Open Education Resources(OER): Introduction to Engineering Thermodynamics

This is a project that extends the original e-textbook written by Dr. Claire Yu Yan in UBC Okanagan campus:Introduction to Engineering Thermodynamics. The main goals for this project are as follows:

(i) To provide students with a list of questions that can be done in their laptops with the help of some Python packages such as CoolProp, numpy and matplotlib.

(ii) To reinforce the concepts learnt from e-textbook and to get a better appreciation of how to use digital tools to do perform thermodynamic calculations.

(iii) To provide a starting point for Thermodynamics courses in other disciplines to adopt a more computational approach for their courses.

Acknowledgements

Funding acknowledgements

This project was made possible by 2023/24 OER Affordability Grant (formerly known as OER Implementation Grant) at UBC Vancouver.

Contributions

Content for various chapers was created by the following students:

Ali Doustahadi, MASc candidate in the Department of Materials Engineering, UBC Vancouver, created material for Chapter 3 and Chapter 5.

Dr.Chun-Sheng Wang, Postdoctoral research fellow in the Department of Mechanical Engineering, UBC Okanagan, created material for Chapter 1 and 6.

Hariharan Umashankar, PhD candidate in the Department of Materials Engineering, UBC Vancouver, created material for Chapter 2 and Chapter 4 and hosted this website on Github pages.

The following faculty members were responsible for providing feedback along the course of this project and acquiring the project funds:

Dr. Amir M. Dehkhoda, Assistant Professor of Teaching, Department of Materials Engineering, UBC Vancouver.

Dr. Claire (Yu) Yan, Associate Professor of Teaching, Department of Mechanical Engineering, UBC Okanagan.

Getting Started: Introduction to Jupyter and Python!

Hey! Welcome to learning about Jupyter notebooks and using it for doing basic thermodynamic calculations.

In this section, we will use the Jupyter notebook interface to show how to add, subtract, multiply and divide numbers. Also, an introduction about python packages will be provided in here.

Basic operations inside Jupyter

Adding a cell above the current one: ‘a’

Adding a cell below the current one: ‘b’

Delete cell entirely: ‘d,d’ (press d twice)

To execute/run a cell: Ctrl + Enter

To execute/run a cell and move to the next one: Shift + enter

Each cell has a type associated with it: “Code”, “Markdown”, “Raw”

Import libraries

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

Basic math

```
b = 1 + 1
print(b)
```

```
2
```

Skip to main content

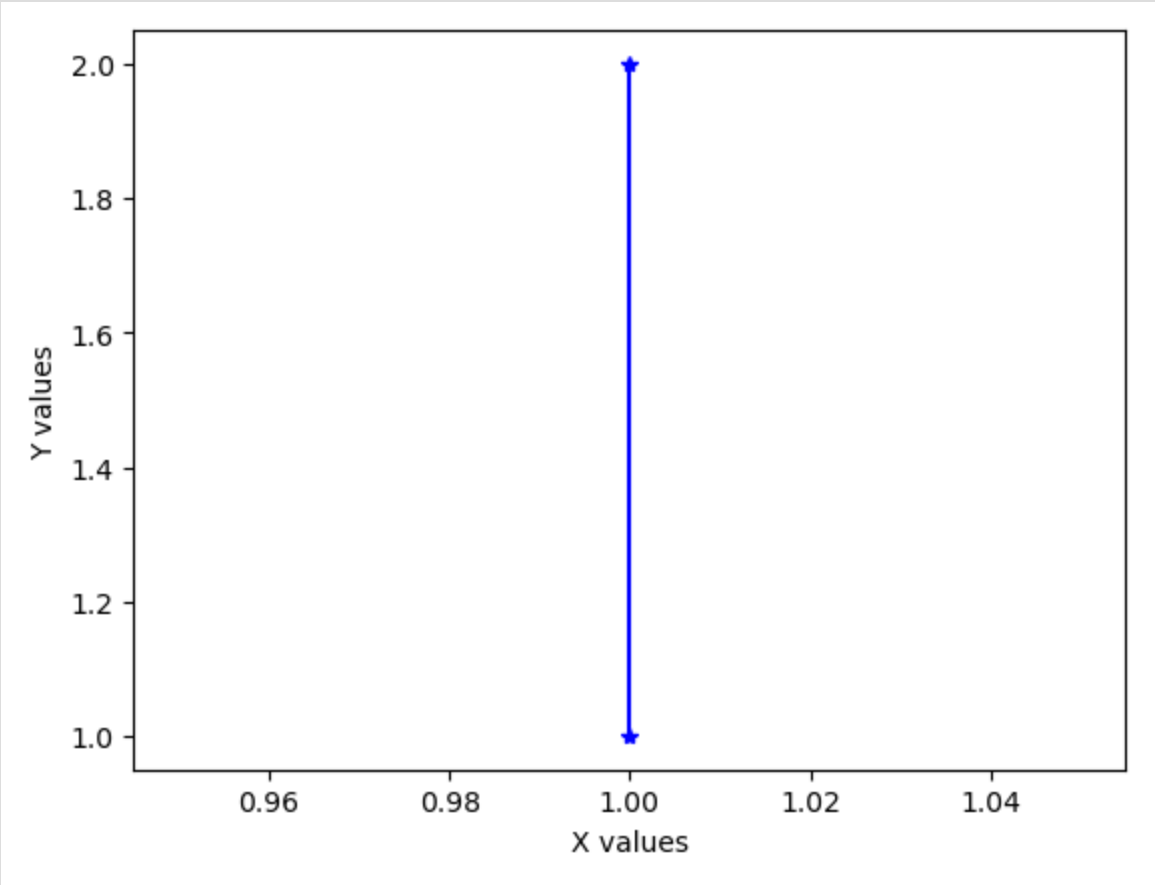
```
c = 4 * 4 # multiply
print(c)
c = 4 **2 # exponentiate
print(c)
```

```
16
16
```

Plotting using `matplotlib` package

```
plt.plot([1,1],[1,2],'-*b') ## x array, y array and line or marker specifications
plt.xlabel('X values')
plt.ylabel('Y values')
```

```
Text(0, 0.5, 'Y values')
```



Numerical operation using `numpy` package

```
a = np.linspace(10,50,num=50) ## create a regular-spaced array from start to end and number of datap
```

```
b = np.linspace(0,0.5,num=50) ## create a regular-spaced array from start to stop (excluding) with s
```

```
np.max(a)
```

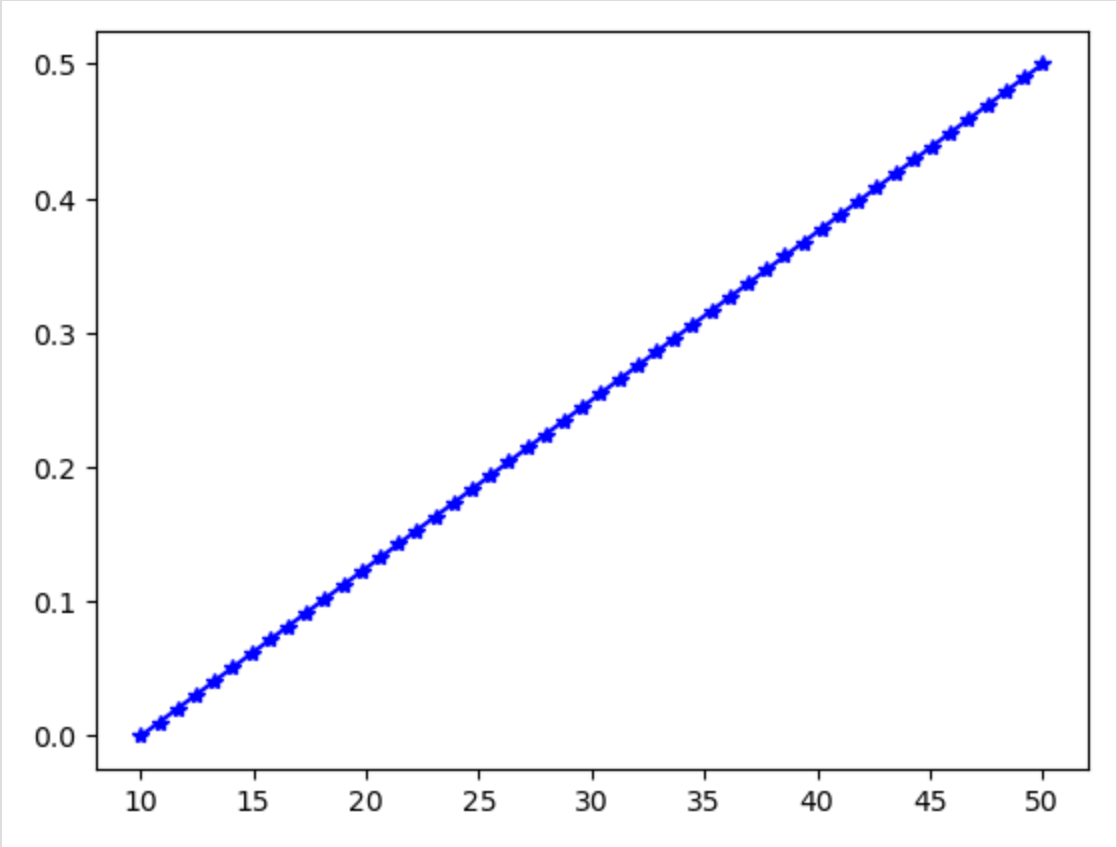
```
50.0
```

```
np.max(b)
```

```
0.5
```

```
plt.plot(a,b,'-*b')
```

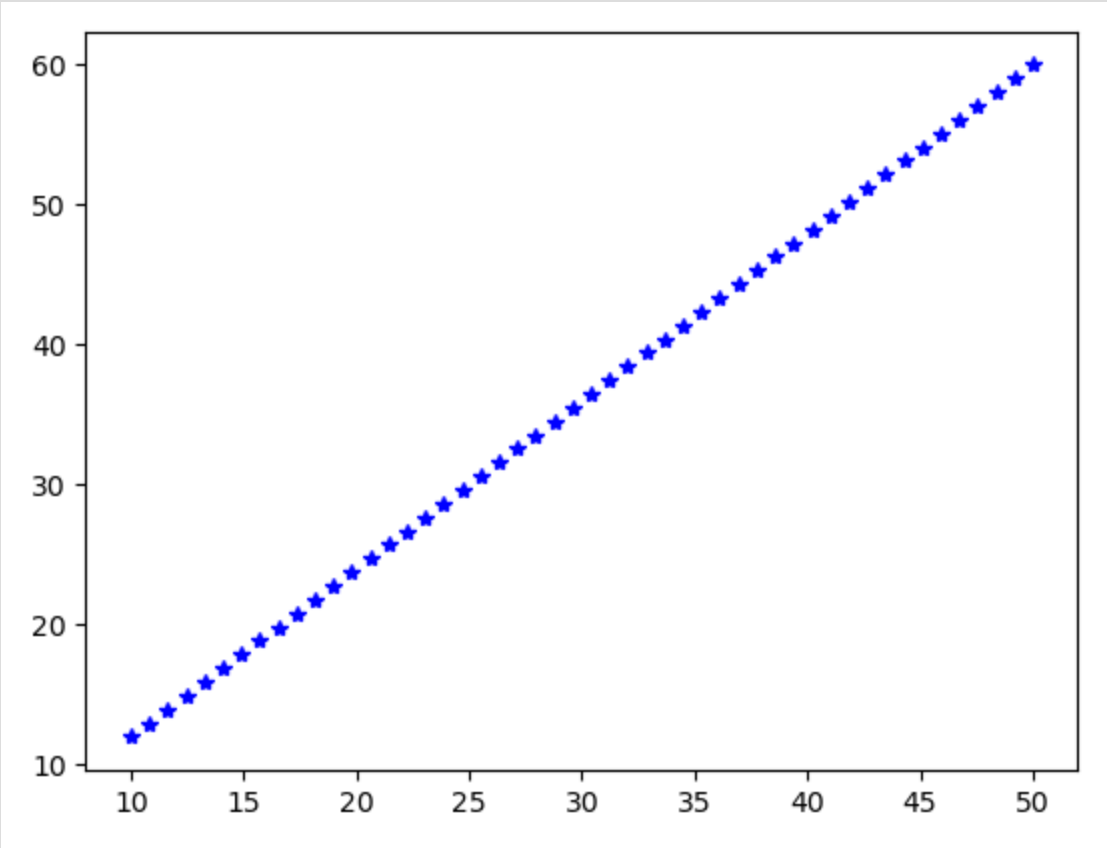
```
[<matplotlib.lines.Line2D at 0x7f6285497d60>]
```



Skip to main content

```
b = np.linspace(12,60,num=50)
plt.plot(a,b,'*b')
```

```
[<matplotlib.lines.Line2D at 0x7f628541f9a0>]
```




```
np.random.rand(15)
```

```
array([0.89342128, 0.22409828, 0.97702249, 0.5533418 , 0.72261383,
       0.63191313, 0.2455794 , 0.588028  , 0.94449222, 0.81295154,
       0.85673411, 0.98242762, 0.70420093, 0.73077149, 0.63828771])
```

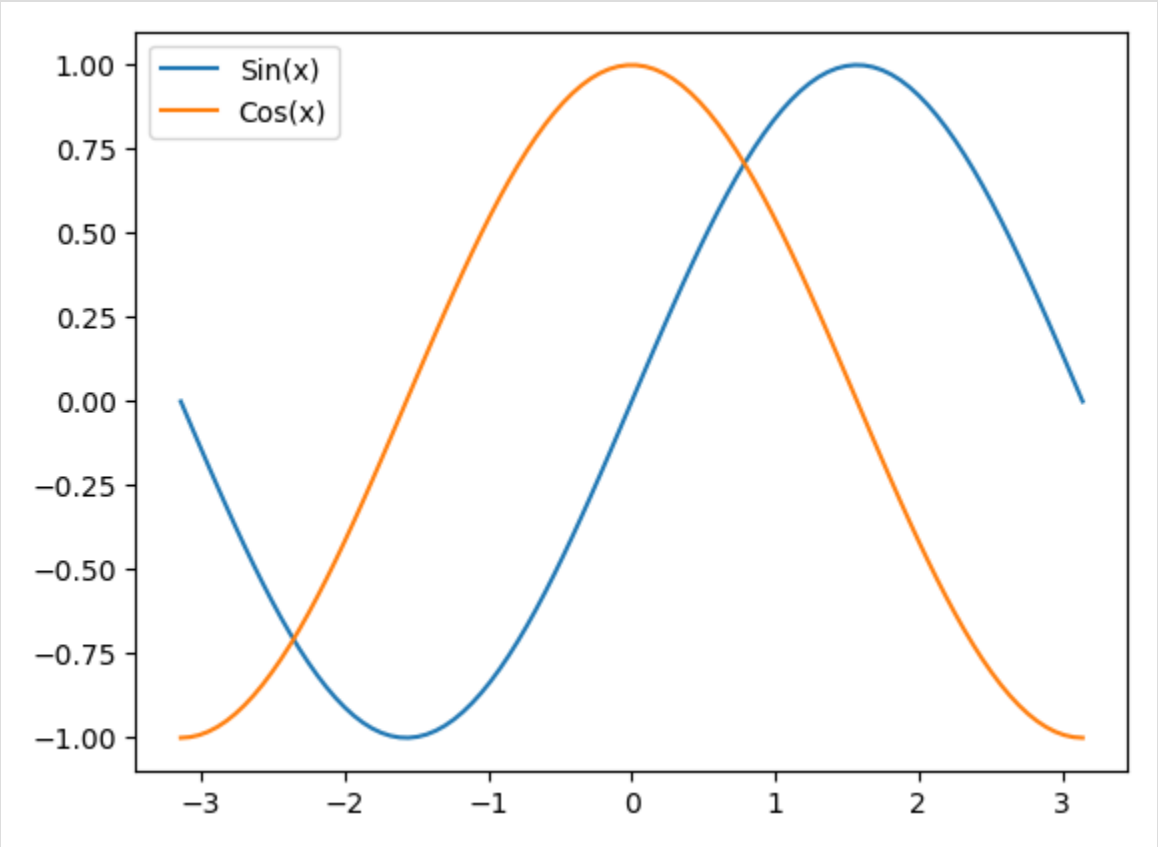
```
np.random.randint(50,high=100,size=12)
```

```
array([59, 83, 54, 95, 99, 89, 88, 75, 55, 65, 76, 68])
```

Skip to main content

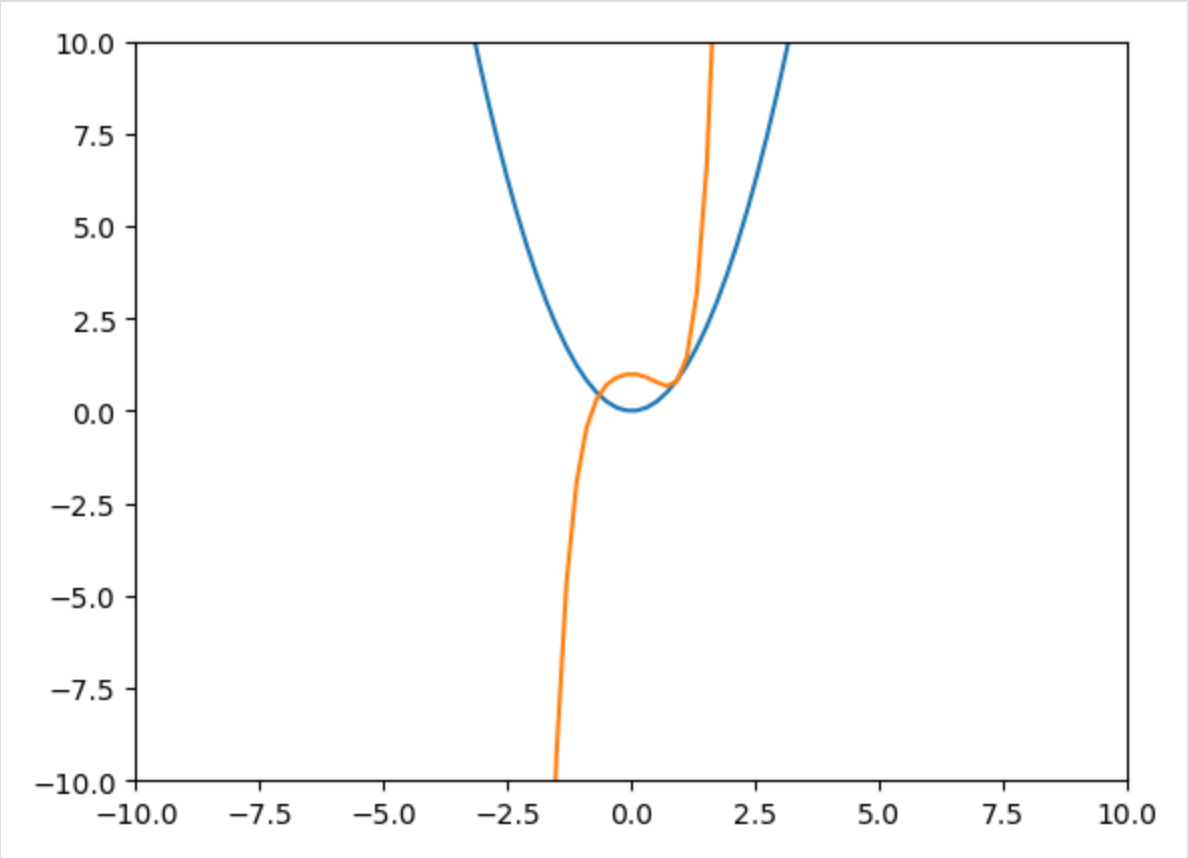
```
## did some plotting with sinx, cosx
x = np.linspace(-np.pi, np.pi, 100)
y = np.sin(x)
plt.plot(x, y,label='Sin(x)')
y = np.cos(x)
plt.plot(x, y, label='Cos(x)')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f62b86e7f70>
```




```
## did some plotting with polynomials
x = np.linspace(-10,10,100)
y = x ** 2
plt.plot(x, y, label ='x^2')
y = x ** 5 - x **2 + 1
plt.plot(x, y, label = 'x^5 - x^2 + 1')

## something that we did not cover in class:
plt.xlim([-10,10]) ## restricts the x-axis limit from -10 to +10
plt.ylim([-10,10]) ## restricts the y-axis limit from -10 to + 10
```

```
(-10.0, 10.0)
```



Installing CoolProp and calculating Enthalpy of Vaporization

install CoolProp by using:

!pip install CoolProp

or

import sys !{sys.executable} -m pip install CoolProp

```
from CoolProp.CoolProp import PropsSI
```

```
# enthalpy of vaporization example
H_L = PropsSI('H', 'P', 101325, 'Q', 0, 'water')
H_V = PropsSI('H', 'P', 101325, 'Q', 1, 'water')
```

```
print("the enthalpy of vaporization at P = 1 atm is:",round((H_V - H_L)/1000), 'kJ/kg')
```

Skip to main content

Defining variables: Properties of Ideal Gases as use case

From: https://pressbooks.bccampus.ca/thermo1/back-matter/properties-of-various-substances/#TG1

Use the above table to look-up the following properties of Air:

Gas constant (R)

C_p (heat capacity at const. pressure)

C_v (heat capacity at const. volume)

k (ratio of Cp and Cv)

```
#Air
R_air = 0.287 # kJ/kgK
Cp_air = 1.005 #kJ/kgK
Cv_air = 0.718 #kJ/kgK
k_air = 1.4
```

```
#Argon

### Please fill in here from looking into the above URL ###
```

R_argon = # kJ/kgK

Cp_argon = #kJ/kgK

Cv_argon = #kJ/kgK

k_argon = # unitless

1. Basic Concepts and Definitions

Let's do some problems based on concepts on Chapter 1 from the e-textbook.

```
import numpy as np
```

2. Thermodynamic Properties of a Pure Substance

This is a problem set based on the concepts from the Chapter here: https://pressbooks.bccampus.ca/thermo1/chapter/2-0-chapter-introduction-and-learning-objectives/

Problem Statement:

A cylindrical tank contains air at a gauge pressure of 3 atm and an ambient atmospheric pressure of 1 atm. Calculate the absolute pressure inside the tank.

```
## Solution:

# Given values
P_gauge = 3 * 101325  # Gauge pressure in Pascals (1 atm = 101325 Pa)
P_atm = 1 * 101325  # Atmospheric pressure in Pascals

# Absolute pressure
P_abs = P_gauge + P_atm

# Output the result
print(f"Absolute pressure inside the tank: {round(P_abs/1e3,1)} kPa")
```

```
Absolute pressure inside the tank: 405.3 kPa
```

Density of a Gas

Problem Statement:

A container with a volume of 2 m³ holds 5 kg of nitrogen gas. Calculate the density of the gas.

```
## Solution:

# Given values
m = 5  # Mass of nitrogen in kg
V = 2  # Volume in m^3

# Density calculation
density = m / V

# Output the result
print(f"Density of the nitrogen gas: {density} kg/m^3")
```

```
Density of the nitrogen gas: 2.5 kg/m^3
```

Energy in a Moving Vehicle

Skip to main content

A car with a mass of 1200 kg is traveling at a speed of 60 km/h. If the height above the ground is 50 m, calculate the total stored energy of the car.

```
## Solution:

# Given values
m = 1200  # Mass of the car in kg
V = 60 * (1000 / 3600)  # Speed in m/s (60 km/h)
g = 9.81  # Acceleration due to gravity in m/s^2
z = 50  # Height in meters

# Total stored energy calculation
E = m * (V**2 / 2 + g * z)

# Output the result
print(f"Total stored energy of the car: {round(E/1e3)} kJ")
```

```
Total stored energy of the car: 755 kJ
```

Enthalpy of a Steam

Problem Statement:

Calculate the specific enthalpy of steam at a pressure of 200 kPa and a specific volume of 0.15 m³/kg.

```
## Solution:

import CoolProp.CoolProp as CP

# Given values
P = 200000  # Pressure in Pascals
v = 0.15  # Specific volume in m^3/kg

# Specific enthalpy calculation
h = CP.PropsSI('H', 'P', P, 'D', 1/v, 'Water')

# Output the result
print(f"Specific enthalpy of steam: {round(h/1e3,1)} kJ/kg")
```

```
Specific enthalpy of steam: 875.4 kJ/kg
```

Quality of a Vapor Mixture

Skip to main content

Problem Statement:

In a vapor mixture at 150°C, the mass of the vapor phase is 3 kg, and the total mass of the mixture is 5 kg. Determine the quality of the mixture.

```
## Solution:

# Given values
m_vapor = 3  # Mass of vapor phase in kg
m_total = 5  # Total mass of the mixture in kg

# Quality calculation
x = m_vapor / m_total

# Output the result
print(f"Quality of the mixture: {x}")
```

```
Quality of the mixture: 0.6
```

Compressed Liquid Approximation

Problem Statement:

For water at 50°C under high pressure, approximate the specific volume, internal energy, enthalpy, and entropy assuming it behaves as a compressed liquid.

```
## Solution:

import CoolProp.CoolProp as CP

# Given values
T = 50 + 273.15  # Temperature in Kelvin

# Approximations for compressed liquid
v = CP.PropsSI('D', 'T', T, 'Q', 0, 'Water')  # Specific volume
u = CP.PropsSI('U', 'T', T, 'Q', 0, 'Water')/1e3  # Specific internal energy
h = CP.PropsSI('H', 'T', T, 'Q', 0, 'Water')/1e3  # Specific enthalpy
s = CP.PropsSI('S', 'T', T, 'Q', 0, 'Water')/1e3  # Specific entropy

# Output the results
print(f"Specific volume: {round(v,2)} m^3/kg")
print(f"Specific internal energy: {round(u,2)} kJ/kg")
print(f"Specific enthalpy: {round(h,2)} kJ/kg")
print(f"Specific entropy: {round(s,2)} kJ/kg*K")
```

Skip to main content

```
Specific volume: 988.0 m^3/kg
Specific internal energy: 209.33 kJ/kg
Specific enthalpy: 209.34 kJ/kg
Specific entropy: 0.7 kJ/kg*K
```

Conversion of Temperature

Problem Statement:

Convert a temperature of 20°C to Kelvin.

```
## Solution:

# Given value
T_Celsius = 20  # Temperature in Celsius

# Conversion to Kelvin
T_Kelvin = T_Celsius + 273.15

# Output the result
print(f"Temperature in Kelvin: {T_Kelvin} K")
```

```
Temperature in Kelvin: 293.15 K
```

Specific Entropy Calculation

Problem Statement:

Calculate the specific entropy of a system with a total entropy of 2500 J/K and a mass of 10 kg.

```
## Solution:

# Given values
S_total = 2500  # Total entropy in J/K
m = 10  # Mass in kg

# Specific entropy calculation
s = S_total / m

# Output the result
print(f"Specific entropy: {round(s/1e3,1)} kJ/(kg*K)")
```

Skip to main content

```
Specific entropy: 0.2 kJ/(kg*K)
```

Complete the following table for Water:

T, degC	P, kPa	u, kJ/kg	Phase description
	560	1470	
240			Saturated vapor
150	2600		
	4200	3100	

Lookup tables from the e-textbook: https://pressbooks.bccampus.ca/thermo1/chapter/thermodynamic-tables/

```
import CoolProp.CoolProp as CP
```

```
########## (a)#################
P = 0.56e6 # in Pa
u = 1470 # in kJ/kg
fluid = "water"
uf = CP.PropsSI("U", "Q", 0, "P", P,fluid)/1e3
ug = CP.PropsSI("U", "Q", 1, "P", P, fluid)/1e3
print("Uf at given pressure: {} kJ/kg".format(round(uf,2)))
print("Ug at given pressure: {} kJ/kg".format(round(ug,2)))
## we are in saturated mixture region; since u_given > uf and u_given < ug!
T = CP.PropsSI("T","P",P,"Q",0,fluid)
print("Phase description: Saturated mixture")
print("Temperature: {}".format(round(T - 273.15,2)),"deg C")
```

```
Uf at given pressure: 658.15 kJ/kg
Ug at given pressure: 2564.51 kJ/kg
Phase description: Saturated mixture
Temperature: 156.15 deg C
```

```
########## (b)#################
fluid = "water"
T = 240 + 273.15 # Kelvin
uf = CP.PropsSI("U", "Q", 0, "T", T,fluid)/1e3
ug = CP.PropsSI("U", "Q", 1, "T", T, fluid)/1e3
print("U at given temperature : {} kJ/kg".format(round(ug,1)))
pressure = CP.PropsSI("P", "T", T, "Q", 1, fluid)
print("Pressure: {} kPa".format(round(pressure /1e3,1)))
```

