EGE UNIVERSITY
SCHOOL OF ENGINEERING
DEPARTMENT OF
MECHANICAL ENGINEERING

MACHINE LAB

	13.15-15.00				15.15-17.00			
	W1	W2	W3	W4	W1	W2	W3	W4
Temperature Calibration	A	B	C	D	E	F	G	H
Specific heat in solids by using Dewar calorimeter	B	C	D	A	H	E	F	G
Pressure Calibration	C	D	A	B	G	H	E	F
Water-water heat pump experiment	D	A	B	C	F	G	H	E

	13.15-15.00				15.15-17.00			
	W5	W6	W7	W8	W5	W6	W7	W8
Humidity Calibration	A	B	C	D	E	F	G	H
Water density measurement	B	C	D	A	H	E	F	G
winter air conditioning	C	D	A	B	G	H	E	F
Double pipe heat exchanger	D	A	B	C	F	G	H	E

	13.15-15.00				15.15-17.00			
	W9	W10	W11	W12	W9	W10	W11	W12
Local Pressure drop mesurements	A	B	C	D	E	F	G	H
Commercial refrigeration system performance	B	C	D	A	H	E	F	G
summer air conditioning	C	D	A	B	G	H	E	F
Shell & Tube heat exchanger	D	A	B	C	F	G	H	E

Ege University, School of Engineering Department of Mechanical Engineering

Name of Experiment	TEMPERATURE CALIBRATION

Aim of Experiment

Thermometer calibration will be carried out.

Experimental set-up

JULABO FK30-SL calibration bath will be used in this experiment. Calibration bath is shown in figüre 1. It has a calibration range of -30°C to 200°C by using a special heat transfer fluid.

Figure 1. Calibration Bath JULABO FK30-SL

Equipment to be used in temperature calibration:

1. Calibration bath JULABO FK30-SL
2. Barometer
3. K Type thermocouple
4. Alcohol termometer
5. Reference PT1000 resistance thermometer an reading unit

Experiment:

Thermocouple and alcohol termometer placed at the top of the calibration bath through the holders. Calibration bath is set to 0° C and when the bath reach to set value it is read **at least 5 different students** and recorded. Reference thermometer is also read and recorded in the same fashion. Then calibration bath set point increase 5° C and this process is repeated until 50 ° C

Evaluation of results

For each measurement point (with the data taken by five different students) avarage and standard deviation is calculated.

Average : $\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}$

Standard deviation $s = \sqrt{\frac{\sum_{i=1}^{n} (X_i - \bar{X})^2}{n-1}}$

Standard deviation of the mean value $s_x = \frac{s}{\sqrt{n}}$

For(K=2) = %95 if 5 (N-1 = 4) sets of data is taken t_{95}=2.776

Table 1 Student's t statistical values for Gauss normal distribution

N-1	%50	%60	%70	%80	%90	%95	%98	%99	%99,5	%99,8
1	1	1,376	1,963	3,078	6,314	12,7100	31,82	63,66	127,3	318,3
2	0,816	1,061	1,386	1,886	2,92	4,3030	6,965	9,925	14,09	22,33
3	0,765	0,978	1,25	1,638	2,353	3,1820	4,541	5,841	7,453	10,21
4	0,741	0,941	1,19	1,533	2,132	**2,7760**	3,747	4,604	5,598	7,173
5	0,727	0,92	1,156	1,476	2,015	2,5710	3,365	4,032	4,773	5,893
6	0,718	0,906	1,134	1,44	1,943	2,4470	3,143	3,707	4,317	5,208
7	0,711	0,896	1,119	1,415	1,895	2,3650	2,998	3,499	4,029	4,785
8	0,706	0,889	1,108	1,397	1,86	2,3060	2,896	3,355	3,833	4,501
9	0,703	0,883	1,1	1,383	1,833	2,2620	2,821	3,25	3,69	4,297
10	0,7	0,879	1,093	1,372	1,812	2,2280	2,764	3,169	3,581	4,144
11	0,697	0,876	1,088	1,363	1,796	2,2010	2,718	3,106	3,497	4,025
12	0,695	0,873	1,083	1,356	1,782	2,1790	2,681	3,055	3,428	3,93
13	0,694	0,87	1,079	1,35	1,771	2,1600	2,65	3,012	3,372	3,852
14	0,692	0,868	1,076	1,345	1,761	2,1450	2,624	2,977	3,326	3,787
15	0,691	0,866	1,074	1,341	1,753	2,1310	2,602	2,947	3,286	3,733
16	0,69	0,865	1,071	1,337	1,746	2,1200	2,583	2,921	3,252	3,686
17	0,689	0,863	1,069	1,333	1,74	2,1100	2,567	2,898	3,222	3,646
18	0,688	0,862	1,067	1,33	1,734	2,1010	2,552	2,878	3,197	3,61
19	0,688	0,861	1,066	1,328	1,729	2,0930	2,539	2,861	3,174	3,579
20	0,687	0,86	1,064	1,325	1,725	2,0860	2,528	2,845	3,153	3,552
21	0,686	0,859	1,063	1,323	1,721	2,0800	2,518	2,831	3,135	3,527
22	0,686	0,858	1,061	1,321	1,717	2,0740	2,508	2,819	3,119	3,505
23	0,685	0,858	1,06	1,319	1,714	2,0690	2,5	2,807	3,104	3,485
24	0,685	0,857	1,059	1,318	1,711	2,0640	2,492	2,797	3,091	3,467
25	0,684	0,856	1,058	1,316	1,708	2,0600	2,485	2,787	3,078	3,45
26	0,684	0,856	1,058	1,315	1,706	2,0560	2,479	2,779	3,067	3,435

27	0,684	0,855	1,057	1,314	1,703	2,0520	2,473	2,771	3,057	3,421
28	0,683	0,855	1,056	1,313	1,701	2,0480	2,467	2,763	3,047	3,408
29	0,683	0,854	1,055	1,311	1,699	2,0450	2,462	2,756	3,038	3,396
30	0,683	0,854	1,055	1,31	1,697	2,0420	2,457	2,75	3,03	3,385
40	0,681	0,851	1,05	1,303	1,684	2,0210	2,423	2,704	2,971	3,307
50	0,679	0,849	1,047	1,299	1,676	2,0090	2,403	2,678	2,937	3,261
60	0,679	0,848	1,045	1,296	1,671	2,0000	2,39	2,66	2,915	3,232
80	0,678	0,846	1,043	1,292	1,664	1,9900	2,374	2,639	2,887	3,195
100	0,677	0,845	1,042	1,29	1,66	1,9840	2,364	2,626	2,871	3,174
120	0,677	0,845	1,041	1,289	1,658	1,9800	2,358	2,617	2,86	3,16
1000	0,674	0,842	1,036	1,282	1,645	1,9600	2,326	2,576	2,807	3,09

Then uncertainity of the measurement can be calculated as:

$U = \overline{X} \pm ts_x \qquad (K = 2)$ This range means assuming data distributed by the gauss distribution 95 % (K=2*σ) of the measured data will be fall into this range. With other Worlds, if we have carried out very high number of measurements (then data fit to gauss distribution) deviation of average compare to avarage data of very high number will be uncertainity.

In uncertainty term the concept biass is also important. When avarage value of each data point is calculated both for our device and reference device, the difference of these two values may not be same and some differences exist between these two average value sets.In Order to take this effect into acount a new uncertainity term is defined as:
$U_{ADD} = B + t_{95} S_x$ This is call addition method for uncertainity, the other method is root square addition method. This will be our preferred uncertainity calculation method.

