H2Down + (((H2Up-H2Down)*(P2-1200))/(1300-1200))
    print("H2 Values:",round(H2,3))
elif P2 == 1300:
    H2 = interpolate(S2,Ssup1300,Hsup1300)
    print(round(H2,3))
elif 1300<P2<1400:
    H2Down = interpolate(S2,Ssup1300,Hsup1300)
    H2Up = interpolate(S2,Ssup1400,Hsup1400)
    H2 = H2Down + (((H2Up-H2Down)*(P2-1300))/(1400-1300))
    print("H2 Values:",round(H2,3))
elif P2 == 1400:
    H2 = interpolate(S2,Ssup1400,Hsup1400)
    print(round(H2,3))
elif 1400<P2<1500:
    H2Down = interpolate(S2,Ssup1400,Hsup1400)
    H2Up = interpolate(S2,Ssup1500,Hsup1500)
    H2 = H2Down + (((H2Up-H2Down)*(P2-1400))/(1500-1400))
    print("H2 Values:",round(H2,3))
elif P2 == 1500:
    H2 = interpolate(S2,Ssup1500,Hsup1500)
    print(round(H2,3))
elif 1500<P2<1600:
    H2Down = interpolate(S2,Ssup1500,Hsup1500)
    H2Up = interpolate(S2,Ssup1600,Hsup1600)
    H2 = H2Down + (((H2Up-H2Down)*(P2-1500))/(1600-1500))
    print("H2 Values:",round(H2,3))
elif P2 == 1600:
    H2 = interpolate(S2,Ssup1600,Hsup1600)
    print(round(H2,3))
elif 1600<P2<1700:
    H2Down = interpolate(S2,Ssup1600,Hsup1600)
```

```
    H2Up = interpolate(S2,Ssup1700,Hsup1700)
    H2 = H2Down + (((H2Up-H2Down)*(P2-1600))/(1700-1600))
    print("H2 Values:",round(H2,3))
elif P2 == 1700:
    H2 = interpolate(S2,Ssup1700,Hsup1700)
    print(round(H2,3))
elif 1700<P2<1800:
    H2Down = interpolate(S2,Ssup1700,Hsup1700)
    H2Up = interpolate(S2,Ssup1800,Hsup1800)
    H2 = H2Down + (((H2Up-H2Down)*(P2-1700))/(1800-1700))
    print("H2 Values:",round(H2,3))
elif P2 == 1800:
    H2 = interpolate(S2,Ssup1800,Hsup1800)
    print(round(H2,3))
elif 1800<P2<1900:
    H2Down = interpolate(S2,Ssup1800,Hsup1800)
    H2Up = interpolate(S2,Ssup1900,Hsup1900)
    H2 = H2Down + (((H2Up-H2Down)*(P2-1800))/(1900-1800))
    print("H2 Values:",round(H2,3))
elif P2 == 1900:
    H2 = interpolate(S2,Ssup1800,Hsup1800)
    print(round(H2,3))
elif 1900<P2<2000:
    H2Down = interpolate(S2,Ssup1900,Hsup1900)
    H2Up = interpolate(S2,Ssup2000,Hsup2000)
    H2 = H2Down + (((H2Up-H2Down)*(P2-1900))/(2000-1900))
    print("H2 Values:",round(H2,3))
elif P2 == 2000:
    H2 = interpolate(S2,Ssup2000,Hsup2000)
    print(round(H2,3))
elif 2000<P2<2100:
    H2Down = interpolate(S2,Ssup2000,Hsup2000)
    H2Up = interpolate(S2,Ssup2100,Hsup2100)
    H2 = H2Down + (((H2Up-H2Down)*(P2-2000))/(2100-2000))
    print("H2 Values:",round(H2,3))



#TODO: Calculate Rated of Heat (QL) and Power Input to the Compressor (Win)
QL = (H1-H4)
print("QL Values:",round(QL,3))
Win = (H2-H1)
print("Win Values:",round(Win,3))
QH = (H2-H3)
print("QH Values:",round(QH,3))
COP = (QL/Win)
print("COP Values:",round(COP,3))
```