Skip to main content

```
U at given temperature : 2603.1 kJ/kg
Pressure: 3346.9 kPa
```

```
############ (c) ##################
fluid = "water"
T = 150 + 273.15 # Kelvin
P = 2600e3 # Pa
Psat = CP.PropsSI("P", "T", T, "Q",0,fluid)/1e3
## P > Psat; hence compressed fluid
print("Phase description: Compressed liquid")
u_state = CP.PropsSI("U", "T", T, "P", P, fluid)
uf = CP.PropsSI("U", "T", T, "Q", 0, fluid)
print("u = {} kJ/kg".format(round(u_state/1e3,2)))
print("uf  = {} kJ/kg".format(round(uf/1e3,2)))
```

```
Phase description: Compressed liquid
u = 630.66 kJ/kg
uf  = 631.66 kJ/kg
```

```
############ (d)#################
fluid = "water"
pressure = 4200e3 # Pa
u_given_state = 3100e3 # in J/kg
# calculating u at q = 0 (sat. liquid)
## add comments to help the audience to help them to understand the arguments that you supply
uf = CP.PropsSI("U", "Q", 0, "P", pressure,fluid)/1e3
ug = CP.PropsSI("U", "Q", 1, "P", pressure, fluid)/1e3
print("Uf at given pressure: {} kJ/kg".format(round(uf,1)))
print("Ug at given pressure: {} kJ/kg".format(round(ug,1)))
## we are in ssuperheated vapor region; u > ug!
T = CP.PropsSI("T", "P", pressure, "U", u_given_state,fluid)
print("Temperature at the final state in deg C: {}".format(round(T-273.15,2)))
```

```
Uf at given pressure: 1096.4 kJ/kg
Ug at given pressure: 2601.0 kJ/kg
Temperature at the final state in deg C: 500.8
```

Completed table:

T, degC	P, kPa	u, kJ/kg	Phase description
156.15	560	1470	**Saturated liq-vapour mixture**
240	**3346.92**	**2603.12**	Saturated vapor
150	2600	**630.66**	**Compressed Liquid**
500.8	4200	3100	**Superheated vapor**

Complete the table: R-134a

T, degC	P, kPa	u, kJ/kg	Phase description
	800	275	
-10			Saturated vapor
30	1200		
	2000	250	

```
from CoolProp import CoolProp as CP
########### (a)================###
P = 0.8e6 # in Pa
u = 275 # in kJ/kg
fluid = "R134a"
uf = CP.PropsSI("U", "Q", 0, "P", P,fluid)/1e3
ug = CP.PropsSI("U", "Q", 1, "P", P, fluid)/1e3
print("Uf at given pressure: {} kJ/kg".format(round(uf,1)))
print("Ug at given pressure: {} kJ/kg".format(round(ug,1)))
## we are in saturated mixture region; since u_given > uf and u_given < ug!
T = CP.PropsSI("T","P",P,"Q",0,fluid)
print("Phase description: Saturated mixture")
print("Temperature: {}".format(round(T - 273.15,2)),"deg C")
```

```
Uf at given pressure: 243.0 kJ/kg
Ug at given pressure: 395.0 kJ/kg
Phase description: Saturated mixture
Temperature: 31.33 deg C
```

Skip to main content

```
########### (b)================###
T = -10 + 273.15 ## K
fluid = "R134a"
press1 = CP.PropsSI("P","T",T,"Q",1,fluid)/1e3
ug = CP.PropsSI("U", "Q", 1, "T", T, fluid)/1e3
print("U = Ug at given temperature: {} kJ/kg".format(round(ug,2)))
print("Pressure = {} kPa".format(round(press1,1)))
```

```
U = Ug at given temperature: 372.69 kJ/kg
Pressure = 200.6 kPa
```

```
########### (c)================###
T = 30 + 273.15 ## K
P = 1200e3 # Pa
fluid = "R134a"
press_at_given_temp = CP.PropsSI("P","T",T,"Q",0,fluid) /1e3
print("Pressure = {} kPa".format(round(press_at_given_temp,1)))
u= CP.PropsSI("U", "P", P, "T", T, fluid)/1e3
print("U = {} kJ/kg".format(round(u,2)))
uf= CP.PropsSI("U", "Q",0, "T", T, fluid)/1e3
print("Uf = {} kJ/kg".format(round(uf,2)))
```

```
Pressure = 770.2 kPa
U = 240.7 kJ/kg
Uf = 241.07 kJ/kg
```

```
########### (d)================###
P = 2000e3 # Pa
fluid = "R134a"
T_sat = CP.PropsSI("T", "P", P,"Q",0,fluid)
print("Temperature = {} C".format(round(T_sat-273.15,1)))
uf= CP.PropsSI("U", "P", P, "Q",0, fluid)/1e3
ug= CP.PropsSI("U", "P", P, "Q",1, fluid)/1e3

print("Uf = {} kJ/kg".format(round(uf,2)))
print("Ug = {} kJ/kg".format(round(ug,2)))
## given U is below u_f; hence compressed liquid
T = CP.PropsSI("T","U",250e3,"P",P,fluid) - 273.15
print("Temperature = {} C".format(round(T,2)))
```

```
Temperature = 67.5 C
Uf = 297.98 kJ/kg
Ug = 409.7 kJ/kg
Temperature = 36.92 C
```

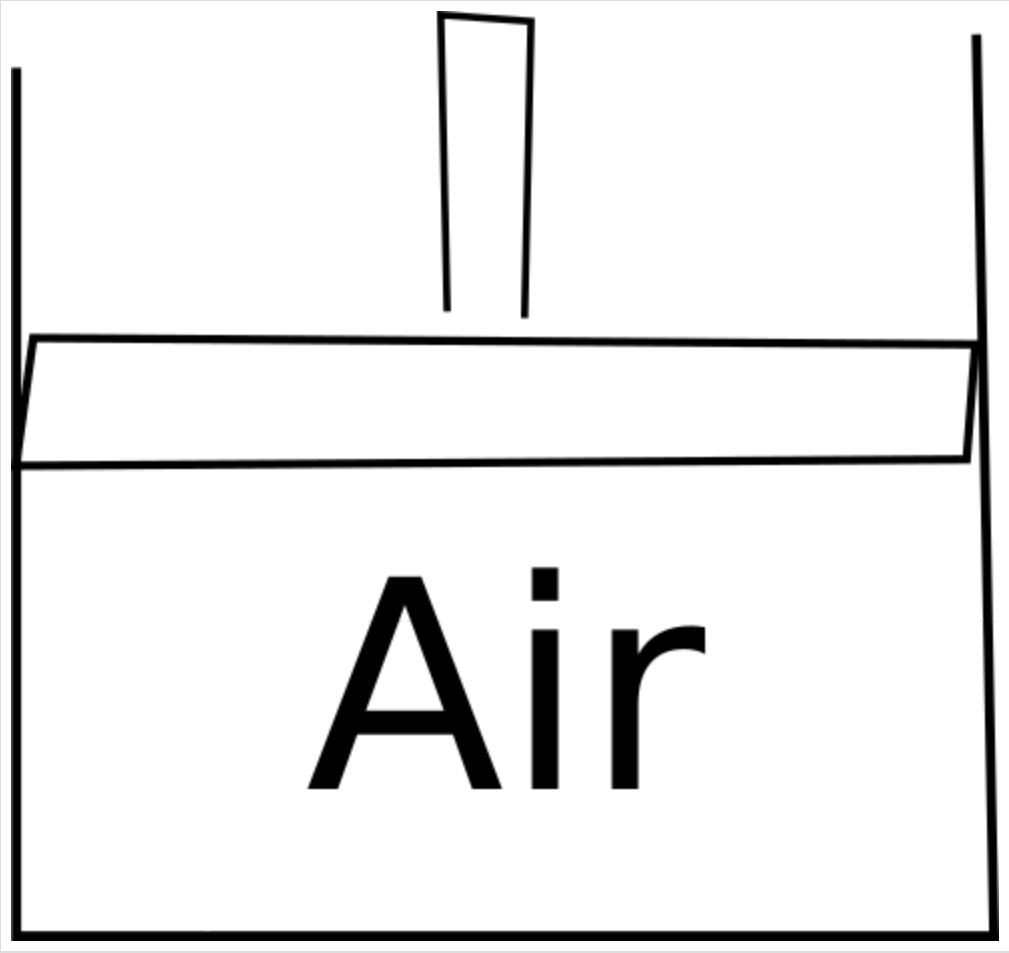
Skip to main content

T, degC	P, kPa	u, kJ/kg	Phase description
31.33	800	275	**Saturated liq-vapour mixture**
-10	**200.6**	**372.687**	Saturated vapor
30	1200	**240.7**	**Compressed liquid**
36.92	2000	250	**Compressed Liquid**

Piston-cylinder: Isobaric process

Problem statement:

A piston–cylinder device contains 0.9 kg of steam at 300°C and 0.8 MPa. Steam is cooled at constant pressure until one-third of the mass condenses. (a) Show the process on a T-v diagram and a P-v diagram (b) Find the final temperature. (c) Determine the volume change



Skip to main content

A function definition to help plot P-v diagram

The arguments of this function are supplied inside the paranthesis:

fluid: “Fluid of interest”

state1: [P,v] list

state2: [P,v] list

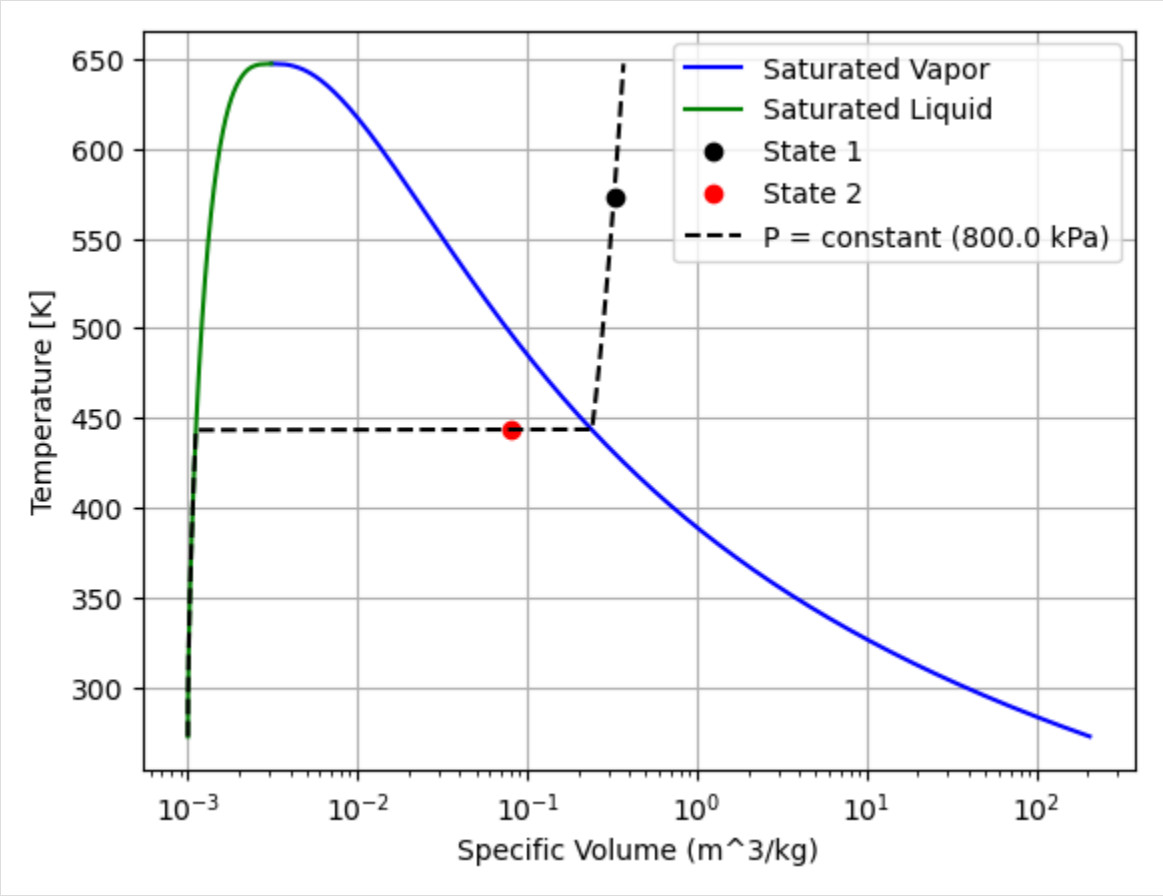
```
def plot_p_v_diagram(fluid, state1, state2, plot_const_press_line=True):
    # import the libraries we'll need
    import CoolProp.CoolProp as CP
    import numpy as np
    import matplotlib.pyplot as plt

    # define variables
    fluid = fluid
    T_min = CP.PropsSI("Tmin", fluid)
    T_max = CP.PropsSI("Tcrit", fluid)
    T_vals = np.linspace(T_min, T_max, 1000)
    Q = 1

    P_saturated_vapor = [CP.PropsSI("P", "T", T, "Q", Q, fluid) for T in T_vals]
    vol_vapor = 1 / np.array([CP.PropsSI("D", "T", T, "Q", Q, fluid) for T in T_vals])
    plt.figure(2)
    plt.plot(vol_vapor, P_saturated_vapor, "-b", label="Saturated Vapor")
    plt.xscale("log")

    Q = 0

    P_saturated_liquid = [CP.PropsSI("P", "T", T, "Q", Q, fluid) for T in T_vals]
    vol_liquid = 1 / np.array([CP.PropsSI("D", "T", T, "Q", Q, fluid) for T in T_vals])

    plt.plot(vol_liquid, P_saturated_liquid, "-g", label="Saturated Liquid")
    plt.xscale("log")

    plt.ylabel("Pressure [Pa]")
    plt.xlabel("Specific Volume (m^3/kg)")

    # Specific states
    P1 = state1[0]
    P2 = state2[0]

    plt.plot(state1[1], P1, "ok", label="State 1")
    plt.plot(state2[1], P2, "or", label="State 2")

    if plot_const_press_line:
        plt.axhline(y=P1, color='k', linestyle='--', label="P = constant ({} kPa)".format(round(P1/1

    plt.legend()
    plt.grid()
    plt.show()
```

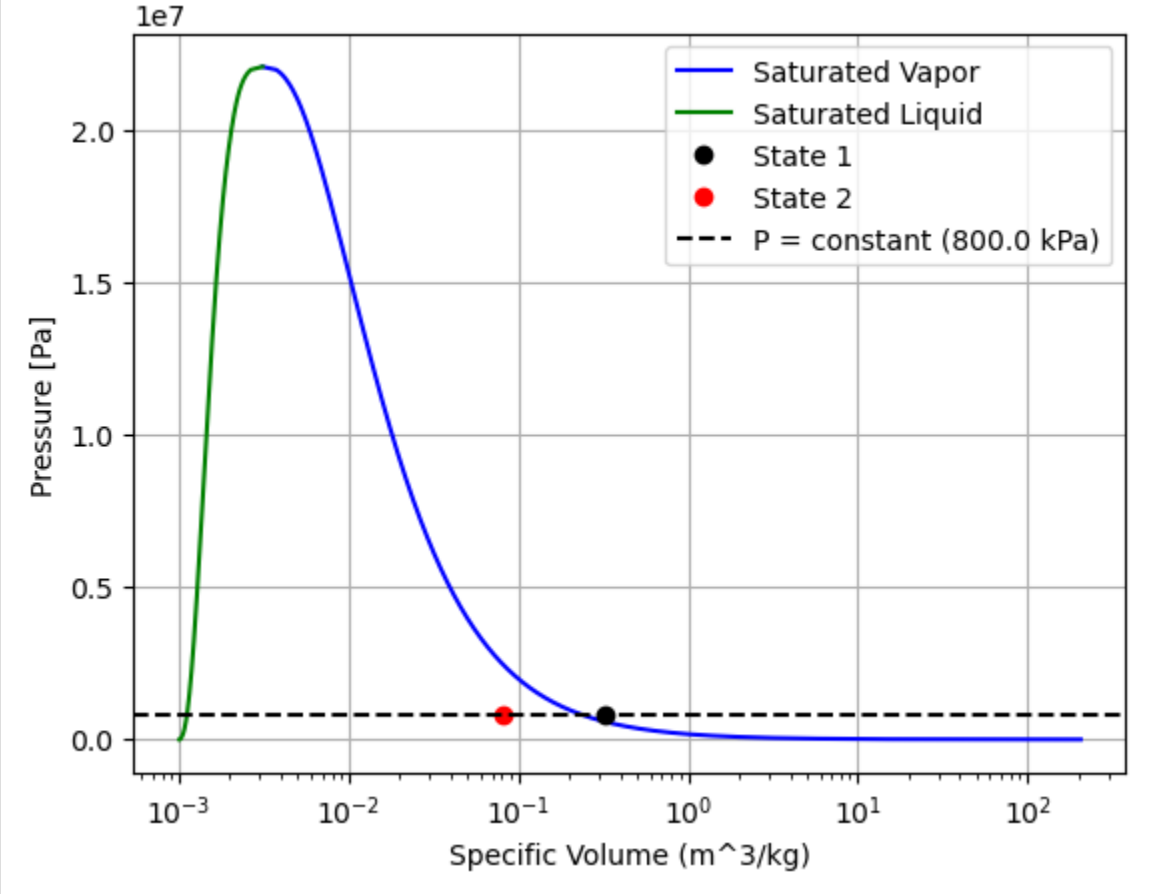
Solution:

Skip to main content

```
import CoolProp.CoolProp as CP
import numpy as np
import matplotlib.pyplot as plt
def plot_T_v_diagram(fluid,state1,state2,plot_const_press_line=True):
    # import the libraries we'll need
    import CoolProp.CoolProp as CP
    import numpy as np
    import matplotlib.pyplot as plt


    # define variables
    fluid = fluid  # define the fluid or material of interest, for full list see CP.Fluidslist()
    T_min = CP.PropsSI("Tmin", fluid)  # this is the min temp we can get data for water
    T_max = CP.PropsSI("Tcrit", fluid)  # this is the max temp we can get data for water
    T_vals = np.linspace(
        T_min, T_max, 1000
    )  # define an array of values from T_min to T_max
    Q = 1  # define the steam quality as 1, which is 100% vapor


    density = [
        CP.PropsSI("D", "T", T, "Q", Q, fluid) for T in T_vals
    ]  # call for density values using CoolProp
    vol = 1 / np.array(density)  # convert density into specific volume

    plt.figure(1)
    plt.plot(vol, T_vals, "-b", label="Saturated Vapor")  # plot temp vs specific vol
    plt.xscale("log")  # use log scale on x axis


    Q = 0  # define the steam quality as 0, which is 100% liquid


    density = [
        CP.PropsSI("D", "T", T, "Q", Q, fluid) for T in T_vals
    ]  # call for density values using CoolProp
    vol = 1 / np.array(density)  # convert density into specific volume


    plt.plot(vol, T_vals, "-g", label="Saturated Liquid")  # plot temp vs specific vol
    plt.xscale("log")  # use log scale on x axis


    plt.ylabel("Temperature [K]")  # give y axis a label
    plt.xlabel("Specific Volume (m^3/kg)")  # give x axis a label
    plt.grid()
    # plot various points on the T-v diagram:

    x = [state1[0], state2[0]]  # specific volume in m3/kg
    y = [state1[1], state2[1]]  # temperature in K
    plt.plot(x[0], y[0], "ok", label="State 1")
    plt.plot(x[1], y[1], "or", label="State 2")
    if plot_const_press_line == True:
        # Plotting the constant pressure line for the given pressure:
        P_const = CP.PropsSI("P", "T", state1[1], "D", 1/state1[0], fluid)
        v_vals_constP = [1 / CP.PropsSI("D", "T", T, "P", P_const, fluid) for T in T_vals]
        plt.plot(v_vals_constP, T_vals, "--k", label="P = constant ({} kPa)".format(round(P_const/1e
        plt.legend()
    else:
        plt.legend()
```

```
########### (a) Show the process on a T-v diagram: ################
m = 0.9  # kg
T = 300 + 273.15  # Kelvin
pressure = 0.8e6  # Pa
fluid = "Water"
final_temp = CP.PropsSI("T", "Q", 0, "P", pressure, "water")
## quality
q = 1/3
# const pressure process:
vol_state1 = 1 / CP.PropsSI("D", "T", T, "P", pressure, "water")
vol_state2 = 1 / CP.PropsSI("D", "P", pressure, "Q", q, "water")
state1 = [vol_state1, T]
state2 = [vol_state2, final_temp]
## use the following function:
plot_T_v_diagram(fluid,state1,state2,plot_const_press_line=True)
plot_p_v_diagram(fluid, [pressure,vol_state1], [pressure,vol_state2])
########### (b) final temperature ################
print("(b) Final temperature : {} °C".format(round(final_temp-273.15,2)))

########### (c) change in Volume ################
delV = m * (vol_state2 - vol_state1)
print("(c) Vol change: {} m³".format(round(delV,4)))
```





Skip to main content

```
(c) Vol change: -0.219 m³
```

Piston-cylinder: Isobaric- Isochoric process

Problem statement:

A piston-cylinder device contains steam initially at 200 C and 200 kPa. The steam is first cooled isobarically to saturated liquid, then isochorically until its pressure reaches 25 kPa.

1. Sketch the whole process on the P−vand T−v diagrams
2. Calculate the specific heat transfer in the whole process

Solution strategy

Identify the States: State 1: Initial state - T1=200 C and P1=200kPa State 2: After isobaric cooling to saturated liquid at P2=200kPa State 3: After isochoric cooling with V3=V2 and P3=25kPa.

Sketch the Process on P-v and T-v Diagrams: The process from State 1 to State 2 is a horizontal line on the P-v diagram and a downward slope on the T-v diagram. The process from State 2 to State 3 is a vertical line downward on the P-v diagram and also a downward slope on the T-v diagram.

Calculate the Specific Heat Transfer: Using the first law for a closed system: $\Delta u=q-w$ Where: Δu = Change in internal energy between states. q = Heat transfer. w = Work done. For the isobaric process (State 1 to State 2), $w=P(V2-V1)$. For the isochoric process (State 2 to State 3), w=0.

Thus, the total heat transfer for the whole process is: $q=\Delta u+w$

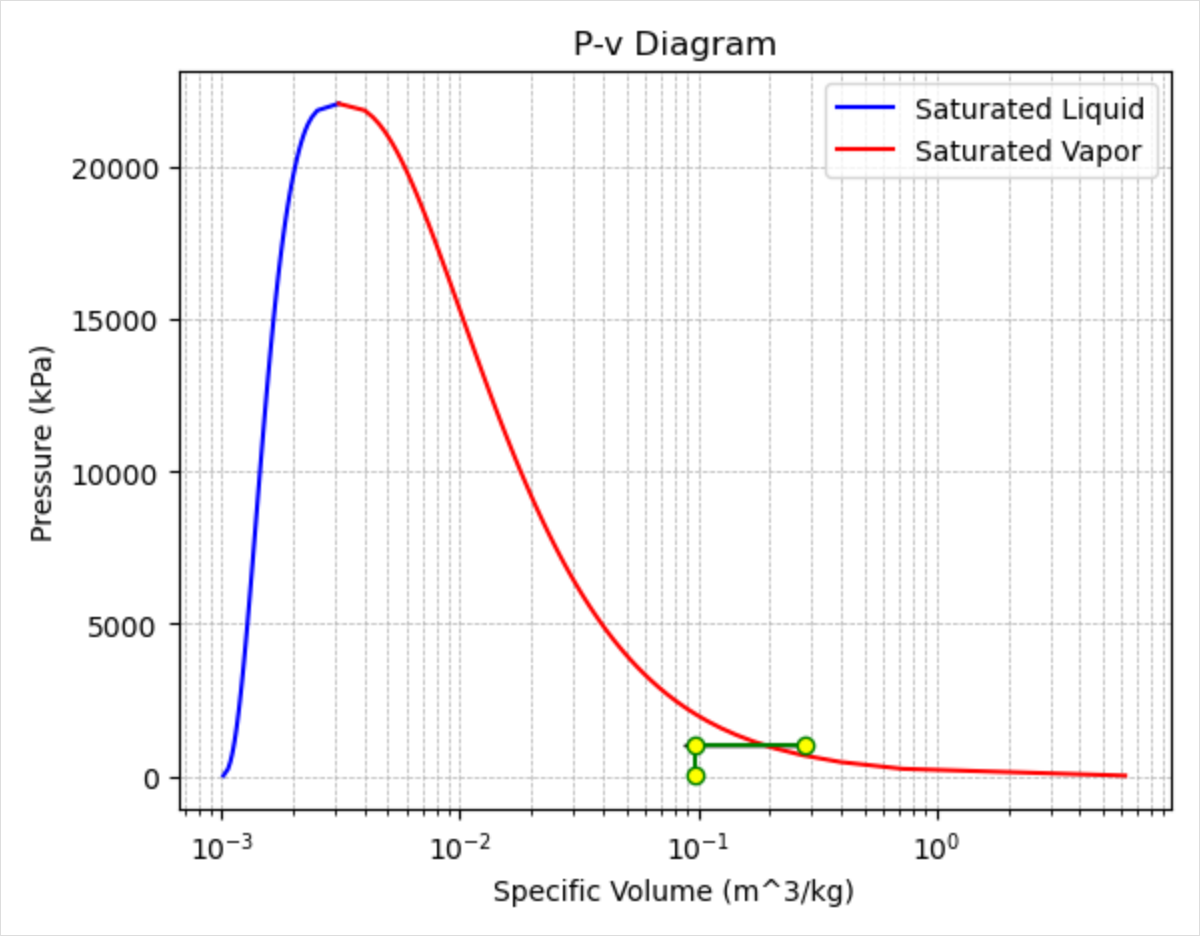
Let’s proceed with the code to sketch the diagrams and calculate the specific heat transfer:

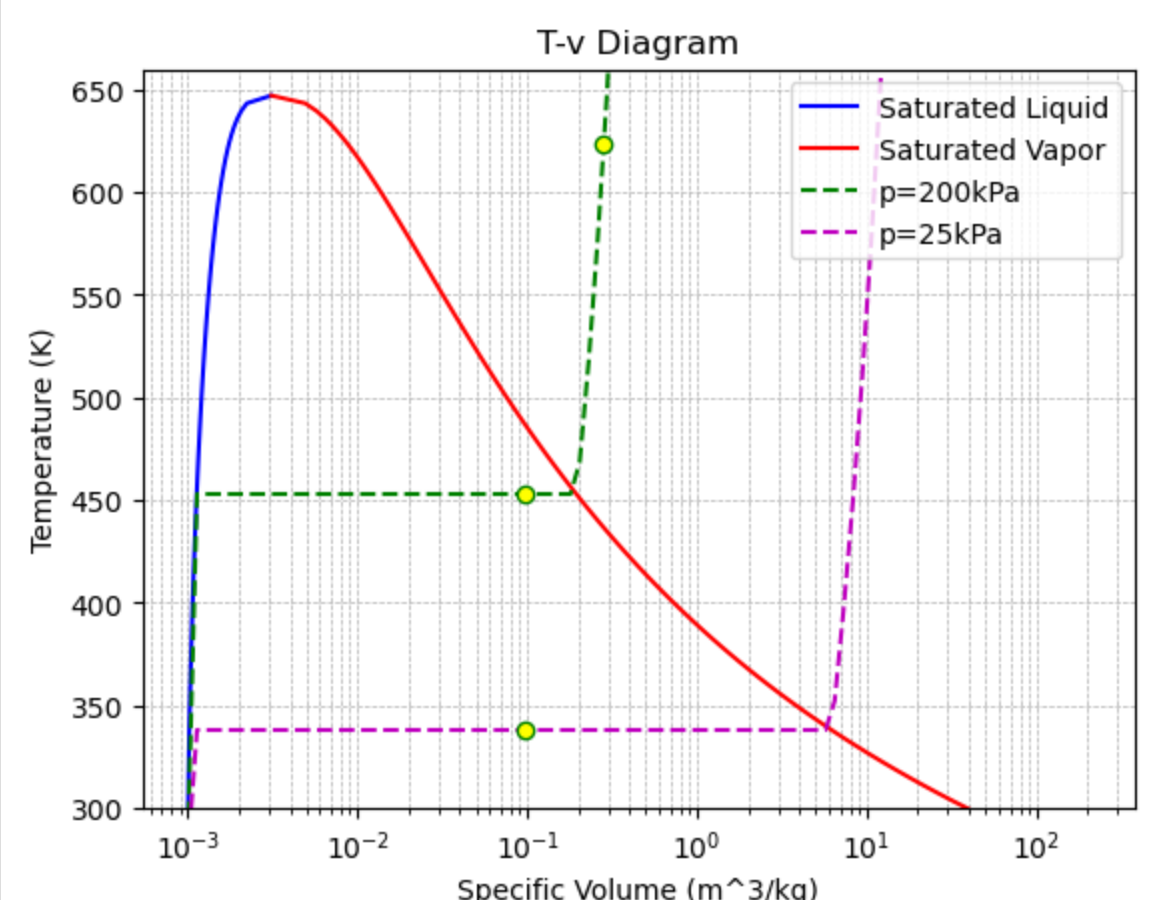
```python
import numpy as np
import matplotlib.pyplot as plt
import CoolProp.CoolProp as CP

# Given data
T1 = 350 + 273.15  # Initial temperature in K
P1 = 1000e3  # Initial pressure in Pa
P2 = 1000e3  # Pressure after isobaric cooling in Pa
P3 = 25e3  # Final pressure after isochoric process in Pa

# State 1 properties
v1 = 1 / CP.PropsSI("D", "T", T1, "P", P1, "Water")
u1 = CP.PropsSI("U", "T", T1, "P", P1, "Water")

# State 2 properties
T2 = CP.PropsSI("T", "P", P2, "Q", 0, "Water")  # Saturated liquid
v2 = 1 / CP.PropsSI("D", "P", P2, "Q", 0.5, "Water")
u2 = CP.PropsSI("U", "P", P2, "Q", 0, "Water")

# State 3 properties (isochoric)
v3 = v2
T3 = CP.PropsSI("T", "D", 1 / v3, "P", P3, "Water")
u3 = CP.PropsSI("U", "D", 1 / v3, "P", P3, "Water")

# Calculate work and heat transfer
w_12 = P1 * (v2 - v1)  # Isobaric process
w_23 = 0  # Isochoric process
q_12 = (u2 - u1) + w_12
q_23 = u3 - u2  # Isochoric process so no work
q_total = (q_12 + q_23)/1e3

print(f"Specific heat transfer for the whole process: {q_total:.2f} kJ/kg")
```

```
Specific heat transfer for the whole process: -2754.35 kJ/kg
```