$U_{RSS} = [B^2 + (t_{95} S_x)^2]^{(1/2)}$

where B is bias and defined as

$B = \overline{X} - \overline{X}_{reference} \qquad (K = 2)$

Biass can be ignored in uncertainity calculation if a curve fitted correction is applied in between (if B=0 then $U_{RSS} = U$

In order to achieve this correlation, polynomial least square curve fitting can be used.
Additional information about simple polynomial curve fitting is given as an appendix :

Requirements in the lab report:

1. Report language will be english
2. Introduction and definition of the experiment
3. Lab calibration measurements
4. Basic information about temperature calibration
5. Calibration report for alcohol termometer
   Average reference thermometer temperatures,
   Avarage alcohol thermometer temperatures
   Reference thermometer uncertainity values
   Alcohol thermometer uncertainity values
   Bias values

Combined U_{RSS} uncertainity values
Least square correction function and it's plot

6. Calibration report for thermocouple
Average reference thermometer temperatures,
Avarage alcohol thermometer temperatures
Reference thermocouple uncertainity values
thermocouple uncertainity values
Bias values
Combined U_{RSS} uncertainity values
Least square correction function and it's plot

Experimental values

	Referance thermometer	**Alcohol thermometer**	**Thermocouple**
Calibration bath T_1			
Calibration bath T_1			
Calibration bath T_1			
Calibration bath T_1			
Calibration bath T_1			
T_1 Average			
T_1 Standard deviation			
T_1 Uncertainity			
Calibration bath T_2			
Calibration bath T_2			
Calibration bath T_2			
Calibration bath T_2			
Calibration bath T_2			
T_2 Average			
T_2 Standard deviation			
T_2 Uncertainity			
Calibration bath T_3			
Calibration bath T_3			
Calibration bath T_3			
Calibration bath T_3			
Calibration bath T_3			
T_3 Average			
T_3 Standard deviation			
T_3 Uncertainity			
Calibration bath T_4			
Calibration bath T_4			
Calibration bath T_4			
Calibration bath T_4			
Calibration bath T_4			
T_4 Average			

T_4 Standard deviation			
T_4 Uncertainity			
Calibration bath T_5			
Calibration bath T_5			
Calibration bath T_5			
Calibration bath T_5			
Calibration bath T_5			
T_5 Average			
T_5 Standard deviation			
T_5 Uncertainity			
Calibration bath T_6			
Calibration bath T_6			
Calibration bath T_6			
Calibration bath T_6			
Calibration bath T_6			
T_6 Average			
T_6 Standard deviation			
T_6 Uncertainity			
Calibration bath T_7			
Calibration bath T_7			
Calibration bath T_7			
Calibration bath T_7			
Calibration bath T_7			
T_7 Average			
T_7 Standard deviation			
T_7 Uncertainity			
Calibration bath T_8			
Calibration bath T_8			
Calibration bath T_8			
Calibration bath T_8			
Calibration bath T_8			
T_8 Average			
T_8 Standard deviation			
T_8 Uncertainity			
Calibration bath T_9			
Calibration bath T_9			
Calibration bath T_9			
Calibration bath T_9			
Calibration bath T_9			
T_9 Average			
T_9 Standard deviation			
T_9 Uncertainity			

Ege University, School of Engineering Department of Mechanical Engineering

Name of Experiment	PRESSURE CALIBRATION

Aim of Experiment
Thermometer calibration will be carried out.

Experimental set-up

GE DRUCK PV622+DPI620+PM620 pressure calibration system in our lab will be used fort his experiment. This device is shown in figure 2. Device operates in environmental temperature range of -10°C to 40°C with an accuracy of ±0.025%. Barometric pressure accuracy is ±0.15 mba..

Figure 2. Pressure calibration device GE DRUCK PV622+DPI620+PM620

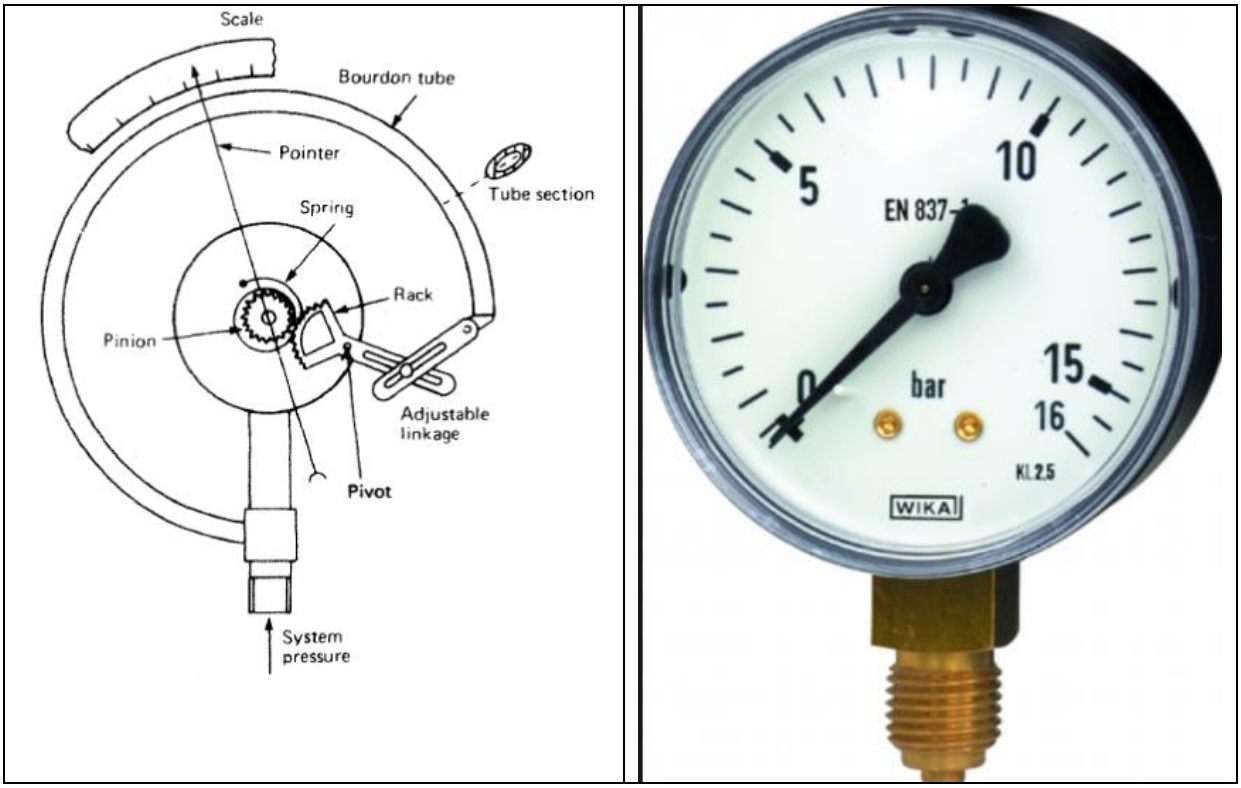


Figure 3. Bourden manometer

Devices to be used in the experiment

1. Pressure calibration device GE DRUCK PV622+DPI620+PM620
2. A Bourden type manometer

Experiment

Pressure calibration device has two types of control a normal air pressure pump and a srew pum for accurate adjustments. Bourden manometer is connected to the calibrator and adjusted to the set pressures required to measure. and after seting pressure value to its required set point, it is read **at least 5 different students** and recorded. Reference pressure read from the calibrator screen is also read and recorded in the same fashion. Then calibrator set point increase 0.2 bar and this process is repeated until Bourden manometer scale is covered.

Evaluation of results

For each measurement point (with the data taken by five different students) avarage and standard deviation is calculated.

Average : $\bar{X} = \dfrac{\sum_{i=1}^{n} X_i}{n}$

Standard deviation $s = \sqrt{\dfrac{\sum_{i=1}^{n} (X_i - \bar{X})^2}{n-1}}$

Standard deviation of the mean value $s_x = \frac{s}{\sqrt{n}}$

For(K=2) = %95 if 5 (N-1 = 4) sets of data is taken t_{95}=2.776 (Table 1)

Then uncertainity of the measurement can be calculated as:

$U = \overline{X} \pm ts_x \qquad (K = 2)$ This range means assuming data distributed by the gauss distribution 95 % (K=2*σ) of the measured data will be fall into this range. With other Worlds, if we have carried out very high number of measurements (then data fit to gauss distribution) deviation of average compare to avarage data of very high number will be uncertainity.

In uncertainty term the concept biass is also important. When avarage value of each data point is calculated both for our device and reference device, the difference of these two values may not be same and some differences exist between these two average value sets.In Order to take this effect into acount a new uncertainity term is defined as:
$U_{ADD} = B + t_{95} S_x$ This is call addition method for uncertainity, the other method is root square addition method. This will be our preferred uncertainity calculation method.

$U_{RSS} = [B^2 + (t_{95} S_x)^2]^{(1/2)}$

where B is bias and defined as

$B = \overline{X} - \overline{X}_{reference} \qquad (K = 2)$

Biass can be ignored in uncertainity calculation if a curve fitted correction is applied in between (if B=0 then $U_{RSS} = U$

In order to achieve this correlation, polynomial least square curve fitting can be used.
Additional information about simple polynomial curve fitting is given as an appendix :

Requirements in the lab report:

1. Report language will be english
2. Introduction and definition of the experiment
3. Lab calibration measurements
4. Basic information about pressure calibration
5. Calibration report for Bourden type manometer
   Average reference pressures,
   Avarage bourden manometer
   Reference reference pressure uncertainity values
   bourden manometer pressure uncertainity values
   Bias values
   Combined U_{RSS} uncertainity values
   Least square correction function and it's plot

Experimental values and results

Atmospheric pressure:

	Referance pressure	**Bourden Manometer**

	(bar)	pressure (bar)
P1		
P1		
P1		
P1		
P1		
P_1 Average		
P_1 Standard deviation		
P_1 Uncertainity		
P2		
P2		
P2		
P2		
P2		
P_2 Average		
P_2 Standard deviation		
P_2 Uncertainity		
P3		
P3		
P3		
P3		
P3		
P_3 Average		
P_3 Standard deviation		
P_3 Uncertainity		
P4		
P4		
P4		
P4		
P4		
P_4 Average		
P_4 Standard deviation		
P_4 Uncertainity		
P5		
P5		
P5		
P5		
P5		
P_5 Average		
P_5 Standard deviation		
P_5 Uncertainity		
P6		

P6		
P6		
P6		
P6		
P_6 Average		
P_6 Standard deviation		
P_6 Uncertainity		
P7		
P7		
P7		
P7		
P7		
P_7 Average		
P_7 Standard deviation		
P_7 Uncertainity		
P8		
P8		
P8		
P8		
P8		
P_8 Average		
P_8 Standard deviation		
P_8 Uncertainity		
P9		
P9		
P9		
P9		
P9		
P_9 Average		
P_9 Standard deviation		
P_9 Uncertainity		
P10		
P10		
P10		
P10		
P10		
P_{10} Average		
P_{10} Standard deviation		
P_{10} Uncertainity		
P11		
P11		
P11		
P11		

P11		
P_{11} Average		
P_{11} Standard deviation		
P_{11} Uncertainity		

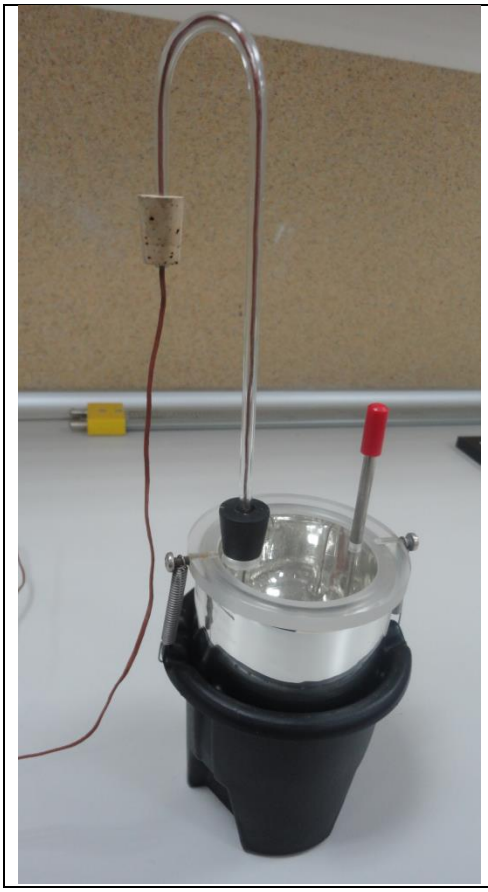
Ege University, School of Engineering **Department of Mechanical Engineering**

Name of Experiment	**DETERMINATION OF SPECIFIC HEAT BY USING DEWAR CALORIMETER**

Aim of Experiment

Determination of Specific heat of different metals by using dewar calorimeter

Devices to be used in the experiment



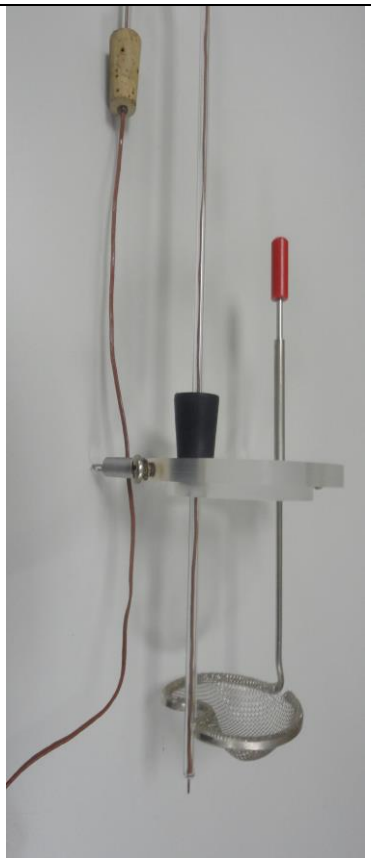


Figure 3. Dewar calorimeter

Figure 4. Nuve EV 018 Owen to heat samples to specific temperatures and sample pieces

In this experiment, dewar calorimeter will be used to measure specific heat of metal samples. Calorimeter is a thermally insulated container. At the top of the container there are openings to hold a thermocouple and stirrer/maple holder. Insulated bottle is located inside of a plastic cup. Internal diameter of the insulated container is 7 cm and the length is about 9 cm. Total water capacity is approximately 300 cc. Top of the calorimeter is connected to the body with two springs. In order to open top, springs are removed.

In our experiment, specific heat of steel and aluminium metal pieces will be measured. In order to heat the samples Nuve EV 018 owen will be used.

Experiment

Sample is measured in weight scale and mass is recorded. Sample is put into the owen and owen temperature is set to 95° C. Calorimeter is weighted as empty and then weighted with filled water. (Calorimeter is filled with approximetly 250 cc of pure water. And temperature of the water is recorded continiously. Sample is taken from the owen and put into the calorimeter and temperature of the water is continued to be observed. When system is reached to the equilibrium temperature, it is recorded. Specific heat can be calculated by using the following energy balance equations

C_s : Specific heat of the sample
C_w : Specific heat of the water
m_s : mass of the sample
m_w : mass of water
T_w : Initial tempearture of water
T_s : Initial temperature of sample
T_M : Equilibrium temperature of water and sample

$C_s = C_w * (m_w/m_s)(T_m - T_w)/(T_s - T_m)$

Requirement of the lab report

- Theory of energy balance and working of the calorimetry
- Measured values
- Full calculations for every steps.

– Comparison of experimentaly found values of specific heat with the table values from the booksand

Repeat this for each sample

Experimental values and results

Name of the sample	
Weight of the sample:	
Weight of the empty Dewar calorimeter:	
Weight of the Dewar calorimeter filled with water:	
Weight of the water in Dewar calorimeter	
Initial temperature of water	
Initial temperature of the sample	
Equilibrium temperature of water and sample	

Name of the sample	
Weight of the sample:	
Weight of the empty Dewar calorimeter:	
Weight of the Dewar calorimeter filled with water:	
Weight of the water in Dewar calorimeter	
Initial temperature of water	
Initial temperature of the sample	
Equilibrium temperature of water and sample	

Name of the sample	
Weight of the sample:	
Weight of the empty Dewar calorimeter:	
Weight of the Dewar calorimeter filled with water:	
Weight of the water in Dewar calorimeter	
Initial temperature of water	
Initial temperature of the sample	
Equilibrium temperature of water and sample	