Process diagrams:

```
# Critical properties
P_critical = CP.PropsSI("Pcrit", "Water")
T_critical = CP.PropsSI("Tcrit", "Water")
T_triple = CP.PropsSI("Ttriple","water")
# P-v diagram with dome
plt.figure()
pressures = np.linspace(CP.PropsSI("P", "Q", 0, "T", T3, "Water"), P_critical, 100)
v_liquid = [1 / CP.PropsSI("D", "Q", 0, "P", P, "Water") for P in pressures]
v_vapor = [1 / CP.PropsSI("D", "Q", 1, "P", P, "Water") for P in pressures]
plt.plot(v_liquid, pressures / 1e3, "b-", label="Saturated Liquid")
plt.plot(v_vapor, pressures / 1e3, "r-", label="Saturated Vapor")
plt.plot(
    [v1, v2, v3],
    np.array([P1, P2, P3]) / 1e3,
    "o-",
    color="green",
    markerfacecolor="yellow",
)
plt.arrow(
    v1, P1 / 1e3, v2 - v1, 0, head_width=5, head_length=0.01, fc="black", ec="black"
)
plt.xscale("log")
plt.xlabel("Specific Volume (m^3/kg)")
plt.ylabel("Pressure (kPa)")
plt.title("P-v Diagram")
plt.legend()
plt.grid(True, which="both", linestyle="--", linewidth=0.5)

# T-v diagram with dome
plt.figure()
temperatures = np.linspace(T_triple, T_critical, 100)
v_liquid_T = [1 / CP.PropsSI("D", "Q", 0, "T", T, "Water") for T in temperatures]
v_vapor_T = [1 / CP.PropsSI("D", "Q", 1, "T", T, "Water") for T in temperatures]
plt.plot(v_liquid_T, temperatures, "b-", label="Saturated Liquid")
plt.plot(v_vapor_T, temperatures, "r-", label="Saturated Vapor")

# Isobaric lines for T-v plot
v_range = np.logspace(np.log10(min(v_liquid_T)), np.log10(max(v_vapor_T)), 100)
T_200kPa = [CP.PropsSI("T", "D", 1 / v, "P", P1, "Water") for v in v_range]
T_25kPa = [CP.PropsSI("T", "D", 1 / v, "P", 25e3, "Water") for v in v_range]
plt.plot(v_range, T_200kPa, "g--", label="p=200kPa")
plt.plot(v_range, T_25kPa, "m--", label="p=25kPa")

plt.plot([v1, v2, v3], [T1, T2, T3], "o", color="green", markerfacecolor="yellow")
# plt.arrow(v1, T1, v2-v1, T2-T1, head_width=0.01, head_length=10, fc='black', ec='black')
# plt.arrow(v2, T2, 0, T3-T2, head_width=0.01, head_length=10, fc='black', ec='black')
plt.xscale("log")
plt.xlabel("Specific Volume (m^3/kg)")
plt.ylabel("Temperature (K)")
plt.title("T-v Diagram")
plt.ylim([300, 660])
plt.legend()
plt.grid(True, which="both", linestyle="--", linewidth=0.5)
plt.show()
```





Skip to main content

3. Ideal and Real Gases

This section covers concepts from Chapter 3 of “Introduction to Engineering Thermodynamics” by Prof. Claire Yu Yan.

Ideal gas law: Air

Imagine $1\ kg$ of air confined in a space of $1\ m^3$ volume. If the air is kept at room temperature, assuming ideal gas law applies to air in this case,

a) calculate air pressure in kPa?

b) what if the temperature increases to $100^\circ C$?

c) what if $1\ kg$ air is added to the same compartment?

d) what if the space within which the air in kept is compressed to decrease the volume to $0.8\ m^3$

Solution Approach for a)

the ideal gas law is an equation of state (EOS) which helps with predicting the state of a materiasl based on its properties Pressure (P), specifiv volume (v), and temperature (T) as following:

$Pv = RT$,which can also be written as
$PV = mRT$ where m is the mass of the material

to calculate pressure:
$P = mRT/V$

```
#defining variables
m = 1 #mass of air in kg
R = 0.287 #air gas constant in kJ/kg.K
V = 1 #volume of air in m3
T = 25 + 273.15 #room temperature in K

# using ideal gas equation of state to calculate pressure
P = m * R * T / V #air pressure in kPa

print('The air pressure is:', f"{P:.1f}", 'kPa')
```

```
The air pressure is: 85.6 kPa
```

Solution Approach for b)

for part b, the temperature increases to $100^\circ C$. As a result, an increase in pressure is expected

Skip to main content

```
T = 100 + 273.15 #increased room temperature in K
P = m * R * T / V #air pressure in kPa
print('The air pressure is:', f"{P:.1f}", 'kPa')
```

```
The air pressure is: 107.1 kPa
```

Solution Approach for c)

for part c, the number of air molecules increases. As a result, an increase in pressure is expected.

```
m = 2 #increased mass of air in kg
P = m * R * T / V #air pressure in kPa
print('The air pressure is:', f"{P:.1f}", 'kPa')
```

```
The air pressure is: 214.2 kPa
```

Solution Approach for d)

for part c, the volume of air decreases resulting in more air molecules per unit volume. As a result, an increase in pressure is expected.

```
V = 0.8 #compressed volume of air in m3
P = m * R * T / V #air pressure in kPa
print('The air pressure is:', f"{P:.1f}", 'kPa')
```

```
The air pressure is: 267.7 kPa
```

Thermo-mechanical Equilibrium: Partitions in a Box

consider a box of $5\ m^3$ volume composed of three isolated compartments A, B and C containing air in three diffrent states. Part A contains $0.5\ kg$ air at $200\ kPa$. Part B measure $25^\circ C$ in temperature, $100\ kPa$ is pressure and occupies $1\ m^3$ of space. Finally, part C contains $2\ kg$ of the same materials at a temperature of $15^\circ C$ and a pressure of $50\ kPa$.

Assuming ideal gas law for air in these states, calculate:

a) volume of air in partition C

Skip to main content

c) mass of air in partition B

d) temperature of air in partition A in $^\circ C$

e) now imagine all the seperators break and air in three compartments mix together. What would be the final pressure if the box is let to get in thermal stability with room temperature?

Solution Approach for a)

ideal gas equation can be used as EOS since the assumption is valid as per the statement, so

$PV = mRT$
$V = mRT/P$

```
#defining variables and looking up tables
R = 0.287 #air gas constant in kJ/kg.K
m_C = 2 #mass of air in partition c in kg
T_C = 15 + 273.15 #temperature of air in partc in K
P_C = 50 #pressure of air in partition c in kPa

#using ideal gas law equation of state
V_C = m_C * R * T_C / P_C #volume of partition c in m3

print('The air volume in partition C is:', f"{V_C:.1f}", 'm3')
```

```
The air volume in partition C is: 3.3 m3
```

Solution Approach for b)

The box is composed of three partitions, therefore:

$V_A + V_B + V_C = 5\, m^3$
$V_A = 5 - V_B - V_C$

```
V_B = 1 #volume of partition B in m3
V_A = 5 - V_B - V_C
print('The air volume in partition A is:', f"{V_A:.3f}", 'm3')
```

```
The air volume in partition A is: 0.692 m3
```

Solution Approach for c)

Skip to main content

$m = PV/RT$

```
#defining state variables
P_B = 100 #pressure of air in partition b in kPa
V_B = 1 #volume of partition b in m3
T_B = 25 + 273.15 #temperature of air in partition b in K

#using ideal gas law equation of state
m_B = P_B * V_B / (R * T_B) #mass of air in partiton b in kg
print('The mass of air in partition B is:', f"{m_B:.1f}", 'kg')
```

```
The mass of air in partition B is: 1.2 kg
```

Solution Approach for d)

from ideal gas law

$T = PV/mR$

```
#defining state variables
P_A = 200 #pressure of air in partition a in kPa
m_A = 0.5 #mass of air in partiton a in kg

#using ideal gas law equation of state
T_A = P_A * V_A / (m_A * R) #temperature of air in partition a in K

#converting from Kelvins to Celsius
T_AC = T_A - 273.15 #temperature of air in partition a in C
print('The temperature of air in partition A is:', f"{T_AC:.1f}", 'Degrees Celcius')
```

```
The temperature of air in partition A is: 691.4 Degrees Celcius
```

Solution Approach for e)

referring to ideal gas law equation of state,

$P = mRT/V$

The temperature would be in $25^{\circ}C$ since the box is in thermal stability with room temperature.

```
#defining/calculating state variables
T = 25 + 273.15 #final temperature of air in K
m = m_A + m_B + m_C #final mass of the mixture
V = 5 #total volume of the box in m3

#using ideal gas law equation of state
P = m * R * T / V #final pressure of the mixture in kPa

print('The final pressure of mixed air is:', f"{P:.1f}", 'kPa')
```

```
The final pressure of mixed air is: 62.8 kPa
```

Piston-cylinder system for Air

Consider $2\,kg$ air at $200\,kPa$ and $25^{\circ}C$ stored in a cylinder-piston system which is in thermal equilibrium with its surrounding. Assuming air at this state as ideal gas,

a) calculate how much space does air occupy in this state

b) if the piston is moved to compress air to half its initial volume at constant temperature, calculate the final air pressurea

c) plot the process on P-V, P-T and T-V diagrams.

Solution Approach for a)

ideal gas equation can be used as EOS since the assumption is valid as per the statement, so

$PV = mRT$
$V = mRT/P$

```
#defining variables and looking up tables
R = 0.287 #air gas constant in kJ/kg.K
m = 2 #mass of air in kg
T = 25 + 273.15 #temperature of air in K
P = 200 #pressure of airc in kPa

#using ideal gas law equation of state
V_1 = m * R * T / P #volume of air in m3

print('The air volume is:', round(V_1,3), 'm3')
```

```
The air volume is: 0.856 m3
```

Skip to main content

The box is compressed to half its initial volume therefore $V_2 = V_1/2$ and temperature remains constant as per the question's statement.

Ideal gas equation solved for pressure (P) is to be used to calculate pressure.

$P = mRT/V$

```
V_2 = V_1 / 2 #volume of air after compression

#using ideal gas law equation of state
P_2 = m * R * T / V_2 #final pressure of the mixture in kPa
print('Secondary air pressure is:', P_2, 'kPa')
```

```
Secondary air pressure is: 400.0 kPa
```

Solution Approach for c)

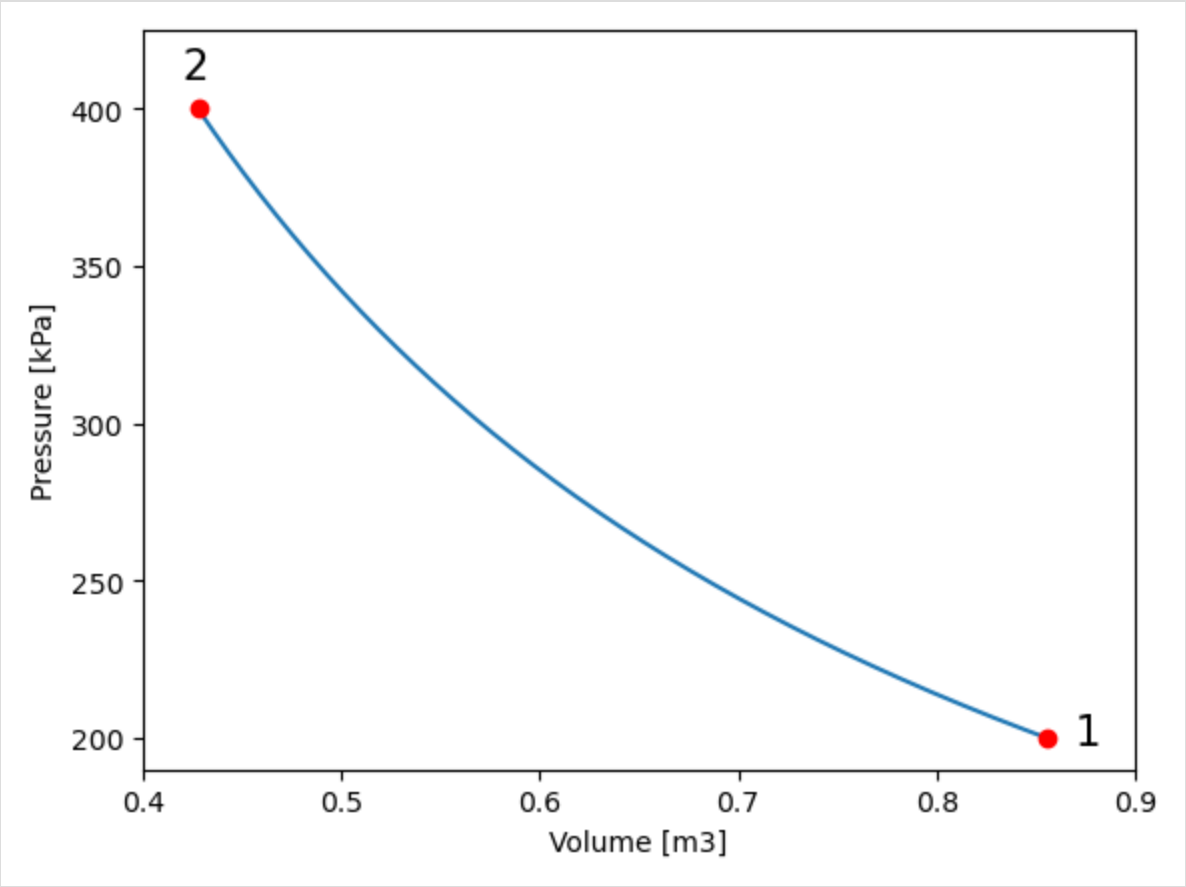
for a P-v diagram, the ideal gas equation is to be used in a form in which pressure(P) and volume(v) are the variables calculated one based on the other one.

$P = mRT(1/V)$

where mRT would be a constant value

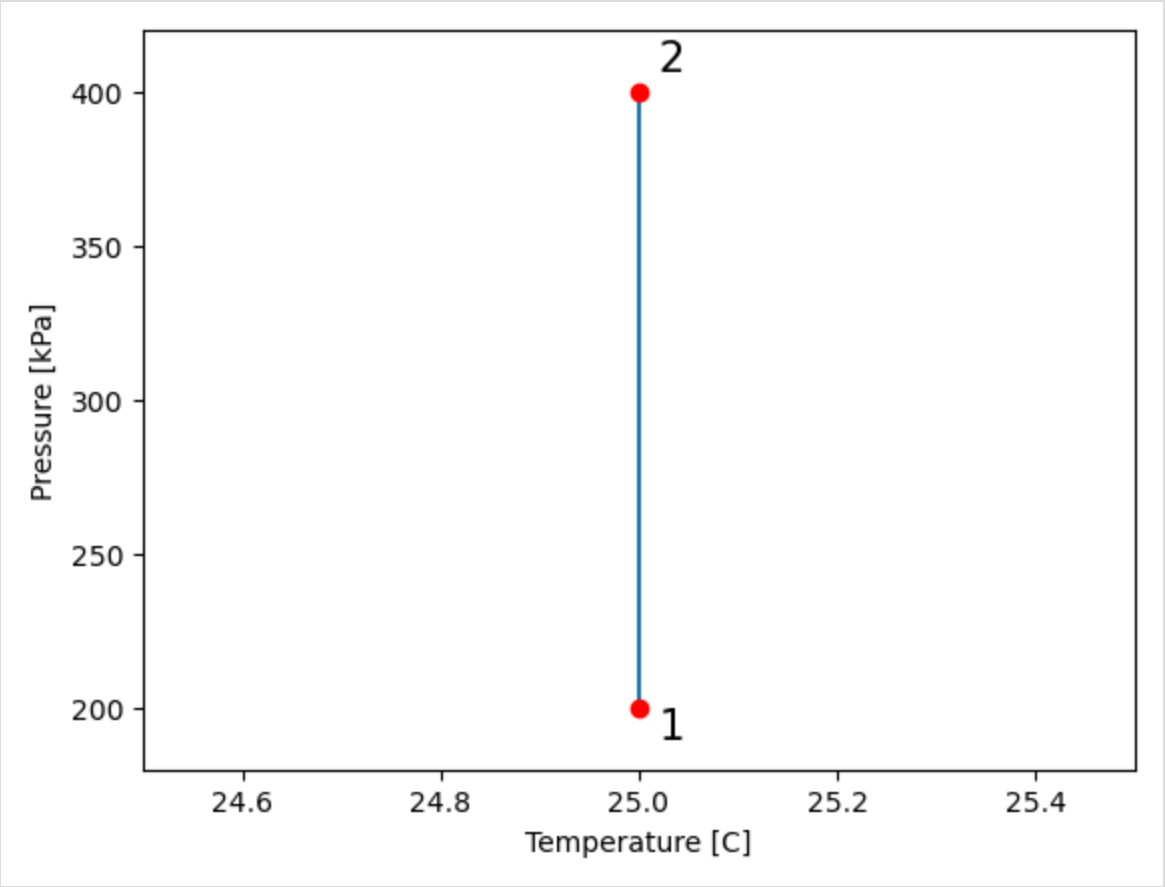
```
#import the libraries we'll need
import numpy as np
import matplotlib.pyplot as plt

#building a range for volume (v) so that pressure (P) is calculated based upon
V_max = V_1 #maximum amount for volume in the process
V_min = V_2 #minimum amount for volume in the process
V_vals = np.linspace(V_min, V_max, 1000)  # define an array of values for volume (v)

#calculating pressure (P) for the array of volume values (V_vals)
P_vals = m * R * T / V_vals

plt.plot(V_vals, P_vals)  # plot pressure vs. volume
plt.ylabel("Pressure [kPa]")  # give y axis a label
plt.xlabel("Volume [m3]")  # give x axis a label


#add-ons to illustrate process path
plt.xlim(0.4, 0.9)
plt.ylim(190, 425)
plt.text(0.87, 198, '1', fontsize = 15)
plt.text(0.42, 410, '2', fontsize = 15)
plt.plot(V_vals[0], P_vals[0], 'ro')
plt.plot(V_vals[-1], P_vals[-1], 'ro')
```

```
[<matplotlib.lines.Line2D at 0x7f6a345b4bb0>]
```



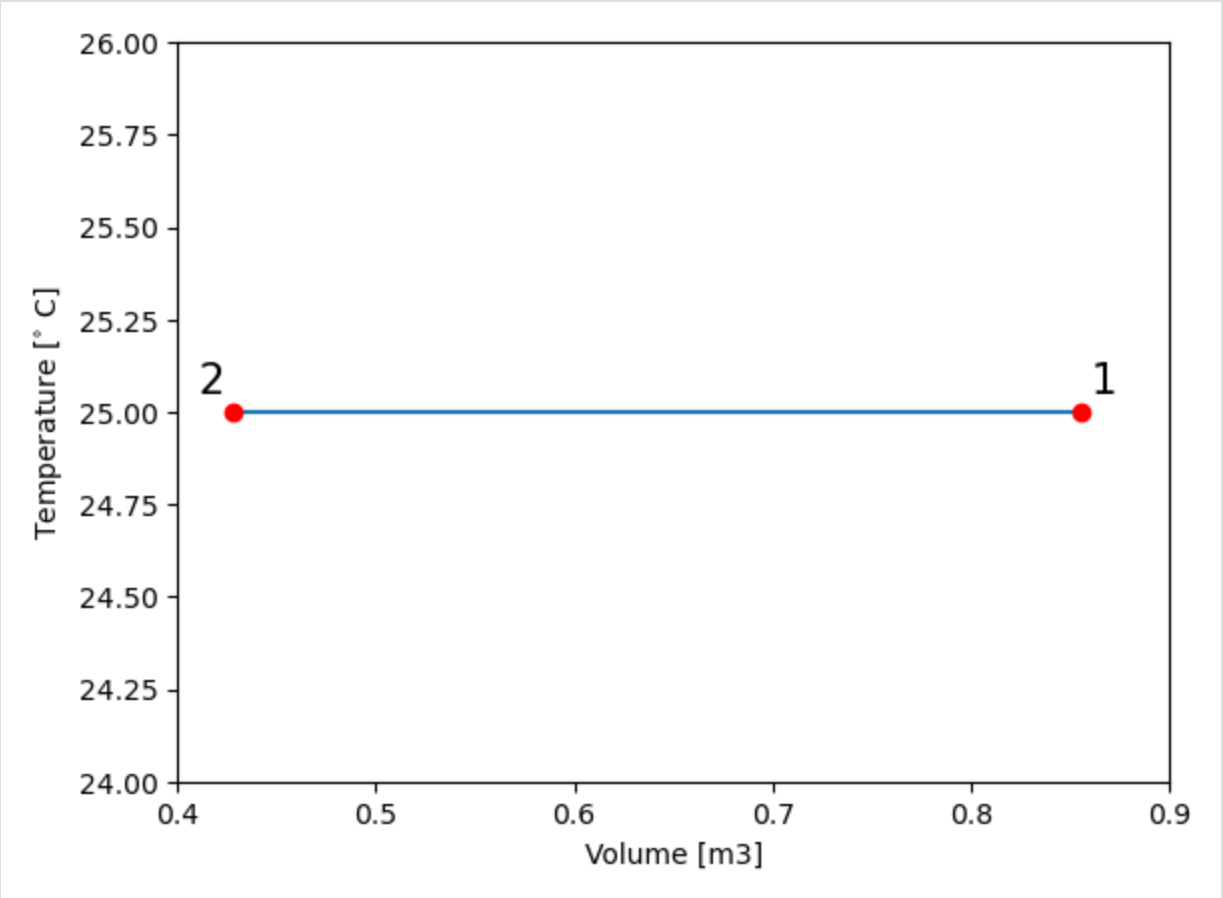

```
#building temperature range to plot P-T and V_T diagrams
T_C = T - 273.15 #process temperature in Celsius
T_vals = np.linspace(T_C, T_C, 1000)  # define an array of values for temperature (T)
plt.plot(T_vals, P_vals)  # plot pressure vs. temperature
plt.ylabel("Pressure [kPa]")  # give y axis a label
plt.xlabel("Temperature [C]")  # give x axis a label

#add-ons to illustrate process path
plt.xlim(24.5, 25.5)
plt.ylim(180, 420)
plt.text(25.02, 190, '1', fontsize = 15)
plt.text(25.02, 407, '2', fontsize = 15)
plt.plot(T_vals[0], P_vals[0], 'ro')
plt.plot(T_vals[-1], P_vals[-1], 'ro')
```

```
[<matplotlib.lines.Line2D at 0x7f6a33cac280>]
```




```
plt.plot(V_vals,T_vals)  # plot volume vs. temperature
plt.ylabel("Temperature [$^{\circ}$ C]")  # give y axis a label
plt.xlabel("Volume [m3]")  # give x axis a label

#add-ons to illustrate process path
plt.xlim(0.4, 0.9)
plt.ylim(24, 26)
plt.text(0.86, 25.05, '1', fontsize = 15)
plt.text(0.41, 25.05, '2', fontsize = 15)
plt.plot(V_vals[0], T_vals[0], 'ro')
plt.plot(V_vals[-1], T_vals[-1], 'ro')
```

```
[<matplotlib.lines.Line2D at 0x7f6a33ca2a00>]
```



Isobaric-Isochoric process: Nitrogen as Ideal gas

Consider $1\ kg$ of Nitrogen stored in a piston-cylinder system sitting at atmospheric pressure and temperature ($101\ kPa$ and $25^{\circ}C$). The system undergoes a thermodynamic cycle where its pressure is doubled through an isothermal process. Following the isothermal compression comes an isobaric expansion followed by an isochoric process to get Nitrogen to its initial state at $101\ kPa$ and $25^{\circ}C$. Assuming Nitrogen to be ideal gas, plot P-V, P-T, and T-V diagrams for this process.

Solution Approach

Let's associate each state with a number to be able to progess in an organized way.

The initial state being (1) where

$P_1 = 101\ kPa$ and $T_1 = 25^{\circ}C$

Followed by an isothermal process where the pressure is doules to reach state (2) where

$P_2 = 2P_1$ and $T_2 = T_1 = 25^{\circ}C$

Skip to main content

Followed by an isobaric process to reach point (3) where

$P_3 = 2P_2$

and

$V_3 = V_1$

because the following process from (3) back to (1) is isochoric where the volume keeps constant.