Ege University, School of Engineering Department of Mechanical Engineering

Name of Experiment	Water to Water Heat pump

Aim of Experiment

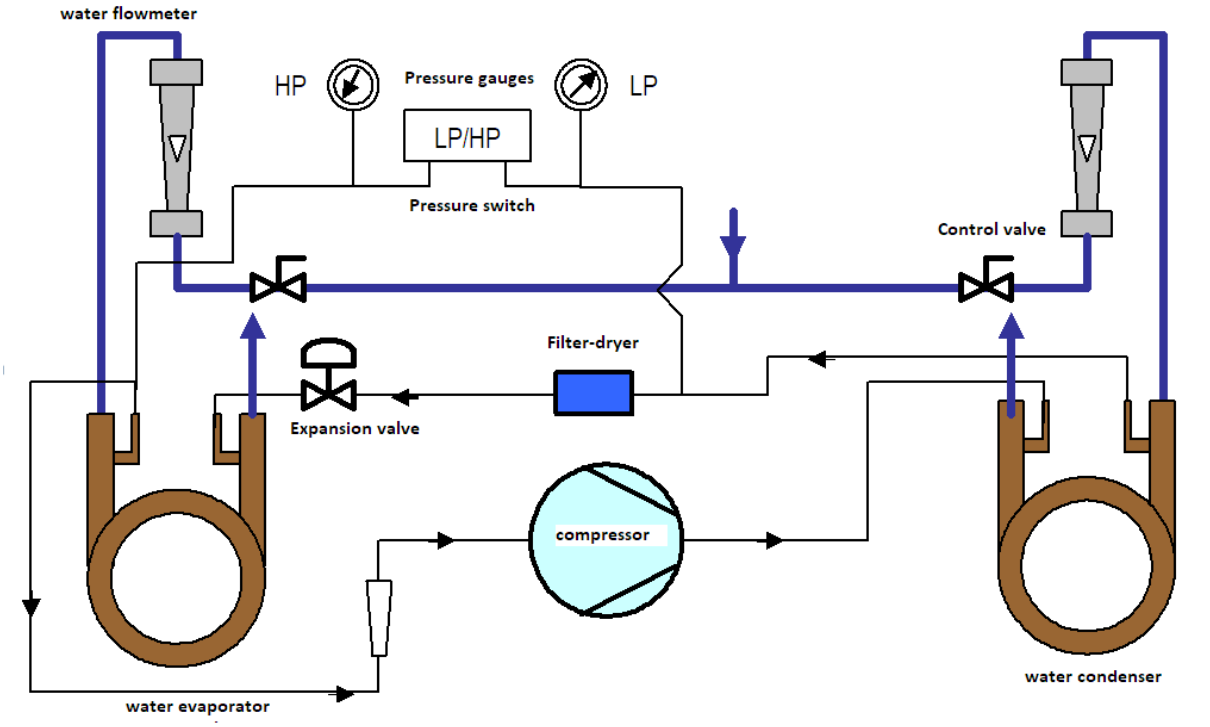
The operating principles of a water-water heat pump will be explained and thermodynamic calculations will be caried out, coeefficient of performance and other thermodynamic properties will be calculated by using experimental values. Cycle will be represanted on P-h and T-s diagrams.

Experimental system:

The experimental system to carry out this test is given by figure 5. System has double pipe evaporator and condenser with water passege in the other side of these heat exchangers.

Figure 5. Water-Water heat pump basic experimental setup

Şekil 1. Temel iklimlendirme eğitim seti deney düzeneği



Devices to be used in the experiment

1. Water to water heat pump experiimental setup
2.

Experiment

1) Set condenser water flow rate to 400 L/h, and evaporator flow rate to 50 L/h and start the system.
2) When you think that the system is in equilibrium reacord the system operating data
   - Water flow rate of condenser
   - Water flow rate of evaporator
   - Refrigerant(R134a) flow rate
   - Water inlet and exit temperatures of condenser
   - Water inlet and exit temperatures of evaporator
   - Compressor power input values(V, I and cos(ϕ))
   -
3) Set condenser water flow rate to 200 L/h, and then 100 L/h. Repeat the previous procedure for each case of flow rates. Deney Sonuçlarının Değerlendirilmesi

Relations to calculate system performance

Power input: $P = U.I_c\ Cos\phi$ [W]

Condenser heat transfer: $\dot{Q}_{cond} = \dot{m}_{cond,water\,flow}.C_{p,water}\,(T_6 - T_5)$ [W]

Condenser heat transfer: $\dot{Q}_{evap} = \dot{m}_{evap,water}.c_{p,water}\,(T_7 - T_8)$ [W]

Coefficient of performance condenser $COP_{cond} = \dfrac{\dot{Q}_{cond}}{P}$

Coefficient of performance evaporator $COP_{evap} = \dfrac{\dot{Q}_{evap}}{P}$

Calculate COP values in different operation conditions and plot the results and discuss these results. REfrigerant used in this cycle is R134a. Show the cycle in Ph and TS diagrams. And analyse the cycle

Isı pompası çevriminde çalışma akışkanı olarak R134a kullanılmaktadır. Farklı çalışma durumları için çevrimi P-h diyagramlarında göstererek analizlerini yapınız.

Requirements from lab report

- DEscription of experiment.
- Mesurement values.
- COP's power input, condenser and evaporator heat transfer and their variarion with load (condenser flow rate) change. Discuss the results.

Experimental results

Number of measurement	**1**	**2**	**3**
Voltage U [Volt]			
Compressor Ic [Amper]			
AC phase angle CosΦ			
Compressor inlet pressure, P_1 [kPa]			
Compressor inlet temperature, [˚C]			
Compressor exit pressure, [kPa]			
Compressor exit temperature, [˚C]			
Condenser exit temperature, [˚C]			
TGV çıkışındaki sıcaklık, [˚C]			
Evaporator water inlet temperature, [˚C]			
Evaporator water exit temperature, [˚C]			
Evaporatorwater mass flow rate, $\dot{m}_{evap,su}$ [L/h]			
Condenser water inlet temperature, [˚C]			
Condenser water exit temperature, [˚C]			

Name of Experiment	HUMIDIDY CALIBRATION

Aim of Experiment
Humidity calibration will be carried out.

Experimental set-up
TESTO HUMINATOR moisture calibration device will be used for this experiments. In this experiment. Equipment operation range is between 15°C to 40°C dry bulb temperature and %5 to %95 relative humidity. Device has an accuracy level of +/- 2 % RH and +/- 0.5 °C temperature.

Figure 1. TESTO HUMINATOR moisture calibration system.

Devices to be used in the experiment
1. Moisture calibration device TESTO HUMINATOR
2. Testo data reader unit.
3. Psychrometric table (www.turhancoban.com)
4. Testo moisture and temperature sensor

Experiment
Moisture Probes to be calibrated is placed into the calibration chamber through enrance holes. Moisture calibration device is set up to desired humidity and moisture values. When the set values reached values of reference (calibration device) and calibrated device is read. (Suggested moisture calibration points 30,40,50 and 60% and temperatures,20,25,30,35 C)
Establishing of equilibrium is taking a while for this experiment some patiens is required. When the system is in equilibrium, moisture and temperature of the device(s) to be calibrated and reference temperature and moisture values (TESTO HUMINATOR) are read . For each caibration point each student will read the value and taken additional sets from four additional students. It is preferable to take additional values from diiferent students in each calibration point to minimize human error. In order to accelarate total time required for calibration device to establih equilibrium, Start with 30% moisture and change the temperatures and then increase the moisture to 40…

Evaluation of experiment results.

For each measurement point (with the data taken by five different students) average and standard deviation is calculated for both temperature and relative humidity values.

Average : $\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}$

Standard deviation $s = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n-1}}$

Standard deviation of the mean value $s_x = \frac{s}{\sqrt{n}}$

For(K=2) = %95 if 5 (N-1 = 4) sets of data is taken t_{95}=2.776 (see table 1 in Temperature calibration section for **Student's t statistical values for Gauss normal distribution**
Then uncertainity of the measurement can be calculated as:

$U = \bar{X} \pm ts_x \qquad (K = 2)$ This range means assuming data distributed by the gauss distribution 95 % (K=2*σ) of the measured data will be fall into this range. With other Worlds, if we have carried out very high number of measurements (then data fit to gauss distribution) deviation of average compare to avarage data of very high number will be uncertainity. Due to dual characteristic (temperature and relative humidity measurements) uncertainity of each component will be calculated seperately.

$U_T = \bar{T} \pm ts_T \qquad (K = 2)$

$U_\phi = \bar{\phi} \pm ts_\phi \qquad (K = 2)$

In uncertainty term the concept biass is also important. When avarage value of each data point is calculated both for our device and reference device, the difference of these two values may not be same and some differences exist between these two average value sets. In Order to take this effect into acount a new uncertainity term is defined as:

$U_{ADD} = B + t_{95} S_x$ This is call addition method for uncertainity, the other method is root square addition method. This will be our preferred uncertainity calculation method.

$U_{RSS} = [B^2 + (t_{95} S_x)^2]^{(1/2)}$

where B is bias and defined as

$B = \bar{X} - \bar{X}_{reference} \qquad (K = 2)$ so in our case Biass term will be calculated for Temperature and relative humidity seperately.

$B_T = \bar{T} - \bar{T}_{reference} \qquad (K = 2)$

$B_\phi = \bar{\phi} - \bar{\phi}_{reference} \qquad (K = 2)$

Biass can be ignored in uncertainity calculation if a curve fitted correction is applied in between (if B=0 then $U_{RSS} = U$

In order to achieve this correlation, polynomial least square curve fitting can be used. Since our measurement is two dimensional a two dimensional (function of temperature and relative humidity) so we will required two Biass correction curve fitting to be applied for this experiment

$\bar{T}_{corrected} = f(\bar{T}_{reference}, \bar{\phi}_{reference})$

$\bar{\phi}_{corrected} = f(\bar{T}_{reference}, \bar{\phi}_{reference})$

Additional information about simple polynomial surface curve fitting is given in an appendix :

Requirements in the lab report:

1. Report language will be english

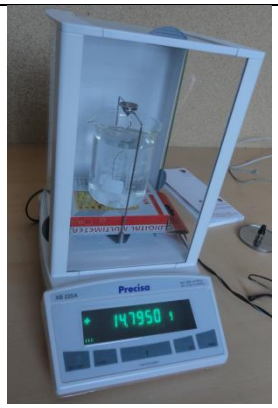
2. Introduction and definition of the experiment
3. Lab calibration measurements
4. Basic information about moisture measurement and calibration
5. Calibration report for given moisture probe(s)
   Average reference temperatures,
   Avarage probe temperatures
   Average reference relative humidities,
   Avarage probe relative humidities.

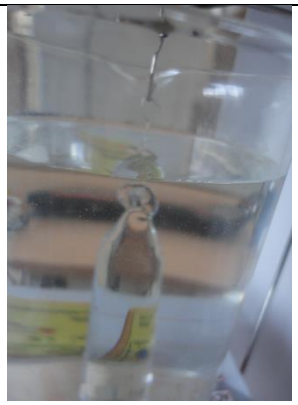
   Reference temperature uncertainity values
   Reference relative humidity uncertainity values
   Probe(s) temperature uncertainity values
   Probe(s) relative humidity uncertainity values
   Temperature Bias values
   Relative humidity Bias values
   Combined U_{RSS} uncertainity values for both temperature and relative humidity
   Least square Biass correction function for temperature and it's plot (3 D)
   Least square Biass correction function for relative humidity and it's plot (3 D)

Name of Experiment	**MEASUREMENT OF LIQUID AND SOLID DENSITIES**

Aim of Experiment
Measuring of densities by using a scale and Archimedes law.

Experimental set-up
Precisa brand scale will be used to carry this experiment. This scale has capability of measuring a mass hang up from the top. In this experiment density of a liquid(water) at the room temperature will be measured. In addition to scale, a barometer to measure athmospheric pressure and a realtive humidity and temperature measuring device is required.



Devices to be used in the experiment

1. Precisa brand scale
2. liquid container
3. Psychrometric table (www.turhancoban.com)
4. Testo moisture and temperature sensor
5. Testo barometric reading sensor

Experiment

Experiment will be carried out as two different experiment. In the first experiment, the scale is adjusted to read zero value when it is included a weight hanger, and then a glass cylinder with a precise 10 ml volume will be hanged from the hanger and the mass value will be read from the scale. Then the glass cylinder will be submerged completely to the liquid and the mass value reading will be taken again. Meanwhile air and water temperature and air mosture and athmospheric pressure will also be recorded. A Buoyancy force equal to

$$F_{B\_air} = \rho_{air} g V_{glass}$$

$F_{B\_liquid} = \rho_{liquid} g V_{glass}$ will be effected to our readings in both first and the second measurement so

$$m_{readed\_in\_air} = m_{glass} - \frac{F_{B\_air}}{g} = \rho_{glass} V_{glass} - \rho_{moist\_air} V_{glass}$$

$$m_{readed\_in\_liquid} = m_{glass} - \frac{F_{B\_liquid}}{g} = \rho_{glass} V_{glass} - \rho_{liquid} V_{glass}$$

In order to find density of atmospheric moist air the data read can be used

$$\rho_{moist\_air} = \rho_{dry\_air}(1+w) = \frac{RT}{P_{atm}}(1+w)$$

The the given equations are enough to calculate density of the liquid.
As a second part of the experiment the density of a steel ball will be calculated. In order to do this, instead of glass cylinder tha ball will be hanged ans measured both in the air and the liquid that density is measured in the first experiment. Similar equations will be valid for this case as well

$$m_{readed\_in\_air} = m_{steel\_ball} - \frac{F_{B\_air}}{g} = \rho_{steel\_ball} V_{steel\_ball} - \rho_{moist\_air} V_{steel\_ball}$$

$$m_{readed\_in\_liquid} = m_{steel\_ball} - \frac{F_{B\_liquid}}{g} = \rho_{steel\_ball} V_{steel\_ball} - \rho_{liquid} V_{steel\_ball}$$

Evaluation of experiment results.
In the experiment determine the density of liquid and then density of steel ball by using the equations given above.

Requirements in the lab report:

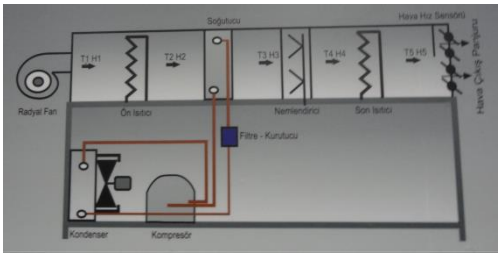
1. Report language will be english
2. Introduction and definition of the experiment
3. Measurements carried out in the experiments
4. Detailed calculations
5. Basic information about Buoyancy and density measurements
6. Results and discussion

Ege University, School of Engineering **Department of Mechanical Engineering**

Name of the experiment	**WINTER AIR CONDITIONING**

Aim of Experiment
Winter heating and air conditioning processes will be explained through the measurements. In an air conditioning unit, preheating, moisturizing and final heating processes wil be shown, and related calculations will be carried out. The process will be shown in psyhrometric chart.

Experimental set-up
Basic air conditioning experimental set-up unit in the lab will be used fort his experiment. The experimental system is shown in the figure below.