To Calculate V_1 the ideal gas equation of state shall be used where

$V = mRT/P$

```
#defining variables and looking up tables
R = 0.2968 #Nitrogen gas constant in kJ/kg.K
m = 1 #mass of Nitrogen in kg
T_1 = 25 + 273.15 #temperature of Nitrogen in K
P_1 = 101 #pressure of Nitrogen in kPa

#using ideal gas law equation of state
V_1 = m * R * T_1 / P_1 #volume of Nitrogen in m3

print('The volume of Nitrogen at its initial state is:', f"{V_1:.3f}", 'm3')
```

```
The volume of Nitrogen at its initial state is: 0.876 m3
```

Then Nitrogen goes through an isothermal compression to double its pressure.

given mass(m), gas constant (R) and temperature (T) are constant,

$P_2V_2 = P_1V_1 = mRT_1 (or T_2)$

therefore

$V_2 = P_1V_1/P_2$

where

$P_2 = 2P_1$

therfore

$V_2 = V_1/2$

```
#calculating variables at the second state
P_2 = 2 * P_1 #pressure of Nitrogen in kPa
T_2 = T_1 #temperature of Nitrogen in K
V_2 = V_1 / 2 #volume of Nitrogen in m3 at the second state
print('The volume of Nitrogen at its second state is:', f"{V_2:.3f}", 'm3')
```

```
The volume of Nitrogen at its second state is: 0.438 m3
```

To calculate temperature at the third state, ideal gas equation of state is to be used as

Skip to main content

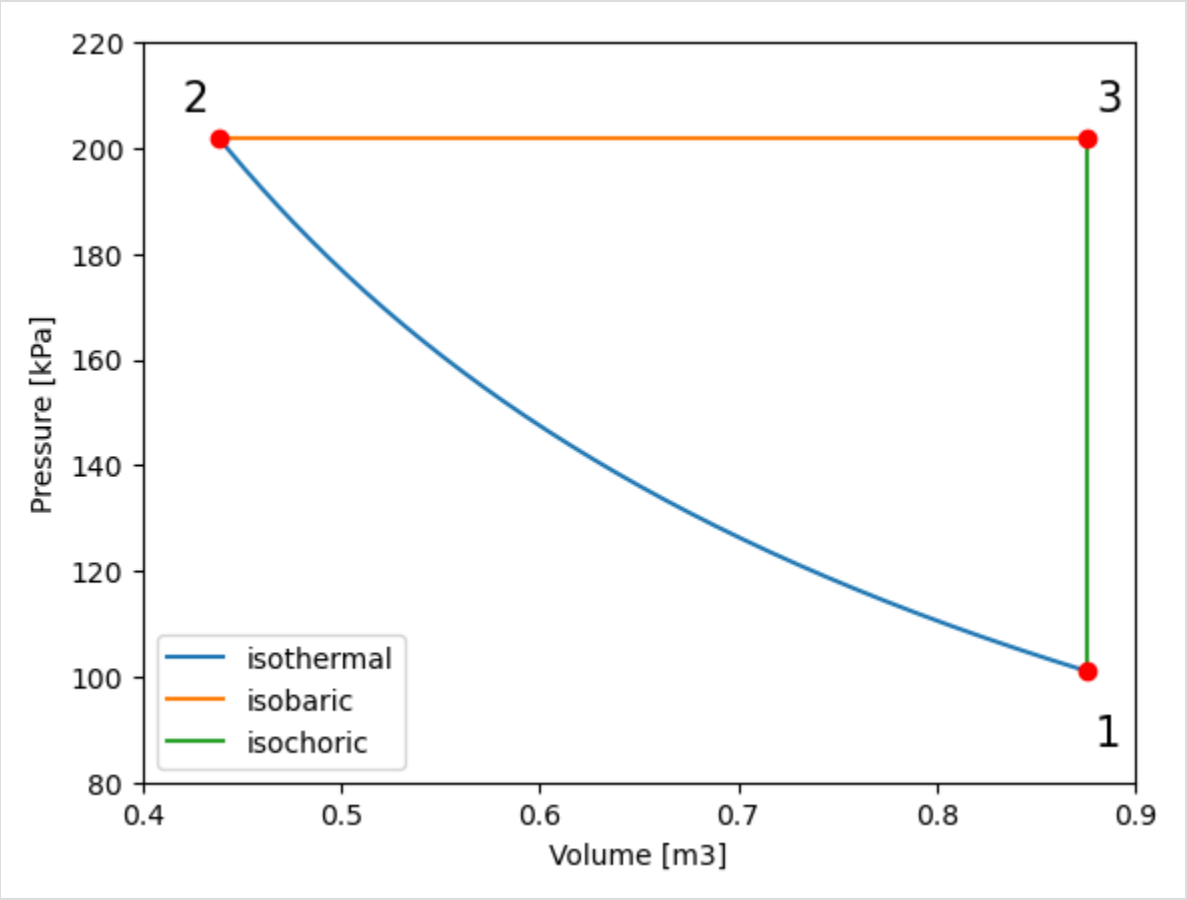
$T_3 = P_3V_3/mR$

```
##calculating variables at the third state
P_3 = P_2 #pressure of Nitrogen in kPa
V_3 = V_1 #volume of Nitrogen in m3 at the third state

#using ideal gas law equation of state
T_3 = P_3 * V_3 / (m * R) #temperature of air in partition a in K
print('The temperature of Nitrogen at the third state is:', T_3, 'K')
```

```
The temperature of Nitrogen at the third state is: 596.3 K
```

```
##plotting P_V
#import the libraries we'll need
import numpy as np
import matplotlib.pyplot as plt

#defining arrays of V values for 1-2
V_vals12 = np.linspace(V_1, V_2, 1000)  # define an array of values for volume (v) for the process 1
#calculating pressure (P) for the array of volume values (V_vals12)
P_vals12 = m * R * T_1 / V_vals12

#defining arrays of V values for 2-3
V_vals23 = np.linspace(V_2, V_3, 1000)  # define an array of values for volume (v) for the process 2
#associated constant pressure for the process 2-3
P_vals23 = np.linspace(P_2, P_3, 1000)

#defining arrays of P values for 3-1
P_vals31 = np.linspace(P_3, P_1, 1000)  # define an array of values for pressure (P) for the process
#associated constant volume for the process 3-1
V_vals31 = np.linspace(V_3, V_1, 1000)

plt.plot(V_vals12, P_vals12,label='isothermal')  # plot pressure vs. volume
plt.plot(V_vals23, P_vals23,label='isobaric')
plt.plot(V_vals31, P_vals31,label='isochoric')
plt.legend()

plt.ylabel("Pressure [kPa]")  # give y axis a label
plt.xlabel("Volume [m3]")  # give x axis a label

#add-ons to illustrate process path
plt.xlim(0.4, 0.9)
plt.ylim(80, 220)
plt.text(0.88, 87, '1', fontsize = 15)
plt.text(0.42, 207, '2', fontsize = 15)
plt.text(0.88, 207, '3', fontsize = 15)
plt.plot(V_vals12[0], P_vals12[0], 'ro')
plt.plot(V_vals23[0], P_vals23[0], 'ro')
plt.plot(V_vals31[0], P_vals31[0], 'ro')
```

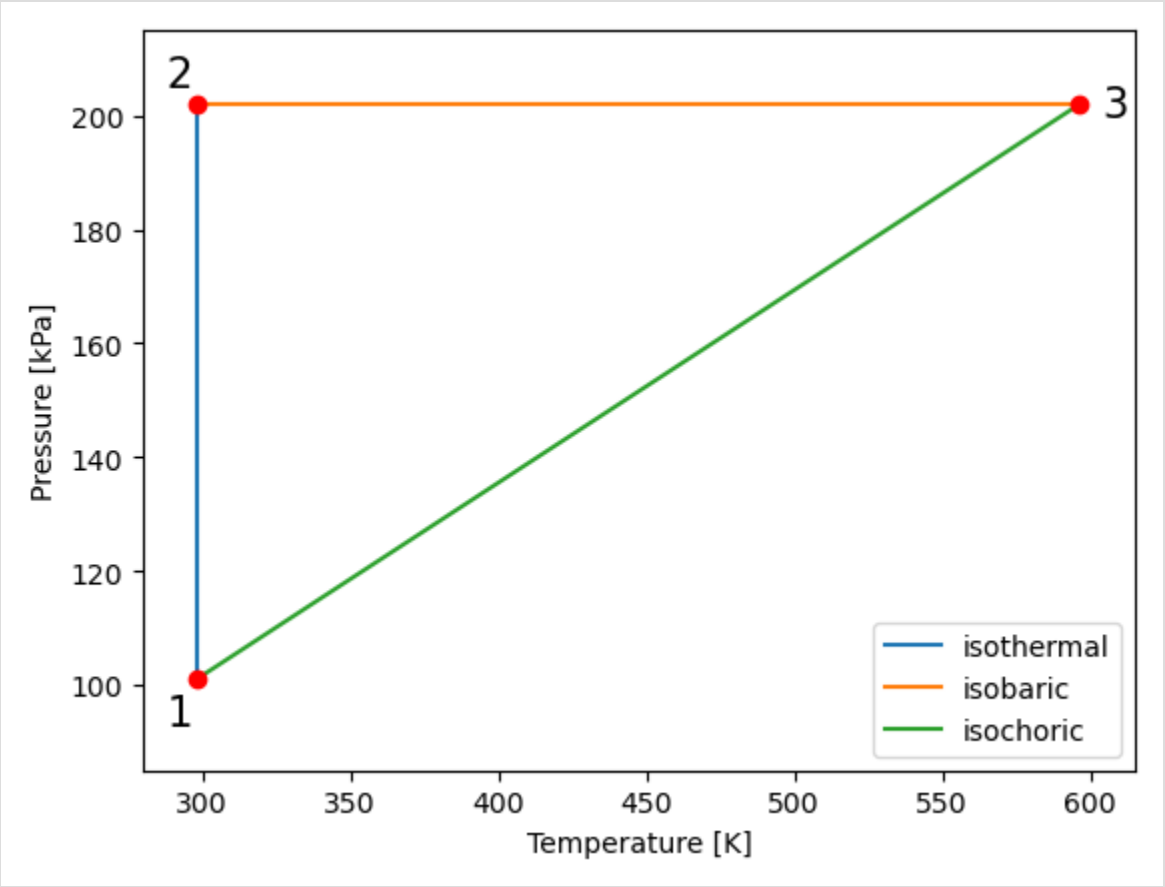
```
[<matplotlib.lines.Line2D at 0x7f28f6191c40>]
```




```
##plotting P_T
#defining arrays of T values for 1-2
T_vals12 = np.linspace(T_1, T_2, 1000)  # define an array of values for temperature (T) for the proc

#defining arrays of T values for 2-3
T_vals23 = np.linspace(T_2, T_3, 1000)  # define an array of values for temperature (T) for the proc

#defining arrays of T values for 3-1
T_vals31 = np.linspace(T_3, T_1, 1000)  # define an array of values for temperature (T) for the proc

plt.plot(T_vals12, P_vals12,label='isothermal')  # plot pressure vs. volume
plt.plot(T_vals23, P_vals23,label='isobaric')
plt.plot(T_vals31, P_vals31,label='isochoric')
plt.legend()

plt.ylabel("Pressure [kPa]")  # give y axis a label
plt.xlabel("Temperature [K]")  # give x axis a label

#add-ons to illustrate process path
plt.xlim(280, 615)
plt.ylim(85, 215)
plt.text(288, 93, '1', fontsize = 15)
plt.text(288, 205, '2', fontsize = 15)
plt.text(604, 200, '3', fontsize = 15)
plt.plot(T_vals12[0], P_vals12[0], 'ro')
plt.plot(T_vals23[0], P_vals23[0], 'ro')
```

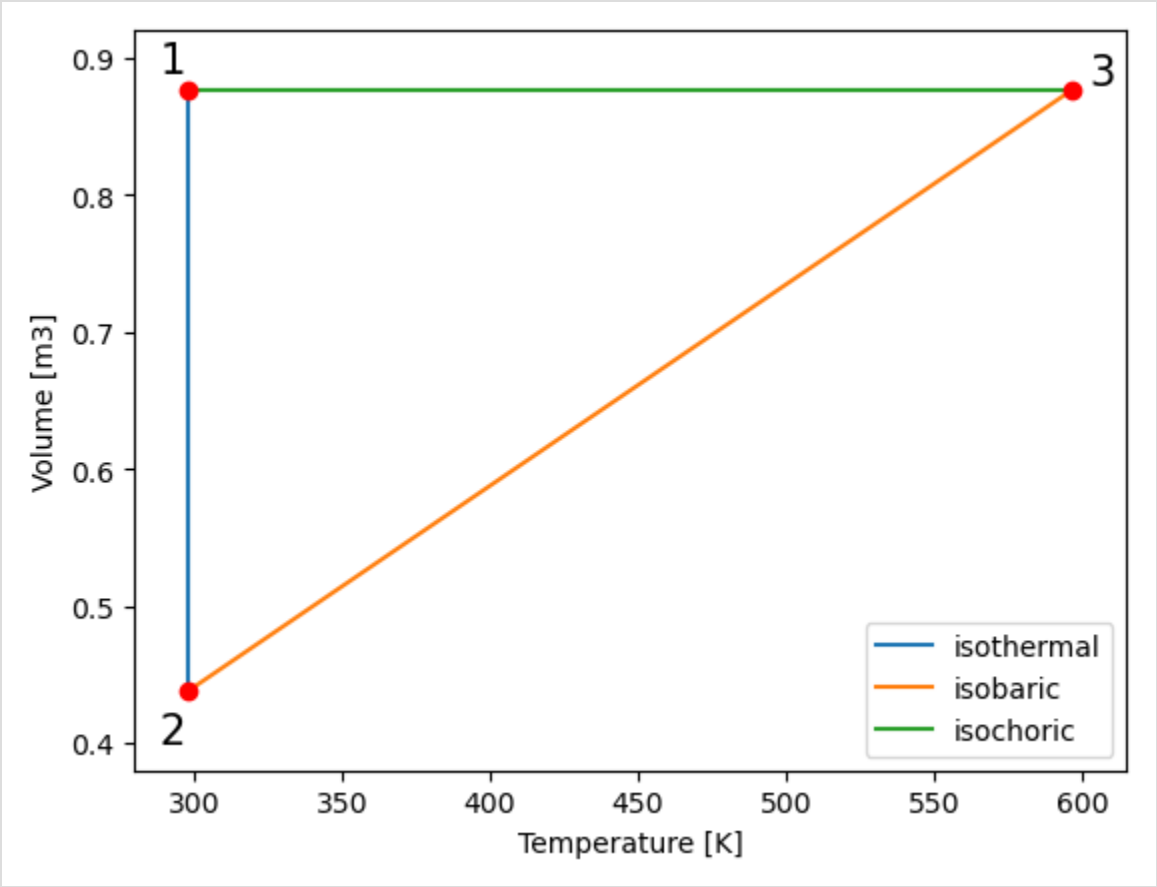
Skip to main content

```
[<matplotlib.lines.Line2D at 0x7f28f58a2d30>]
```




```
##plotting T_V

plt.plot(T_vals12, V_vals12,label='isothermal')  # plot pressure vs. volume
plt.plot(T_vals23, V_vals23,label='isobaric')
plt.plot(T_vals31, V_vals31,label='isochoric')
plt.legend()

plt.ylabel("Volume [m3]")  # give y axis a label
plt.xlabel("Temperature [K]")  # give x axis a label

#add-ons to illustrate process path
plt.xlim(280, 615)
plt.ylim(0.38, 0.92)
plt.text(288, 0.89, '1', fontsize = 15)
plt.text(288, 0.40, '2', fontsize = 15)
plt.text(602, 0.88, '3', fontsize = 15)
plt.plot(T_vals12[0], V_vals12[0], 'ro')
plt.plot(T_vals23[0], V_vals23[0], 'ro')
plt.plot(T_vals31[0], V_vals31[0], 'ro')
```

```
[<matplotlib.lines.Line2D at 0x7f28f57c7a60>]
```



Lee - Kesler Compressibility factor

The following reference is the paper upon which the Lee-Kesler compressibility factor is generated.

[Reference for Lee-Kesler compressibility factor]

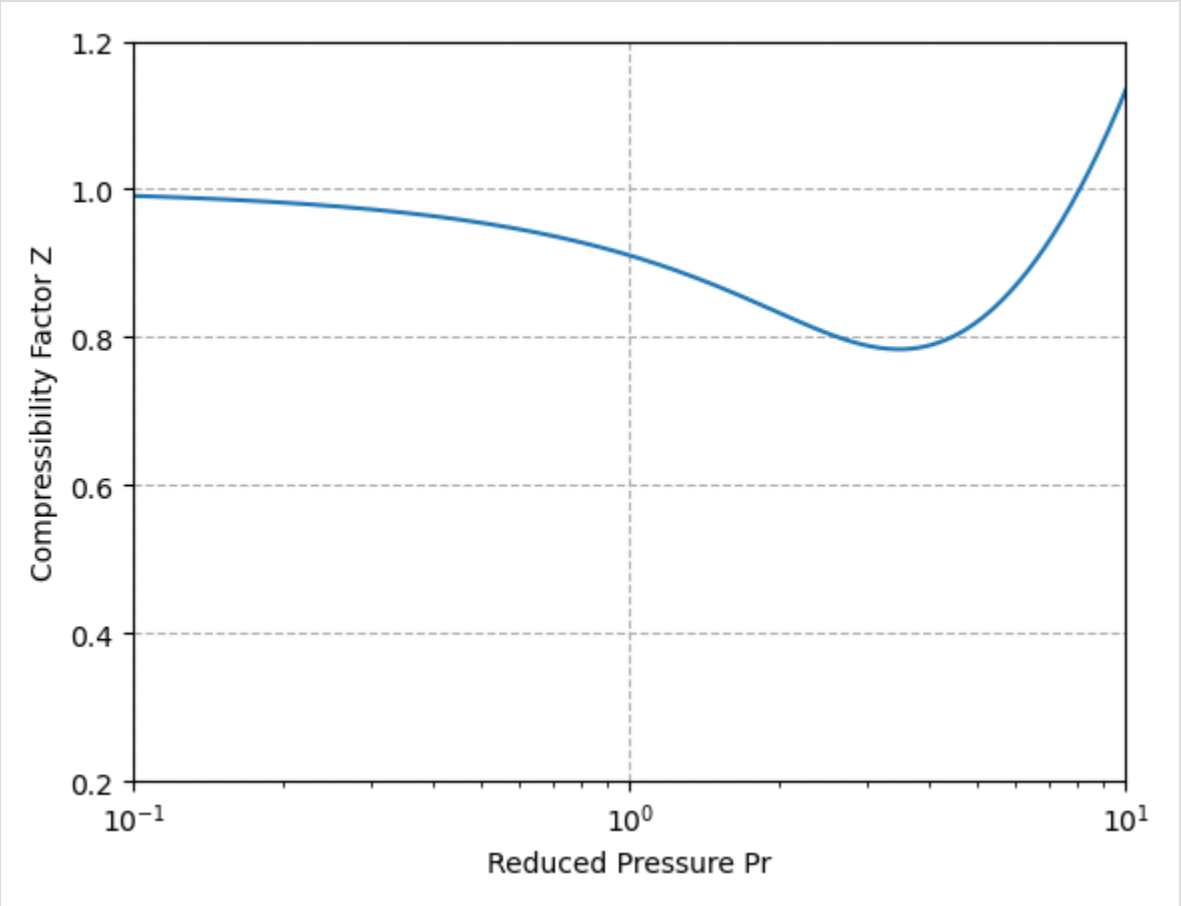
a)Try to reproduce a python code to regenerate the compressibility factor curve based on reduced pressure (P_r) for reduced temperature (T_r) equal to 1.5.

b) Calculate the compressibility factor for $T_r = 1.5$ and $P_r = 3$

Skip to main content

```
#importing required library
import numpy as np
import matplotlib.pyplot as plt

#introducing P_r and T_r
T_r = 1.5

#introducing constants
b1 = 0.1181193
b2 = 0.265728
b3 = 0.154790
b4 = 0.030323
c1 = 0.0236744
c2 = 0.0186984
c3 = 0
c4 = 0.042724
d1 = 0.155488E-4
d2 = 0.623689E-4
betha = 0.65392
gamma = 0.060167

B = b1 - b2/T_r - b3/T_r**2 - b4/T_r**3
C = c1 - c2/T_r + c3/T_r**3
D = d1 + d2/T_r

#V_r in an array structure
# an array of V_r is to be built so that Z is calculated based upon. Otherwise, the equation can't b
V_r = np.logspace(-0.9, 1.65, 10000)
Z_array = 1 + B/V_r + C/V_r**2 + D/V_r**5 + c4*(betha + gamma/V_r**2)*np.exp(-gamma/V_r**2)/(T_r**3
P_r_array = Z_array * T_r / V_r #calculating P_r based on the array built for V_r

#Plotting
plt.plot(P_r_array,Z_array)
plt.xlim(0.1, 10)
plt.ylim(0.2, 1.2)
plt.xscale('log')
plt.grid(ls='--')
plt.xlabel('Reduced Pressure Pr')
plt.ylabel('Compressibility Factor Z')
```

```
Text(0, 0.5, 'Compressibility Factor Z')
```




```
P_r = 3

#finding the index of the array element in P_r_array which is closest to the desired P_r value
difference_array = np.absolute(P_r_array-P_r)
index = difference_array.argmin()
Z = Z_array[index]
print('The compressibility factor value for P_r =',P_r, 'and T_r =', T_r, 'is', f"{Z:.3f}")
```

```
The compressibility factor value for P_r = 3 and T_r = 1.5 is 0.789
```

Ideal gas assumption for Water

Imagine $1\ kg$ of water vapor at $2\ MPa$ and $400^{\circ}C$. Calculate its volume based on the following. Calculate the error in percents.

a) thermodynamic tables using coolprop

b) ideal gas assumption

Skip to main content

d) pinpoint the water at this state on its phase diagram and monitor the ideal gas assumption error based on the distance from the triple point

Solution Approach for a)

```
#importing the required library
import numpy as np
import CoolProp.CoolProp as CP
P = 2E+6 #pressure of wator vapor in Pa
T = 400 + 273.15 #water vapor temperature in K
m = 1 #mass of water vapor in kg
D = CP.PropsSI('D', 'P', P, 'T', T, 'Water') #calculating density using coolprop kg/m
V = m / D #volume occupied m3
print('The volume of water wapor based on thermodynamic tables is',f"{V:.3f}",'m3')

## this value is treated as reference for error calculation since it's based on experiments
V_ref = V
```

```
The volume of water wapor based on thermodynamic tables is 0.151 m3
```

Solution Approach for b)

To Calculate V the ideal gas equation of state shall be used where

$V = mRT/P$

```
#introducing constant R
R = 0.4615 #Steam gas constant in kJ/kg.K
P_kpa = P / 1000 #pressure need to be in KPa to be consistent in ideal gas equation
V = m * R * T / P_kpa
print('The volume of water wapor based on ideal gas correlation is',f"{V:.3f}",'m3')

#calculating error
E = np.abs(V-V_ref)/V_ref * 100
print('The error based on ideal gas correlation is',f"{E:.3f}",'%')
```

```
The volume of water wapor based on ideal gas correlation is 0.155 m3
The error based on ideal gas correlation is 2.721 %
```

Solution Approach for c)

Skip to main content

```
#Introducing critical values
P_crit = 22.06 #critical pressure for water in MPa
T_crit = 647.1 #critical temperature for water in k

#calculating reduced pressure and temperature
P_r = P / 22.06E+6 #reduced pressure
T_r = T / T_crit  #reduced temperature

#now the code for Question#5 in this chapter is used to calculate compressibility factor (Z)
#importing required library
import numpy as np
import matplotlib.pyplot as plt

#introducing constants
b1 = 0.1181193
b2 = 0.265728
b3 = 0.154790
b4 = 0.030323
c1 = 0.0236744
c2 = 0.0186984
c3 = 0
c4 = 0.042724
d1 = 0.155488E-4
d2 = 0.623689E-4
betha = 0.65392
gamma = 0.060167

B = b1 - b2/T_r - b3/T_r**2 - b4/T_r**3
C = c1 - c2/T_r + c3/T_r**3
D = d1 + d2/T_r

#V_r in an array structure
# an array of V_r is to be built so that Z is calculated based upon. Otherwise, the equation can't b
V_r = np.logspace(-0.9, 1.65, 10000)
Z_array = 1 + B/V_r + C/V_r**2 + D/V_r**5 + c4*(betha + gamma/V_r**2)*np.exp(-gamma/V_r**2)/(T_r**3
P_r_array = Z_array * T_r / V_r #calculating P_r based on the array built for V_r

#finding the index of the array element in P_r_array which is closest to the desired P_r value
difference_array = np.absolute(P_r_array-P_r)
index = difference_array.argmin()
Z = Z_array[index]
print('The compressibility factor value for P_r =',f"{P_r:.3f}", 'and T_r =', f"{T_r:.3f}", 'is', f'
```

```
The compressibility factor value for P_r = 0.091 and T_r = 1.040 is 0.973
```

Looking at compressibility factor,

$$Z = Pv/RT = PV/mRT$$

rearranging for V,

$$V = ZmRT/P$$

Skip to main content

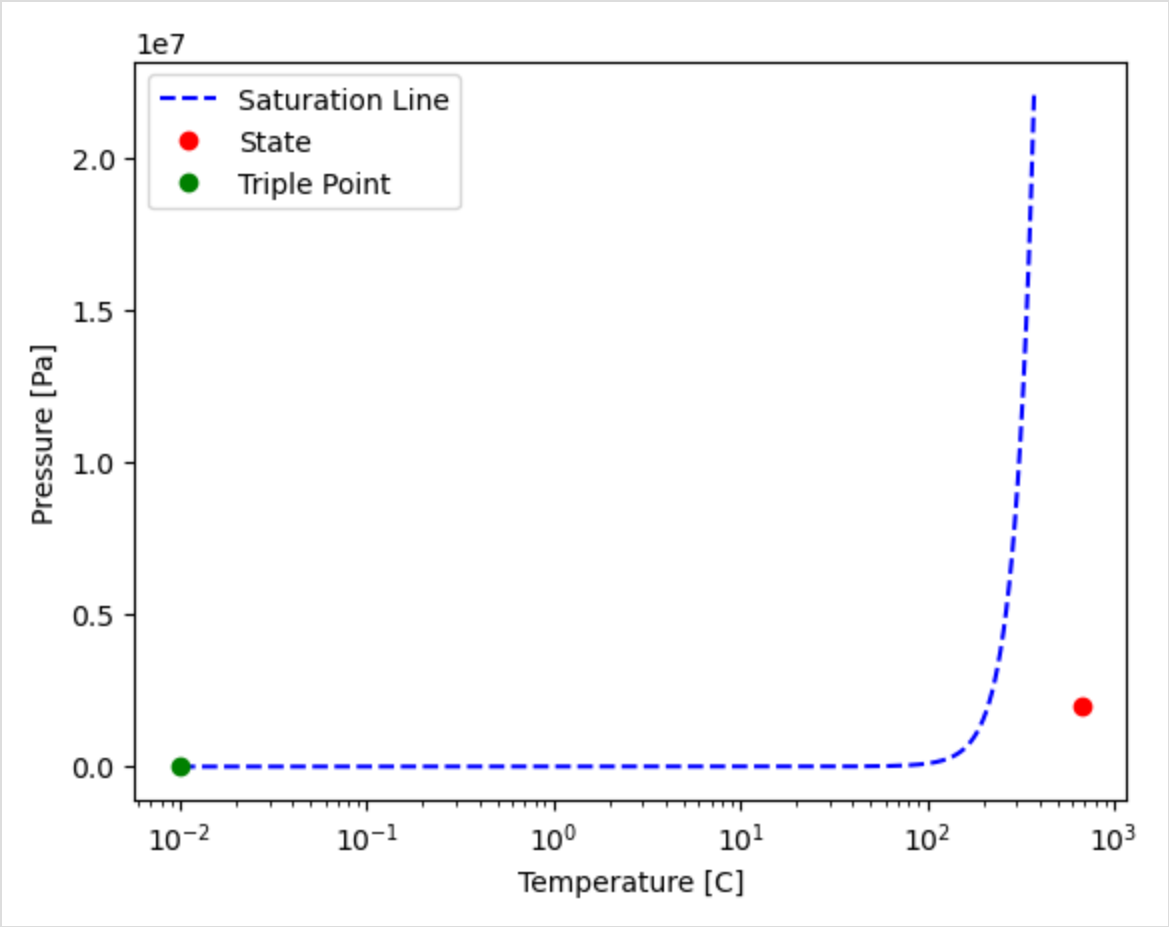
```
V = Z * m * R * T / P_kpa
print('The volume of water wapor based on ideal gas correlation coupled with compressibiity factor i

#calculating error
E = np.abs(V-V_ref)/V_ref * 100
print('The error based on ideal gas correlation coupled with compressibiity factor is',f"{E:.3f}",'%
```

```
The volume of water wapor based on ideal gas correlation coupled with compressibiity factor is 0.151
The error based on ideal gas correlation coupled with compressibiity factor is 0.082 %
```

Solution Approach for d)

```
# define variables
Q = 1  # define the steam quality as 1, which is 100% vapor
fluid = "water"  # define the fluid or material of interest, for full list see CP.Fluidslist()
T_min = CP.PropsSI("Tmin", fluid)  # this is the triple-point temp we can get data for water
P_min = CP.PropsSI("P", "T", T_min, "Q", Q, fluid)  # triple-point temp for pressure
T_max = CP.PropsSI("Tcrit", fluid)  # this is the max temp we can get data for water
T_vals = np.linspace(T_min, T_max, 1000)  # define an array of values from T_min to T_max

pressure = [CP.PropsSI("P", "T", T, "Q", Q, fluid) for T in T_vals]  # call for pressure values usir

plt.plot(T_vals-273.15, pressure, "--b", label="Saturation Line")  # plot temp vs specific vol
plt.xscale("log")  # use log scale on x axis

## something interesting happenning -- why does Saturated liq and vapor P-T curve fall into the same

plt.xlabel("Temperature [C]")  # give x axis a label
plt.ylabel("Pressure [Pa]")  # give y axis a label

# plot various points on the T-v diagram:
plt.plot(T,P,'or', label = 'State')
plt.plot(T_min-273.15,P_min,'og', label = 'Triple Point')

plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fec037114f0>
```



4. The First Law of Thermodynamics for Closed Systems

This Chapter goes through concepts from Chapter 4 from here: https://pressbooks.bccampus.ca/thermo1/chapter/4-0-chapter-introduction-and-learning-objectives/ Using the first law, internal energy, work done and heat supplied/rejected are calculated for various senarios.

Piston-cylinder: isothermal expansion

Problem Statement:

A cylinder fitted with a frictionless piston contains 1 kg of oxygen gas at an initial temperature of 20°C and a volume of 0.8 m³. The gas undergoes an isothermal expansion until its volume doubles. During the process, the cylinder is in thermal contact with a heat reservoir at 20°C.

Skip to main content

1. The specific boundary work done by the oxygen gas during the expansion.
2. The amount of heat transferred to the gas during this process.
3. Assuming the specific internal energy of oxygen changes due to the process, calculate the change in specific internal energy.

For the calculations, assume that oxygen behaves as an ideal gas and use its specific properties. Also, consider that the process is isothermal, meaning the temperature remains constant throughout the process.