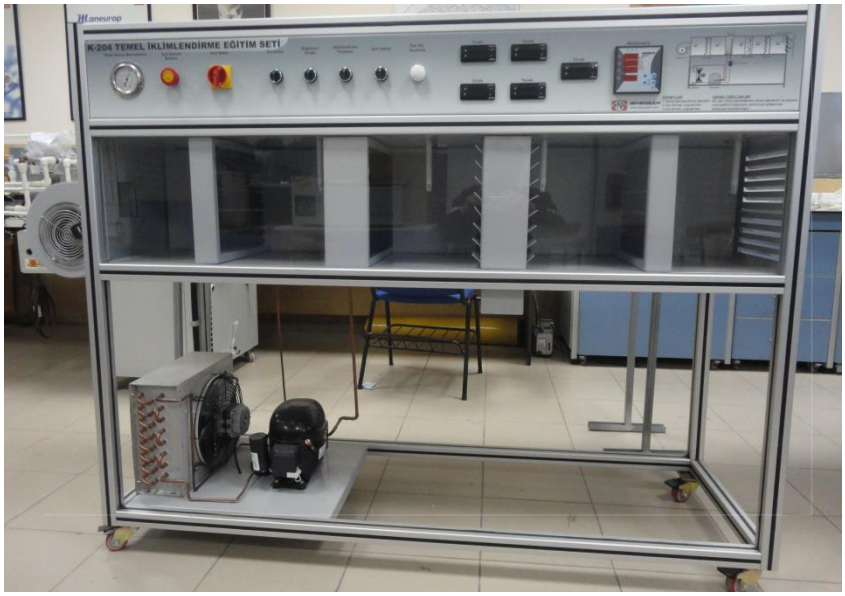


Figure 1. Basic air conditioning units

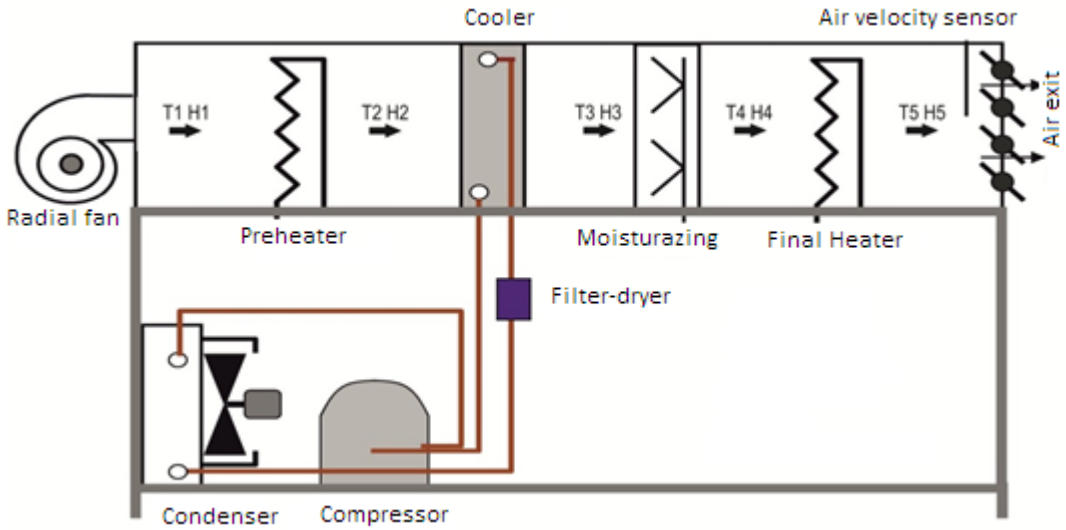


Figure2. Basic air conditioning units flow diagram

Devices to be used in the experiment

1. Basic air conditioning experiment set
2. Anemometer
3. Atmospheric air moisture reading

Experiment

1) Startup the fan.
2) Startup preheater and final heater.
3) Starup moisturizer.
4) When the system reached to equilibrium, record the system parameters and outside air parameters.

Evaluation of results

Winter air air conditioning process is measured when the system come into equilibrium condition. The basic parameters to be used in calculations are listed below:

Volumetric air flow rate: $\dot{V}_h = A.v$ $[m^3/s]$

A: cross sectional area (m^2)
v: velocity(m/s)

Mas flow rate of air $\dot{m}_h = \frac{V_h}{v_g}$

v_g : specific volume of inlet air (m^3/kg)

Preheater heat transfer: $\dot{Q}_{preheater} = \dot{m}_h (h_2 - h_1)$

Amount of water addition (moisturizing): $\dot{m}_s = \dot{m}_h (w_4 - w_3)$

Last heater heat transfer: $\dot{Q}_{last\,heater} = \dot{m}_h (h_5 - h_4)$

Winter air conditioning process should be shown in psyhchrometric diagram.

Requirements of the lab report

- Definition of the experiment and
- Knowledge and explanatio of winter air conditioning process.
- Heat transfer characteristics and detailed calculations of the process
- Show the process in psychrometric diagram.

You can utilize the related codes given in the appendiz and SCO1 library

Experimental measurements and their results

Number of measurements	**1**	**2**	**3**
T_1 [^{0}C]			
ϕ_1 [%]			
T_2 [^{0}C]			
ϕ_2 [%]			
T_3 [^{0}C]			
ϕ_3 [%]			
T_4 [^{0}C]			
ϕ_4 [%]			
T_5 [^{0}C]			
ϕ_5 [%]			

Ege University, School of Engineering
Department of Mechanical Engineering

Name of the experiment	**DOUBLE TUBE HEAT EXCHANGER**

Aim of Experiment
Measuring and calculating of heat transfer characteristics of double pipe heat exchanger

Experimental set-up
Heat exchanger experimental set up in the lab will be used to carry out this experiment. The system Picture is given below.

Figure 1. Heat exchanger experimental setup

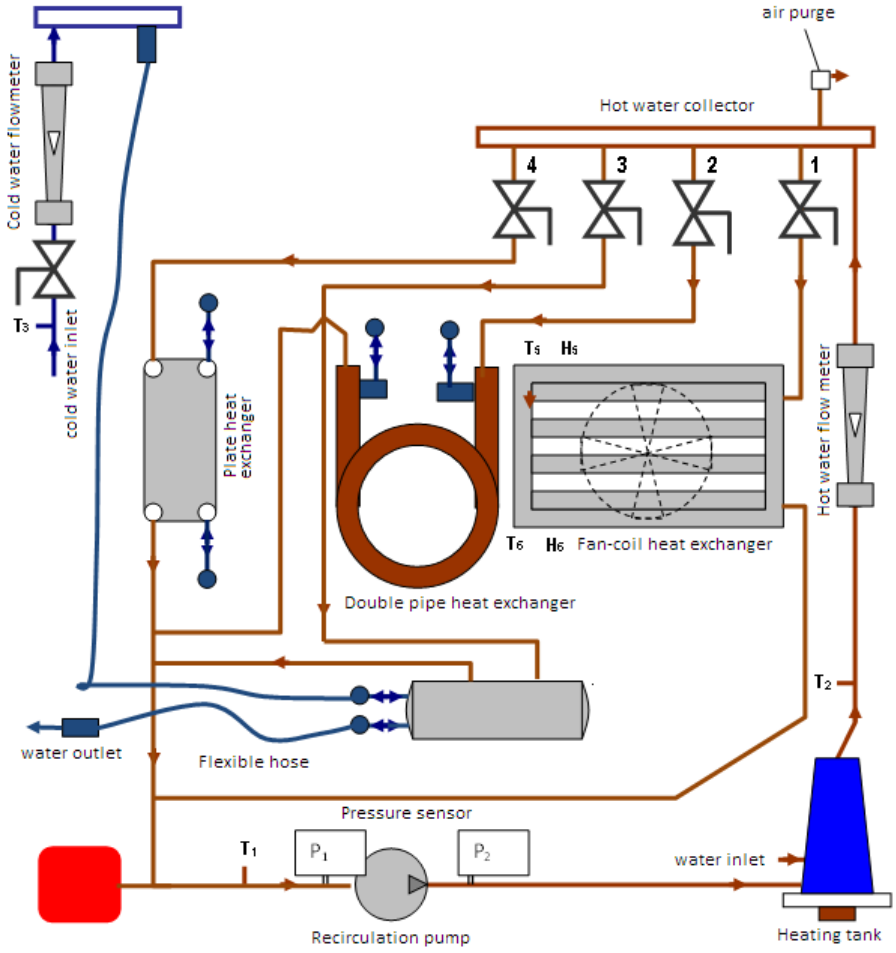


Figure 2. Heat exchanger setup flow diagram

Instruments to be used in the experiment

1. Experimental heat exchanger setup

Experiment

1) Turn on the power switch.
2) Operate the pump. Open the valve #2 in the hot water collector. Adjust the water flow rates 500 L/h'ye ve 400 L/h .
3) Turn on the switch of heater.
4) When the system equilibrium is established, record the system parameters.

Evaluation of results

By using experimental data and equations given blow calculate heating capacities of the heat exchanger. Calculate also heat capacities by using heat transfer theories and compare the results.

Hot water heat transfer: $\dot{Q}_1 = \dot{m}_{hot\_water}\, C_{p\_hot\_water} (T_2 - T_1)$

$\dot{Q}_1 = U.I/1000$ [kW]

Cold water heat transfer: $\dot{Q}_2 = \dot{m}_{cold\_water}\, C_{p\_cold\_water} (T_4 - T_3)$

Experimental Heat transfer coefficient: $U = \frac{Q_1}{A.\Delta T_{\ln}}$ [W/m^{2}K]

Logarithmic temperature difference: $\Delta T_{\ln} = \frac{\Delta T_1 - \Delta T_2}{\ln\left(\frac{\Delta T_1}{\Delta T_2}\right)}$ $\Delta T_1 = T_2 - T_3$

$\Delta T_2 = T_1 - T_4$

Double pipe heat exhanger properties: Di=5/8"- Do=7/8"

A=πDL=3.14x0,016x2.26=0,1135 m^2

The experimental results can be cheched out by using theoretical calculations as well

$$UA = \frac{1}{R_t} = \frac{1}{\frac{1}{h_i A} + \frac{t}{kA} + \frac{1}{h_0 A}}$$

$$U_0 A_0 = U_i A_i = \frac{1}{R_t} = \frac{1}{\frac{1}{h_i A_i} + \frac{\ln(r_0/r_i)}{2\pi kL} + \frac{1}{h_0 A_0}}$$

Heat transfer equation in a pipe:

Laminar region (Re<2100) $Nu_D = 3.66$

Transitional turbulent region (2100<Re<3300) Abraham, Sparrow Tong [55] denklemini kullanabiliriz.