Solution:

```
## Solution:

import CoolProp.CoolProp as CP
import math

# Gas choice
gas = "Oxygen"

# Given values
m = 1.0  # Mass of the gas in kg
T1 = 20 + 273.15  # Initial temperature in Kelvin (converted from 20°C)
V1 = 0.8  # Initial volume in m^3
V2 = 2 * V1  # Final volume (double the initial volume)
R = CP.PropsSI('GAS_CONSTANT', gas) / CP.PropsSI('MOLAR_MASS', gas)  # Specific gas constant for Oxy

# 1. Specific Boundary Work (w_boundary)
# For an isothermal process, P1 * V1 = P2 * V2 (Ideal Gas Law), and P1 can be found from P1 = m * R
P1 = m * R * T1 / V1
# Calculating the boundary work
w_boundary = P1 * V1 * math.log(V2 / V1)

# 2. Heat Transfer (Q)
# For an isothermal process, the heat transfer is equal to the boundary work done
Q = w_boundary

# 3. Change in Specific Internal Energy (Δu)
# For an isothermal process of an ideal gas, Δu = 0
delta_u = 0

# Output the results
print(f"Specific Boundary Work (w_boundary): {round(w_boundary/1e3,1)} kJ")
print(f"Heat Transfer (Q): {round(Q/1e3,1)} kJ")
print(f"Change in Specific Internal Energy (Δu): {round(delta_u/1e3,1)} kJ")
```

```
Specific Boundary Work (w_boundary): 52.8 kJ
Heat Transfer (Q): 52.8 kJ
Change in Specific Internal Energy (Δu): 0.0 kJ
```

Skip to main content

Problem Statement:

Consider a container with a fixed volume of 0.3 m^3 that initially contains helium gas at a temperature of 15°C and a pressure of 100 kPa. The gas undergoes an isochoric (constant volume) heating process until its pressure triples. Calculate:

1. The final temperature of the helium gas.
2. The change in internal energy of the gas during this process, assuming the specific heat at constant volume (Cv) is known.
3. The total heat transfer to the helium gas.

Use the ideal gas law and assume helium behaves as an ideal gas.

Solution:

```
import CoolProp.CoolProp as CP
import math

# Given values
V = 0.3  # Volume in m^3
T1 = 15 + 273.15  # Initial temperature in Kelvin
P1 = 100000  # Initial pressure in Pa (100 kPa)
P2 = 3 * P1  # Final pressure (tripled)
gas = 'Helium'

# 1. Final Temperature (T2)
# Since the process is isochoric, P1/T1 = P2/T2 (Ideal Gas Law)
T2 = P2 * T1 / P1

# 2. Change in internal energy (Δu)
Cv = CP.PropsSI('Cvmass', 'T', T1, 'P', P1, gas)  # Cv for helium
delta_u = Cv * (T2 - T1)

# 3. Total heat transfer (Q)
# For an isochoric process, Q = Δu (First Law of Thermodynamics)
Q = delta_u

# Output the results
print(f"Final Temperature (T2): {round(T2,1)} K")
print(f"Change in internal energy (Δu): {round(delta_u/1e3,1)} kJ")
print(f"Total heat transfer (Q): {round(Q/1e3,1)} kJ")
```

```
Final Temperature (T2): 864.5 K
Change in internal energy (Δu): 1795.8 kJ
Total heat transfer (Q): 1795.8 kJ
```

Adiabatic Compression of an Ideal Gas

Skip to main content

Problem Statement:

A piston-cylinder device initially contains 2 kg of air at 25°C and 1 atm. The air is compressed adiabatically to one-eighth of its original volume. Calculate:

1. The final temperature and pressure of the air.
2. The work done during this adiabatic compression process.
3. Assuming air behaves as an ideal gas with a specific heat ratio (γ), determine the change in internal energy.

Solution:

```
import CoolProp.CoolProp as CP
import math

# Given values
m = 2.0  # Mass of air in kg
T1 = 25 + 273.15  # Initial temperature in Kelvin
P1 = 101325  # Initial pressure in Pa (1 atm)
V1 = 1.0  # Initial volume (arbitrary value)
V2 = V1 / 8  # Final volume (one-eighth of initial)
gamma = CP.PropsSI('Cpmass', 'T', T1, 'P', P1, 'Air') / CP.PropsSI('Cvmass', 'T', T1, 'P', P1, 'Air'

# 1. Final Temperature and Pressure (T2, P2)
# For adiabatic process, T1 * V1^(gamma - 1) = T2 * V2^(gamma - 1)
T2 = T1 * (V1 / V2) ** (gamma - 1)
# P2 using the adiabatic relation P2 * V2^gamma = P1 * V1^gamma
P2 = P1 * (V1 / V2) ** gamma

# 2. Work done (W)
# For adiabatic process, W = (P1 * V1 - P2 * V2) / (gamma - 1)
W = (P1 * V1 - P2 * V2) / (gamma - 1)

# 3. Change in internal energy (Δu)
# Δu = Q - W, but for adiabatic process, Q = 0
delta_u = -W

# Output the results
print(f"Final Temperature (T2): {round(T2,1)} K")
print(f"Final Pressure (P2): {round(P2/1e3,1)} kPa")
print(f"Work done (W): {round(W/1e3,1)} kJ")
print(f"Change in internal energy (Δu): {round(delta_u/1e3,1)} kJ")
```

```
Final Temperature (T2): 687.5 K
Final Pressure (P2): 1869.1 kPa
Work done (W): -329.3 kJ
Change in internal energy (Δu): 329.3 kJ
```

Heat Transfer in a Rigid Container

Skip to main content

Problem Statement:

A rigid container is initially filled with 0.5 kg of carbon dioxide at 20°C and 100 kPa. The container is then heated until the temperature of the gas reaches 80°C. Calculate the amount of heat transferred to the carbon dioxide and the change in its internal energy.

Solution:

```
import CoolProp.CoolProp as CP

# Given values
m = 0.5  # Mass of carbon dioxide in kg
T1 = 20 + 273.15  # Initial temperature in Kelvin
T2 = 80 + 273.15  # Final temperature in Kelvin
gas = 'CarbonDioxide'

# 1. Change in internal energy (Δu)
Cv = CP.PropsSI('Cvmass', 'T', T1, 'P', 100000, gas)  # Cv for carbon dioxide
delta_u = Cv * (T2 - T1) * m

# Convert delta_u to kJ and round to one decimal point
delta_u_kJ = round(delta_u / 1000, 1)

# 2. Total heat transfer (Q)
# For a rigid container, Q = Δu (First Law of Thermodynamics)
Q = delta_u

# Convert Q to kJ and round to one decimal point
Q_kJ = round(Q / 1000, 1)

# Output the results
print(f"Change in internal energy (Δu): {delta_u_kJ} kJ")
print(f"Total heat transfer (Q): {Q_kJ} kJ")
```

```
Change in internal energy (Δu): 19.6 kJ
Total heat transfer (Q): 19.6 kJ
```

Helium in a Spring-Loaded Cylinder

Problem Statement:

A spring-loaded piston-cylinder device contains 0.3 kg of helium. Initially, the helium is at 25°C and 150 kPa, and the spring is uncompressed. The system is heated until the pressure doubles and the volume increases by 20%. Calculate the work done by the helium on the spring and the total heat added to the system.

Skip to main content

Solution:

```
import CoolProp.CoolProp as CP
import math

# Given values
m = 0.3  # Mass of helium in kg
T1 = 25 + 273.15  # Initial temperature in Kelvin
P1 = 150000  # Initial pressure in Pa

# Calculate the initial volume using the density
V1 = m / CP.PropsSI('D', 'T', T1, 'P', P1, 'Helium')
V2 = V1 * 1.2  # Final volume (20% increase)
P2 = 2 * P1  # Final pressure (doubled)

# Polytropic exponent for helium
n = CP.PropsSI('Cpmass', 'T', T1, 'P', P1, 'Helium') / CP.PropsSI('Cvmass', 'T', T1, 'P', P1, 'Heliu

# Calculate final temperature T2 using the polytropic process relation
T2 = T1 * (P2 / P1) * (V1 / V2)**n

# Calculate work done by the helium (W)
if n != 1:
    W = (P2 * V2 - P1 * V1) / (1 - n)
else:
    W = P1 * V1 * math.log(V2 / V1)

# Convert W to kJ and round to 1 decimal place
W_kJ = round(W / 1000, 1)

# Calculate the total heat added (Q)
Cv = CP.PropsSI('Cvmass', 'T', T1, 'P', P1, 'Helium')
delta_U = Cv * (T2 - T1) * m

# Convert delta_U to kJ and round to 1 decimal place
delta_U_kJ = round(delta_U / 1000, 1)

# Q is the sum of delta_U and W, converted to kJ and rounded
Q_kJ = round((delta_U + W) / 1000, 1)

# Output the results
print(f"Final Temperature (T2): {round(T2, 1)} K")
print(f"Work done by the helium (W): {W_kJ} kJ")
print(f"Total heat added (Q): {Q_kJ} kJ")
print(f"Change in Internal Energy (ΔU): {delta_U_kJ} kJ")
```

```
Final Temperature (T2): 440.1 K
Work done by the helium (W): -390.6 kJ
Total heat added (Q): -257.9 kJ
Change in Internal Energy (ΔU): 132.7 kJ
```

Isothermal Expansion of Nitrogen

Skip to main content

Problem Statement:

A cylinder with a movable piston contains 1 kg of nitrogen at 100 kPa and 300 K. It undergoes an isothermal expansion until the volume triples. Calculate the boundary work done during this process and the heat transfer involved.

Solution:

```
import CoolProp.CoolProp as CP
import math

# Given values
m = 1.0  # Mass of nitrogen in kg
T = 300  # Temperature in Kelvin (constant)
P1 = 100000  # Initial pressure in Pa
V1 = m * CP.PropsSI('Dmolar', 'T', T, 'P', P1, 'Nitrogen')  # Initial volume using density
V2 = 3 * V1  # Final volume (tripled)

# 1. Boundary work (W)
# For isothermal process, W = nRT ln(V2/V1)
R = CP.PropsSI('GAS_CONSTANT', 'Nitrogen') / CP.PropsSI('MOLAR_MASS', 'Nitrogen')
W = m * R * T * math.log(V2 / V1)

# 2. Heat transfer (Q)
# For an isothermal process, Q = W
Q = W

# Output the results
print(f"Boundary work done (W): {round(W/1e3,1)} kJ")
print(f"Heat transfer (Q): {round(Q/1e3,1)} kJ")
```

```
Boundary work done (W): 97.8 kJ
Heat transfer (Q): 97.8 kJ
```

Adiabatic Compression in a Rigid Container

Problem Statement:

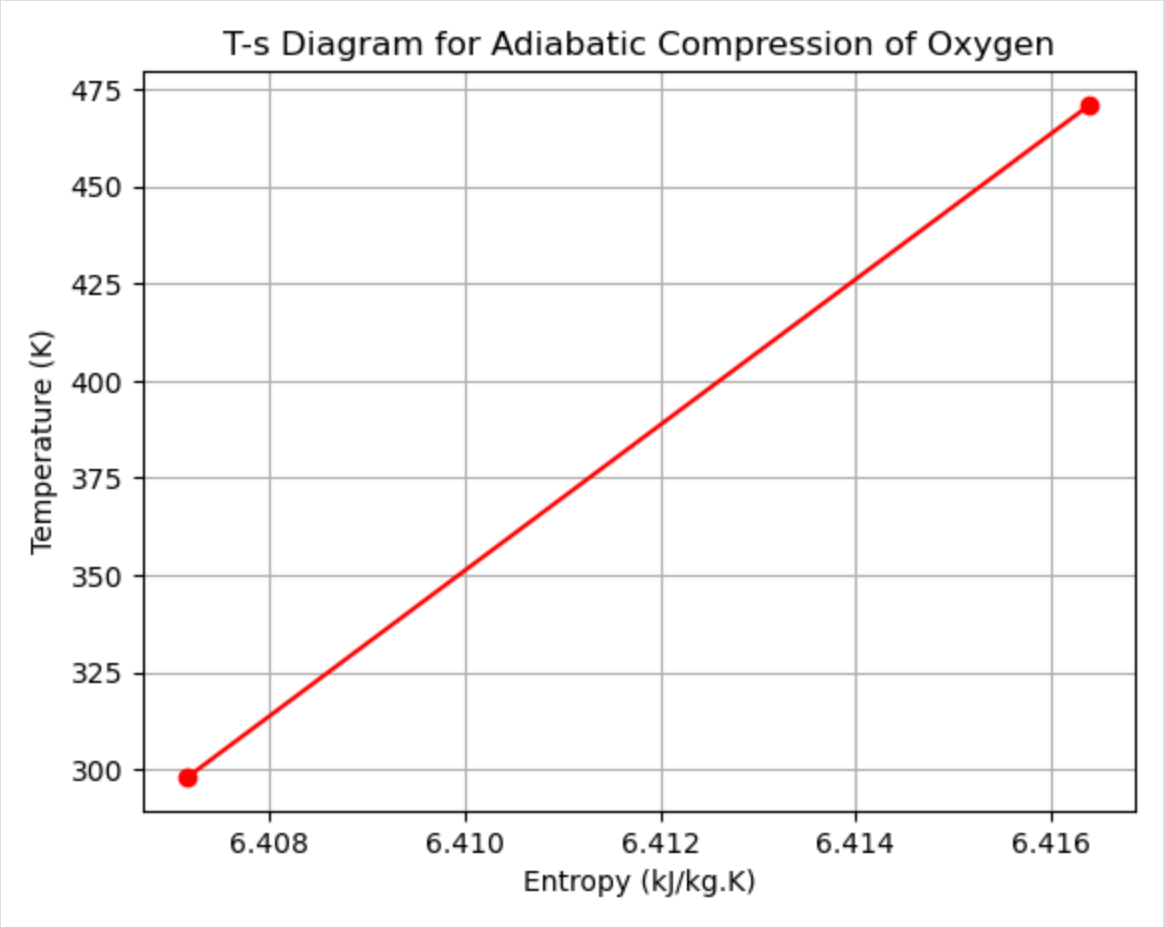
A rigid container with 2 kg of oxygen is initially at 1 atm and 25°C. The oxygen is compressed adiabatically until the pressure increases to 5 atm. Calculate the final temperature, the work done on the gas, and the change in internal energy.

Skip to main content

```
import CoolProp.CoolProp as CP
import math

# Given values
m = 2.0  # Mass of oxygen in kg
T1 = 25 + 273.15  # Initial temperature in Kelvin
P1 = 101325  # Initial pressure in Pa (1 atm)
P2 = 5 * P1  # Final pressure (5 atm)
gamma = CP.PropsSI('Cpmass', 'T', T1, 'P', P1, 'Oxygen') / CP.PropsSI('Cvmass', 'T', T1, 'P', P1, 'O

# Final Temperature (T2) for adiabatic process
T2 = T1 * (P2 / P1) ** ((gamma - 1) / gamma)

# Calculate specific volumes V1 and V2 using ideal gas law
R = CP.PropsSI('GAS_CONSTANT', 'Oxygen') / CP.PropsSI('MOLAR_MASS', 'Oxygen')  # Specific gas consta
V1 = m * R * T1 / P1
V2 = m * R * T2 / P2

# Calculate work done (W) during adiabatic process
W = (P1 * V1 - P2 * V2) / (gamma - 1)

# Change in internal energy (ΔU) for adiabatic process
delta_U = -W

# Output the results (W and ΔU in kJ)
print(f"Final Temperature (T2): {round(T2)} K")
print(f"Work done (W): {round(W/1000,1)} kJ")
print(f"Change in internal energy (ΔU): {round(delta_U/1e3,1)} kJ")
```

```
Final Temperature (T2): 471 K
Work done (W): -226.3 kJ
Change in internal energy (ΔU): 226.3 kJ
```

```
fluid = 'Oxygen'
import matplotlib.pyplot as plt
# Calculate entropy at initial and final states
S1 = CP.PropsSI('S', 'T', T1, 'P', P1, fluid)
S2 = CP.PropsSI('S', 'T', T2, 'P', P2, fluid)
plt.plot([S1 / 1000, S2 / 1000], [T1, T2], 'ro-')  # Adiabatic process
plt.xlabel('Entropy (kJ/kg.K)')
plt.ylabel('Temperature (K)')
plt.title('T-s Diagram for Adiabatic Compression of Oxygen')
plt.grid(True)
plt.show()
```



Linear interpolation for Internal energy of Superheated water

A function named “linear_interpolation” is defined, arguments of the same are T1, T2 (the two ends of the temperatures), T (the temperature at which a property needs to be interpolated) and Prop1, Prop2 are the proeprty values at T1 and T2.

```
def linear_interpolation(x, x1, x2, y1, y2):
    # Function to interpolate between two known points
    return y1 + (x - x1) / (x2 - x1) * (y2 - y1)
```

A function named “calculate_relative_error” is defined, arguments of the same are x1, x2 (the two ends of the input variable), x (the x-value at which a property needs to be interpolated) and y1, y2 are the property values at x1 and x2.

Skip to main content

```
def calculate_relative_error(x, x1, x2, y1, y2, fluid):
    # Calculate the interpolated value
    y_interpolated = linear_interpolation(x, x1, x2, y1, y2)

    # Get the value from CoolProp
    y_coolprop = CP.PropsSI("U", "P", P, "T", x, fluid) / 1e3  # Convert from J/kg to kJ/kg

    # Calculate absolute and relative errors
    absolute_error = abs(y_coolprop - y_interpolated)
    relative_error = (absolute_error / y_coolprop) * 100

    return relative_error
```

```
import CoolProp.CoolProp as CP
# Example usage from Superheated water:
# https://pressbooks.bccampus.ca/thermo1/back-matter/thermodynamic-properties-of-water/#TA2
T1, T2 = 273.15 + 100, 273.15 + 150  # Temperatures in K
P = 10e3  # in Pa
U1, U2 = 2515.49, 2587.91  # Properties in SI units
fluid = "water"
T = 273.15 + 133  # Temperature at which we want the interpolated property
Prop_interpolated = linear_interpolation(T, T1, T2, U1, U2)
print("Interpolated property at {} K: {} kJ/kg".format(T, round(Prop_interpolated, 2)))


cool_prop = CP.PropsSI("U", "P", P, "T", T, fluid) / 1e3  ## in kJ/kg
print("Property from CoolProp at {} K: {} kJ/kg".format(T, round(cool_prop, 2)))

absolute_difference = abs(cool_prop - Prop_interpolated)
percentage_difference = (absolute_difference / Prop_interpolated) * 100
print("Relative difference :{} %".format(round(percentage_difference, 4)))
```

```
Interpolated property at 406.15 K: 2563.29 kJ/kg
Property from CoolProp at 406.15 K: 2563.19 kJ/kg
Relative difference :0.0036 %
```

Linear interpolation for Internal energy of R-134a refrigerant

Skip to main content

```
import CoolProp.CoolProp as CP


# Example usage from Superheated R134a:
# https://pressbooks.bccampus.ca/thermo1/back-matter/thermodynamic-properties-of-r134a/#TC2
T1, T2 = 273.15 + 40, 273.15 + 50  # Temperatures in K
P = 100e3  # in Pa
U1, U2 = 412.4, 420.37  # Properties in SI units
fluid = "R134a"
T = 273.15 + 43  # Temperature at which we want the interpolated property
Prop_interpolated = linear_interpolation(T, T1, T2, U1, U2)
print("Interpolated property at {} K: {} kJ/kg".format(T, round(Prop_interpolated,2)))


cool_prop = CP.PropsSI("U", "P", P, "T", T, fluid) / 1e3  ## in kJ/kg
print("Property from CoolProp at {} K: {} kJ/kg".format(T, round(cool_prop,2)))

absolute_difference = abs(cool_prop - Prop_interpolated)
percentage_difference = (absolute_difference / Prop_interpolated) * 100
print("Relative difference :{} %".format(round(percentage_difference,4)))
```

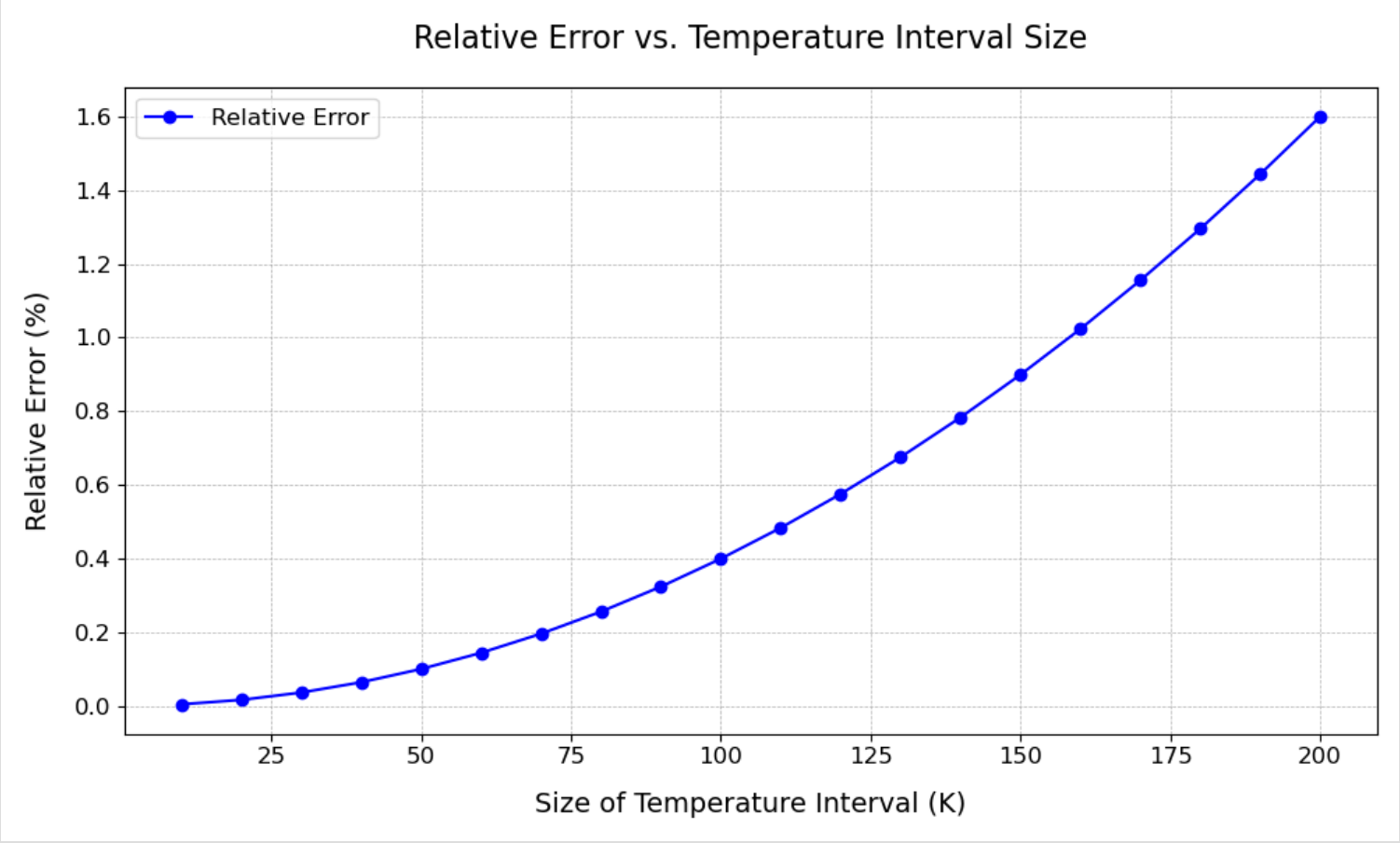
```
Interpolated property at 316.15 K: 414.79 kJ/kg
Property from CoolProp at 316.15 K: 414.77 kJ/kg
Relative difference :0.0046 %
```

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

# Constants
P = 10e3  # Pressure in Pa
fluid = "R134a"
T_target = 273.15 + 135  # Target temperature for property evaluation

# Range of interval sizes
interval_sizes = np.linspace(10,200,20)
relative_errors = []

# Loop over interval sizes and calculate relative errors
for interval in interval_sizes:
    T1 = T_target - interval / 2
    T2 = T_target + interval / 2

    # Get properties from CoolProp for the interval boundaries
    U1 = CP.PropsSI("U", "P", P, "T", T1, fluid) / 1e3
    U2 = CP.PropsSI("U", "P", P, "T", T2, fluid) / 1e3

    # Calculate relative error
    error = calculate_relative_error(T_target, T1, T2, U1, U2, fluid)
    relative_errors.append(error)
```

A plot to illustrate the relative error as a function of size of interval

Skip to main content

```
# Plotting
# Plotting with logarithmic scale and improved aesthetics
plt.figure(figsize=(10, 6))  # Sets the figure size
plt.plot(interval_sizes, relative_errors, marker='o', linestyle='-', color='blue', label='Relative E

plt.xlabel('Size of Temperature Interval (K)', fontsize=14, labelpad=12)
plt.ylabel('Relative Error (%)', fontsize=14, labelpad=12)
plt.title('Relative Error vs. Temperature Interval Size', fontsize=16, pad=20)

#plt.xscale('log')  # Logarithmic scale for the x-axis
#plt.yscale('log')  # Logarithmic scale for the y-axis

plt.legend(fontsize=12)
plt.grid(True, which="both", linestyle='--', linewidth=0.5)  # Adds gridlines for both major and mir
plt.tick_params(labelsize=12)  # Adjust the size of the axis ticks labels

plt.tight_layout()  # Adjusts subplot params so that the subplot(s) fits in to the figure area

plt.show()
```



Reference States in Thermodynamics

In thermodynamics, a reference state serves as a baseline or zero point for measuring thermodynamic properties such as enthalpy (H) and entropy (S). Choosing different reference states doesn't affect the differences in properties calculated

Skip to main content

For substances such as refrigerants (e.g., R-134a), commonly used reference states include the Normal Boiling Point (NBP) and the International Institute of Refrigeration (IIR) standard:

```
NBP: It sets the enthalpy and entropy at the boiling point of the substance under 1 atmosphere to ze

IIR: This reference state, set by the International Institute of Refrigeration, defines the enthalpy

For refrigerants, the ASHRAE reference state typically sets a specific point where the enthalpy and
```

Despite the reference state chosen, the differences in enthalpy (ΔH) and entropy (ΔS) between two defined states of R-134a remain the same. These differences are intrinsic and are not influenced by the reference points, as you'll observe in the CoolProp calculations that follow.

Problem statement:

Use threee different references and calculate absolute and relative differences of S, H for R134a refigerant fluid.

```
## add internal energy as another property
## make it clear what the textbook uses, in this case it is ASHRAE. A note at the end would be great
```

Solution

```
import CoolProp.CoolProp as CP

# Define two states for R-134a
T1 = 280  # Temperature in K for state 1
P1 = 101325  # Pressure in Pa for state 1 (1 atm)
T2 = 300  # Temperature in K for state 2
P2 = 200000  # Pressure in Pa for state 2

# Set the IIR reference state
CP.set_reference_state('R134a', 'IIR')

# Calculate enthalpy and entropy for both states with the IIR reference
h1_iir = CP.PropsSI('H', 'T', T1, 'P', P1, 'R134a')
s1_iir = CP.PropsSI('S', 'T', T1, 'P', P1, 'R134a')
h2_iir = CP.PropsSI('H', 'T', T2, 'P', P2, 'R134a')
s2_iir = CP.PropsSI('S', 'T', T2, 'P', P2, 'R134a')

# Calculate differences with IIR reference
dh_iir = (h2_iir - h1_iir) / 1000
ds_iir = (s2_iir - s1_iir) / 1000

# Set the NBP reference state
CP.set_reference_state('R134a', 'NBP')

# Calculate enthalpy and entropy for both states with the NBP reference
h1_nbp = CP.PropsSI('H', 'T', T1, 'P', P1, 'R134a')
s1_nbp = CP.PropsSI('S', 'T', T1, 'P', P1, 'R134a')
h2_nbp = CP.PropsSI('H', 'T', T2, 'P', P2, 'R134a')
s2_nbp = CP.PropsSI('S', 'T', T2, 'P', P2, 'R134a')

# Calculate differences with NBP reference
dh_nbp = (h2_nbp - h1_nbp) / 1000
ds_nbp = (s2_nbp - s1_nbp) / 1000

# Set the ASHRAE reference state
CP.set_reference_state('R134a', 'ASHRAE')

# Calculate enthalpy and entropy for both states with the ASHRAE reference
h1_ashrae = CP.PropsSI('H', 'T', T1, 'P', P1, 'R134a')
s1_ashrae = CP.PropsSI('S', 'T', T1, 'P', P1, 'R134a')
h2_ashrae = CP.PropsSI('H', 'T', T2, 'P', P2, 'R134a')
s2_ashrae = CP.PropsSI('S', 'T', T2, 'P', P2, 'R134a')

# Calculate differences with ASHRAE reference
dh_ashrae = (h2_ashrae - h1_ashrae) / 1000
ds_ashrae = (s2_ashrae - s1_ashrae) / 1000

print("Properties at T = 280 K and P = 1 bar")
# Print statements for each state
print("IIR Reference State:")
print(f"State 1 Enthalpy (kJ/kg): {h1_iir / 1000:.3f}")
print(f"State 1 Entropy (kJ/kgK): {s1_iir / 1000:.3f}")
print(f"State 2 Enthalpy (kJ/kg): {h2_iir / 1000:.3f}")
print(f"State 2 Entropy (kJ/kgK): {s2_iir / 1000:.3f}")
print(f"Enthalpy Difference (kJ/kg): {dh_iir:.3f}")
print(f"Entropy Difference (kJ/kgK): {ds_iir:.3f}\n")

print("NBP Reference State:")
print(f"State 1 Enthalpy (kJ/kg): {h1_nbp / 1000:.3f}")
print(f"State 1 Entropy (kJ/kgK): {s1_nbp / 1000:.3f}")
```

Skip to main content

```
print(f"Enthalpy Difference (kJ/kg): {dh_nbp:.3f}")
print(f"Entropy Difference (kJ/kgK): {ds_nbp:.3f}\n")

print("ASHRAE Reference State:")
print(f"State 1 Enthalpy (kJ/kg): {h1_ashrae / 1000:.3f}")
print(f"State 1 Entropy (kJ/kgK): {s1_ashrae / 1000:.3f}")
print(f"State 2 Enthalpy (kJ/kg): {h2_ashrae / 1000:.3f}")
print(f"State 2 Entropy (kJ/kgK): {s2_ashrae / 1000:.3f}")
print(f"Enthalpy Difference (kJ/kg): {dh_ashrae:.3f}")
print(f"Entropy Difference (kJ/kgK): {ds_ashrae:.3f}\n")
```

```
Properties at T = 280 K and P = 1 bar
IIR Reference State:
State 1 Enthalpy (kJ/kg): 409.317
State 1 Entropy (kJ/kgK): 1.848
State 2 Enthalpy (kJ/kg): 424.282
State 2 Entropy (kJ/kgK): 1.846
Enthalpy Difference (kJ/kg): 14.965
Entropy Difference (kJ/kgK): -0.002

NBP Reference State:
State 1 Enthalpy (kJ/kg): 243.507
State 1 Entropy (kJ/kgK): 0.979
State 2 Enthalpy (kJ/kg): 258.472
State 2 Entropy (kJ/kgK): 0.977
Enthalpy Difference (kJ/kg): 14.965
Entropy Difference (kJ/kgK): -0.002

ASHRAE Reference State:
State 1 Enthalpy (kJ/kg): 261.173
State 1 Entropy (kJ/kgK): 1.052
State 2 Enthalpy (kJ/kg): 276.138
State 2 Entropy (kJ/kgK): 1.050
Enthalpy Difference (kJ/kg): 14.965
Entropy Difference (kJ/kgK): -0.002
```

Reference	State	Enthalpy (kJ/kg)	Entropy (kJ/(kg·K))	ΔH (kJ/kg)	ΔS (kJ/(kg·K))
IIR	1(280 K, 1bar)	409.317	1.848		
IIR	2	424.282	1.846	14.965	-0.002
NBP	1	243.507	0.979		
NBP	2	258.472	0.977	14.965	-0.002
ASHRAE	1	261.173	1.052		
ASHRAE	2	276.138	1.050	14.965	-0.002

Skip to main content

```
import numpy as np

# Arrays of known points
x_points = np.array([0, 10])
y_points = np.array([1.0333, 1.0628])

# Point to interpolate
x= 280 - 273.15
# Perform interpolation
y = np.interp(x, x_points, y_points)

print("Interpolated value at x = {}: y = {}".format(round(x,3), round(y,3)))
```

```
Interpolated value at x = 6.85: y = 1.054
```

```
import numpy as np

# Arrays of known points
x_points = np.array([20, 30])
y_points = np.array([270.20, 278.91])

# Point to interpolate
x= 300 - 273.15
# Perform interpolation
y = np.interp(x, x_points, y_points)

print("Interpolated value at x = {}: y = {}".format(round(x,2), round(y,2)))
```

```
Interpolated value at x = 26.85: y = 276.17
```

Comparison of work done in different processes

Problem Statement

Consider an ideal gas undergoing a polytropic process. At the initial state, the pressure (P_1 = 200 kPa) and specific volume (v_1 = 0.05 m3/kg). At the final state, the specific volume (v_2 = 0.1 , ext{m}^3/ ext{kg}). Analyze the process for polytropic exponents (n = 1.3) and (n = 1.0) (isothermal process).

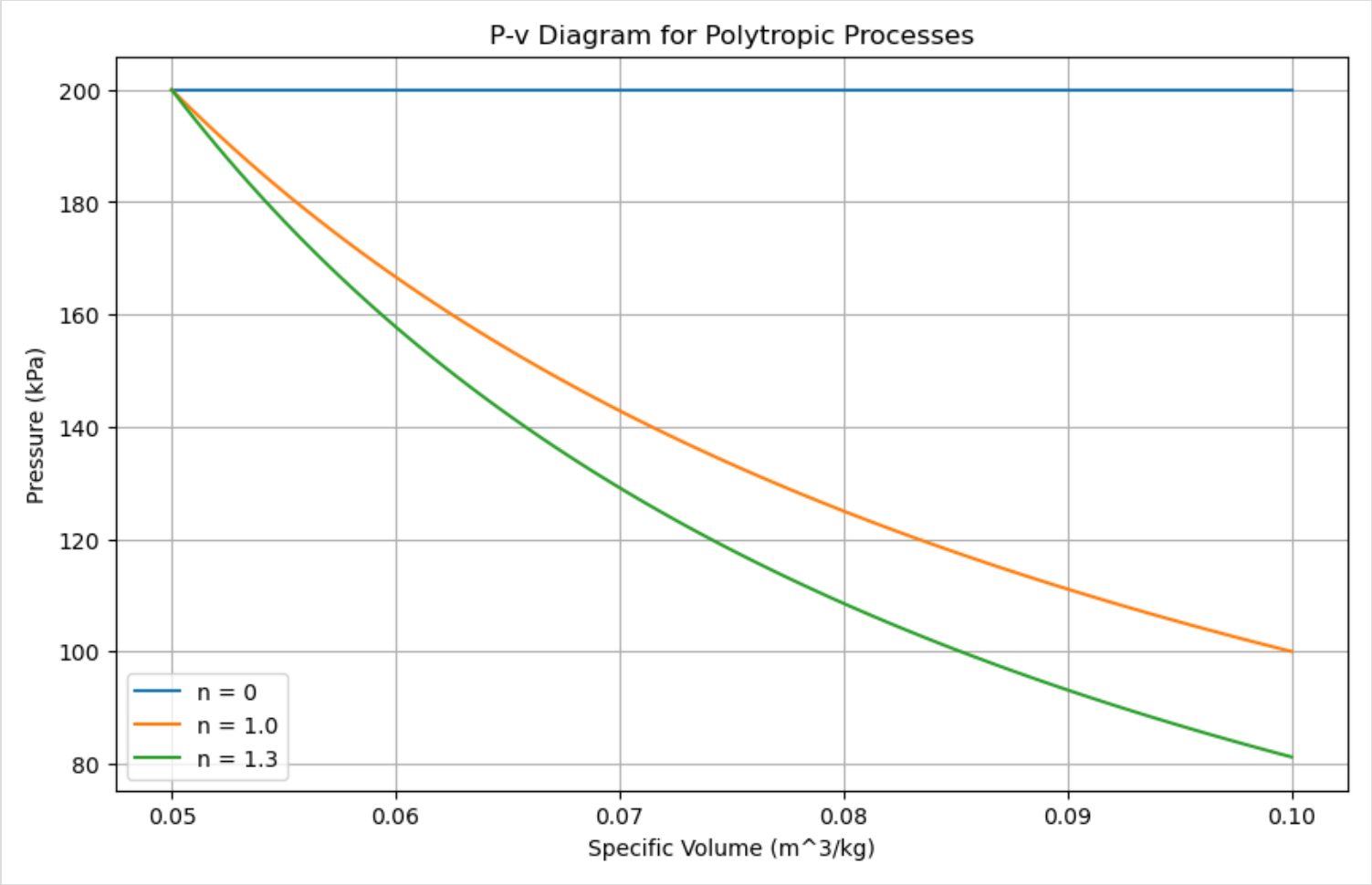
1. Sketch the two processes on a P-v diagram. Which process has a larger specific boundary work?
2. Calculate the specific boundary work for both processes.
3. Compare the work for an isobaric (n=0), n=1 (isothermal), and n=1.3 polytropic processes.

Skip to main content

Solution

```
import matplotlib.pyplot as plt
import numpy as np
# Given values
P1 = 200  # kPa
v1 = 0.05  # m^3/kg
v2 = 0.1  # m^3/kg
n_values = [0, 1.0, 1.3]  # Polytropic exponents

def polytropic_process(P1, v1, v2, n):
    if n == 0:  # Isobaric
        P2 = P1
    else:  # Polytropic or Isothermal
        P2 = P1 * (v1 / v2)**n
    return P2

def specific_work(P1, v1, v2, n):
    if n == 0:  # Isobaric
        return P1 * (v2 - v1)
    elif n == 1:  # Isothermal
        return P1 * v1 * np.log(v2 / v1)
    else:  # Polytropic
        return (P1 * v1 - polytropic_process(P1, v1, v2, n) * v2) / (1 - n)

# Plotting the P-v diagram
plt.figure(figsize=(10, 6))
v = np.linspace(v1, v2, 100)
for n in n_values:
    P = [polytropic_process(P1, v1, vi, n) for vi in v]
    plt.plot(v, P, label=f'n = {n}')
    work = specific_work(P1, v1, v2, n)
    print(f'Specific boundary work for n = {n}: {work:.1f} kJ/kg')

plt.xlabel('Specific Volume (m^3/kg)')
plt.ylabel('Pressure (kPa)')
plt.title('P-v Diagram for Polytropic Processes')
plt.legend()
plt.grid(True)
plt.show()
```

```
Specific boundary work for n = 0: 10.0 kJ/kg
Specific boundary work for n = 1.0: 6.9 kJ/kg
Specific boundary work for n = 1.3: -6.3 kJ/kg
```



5. The First Law of Thermodynamics for a Control Volume

Practice problems on Chapter 5 from Prof. Claire Yu Yan's textbook.

Energy balance for Stream in Control Volume

1kg of steam is confined in a space of the size $1\ m^3$ where the thermometer measures $400^\circ C$. Using CoolProp as a tool to extract thrmodynamic properties, determine:

a)the pressure reading on a pressure guage at atmospheric pressure

the steam is then heated to double the pressure reading while the space is kept from expanding. determine:

Skip to main content

c)how much heat is require to do so

d)what would be the asnwer if C_v is used instead of CoolProp? What assumptions are made?

Solution Approach for a)

Density D and temperature T are the two known parameters that can be used to extract properties.

$D = m/V$

```
#importing the required library
import CoolProp.CoolProp as CP

# define the given inputs:
T = 400 + 273.15 #temperature in K
D = 1 / 1 #density in kg/m3
P = CP.PropsSI('P', 'D', D, 'T', T, 'Water') #calculating pressure using coolprop Pa
P_g = P - 101325 #gauge pressure in Pa
print('The gauage pressure of steam at the given initial properties is',f"{P_g:.1f}",'Pa')
```

```
The gauage pressure of steam at the given initial properties is 208099.6 Pa
```

Solution Approach for b)

The known parameters from the secondary state are pressure

$P_{g2} = 2P_g$

and density

$D_2 = D$

since the mass and the volume of the space remains constant

```
# define the inputs:
P_g2 = P_g * 2 #gauge pressure in secondary state in Pa
P_2 = P_g2 + 101325 #absolute pressure in secondary state in Pa
D_2 = D #density in the secondary state in kg/m3
T_2 = CP.PropsSI('T', 'D', D_2, 'P', P_2, 'Water') #calculating secondary temperature using coolprop
T_2C = T_2 - 273.15 #temperature at secondary state in C
print('The temperature of steam at the secondary state is',f"{T_2C:.1f}",'C')
```

```
The temperature of steam at the secondary state is 849.0 C
```

Solution Approach for c)

Skip to main content

Given the first law of thermodynamics,

$Q = \Delta U + W$

and $W = 0$ since the boundaries of the space are fixed. Therefore,

$Q = \Delta U = U_2 - U_1 = m(u_2 - u_1)$

```
# define the inputs using coolprop:
m = 1 #steam mass in kg
u_1 = CP.PropsSI('U', 'D', D, 'T', T, 'Water') #calculating initial internal energy in J/kg
u_2 = CP.PropsSI('U', 'D', D_2, 'P', P_2, 'Water')  #calculating secondary internal energy in J/kg
Q = m * (u_2 - u_1) / 1000 #heat required in kJ
print('The heat required to double the guage presure is',f"{Q:.1f}",'kJ')
```

```
The heat required to double the guage presure is 790.8 kJ
```

Solution Approach for d)

To use C_v and C_p values to calculate changes in enthalpy and internal energy, ideal gas assumption is to be made for steam.

$Q = \Delta U = m\Delta u = mC_v\Delta T$

```
#define the constants
C_v = 1.4108 #C_v of steam in kJ/kg.k
Q = m * C_v * (T_2 - T) #heat required in kJ
print('The heat required to double the guage presure is',f"{Q:.1f}",'kJ','using C_v')
```

```
The heat required to double the guage presure is 633.5 kJ using C_v
```

Piston-cylinder: Carbon dioxide

Consider carbon-dioxide stored in a cylinder-and-piston system at $1\ MPa$ and $0^\circ C$ with a volume of $1\ m^3$. The piston is free to move keeping the pressure of carbon-diaxide constant. The system is then heated to $25^\circ C$. Using CoolProp as a tool, determine:

a)the mass of carbon-dioxide

b)heat required to do so

c)the required heat if the weight of the 500g aluminum piston and cylinder are to be considered

d)the required heat in b if C_p is to be used to calculate Δh instead of CoolProp? what assumptions are made in this case?

Skip to main content

Solution Approach for a)

the two knowns to extract thermodynamic properties are

$P_1 = 1\,MPa$

$T_1 = 0^\circ C$

to calculate mass, density is to be extracted

$m = D_1 V_1$

```
#importing the required library
import CoolProp.CoolProp as CP

#define state variables
V_1 = 1 #ninitial volume in m3
P_1 = 1e+6 #initial pressure in Pa
T_1 = 0 + 273.15 #initial temperature in K
D_1 = CP.PropsSI('D', 'P', P_1, 'T', T_1, 'CO2') #calculating density using coolprop kg/m3
m_co2 = D_1 * V_1 #mass of carbon-dioxide in kg
print('The mass of carbon-dioxide is',f"{m_co2:.1f}",'kg')
```

```
The mass of carbon-dioxide is 20.8 kg
```

Solution Approach for b)

based on the first law,

$Q = \Delta U + W = \Delta H = m(h_2 - h_1)$

```
#define state variable
T_2 = 25 + 273.15 #secondary temperature in K
P_2 = P_1 #constant pressure process
#extracting enthalpy
h_1 = CP.PropsSI('H', 'P', P_1, 'T', T_1, 'CO2') #initial enthalpy in J/kg
h_2 = CP.PropsSI('H', 'P', P_2, 'T', T_2, 'CO2') #secondary enthalpy in J/kg
Q = m_co2 * (h_2-h_1) / 1000 #heat required in KJ
print('The required heat for this process is',f"{Q:.1f}",'kJ')
```

```
The required heat for this process is 481.1 kJ
```

Solution Approach for c)

for solids and liquids

Skip to main content

```
#define constants
m_alm = 0.5 #mass of aluminum in kg
C_p_alm = 0.897 #C_p of aluminum in kJ/kg.K
Q_alm = m_alm * C_p_alm * (T_2-T_1)
Q_total = Q_alm + Q #total heat required for the process
print('The required total heat for this process is',f"{Q_total:.1f}",'kJ')
```

```
The required total heat for this process is 492.4 kJ
```

Solution Approach for d)

in order to use C_p to calculate Δh, carbon-dioxide is assumed to be ideal gas; then from the first law,

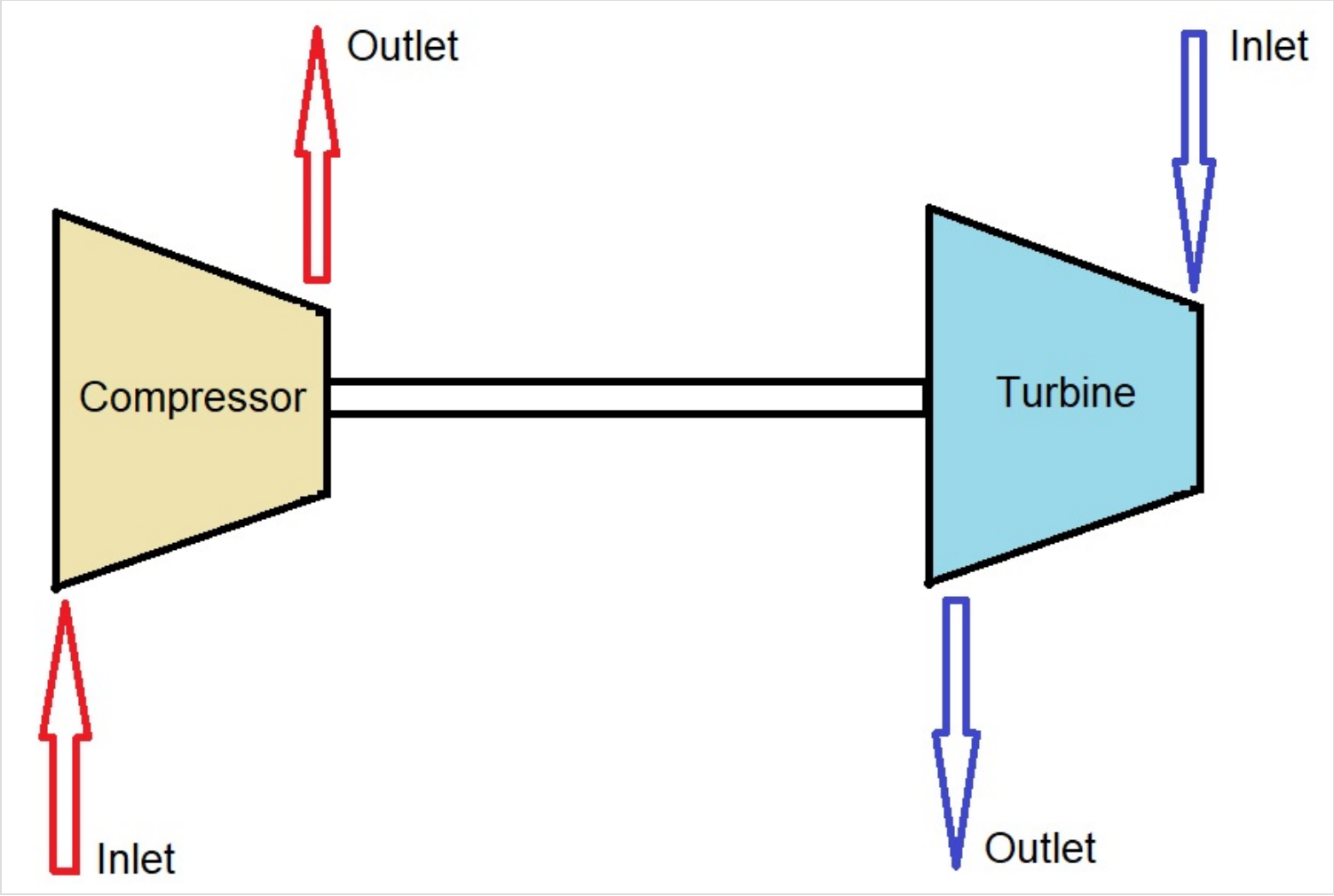
$Q = \Delta U + W = \Delta H = mC_p\Delta T$

```
#define constants
C_p_co2 = 0.846 #Cp of CO2 in kJ/kg.K
Q = m_co2 * C_p_co2 * (T_2-T_1)
print('The required heat for this process using Cp assumption is',f"{Q:.1f}",'kJ')
```

```
The required heat for this process using Cp assumption is 440.7 kJ
```

Air Compressor and Turbine

Imagine a turbine whose work output is used to drive a compressor to compress air from atmospheric pressure and temperature to $1\ MPa$. The turbine operates with steam enerting the turbine at $1.2\ MPa$ and $600^{\circ}C$ and exits as satirated vapor at atmospheric pressure. Assuming air to follow ideal-gas law going through a polytropic process with $n = 1.3$, determine the flow-rate of steam per $1\ kg/s$ of air flow-rate.



Solution Approach

Since the turbine and the compressor are couples, the work output from the turbine is considered the work input for the compressor. Therefore, calculating the work output for unit mass of turbine flow-rate as well as the work input for compressing $1\ kg$ of air build the bridge to connect the turbune and the compressor and to calculate the required flow-rate.

From the first law of thermodynamics for the trubine assuming no changes in velocity and elevation:

$w = \Delta h = h_{in} - h_{out}$

Skip to main content

```
#importing the required library
import CoolProp.CoolProp as CP

#define thermodynamic variables
P_atm = 101325 #atmospheric pressure in Pa
P_it = 1.2E+6 #turbine inlet pressure in Pa
T_it = 600 + 273.15 #turbine inlet temperatue in K
P_ot = P_atm #turbine outlet pressure in Pa
x_ot = 1#turbine outlet quality

h_it = CP.PropsSI('H', 'P', P_it, 'T', T_it, 'Water') #turbine inlet enthapy J/kg
h_ot = CP.PropsSI('H', 'P', P_ot, 'Q', x_ot, 'Water') #turbine inlet enthapy J/kg

w_t = h_it - h_ot #turbine work output per unit mass of steam in J
```

For a polytropic process of an ideal gas with a polytropic constant of n:

$T_2 = T_1(P_2/P_1)^{(n-1)/n}$

for reference look at Example 1 from Chapter 5 in the textbook

```
n = 1.3 #polytropic constant of the process
T_ic = 25 + 273.15 #inlet temperature of compressor in K
P_ic = P_atm #inlet pressure of compressor in Pa
P_oc = 1E+6 #outlet pressure of compressor in Pa
T_oc = T_ic * (P_oc/P_ic)**((n-1)/n) #outlet temperature of compressor air in K
```

Now assuming air as an ideal gas, its C_p value can be used to calculate changes in enthalpy

$w = \Delta h = h_{out} - h_{in} = C_p(T_{out} - T_{in})$

```
#define constants
C_p = 100.5 #Cp of air in J/kg.K
w_c = C_p * (T_oc - T_ic) #specific work input to compress air in J
```

Now, to correlate mass flow-rates, the rate of work output in the turbine equals to the rate of work input into the compressor

$\dot{W}_{turbine} = \dot{W}_{compressor}$

$\dot{m}_{turbine} w_{turbine} = \dot{m}_{compressor} w_{compressor}$

$\dot{m}_{turbine} = \dot{m}_{compressor} w_{compressor} / w_{turbine}$

```
m_c = 1 #flow-rate of compressor required in kg/s
m_t = m_c * w_c / w_t
print('The steam mass flow-rate required to compress 1 kg/s of air is',f"{m_t:.3f}",'kg/s')
```

```
The steam mass flow-rate required to compress 1 kg/s of air is 0.020 kg/s
```

Skip to main content

P-h diagram for R-134a refrigerant

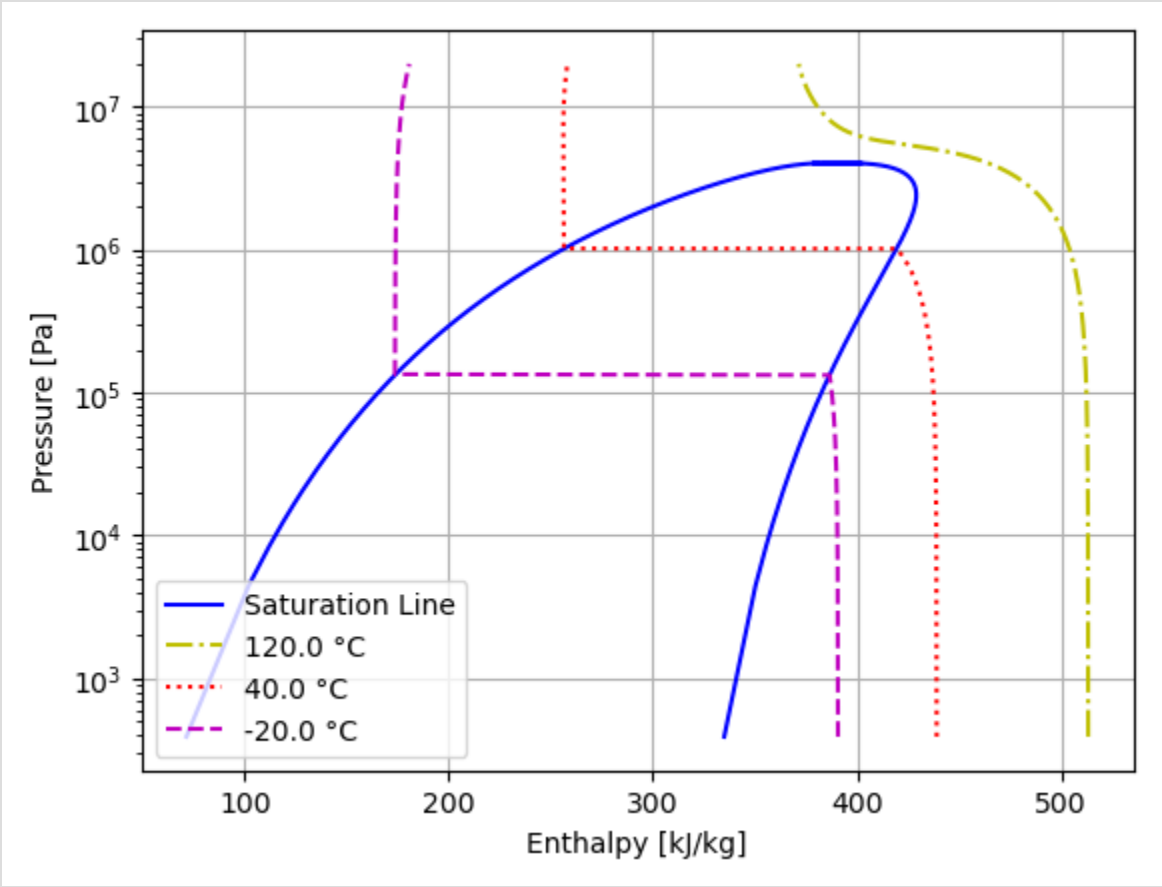
Consider R134-a as a refrigrant fluid. Build the P-h (Pressure vs. Specific Enthalpy) for R134-a knowing the critical pressure is around $4.03\ MPa$. Build three constant temperature curves for $T = 120^{\circ}C$, $0\ ^{\circ}C$ and $-20^{\circ}C$.

Solution approach:

Skip to main content

```
# Plot a P-h diagram for a fluid of choice


# import the libraries we'll need
import CoolProp.CoolProp as CP
import numpy as np
import matplotlib.pyplot as plt


# define variables
fluid = "R134A"  # define the fluid or material of interest, for full list see CP.Fluidslist()
T_min = CP.PropsSI("Tmin", fluid)  # triple-point temp for the fluid
P_min = CP.PropsSI("P", "T", T_min, "Q", 0, fluid)  # triple-point pressure for the fluid
P_max = 4.03E+6 #approximate critical pressure

P_vals = np.linspace(P_min, P_max, 1000)  # define an array of values from P_min to P_max
Q = 1  # define the fluid quality as 1, which is 100% vapor

enthalpy = [CP.PropsSI("H", "P", P, "Q", Q, fluid)/1000 for P in P_vals]  # call for enthalpy values

plt.plot(enthalpy, P_vals, "-b", label="Saturation Line")  # plot pressure vs enthalpy

Q = 0  # define the fluid quality as 0, which is 100% liquid

enthalpy = [CP.PropsSI("H", "P", P, "Q", Q, fluid)/1000 for P in P_vals] # call for enthalpy values

plt.plot(enthalpy, P_vals, "-b")  # plot pressure vs enthalpy


plt.yscale("log")  # use log scale on y axis
plt.ylabel("Pressure [Pa]")  # give y axis a label
plt.xlabel("Enthalpy [kJ/kg]")  # give x axis a label
plt.grid()
plt.legend()