$$Nu_D = 2.2407\left(\frac{\mathrm{Re}}{1000}\right)^4 - 29.499\left(\frac{\mathrm{Re}}{1000}\right)^3 + 142.32\left(\frac{\mathrm{Re}}{1000}\right)^2 - 292.51\left(\frac{\mathrm{Re}}{1000}\right) + 219.88$$

Turbulant region (Re>3300) Gnielinski[56] equation

$$Nu_D = \frac{(f/8)(\mathrm{Re}_D - 1000)\,\mathrm{Pr}}{1 + 12.7(f/8)(\mathrm{Pr}^{2/3} - 1)}$$

Goudar- Sonnad equation for friction factor

$$a = \frac{2}{\ln(10)}$$

$$b = \frac{(\varepsilon / D)}{3.7}$$

$$d = \frac{\ln(10)}{5.02}\mathrm{Re}$$

$$s = bd + \ln(d)$$

$$q = s^{(s/(s+1))}$$

$$\delta_{LA} = \frac{g}{g+1}z$$

$$\delta_{CFA} = \delta_{LA}\left(1 + \frac{z/2}{(g+1)^2 + (z/3)(2g-1)}\right)$$

$$\frac{1}{\sqrt{f}} = a\left[\ln\left(\frac{d}{q}\right) + \delta_{CFA}\right] \qquad \textbf{(3.2-3C)}$$

Requirements in the lab experiment

- Description of the experiment.
- Measurement values
- Experimental calculations of heat transfer characterisitcs of the double pipe heat exchanger
- Calculations of heat transfer characterisitcs of the double pipe heat exchanger from heat transfer formulations
- Comparison of theoretical calculations with experimental results.

Experimental values

Number of measurements		**1**	**2**	**3**
U	**[Volt]**			
I	**[Amper]**			
cos(ϕ)				
$\mathbf{T_1}$	**[˚C]**			
$\mathbf{T_2}$	**[˚C]**			
$\mathbf{T_3}$	**[˚C]**			
$\mathbf{T_4}$	**[˚C]**			
$\dot{V}_{hot\_water}$	**[L/h]**			
$\dot{V}_{cold,\ su}$	**[L/h]**			

Ege University, School of Engineering
Department of Mechanical Engineering

Name of the experiment	**LOCAL PRESSURE DROP MESUREMENTS**

Aim of Experiment
Measuring and calculating of Local pressure drops of slide valve and ball valve.

Experimental set-up
Pressure drop experimental set up in the lab will be used to carry out this experiment. The system Picture is given below.

Figure 1. Pressure drop exchanger experimental setup

Instruments to be used in the experiment

1. Experimental pressure drop setup

Experiment

1) Turn on the power switch.
2) Operate the pump.
3) Connect pressure gauge outputs at the inlet and outlet of the slide valve(şiber vana)
4) Take the pressure drop for different flow rates and valve opening angles(or turns)
5) Connect pressure gauge outputs at the inlet and outlet of the ball valve(küresel vana)
6) Take the pressure drop for different flow rates and valve opening angles.

Evaluation of results

By using experimental data and equations given blow calculate Local pressure drop coefficients of the valves as a function of valve opening angles.

Pressure drop: $\Delta P = \rho g h = \rho K_L \frac{V^2}{2}$

Mass flow rate $m = \rho Q = \rho V A$ [kg/s] Q is volumetric flow rate(m^3/s)

Find book data for this type of valves and compare your results with the data given in printed or internet media.

Requirements in the lab experiment

- Description of the experiment.
- Measurement values
- Experimental calculations of local pressure loss coefficients at different opening positions (give details of your calculations)
- Comparison of printed values with your experimental results.

Ege Üniversitesi Mühendislik Fakültesi Makina Mühendisliği Bölümü Makina Laboratuvarı Deneyleri

Deneyin Adı	**COMMERTIAL REFRIGERATION SYSTEM PERFORMANCE**

Purpose of the experiment

A commertial refrigeration cycle with two different evaporation pressure and two compressor will be investigated and operation principles will be explained, results of system performance will be calculated and be shown in P-h diagram.

Experimental set-up

Commertial refrigeration system in our department will be used to carry out the experiment.