# Building constant temperature curves

T_up = 120 + 273.15
T_mid = 40 + 273.15
T_down = -20 + 273.15

P_max = 20E+6  # max pressure in the plot set to 20MPa
P_vals = np.linspace(P_min, P_max, 10000)  # define an array of values from P_min to P_max
enthalpy_up = [CP.PropsSI("H", "P", P, "T", T_up, fluid)/1000 for P in P_vals] # call for enthalpy v
enthalpy_mid = [CP.PropsSI("H", "P", P, "T", T_mid, fluid)/1000 for P in P_vals] # call for enthalpy
enthalpy_down = [CP.PropsSI("H", "P", P, "T", T_down, fluid)/1000 for P in P_vals] # call for enthal

plt.plot(enthalpy_up, P_vals, "-.y", label="{} °C".format(T_up-273.15))  # plot pressure vs enthalpy
plt.plot(enthalpy_mid, P_vals, ":r", label="{} °C".format(T_mid-273.15))  # plot pressure vs enthalp
plt.plot(enthalpy_down, P_vals, "--m", label="{} °C".format(T_down-273.15))  # plot pressure vs enth
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f5d64166100>
```



Refigeration cycle: R134a Part 2

consider a refrigration cycle working with R134-a as coolant. The refrigrnt absorbs heat at $-20^{\circ}C$ during evaporation and enters the compressor as saturated vapor. The compressor then pressurizes R134-a while the its temperature increases to $120^{\circ}C$. The refrigrant is then cooled down to saturated liquid at $40^{\circ}C$ in a condensor in constant pressure before entering a throttling valve. The refrigrant is then throttled to $-20^{\circ}C$ to provide refrigrating fluid for the evaporator. Calculate,

a) specific enthalpy of the refrigrant entering the throttling valve

b) specific enthalpy, quality and pressure of the refrigrant exiting the throttling valve

c) specific enthalpy and pressure at compressor inlet

d) specific enthalpy at compressor outlet

e) how much heat is absorbed in the evaporator per unit mass of the refrigrant

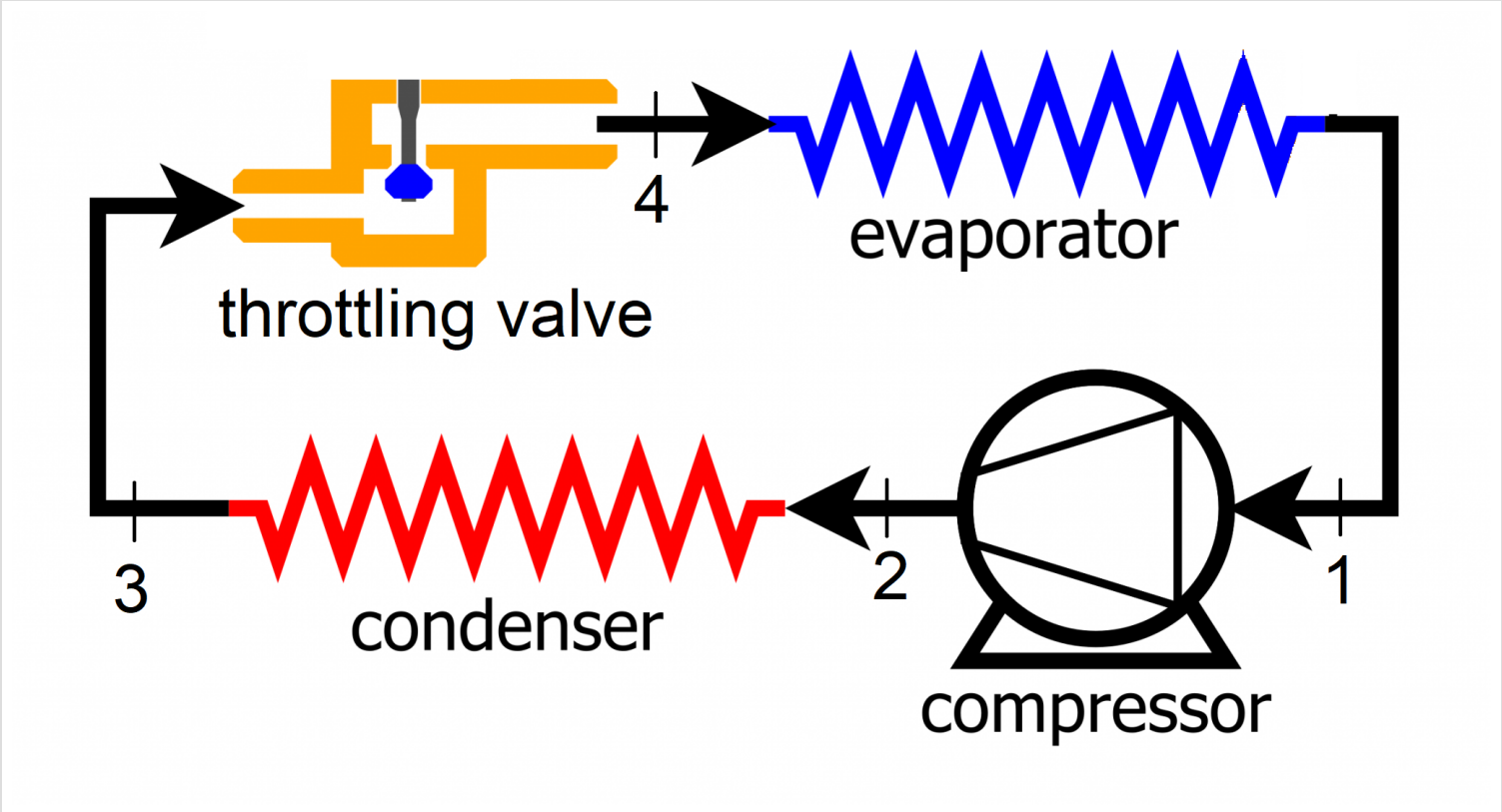
f) how much heat is rejected in the condenser per unit mass of the refrigrant

g) how much work is required by the compresor to elevate $1\ kg$ of the refrigrant's pressure; what would be the coefficient of performance (COP)?

Skip to main content

i) verify the first law of thermodynamics for the system as a whoole

j) draw the process on a P-h diagram developed previously for Q4 at this chapter



Solution Approach for a)

The refrigrant entering the throttling valve comes from the condenser; therefore the state 3 enthalpy is desired. The temperature at this state is guven as $T_3 = 0°C$ and the refrigrant is at its satuarated liquid state with quality equal to $x = 0$.

```
# import the libraries we'll need
import CoolProp.CoolProp as CP

# define variables
fluid = "R134A"  # define the fluid or material of interest
T_3 = 40 + 273.15 #stete #3 temperature in K
h_3 = CP.PropsSI("H", "T", T_3, "Q", 0, fluid)/1000  # enthalpy of the refrigrant at state #3 in kJ/
print('Th specific of the refrigrant at state #3 is:', f"{h_3:.1f}", 'kJ/kg')
```

```
Th specific of the refrigrant at state #3 is: 256.4 kJ/kg
```

Solution Approach for b)

Skip to main content

A throttling valve is ideally assumed to conserve enthalpy considering the first law of thermodynamics. Therefore,

$h_3 = h_4$

The refrigrant enters the evaporator as saturated liquid at $-20^\circ C$, therefore

$h_4 = h_{f@-20^\circ C} + x h_{fg@-20^\circ C}$

$x = (h_4 - h_{f@-20^\circ C})/h_{fg@-20^\circ C}$

The pressure at the throttling valve exit, which is same as evaporator inlet, would be satuartion pressure at the evaporator's working temperature since the refrigrant enters the evaporator as saturated fluid.

$P_4 = P_{sat@-20^\circ C}$

```
h_4 = h_3   #constant enthalpy through a throttling valve

T_4 = -20 + 273.15   #temperature of refrigrant at state #4 in K
h_gN20 = CP.PropsSI("H", "T", T_4, "Q", 1, fluid)/1000   #enthalpy of sat vap at -20C in kJ/kg
h_fN20 = CP.PropsSI("H", "T", T_4, "Q", 0, fluid)/1000   #enthalpy of sat lig at -20C in kJ/kg
h_fgN20 = h_gN20 - h_fN20   #h_fg at -20C in kJ/kg
x = (h_4 - h_fN20)/h_fgN20  #qaulity of refrigrant entering the evaporator

P_4 = CP.PropsSI("P", "T", T_4, "Q", x, fluid)  # pressure of the refrigrant at state #1 in Pa

print('The specific of the refrigrant at state #4 is:', f"{h_4:.1f}", 'kJ/kg')
print('The quality of the refrigrant at state #4 is:', f"{x:.3f}")
print('The pressure of the refrigrant at state #4 is:', f"{P_4:.1f}", 'Pa')
```

```
The specific of the refrigrant at state #4 is: 256.4 kJ/kg
The quality of the refrigrant at state #4 is: 0.389
The pressure of the refrigrant at state #4 is: 132735.0 Pa
```

Solution Approach for c)

The refrigrant enters the compressor as saturated vapor, therefore it has gone through constant pressure(and temperature) heating in the evaporator whose temperature is $-20^\circ C$. So,

$h_1 = h_v @ -20^\circ C$

$P_1 = P_{sat@-20^\circ C}$

```
T_1 = T_4 #temperature at state #1 in K
h_1 = CP.PropsSI("H", "T", T_1, "Q", 1, fluid)/1000  # enthalpy of the refrigrant at state #1 in kJ/
P_1 = CP.PropsSI("P", "T", T_1, "Q", 1, fluid)  # pressure of the refrigrant at state #1 in Pa
print('Th specific enthalpy of the refrigrant at state #1 is:', f"{h_1:.1f}", 'kJ/kg')
print('Th pressure of the refrigrant at state #1 is:', f"{P_1:.1f}", 'Pa')
```

```
Th specific enthalpy of the refrigrant at state #1 is: 386.6 kJ/kg
Th pressure of the refrigrant at state #1 is: 132735.0 Pa
```

Skip to main content

Solution Approach for d)

The refrigrant is heated to $T_2 = 120^\circ C$. The pressure to which the refrigrant is pressurized to, however, is unknown. The condensor is assumed to be operating with constant pressure; therefore, the pressure keeps constant and the pressure at condensor outlet would be same as pressure at its inlet which is same as compressor outlet. The pressure at condensor outlet is calculated based on the refrigrant being at saturation state at $0^\circ C$.

$P_2 = P_3$

$P_3 = P_{satR134a@-20^\circ C}$

The pressure and temperature at the compressor outlet are then used to calculate enthalpy at this state.

```
T_2 = 120 + 273.15 #temperature at state #2 in K
P_3 = CP.PropsSI("P", "T", T_3, "Q", 0, fluid)  # pressure of the refrigrant at state #3 in Pa
P_2 = P_3 # pressure of the refrigrant at state #2 in Pa
h_2 = CP.PropsSI("H", "T", T_2, "P", P_2, fluid)/1000  # enthalpy of the refrigrant at state #2 in k
print('The specific of the refrigrant at state #2 is:', f"{h_2:.1f}", 'kJ/kg')
```

```
The specific of the refrigrant at state #2 is: 504.1 kJ/kg
```

Solution Approach for e)

Considerin the first law of thermodynamics for the evaporator,

$q_c = h_1 - h_4$

```
q_c = h_1 - h_4
print(f"{q_c:.1f}", 'kJ of heat is absorbed by the refrigrant in the evaporator per kg of refrigrant
```

```
130.1 kJ of heat is absorbed by the refrigrant in the evaporator per kg of refrigrant
```

Solution Approach for f)

Considerin the first law of thermodynamics for the condenser,

$q_h = h_2 - h_3$

```
q_h = h_2 - h_3
print(f"{q_h:.1f}", 'kJ of heat is rejected to the environment in the condenser per kg of refrigrant
```

Skip to main content

```
247.6 kJ of heat is rejected to the environment in the condenser per kg of refrigrant
```

Solution Approach for g)

Considerin the first law of thermodynamics for the compressor,

$w = h_2 - h_1$

$COP = q_c / w$

```
w = h_2 - h_1
cop = q_c / w
print(f"{w:.1f}", 'kJ of energy is required to compress 1kg of R134-a to the desired pressure')
print('The COP of the cycle is:', f"{cop:.1f}")
```

```
117.5 kJ of energy is required to compress 1kg of R134-a to the desired pressure
The COP of the cycle is: 1.1
```

Solution Approach for h)

for a polytropic process,

$Pv^k = constant$

$P_1 v_1^k = P_2 v_2^k$

$P_1 / P_2 = (v_2 / v_1)^k$

$d(density) = 1/v$

$P_1 / P_2 = (d_1 / d_2)^k$

$log(P_1 / P_2) = k\, log(d_1 / d_2)$

$k = log(P_1 / P_2) / log(d_1 / d_2)$

```
#importing required libraries
import numpy as np

#evaluating density
d_1 = CP.PropsSI("D", "T", T_1, "Q", 1, fluid)  # density of the refrigrant at state #1 in kg/m3 bas
d_2 = CP.PropsSI("D", "T", T_2, "P", P_2, fluid)  # density of the refrigrant at state #2 in kg/m3 b
k = np.log(P_1/P_2)/np.log(d_1/d_2)
print('Th polytropic constant for the compression process is:', f"{k:.3f}")
```

```
Th polytropic constant for the compression process is: 1.254
```

Skip to main content

Solution Approach for i)

Considerin the first law of thermodynamics for the whole system,

$w = q_h - q_c$

```
w_test =  q_h - q_c
w_test == w #w_test is the value calculated from the first law and w is the value calculated using e
```

```
True
```

Solution Approach for j)

Except the compression process, other are straight lines of constant pressure or constant enthalpy processes. Therefore, the curve for the compression is to be built based on polytropic process and thermodynamic properties obtained from CoolProp.

from h)

$P/P_1 = (d/d_1)^k$

$P = P1\,(d/d_1)^k$

values of P and d are generic pressure and density values for a polytropic process.

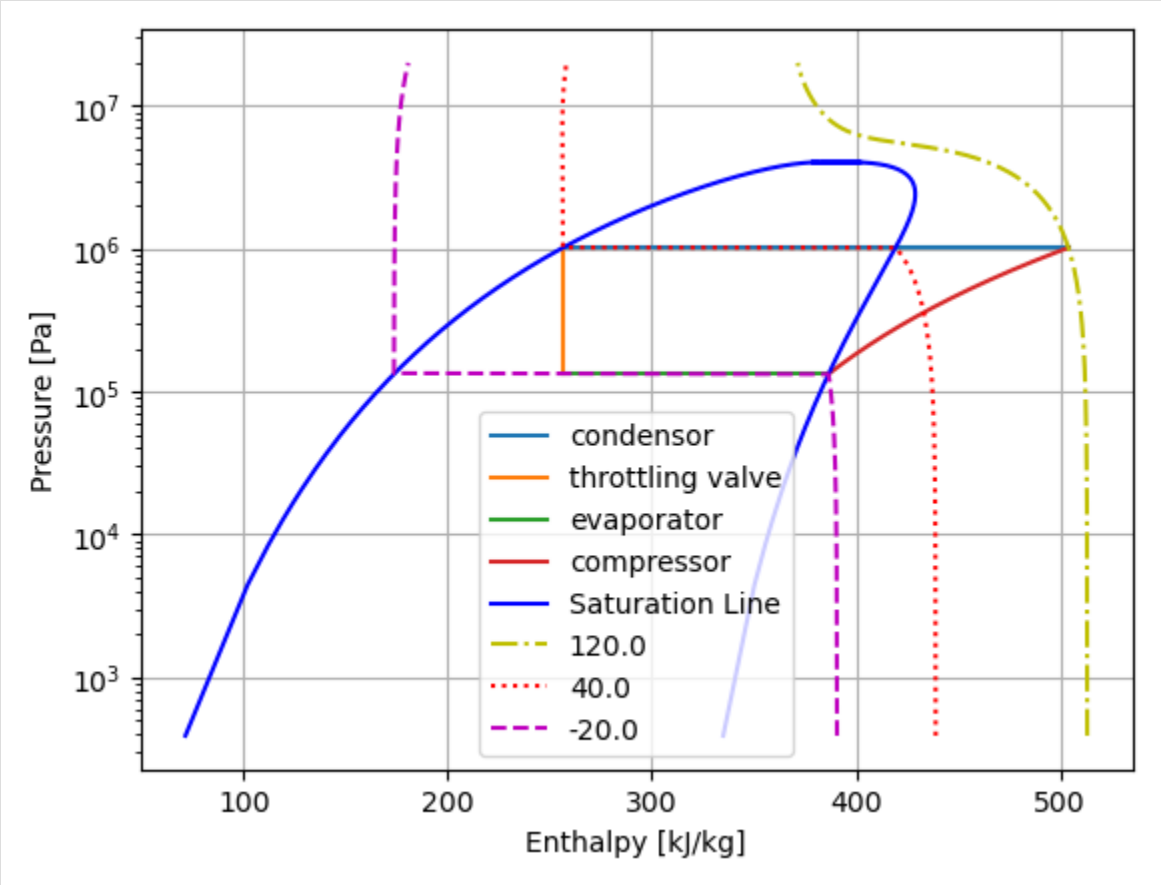
then an array of densities ranging from d_1 to d_2 is used to calculate pressure and enthalpy accordingly.

Skip to main content

```
# import the libraries we'll need
import CoolProp.CoolProp as CP
import numpy as np
import matplotlib.pyplot as plt

#for the compression process 1-2
#building an array of densities
e = 1000   #number of data points for the polytropic process
d_12 = np.linspace(d_1, d_2, e)  # define an array of values from d_1 to d_2
P_12 = P_1 * (d_12/d_1) ** k   #array of pressure based on density for the polytropic compression
h_12 = np.zeros(e)   #an empty array to store enthalpy values

for i in range(e):
   h_12[i] = CP.PropsSI("H", "P", P_12[i], "D", d_12[i], fluid)/1000  # enthalpy values for the poly

#for the cooling process in the condensor in constant pressure
P_23 = np.linspace(P_2, P_3, 1000)  # define an array of pressure values from 2 to 3
h_23 = np.linspace(h_2, h_3, 1000)  # define a linear array of enthalpy values from 2 to 3

#for the process through the throttling valve at constant enthalpy
P_34 = np.linspace(P_3, P_4, 1000)  # define an array of pressure values from 3 to 4
h_34 = np.linspace(h_3, h_4, 1000)  # define a linear array of enthalpy values from 3 to 4

#for the process through the evaporatot at constant pressure
P_41 = np.linspace(P_4, P_1, 1000)  # define an array of pressure values from 4 to 1
h_41 = np.linspace(h_4, h_1, 1000)  # define a linear array of enthalpy values from 4 to 1

#plotting the process on the P-h diagram
plt.plot(h_23, P_23, "-", label="condensor")
plt.plot(h_34, P_34, "-", label="throttling valve")
plt.plot(h_41, P_41, "-", label="evaporator")
plt.plot(h_12, P_12, "-", label="compressor")
plt.legend()


#building the P-h diagram


# define variables
fluid = "R134A"  # define the fluid or material of interest, for full list see CP.Fluidslist()
T_min = CP.PropsSI("Tmin", fluid)  # triple-point temp for the fluid
P_min = CP.PropsSI("P", "T", T_min, "Q", 0, fluid)  # triple-point pressure for the fluid
P_max = 4.03E+6 #approximate critical pressure

P_vals = np.linspace(P_min, P_max, 1000)  # define an array of values from P_min to P_max
Q = 1  # define the fluid quality as 1, which is 100% vapor

enthalpy = [CP.PropsSI("H", "P", P, "Q", Q, fluid)/1000 for P in P_vals]  # call for enthalpy values

plt.plot(enthalpy, P_vals, "-b", label="Saturation Line")  # plot pressure vs enthalpy

Q = 0  # define the fluid quality as 0, which is 100% liquid

enthalpy = [CP.PropsSI("H", "P", P, "Q", Q, fluid)/1000 for P in P_vals] # call for enthalpy values

plt.plot(enthalpy, P_vals, "-b")  # plot pressure vs enthalpy


plt.yscale("log")  # use log scale on y axis
plt.ylabel("Pressure [Pa]")  # give y axis a label
```

```
# Building constant temperature curves

T_up = T_2
T_mid = T_3
T_down = T_4

P_max = 20E+6  # max pressure in the plot set to 20MPa
P_vals = np.linspace(P_min, P_max, 10000)  # define an array of values from P_min to P_max
enthalpy_up = [CP.PropsSI("H", "P", P, "T", T_up, fluid)/1000 for P in P_vals] # call for enthalpy v
enthalpy_mid = [CP.PropsSI("H", "P", P, "T", T_mid, fluid)/1000 for P in P_vals] # call for enthalpy
enthalpy_down = [CP.PropsSI("H", "P", P, "T", T_down, fluid)/1000 for P in P_vals] # call for enthal

plt.plot(enthalpy_up, P_vals, "-.y", label=T_up-273.15)  # plot pressure vs enthalpy
plt.plot(enthalpy_mid, P_vals, ":r", label=T_mid-273.15)  # plot pressure vs enthalpy
plt.plot(enthalpy_down, P_vals, "--m", label=T_down-273.15)  # plot pressure vs enthalpy
plt.legend()
```

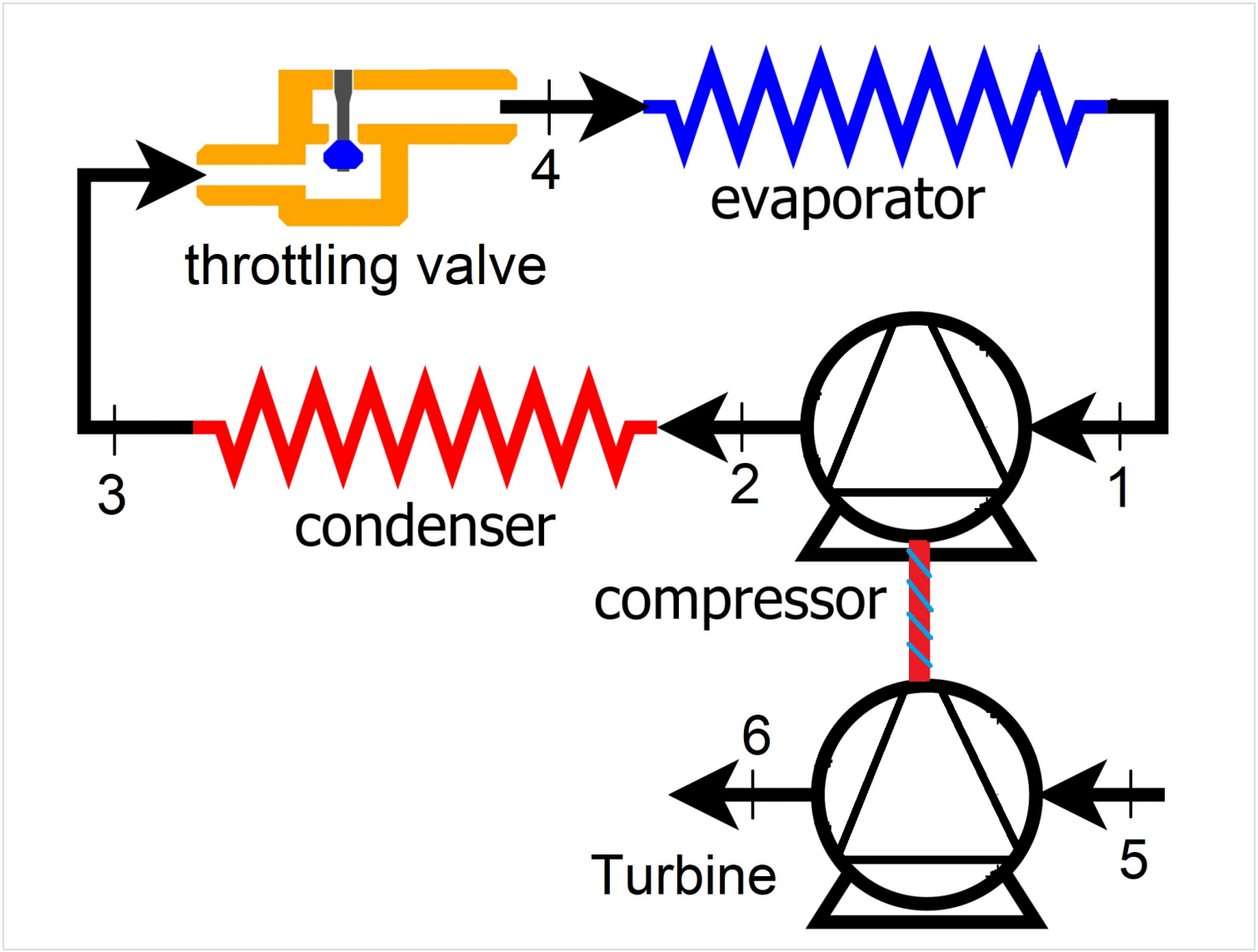
```
<matplotlib.legend.Legend at 0x7f90685ef940>
```



Refrigeration cycle: R134a

consider a refrigration cycle working with R134-a as coolant to support $100\,kW$ of cooling load to a cold storage. The refrigrnt absorbs heat at $-20^{\circ}C$ during evaporation and enters the compressor as saturated vapor. The compressor is

$120^\circ C$. The refrigrant is then cooled down to saturated liquid at $40^\circ C$ in a condensor in constant pressure before entering a throttling valve. The refrigrant is then throttled to $-20^\circ C$ to provide refrigrating fluid for the evaporator. Given the turbine works with compressed air at $1\ MPa$ guage pressure and room temperature, determine the air flow-rate required to provide the required cooling load given the air is discharched to the atmosphere at the turbine outlet and goes through a polytropic process with $n = 1.5$.



Solution Approach

the enthalpy change through the evaporator per kg of the refrigrant is to be calculated. Afterwards, the flow-rate of refrigrant is calculated based on the cooling load, followed by calculation of compressor load based on flow-rate and work calculated per kg of refrigrant. From Q#5 of this chapter:

Skip to main content

```
# import the libraries we'll need
import CoolProp.CoolProp as CP

# define variables
fluid = "R134A"  # define the fluid or material of interest
T_3 = 40 + 273.15 #stete #3 temperature in K
h_3 = CP.PropsSI("H", "T", T_3, "Q", 0, fluid)/1000  # enthalpy of the refrigrant at state #3 in kJ/

h_4 = h_3   #constant enthalpy through a throttling valve

T_4 = -20 + 273.15   #temperature of refrigrant at state #4 in K

P_4 = CP.PropsSI("P", "T", T_4, "Q", 1, fluid)  # pressure of the refrigrant at state #1 in Pa (qual

T_1 = T_4 #temperature at state #1 in K
h_1 = CP.PropsSI("H", "T", T_1, "Q", 1, fluid)/1000  # enthalpy of the refrigrant at state #1 in kJ/

T_2 = 120 + 273.15 #temperature at state #2 in K
P_3 = CP.PropsSI("P", "T", T_3, "Q", 0, fluid)  # pressure of the refrigrant at state #3 in Pa
P_2 = P_3 # pressure of the refrigrant at state #2 in Pa
h_2 = CP.PropsSI("H", "T", T_2, "P", P_2, fluid)/1000  # enthalpy of the refrigrant at state #2 in k

q_c = h_1 - h_4

q_h = h_2 - h_3

w = h_2 - h_1
```

The total cooling load $\dot{Q}_c$ is given to be $100\ kW$; therefore the refrigrant flow-rate is calculated as

$\dot{m}_{R134a} = \dot{Q}_c / q_c$

```
Q_c = 100    #cooling load in kW
m_r134a = Q_c / q_c   #refrigrant flow-rate in kg/s
```

The total work done by the compressor then would be,

$\dot{W}_{R134a} = \dot{m}_{R134a}\, w$

```
W_r134a = m_r134a * w   #the work input by the compressor in kW
```

Now, looking at the turbine-compressor coupling, the work required by the compressor is supported by the turbine in which air goes through a polytropic process. For a polytropic process,

$Pv^n = constant$

$P_1 v_1^k = P_2 v_2^k$

$d(density) = 1/v$

$d_2/d_1 = (P_2/P_1)^{(1/k)}$

$d_2 = d_1\,(P_2/P_1)^{(1/k)}$

Skip to main content

```
#thermodynamic properties and constants for air at state #5
fluid = "Air"
R = 0.287    #air gas constant in kJ/kg.k
P_atm = 101.325   #atmospheric pressure in kPa
P_guage = 1000   #guage presssure at turbine inlet in kPa
P_5 = P_guage + P_atm
T_5 = 25 + 273.15   #compressed air temperature in K
D_5 = CP.PropsSI("D", "T", T_5, "P", P_5, fluid)  #air density at turbine inlet

P_6 = P_atm   #pressure at turbine outlet in kPa
n = 1.5
D_6 = D_5 * (P_6/P_5) ** (1/n)
```

Now, considering the first law of thermodynamics,

$w_turbine = h_5 - h_6$

$\dot{W}_{turbine} = \dot{m}_{air} \, (h_5 - h_6)$

and from the coupling

$\dot{W}_{turbine} = \dot{W}_{R134a}$

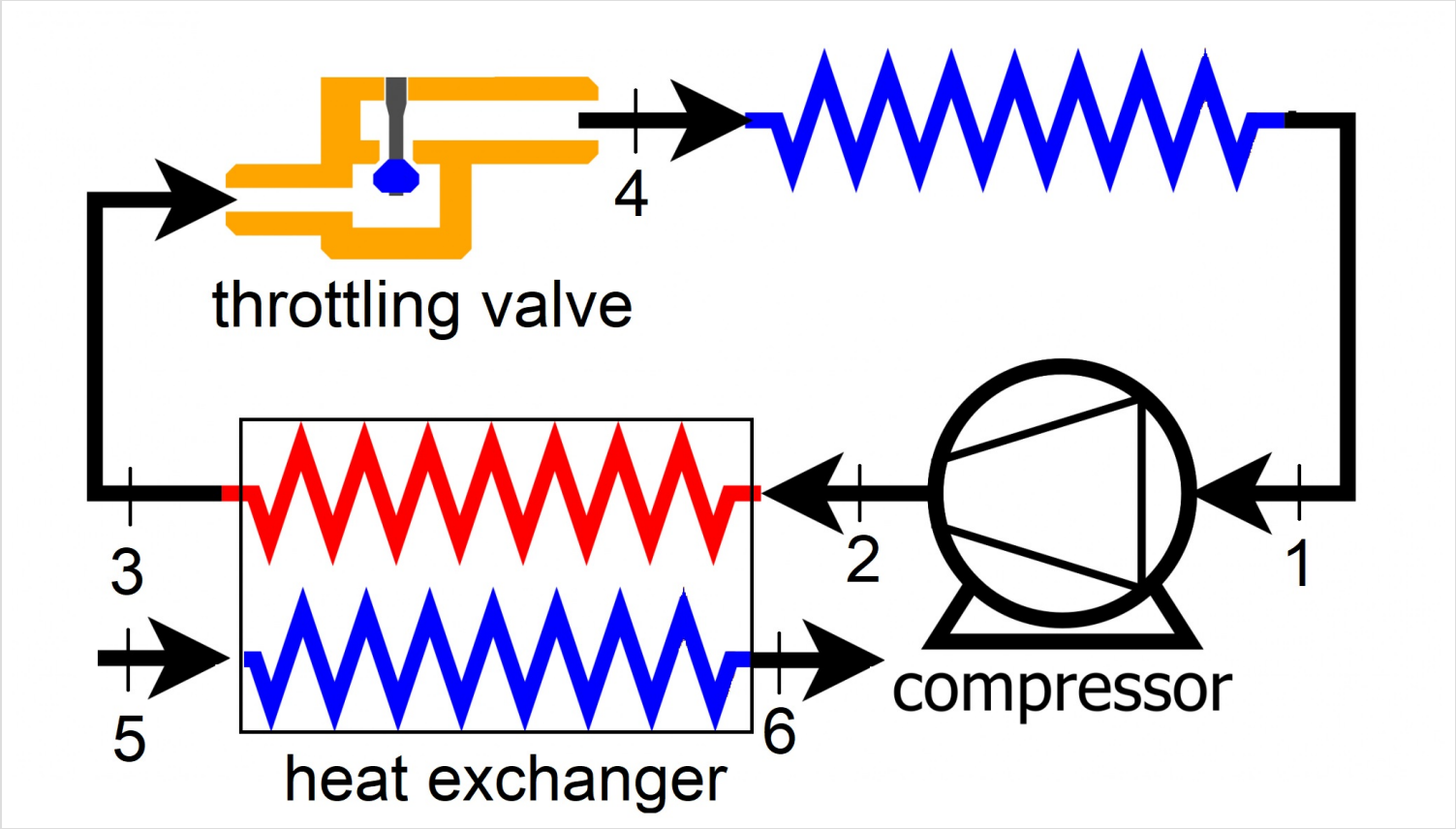
$\dot{m}_{air} = \dot{W}_{R134a} / (h_5 - h_6)$

```
fluid = "Air"
h_5 = CP.PropsSI("H", "T", T_5, "P", P_5, fluid)/1000   #air enthalpy at turbine inlet in kJ/kg
h_6 = CP.PropsSI("H", "D", D_6, "P", P_6, fluid)/1000   #air enthalpy at turbine outlet in kJ/kg
m_air = W_r134a / (h_5 - h_6)   #required air flow-rate in kg/s

print('The required air flow-rate to run the compressor is:', f"{m_air:.1f}", 'kg/s')
```

```
The required air flow-rate to run the compressor is: 0.6 kg/s
```

Refrigeration cycle: Water flow-rate

consider a refrigration cycle working with R134-a as coolant. The refrigrnt absorbs heat at $-20^{\circ}C$ during evaporation and enters the compressor as saturated vapor. The compressor then pressurizes R134-a while the its temperature increases to $120^{\circ}C$. The refrigrant is then cooled down to saturated liquid at $40^{\circ}C$ in constant pressure in a heat exchanger before entering a throttling valve. The refrigrant is then throttled to $-20^{\circ}C$ to provide refrigrating fluid for the evaporator. Assuming the refrigrant exchanges heat with pressurized water in the heat exchanger to heat it up from room temperature to saturated vapor at $110^{\circ}C$, calculate how much water is required per kg of R134a to support cooling power in the heat exhanger.



Solution Approach

based on the first law of thermodynamics, for the heat exchanger

$Q + \dot{m}_i h_i = \dot{m}_e h_e$

Assuming an isolated heat exchanger,

$\dot{m}_5 h_5 + \dot{m}_2 h_2 = \dot{m}_3 h_3 + \dot{m}_6 h_6$

given

$\dot{m}_5 = \dot{m}_6 = \dot{m}_{water}$

and

$\dot{m}_2 = \dot{m}_3 = \dot{m}_{R134a}$

water flow-rate per $1\ kg/s$ of $R134a$ flow-rate would be

$\dot{m}_{water} = (h_2 - h_3)/(h_6 - h_5)$

from Q#5 of this chapter

Skip to main content

```
## import the libraries we'll need
import CoolProp.CoolProp as CP

# define variables
fluid = "R134A"  # define the fluid or material of interest
T_3 = 40 + 273.15 #stete #3 temperature in K
h_3 = CP.PropsSI("H", "T", T_3, "Q", 0, fluid)/1000  # enthalpy of the refrigrant at state #3 in kJ/

h_4 = h_3   #constant enthalpy through a throttling valve

T_4 = -20 + 273.15   #temperature of refrigrant at state #4 in K

P_4 = CP.PropsSI("P", "T", T_4, "Q", 1, fluid)  # pressure of the refrigrant at state #1 in Pa (qual

T_1 = T_4 #temperature at state #1 in K
h_1 = CP.PropsSI("H", "T", T_1, "Q", 1, fluid)/1000  # enthalpy of the refrigrant at state #1 in kJ/

T_2 = 120 + 273.15 #temperature at state #2 in K
P_3 = CP.PropsSI("P", "T", T_3, "Q", 0, fluid)  # pressure of the refrigrant at state #3 in Pa
P_2 = P_3 # pressure of the refrigrant at state #2 in Pa
h_2 = CP.PropsSI("H", "T", T_2, "P", P_2, fluid)/1000  # enthalpy of the refrigrant at state #2 in k

#for the water side
fluid = "water"
T_6 = 110 + 273.15   #temperature of water at heat exchanger exit in K
T_5 = 25 + 273.15   #temperature of water at heat exchanger inlet in K
h_6 = CP.PropsSI("H", "T", T_6, "Q", 1, fluid)/1000   #enthalpy of water at heat exchanger exit in k
P_6 = CP.PropsSI("P", "T", T_6, "Q", 1, fluid)   #pressure of water at heat exchanger exit in Pa
P_5 = P_6   #pressure of water at heat exchanger inlet in Pa assuming constant pressure heating
h_5 = CP.PropsSI("H", "T", T_5, "P", P_5, fluid)/1000   #enthalpy of water at heat exchanger exit in

m_water = (h_2 - h_3) / (h_6 - h_5)

print('The required water flow-rate to support cooling for condensor per kg/s of R134a:', f"{m_water
```

```
The required water flow-rate to support cooling for condensor per kg/s of R134a: 0.1 kg/s
```

Chapter 5

Question #8

Consider a multi-evaporator refrigration system operating with two evaporators to support cooling powers for two seperate compartments set at different temperatures. The low-temperature evaporator1 operates at a constant $-20^{\circ}C$ and the high-temperature evaporator2 operates at $0^{\circ}C$. To optimize the size of the condensor and the evaporators, the refrigrant R134-a is cooled down to saturated liquid at $40^{\circ}C$ in the condensor and is heated up to saturated vapor in the evaporators. The evaporator outputs are then mixed after regulating the pressure for the high-temperature evaporator using a valve and then pressurized in the compressor while increasing to $120^{\circ}C$ in temperature. Given the cooling loads

a) the specific enthalpy of the refrigrant exiting the condenser

b) the specific enthalpy of the refrigrant exiting each of the evaporators

c) the flow-rate of the refrigrant through each evaporator

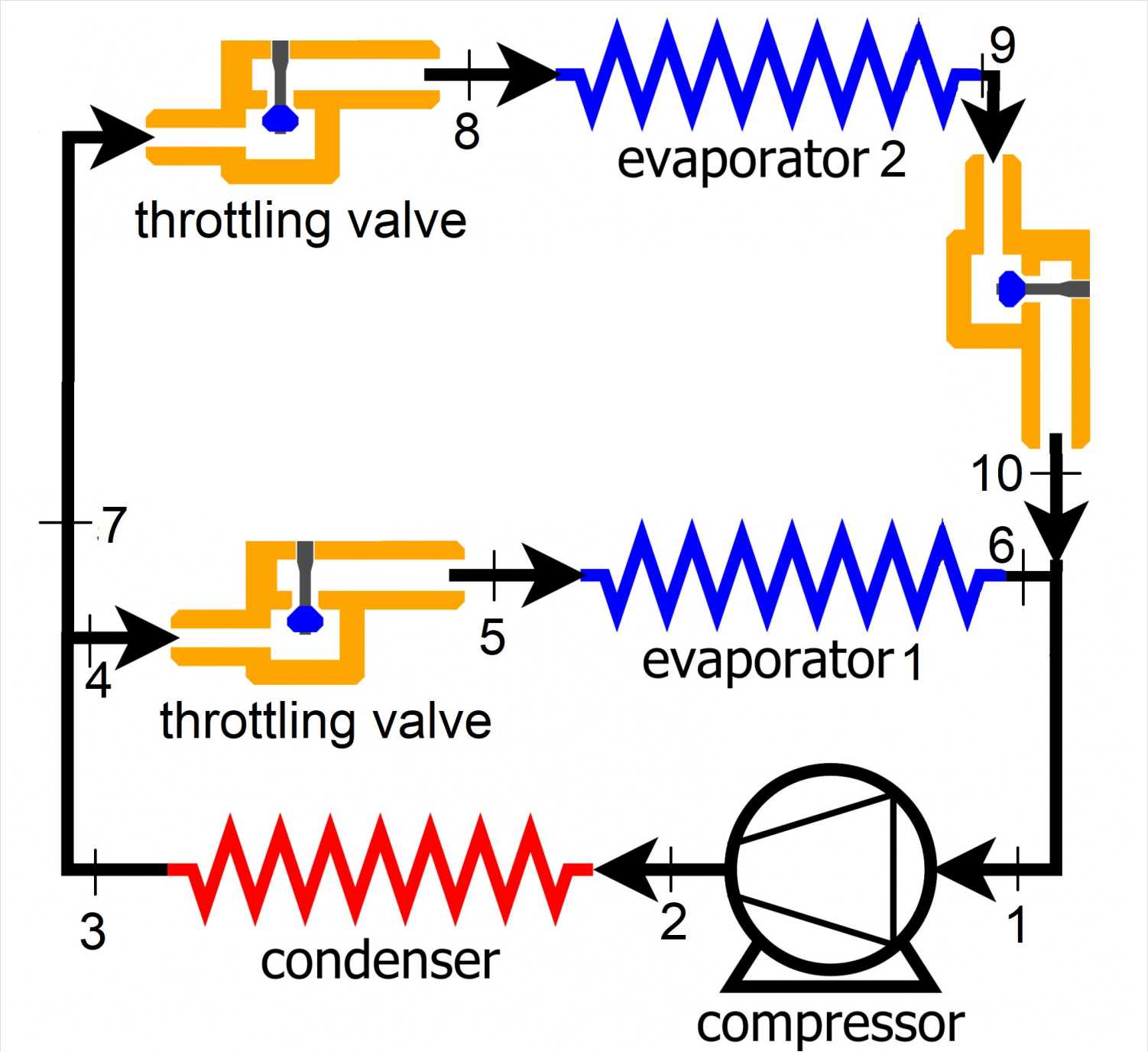
d) the specific enthalpy of the refrigrant entering the compressor

e) the pressure ratio of the compressor

f) the specific enthalpy of the refrigrant exiting the compressor

g) the work required in the compressor

h) the coefficient of performance (COP)



Skip to main content

Solution Approach for a)

the refrigrant exiting the condenser would be a sat liquid at $40^{\circ}C$, therefore

$h_3 = h_{f@40^{\circ}C}$

```
# import the libraries we'll need
import CoolProp.CoolProp as CP

# define variables
fluid = "R134A"  # define the fluid or material of interest
T_3 = 40 + 273.15 #stete #3 temperature in K
h_3 = CP.PropsSI("H", "T", T_3, "Q", 0, fluid)/1000  # enthalpy of the refrigrant at state #3 in kJ/
print('The specific enthalpy of the refrigrant exiting the condenser is:', f"{h_3:.1f}", 'kJ/kg')
```

```
The specific enthalpy of the refrigrant exiting the condenser is: 256.4 kJ/kg
```

Solution Approach for b)

the low-temperature evaporator operates at $-20^{\circ}C$ and the high-temperature one operates at $0^{\circ}C$ and the refrigrant exiting those are at their saturated liquid state to optimize evaporators' performance, therefore

$h_6 = h_{g@-20^{\circ}C}$

$h_9 = h_{g@0^{\circ}C}$

```
T_6 = -20 + 273.15 #stete #6 temperature in K
T_9 = 0 + 273.15 #stete #9 temperature in K

h_6 = CP.PropsSI("H", "T", T_6, "Q", 1, fluid)/1000  # enthalpy of the refrigrant at state #6 in kJ/
h_9 = CP.PropsSI("H", "T", T_9, "Q", 1, fluid)/1000  # enthalpy of the refrigrant at state #9 in kJ/

print('The specific enthalpy of the refrigrant exiting the evaporator 1 is:', f"{h_6:.1f}", 'kJ/kg')
print('The specific enthalpy of the refrigrant exiting the evaporator 2 is:', f"{h_9:.1f}", 'kJ/kg')
```

```
The specific enthalpy of the refrigrant exiting the evaporator 1 is: 386.6 kJ/kg
The specific enthalpy of the refrigrant exiting the evaporator 2 is: 398.6 kJ/kg
```

Solution Approach for c)

to calculate the flow-rate based on cooling load, the cooling capacities for the evaporators are to be calculated first. For Evaporator 1,

$h_5 = h_4$ assuming a constant enthalpy expansion valve

and

Skip to main content

therefore,

$q_{c1} = h_6 - h_5$ the cooling capacity for evaporator 1

and

$\dot{m}_{e1} = \dot{Q}_{c1}/q_{c1}$ the flow-rate for evaporator 1

similarly for evaporator 2,

$h_8 = h_7$ assuming a constant enthalpy expansion valve

and

$h_7 = h_3$

therefore,

$q_{c2} = h_9 - h_8$ the cooling capacity for evaporator 1

and

$\dot{m}_{e2} = \dot{Q}_{c2}/q_{c2}$ the flow-rate for evaporator 1

```
#cooling loads
Q_c1 = 100   #cooling load for evaporator 1 in kW
Q_c2 = 100   #cooling load for evaporator 1 in kW

#for evaporator 1
h_4 = h_3
h_5 = h_4
q_c1 = h_6 - h_5   #cooling capacity for evaporator 1
m_e1 = Q_c1 / q_c1   #refrigrant flow-rate in evaporator 1 in kg/s

#for evaporator 2
h_7 = h_3
h_8 = h_7
q_c2 = h_9 - h_8   #cooling capacity for evaporator 2
m_e2 = Q_c2 / q_c2   #refrigrant flow-rate in evaporator 2 in kg/s

print('The refrigrant flow-rate in evaporator 1 is:', f"{m_e1:.1f}", 'kg/s')
print('The refrigrant flow-rate in evaporator 2 is:', f"{m_e2:.1f}", 'kg/s')
```

```
The refrigrant flow-rate in evaporator 1 is: 0.8 kg/s
The refrigrant flow-rate in evaporator 2 is: 0.7 kg/s
```

Solution Approach for d)

applying the first law and conservation of mass to the mixing point where two outlets from evaporators come together,

$\dot{m}_{10} + \dot{m}_6 = \dot{m}_1$

$\dot{m}_{10}h_{10} + \dot{m}_6 h_6 = \dot{m}_1 h_1$

so

Skip to main content

while

$h_{10} = h_9$ assuming a constant enthalpy expansion

and

$\dot{m}_{10} = \dot{m}_{e2}$

$\dot{m}_6 = \dot{m}_{e1}$

```
#mass flow-rate
m_10 = m_e2
m_6 = m_e1
m_1 = m_10 + m_6

h_10=h_9

h_1 = (m_10 * h_10 + m_6 * h_6)/m_1
print('The specific enthalpy of the refrigrant entering the compressor is:', f"{h_1:.1f}", 'kJ/kg')
```

```
The specific enthalpy of the refrigrant entering the compressor is: 392.3 kJ/kg
```

Solution Approach for e)

For compressor inlet, the pressure would be same pressure as is in evaporator 1 which is the saturation pressure at the low temperature($-20^{\circ}C$)

$P_1 = P_6 = P_{sat@-20^{\circ}C}$

For compressor outlet, the pressure would be same as working pressure for the condenser which is the saturation pressure at $40^{\circ}C$

$P_2 = P_3 = P_{sat@40^{\circ}C}$

$R(pressure\ ratio) = P_2/P_1$

```
P_6 = CP.PropsSI("P", "T", T_6, "Q", 1, fluid)  # sat pressure at state #6 in Pa
P_1 = P_6

P_3 = CP.PropsSI("P", "T", T_3, "Q", 1, fluid)  # sat pressure at state #3 in Pa
P_2 = P_3

R = P_2 / P_1   #the compressor pressure ratio
print('The pressure ratio of the compressor is:', f"{R:.1f}")
```

```
The pressure ratio of the compressor is: 7.7
```

Skip to main content

The specific enthalpy of he refrigrant at the compressor outlet is evaluated based on temperature and pressure

```
#define variables
T_2 = 120 + 273.15 #temperature of R134a at compressor outlet
h_2 = CP.PropsSI("H", "T", T_2, "P", P_2, fluid)/1000  # refrigrant enthalpy at compressor outlet ir
print('The specific enthalpy of the refrigrant exiting the compressor is:', f"{h_2:.1f}", 'kJ/kg')
```

```
The specific enthalpy of the refrigrant exiting the compressor is: 504.1 kJ/kg
```

Solution Approach for g)

Based on the first law,

$\dot{W} = \dot{m}\,(h_2 - h_1)$e

```
W = m_1 * (h_2 - h_1)   #work input for compressor in kW
print('The compressor work input is:', f"{W:.1f}",'kW')
```

```
The compressor work input is: 164.4 kW
```

Solution Approach for h)

$COP = Q_c(cooling\,load)/W(compressor\,work\,input)$

```
cop = (Q_c1 + Q_c2) / W
print('The COP of the cycle is:', f"{cop:.1f}")
```

```
The COP of the cycle is: 1.2
```

Chapter 5

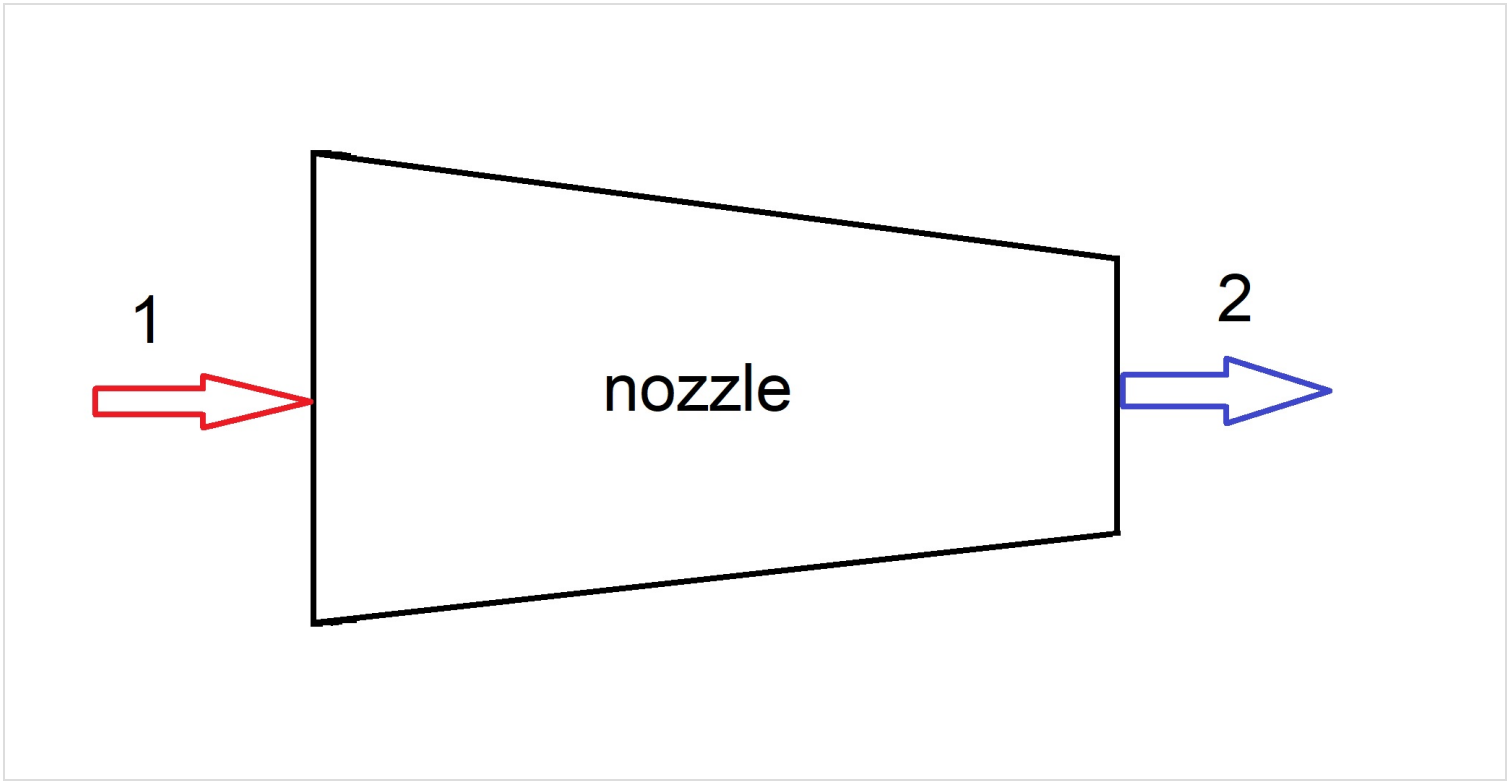
Question #9

$5\ g/s$ of hydrogen at $-20^{\circ}C$ and $5\ bar$ enters a nozzle with inlet and outlet diameters of $40\ mm$ and $15\ mm$. Given the outlet pressure of nozzle to be $2\ bar$, calculate

Skip to main content

b) outlet temperature and velocity of hydrogen assuming ideal gas application

c) outlet temperature and velocity of hydrogen using CoolProp module using the answers from b as the first guess



Solution Approach for a)

from mass flow-rate correlation

$$\dot{m} = \rho_1 A_1 V_1$$

so

$$V_1 = \dot{m}/(\rho_1 A_1)$$

Skip to main content

```python
# import the libraries we'll need
import CoolProp.CoolProp as CP
import numpy as np

# define variables
fluid = "hydrogen"  # define the fluid or material of interest
R = 4124   #fluid gas constant in J/kg.K
C_p = 14307   #fluid Cp in J/kg.K
T_1 = -20 + 273.15   #inlet temperature in K
P_1 = 5e+5   #inlet pressure in Pa
P_2 = 2e+5   #outlet pressure in Pa
m = 0.005   #fluid mass flow rate in kg/s
D_1 = 0.04   #inlet diameter in m
A_1 = np.pi * D_1 **2 /4   #inlet area in m2
rho_1 = CP.PropsSI("D", "T", T_1, "P", P_1, fluid)   #density of fluid at inlet in kg/m3

V_1 = m / (rho_1 * A_1)   #velocity of fluid at inlet

print('The velocity of the fluid at inlet is:', f"{V_1:.1f}", 'm/s')
```

```
The velocity of the fluid at inlet is: 8.3 m/s
```

Solution Approach for b)

from mass conservation

$$\dot{m} = \rho_2 A_2 V_2$$

so

$$V_2 = \dot{m}/(\rho_2 A_2)$$

while from ideal gas assumption

$$P = \rho R T$$

therefore

$$\rho_2 = P_2/(R T_2)$$

so

$$V_2 = \dot{m} R T_2/(P_2 A_2) = \alpha T_2$$

while

$$\alpha = \dot{m} R/(P_2 A_2)$$

from energy conservation

$$h_1 + 1/2 V_1^2 = h_2 + 1/2 V_2^2$$

and from ideal gas assumption

$$\Delta h = C_p (T_2 - T_1)$$

Skip to main content

$C_pT_1 + 1/2V_1^2 = C_pT_2 + 1/2V_2^2$

substituting V_2

$C_pT_1 + 1/2V_1^2 = C_pT_2 + 1/2\alpha^2T_2^2$

organizing for T_2

$(1/2\alpha^2)T_2^2 + C_pT_2 - (C_pT_1 + 1/2V_1^2) = 0$

```
# define variables
D_2 = 0.015   #outlet diameter in m
A_2 = np.pi * D_2 ** 2 /4   #outlet area in m2
alpha = m * R / (P_2 * A_2)

# Coefficients of the quadratic equation ax^2 + bx + c = 0
a = 0.5 * alpha **2
b = C_p
c = -1 * (C_p * T_1 + 0.5 * V_1 ** 2)

# Calculate the discriminant (the value inside the square root)
discriminant = b**2 - 4*a*c
# Two real solutions
x1 = (-1*b + np.sqrt(discriminant)) / (2*a)
x2 = (-1*b - np.sqrt(discriminant)) / (2*a)

# to pick the correct positive value for temperature at outlet
if x1 > 0:
   T_2 = x1
else:
   T_2 = x2

V_2 = alpha * T_2   # velocity at outlet
T_2C = T_2 - 273.15   #temperature at outlet in celcius
print('The velocity of the fluid at outlet is:', f"{V_2:.2f}", 'm/s')
print('The temperature of the fluid at outlet is:', f"{T_2C:.2f}", 'celcius')
```

```
The velocity of the fluid at outlet is: 147.25 m/s
The temperature of the fluid at outlet is: -20.76 celcius
```

Solution Approach for b)

again from energy conservarion

$h_1 + 1/2V_1^2 = h_2 + 1/2V_2^2$

while h_2 is obtained from coolprop based on pressure and temperature and

$V_2 = \dot{m}/(\rho_2 A_2)$

while ρ_2 is obtained from coolprop likewise, therefore

$h_2 + (\dot{m}^2/(2A_2^2))(1/\rho_2^2) = h_1 + 1/2V_1^2$

setting β as

the difference between β and the left hand side of the previous equation calculated by CoolProp would be the error of calculations used along with a tunning parameter (z) in a tial-error method.

now, a trial and error method is to be used to obtain temperature (and veocity) using coolprop and the previous equation

```
h_1 = CP.PropsSI("H", "T", T_1, "P", P_1, fluid)   #enthalpy of fluid at inlet in J/kg
betha = h_1 + 0.5 * V_1 **2

#obtaining enthalpy and density based on the guess value
h_2 = CP.PropsSI("H", "T", T_2, "P", P_2, fluid)   #guess enthalpy of fluid at outlet in J/kg
rho_2 = CP.PropsSI("D", "T", T_2, "P", P_2, fluid)   #guess density of fluid at outlet in kg/m3
e = betha - h_2 - 0.5 * (m / (rho_2 * A_2)) ** 2   #error based on the guess

#now starting a loop to implement the trila and error method
i = 1
z = 0.0001    #tunning parameter to tune calculation error and apply to the new temperature guess

while np.absolute(e) > 1 :
    T_2 = T_2 + e * z    #new temperature guess based on the error (e) and the tunning parameter (z)
    h_2 = CP.PropsSI("H", "T", T_2, "P", P_2, fluid)   #new guess enthalpy of fluid at outlet in J/k
    rho_2 = CP.PropsSI("D", "T", T_2, "P", P_2, fluid)   #new guess density of fluid at outlet in kg
    e = betha - h_2 - 0.5 * (m / (rho_2 * A_2)) ** 2   #new error based on the new guess
    i += 1
    if i == 1000:
        break

# note the difference between two consecutive temperature guesses is e * z therefore an e < 1 means

V_2 = m / (rho_2 * A_2)   #velocity calculated based on temperature
T_2C = T_2 - 273.15   #temperature at outlet in celcius

print('The velocity of the fluid at outlet based on CoolProp is:', f"{V_2:.2f}", 'm/s')
print('The temperature of the fluid at outlet based on CoolProp is:', f"{T_2C:.2f}", 'celcius')
```

```
The velocity of the fluid at outlet based on CoolProp is: 147.49 m/s
The temperature of the fluid at outlet based on CoolProp is: -20.71 celcius
```

6. Entropy and the Second Law of Thermodynamics

Let's do some problems based on concepts on Chapter 6 from the e-textbook.

```
import numpy as np
```