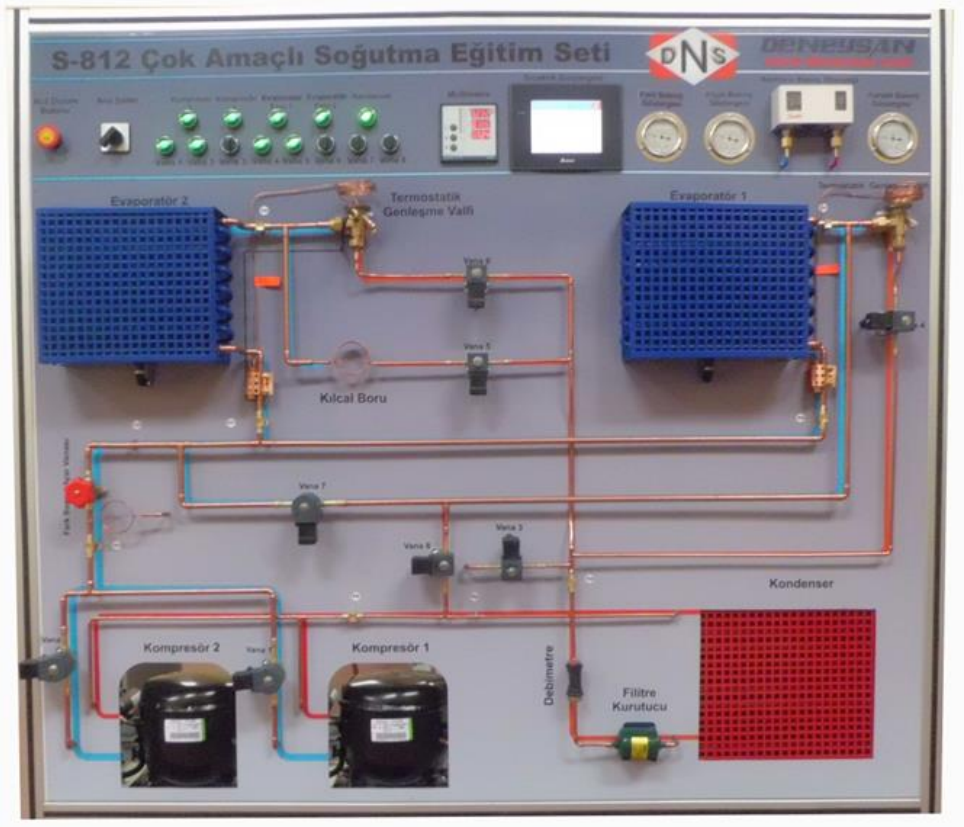


Figure 1. Commertial refrigeration system

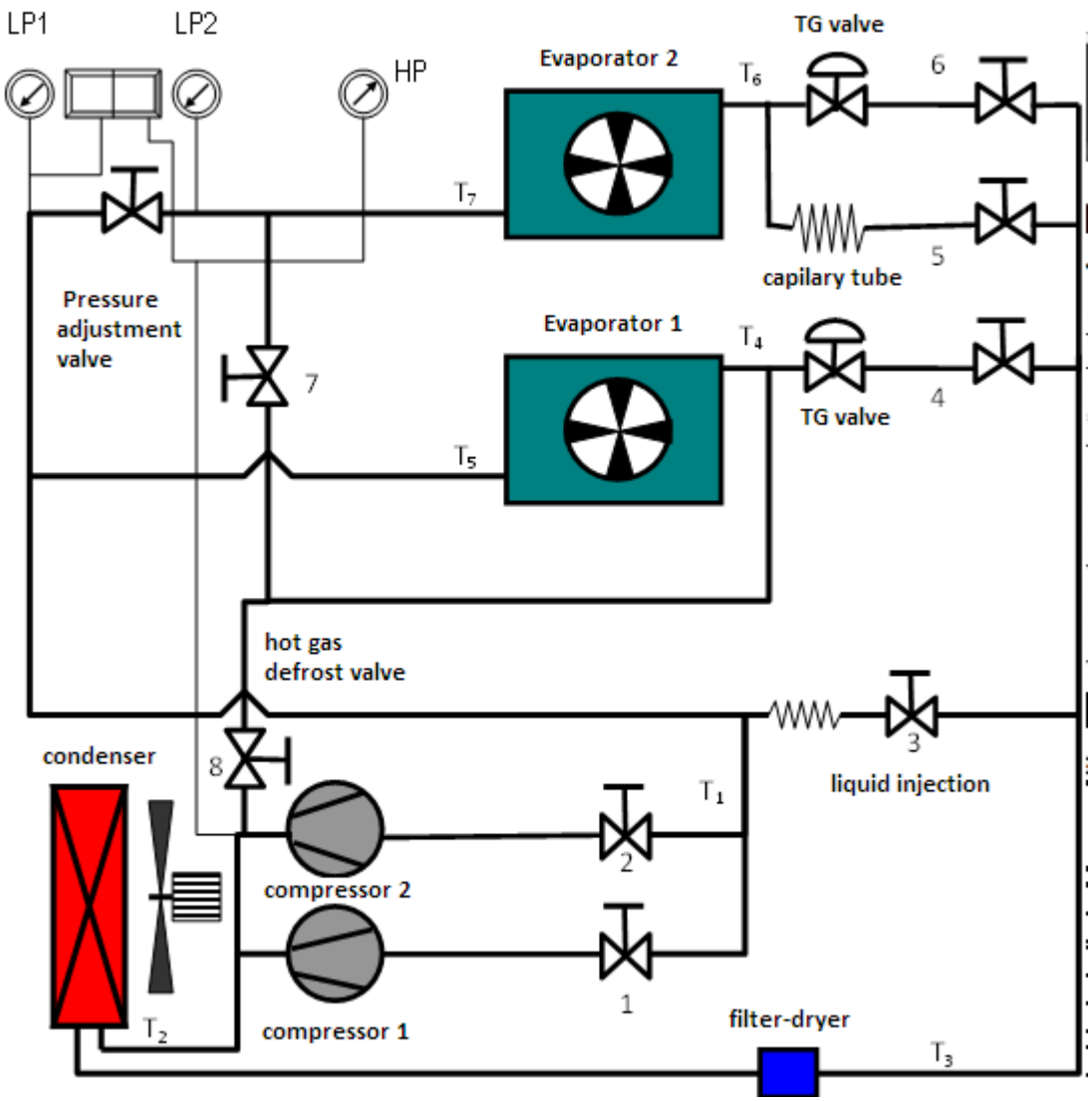


Figure 2. Commertial refrigeration system flow diagram

Equipment to be used in the experiment

1. TCommertial refrigeration educational set

Carrying out the experiment

1) open valves 1, 2, 4 and 6, and close the all remaining valves
2) Operate fans and compressors
3) Adjust pressure adjustment valve(number 7) by observing pressures difference of 1 bar in between LP1 and LP2. This will cause two different evaporation pressure for two evaporators in the system.
4) When the system is in steady state record the sytem readings (tempreatures, pressures, flow rates etc.)
5) Re-adjust pressure adjustment valve(number 7) by observing pressures difference of 2 bar in between LP1 and LP2.
6) When the system is in steady state record the sytem readings (tempreatures, pressures, flow rates etc.)

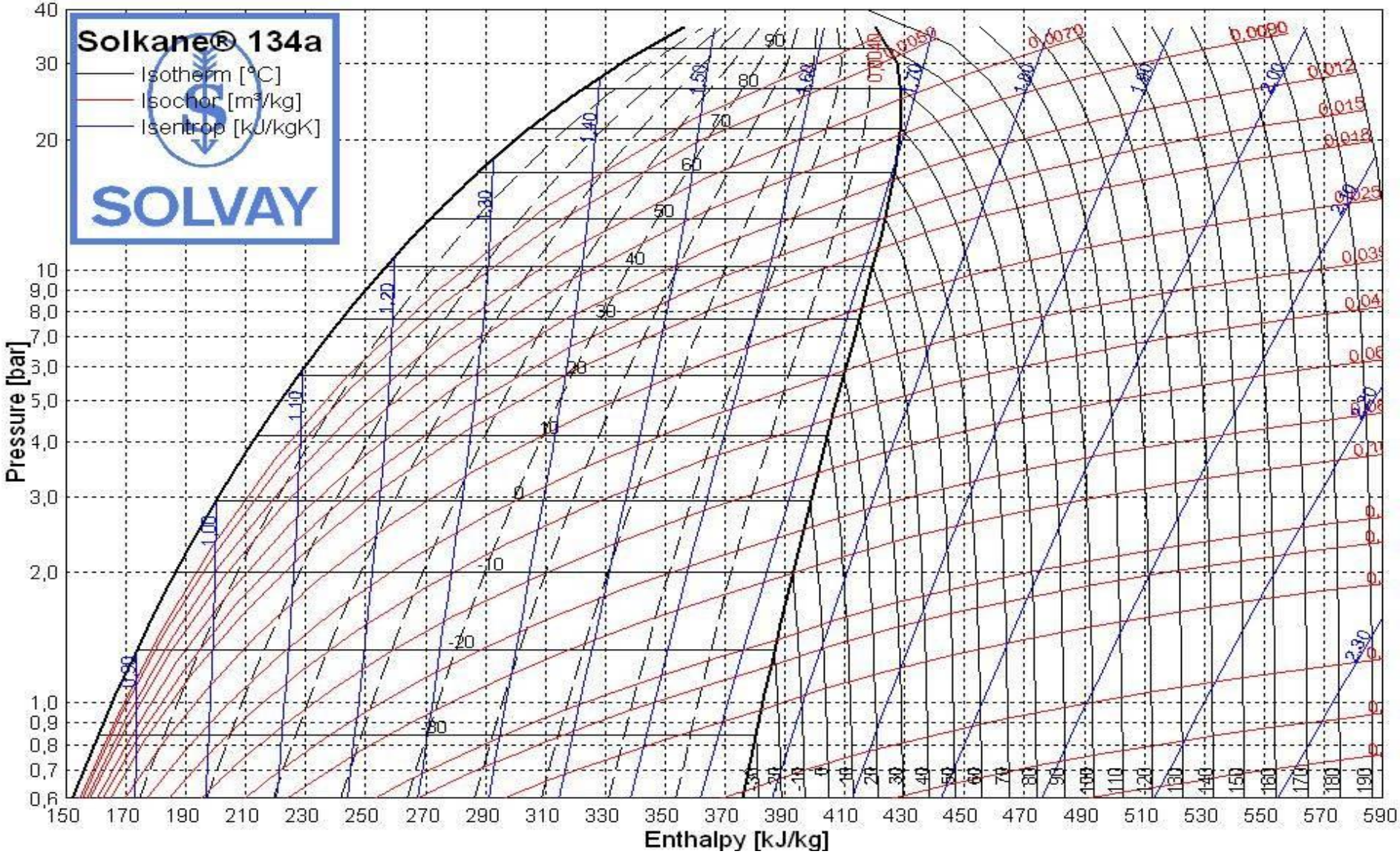
Evaluationof experimental results
All termodynamic cycle will be evaluated for the values obtained in the experiment. The results will be shown in P-h diagram. Working refrigerant of the cycle is R134a.

Requirements of the lab report

- Full description of the experiment
- Measured set of values
- Thermodynamic claculations of the cycle for both pressure differences.
- Show the cycle in P-h diagram

Experimental values

Number of measurements	**Okunacağı gösterge**	**1**	**2**
Condenser pressure, P_2 [kPa]	**HP**		
Condensing saturated temperature, T_y [˚C]	**HP**		
Evaporator 1 pressure, P_{e1} [kPa]	**LP1**		
Evaporator 2 pressur, P_{e2} [kPa]	**LP2**		
Evaporator 1 saturation temperature, T_{e1} [˚C]	**T_4**		
Evaporator 2 saturation temperature, T_{e2} [˚C]	**T_6**		
Evaporator 1 exit temperature, $T_{eç1}$ [˚C]	**T_5**		
Evaporator 2 exit temperature,, $T_{eç2}$ [˚C]	**T_7**		
Evaporator 1 degree of superheat, ($T_{eç1}$-T_{e1}) [˚C]	**T_5 - T_4**		
Evaporator 1 degree of superheat, ($T_{eç2}$-T_{e2}) [˚C]	**T_7 - T_6**		


Solkane® 134a
Isotherm [°C]
Isochor [m³/kg]
Isentrop [kJ/kgK]
SOLVAY
Pressure [bar]
Enthalpy [kJ/kg]

Ege University, School of Engineering Department of Mechanical Engineering

Name of the experiment	SUMMER AIR CONDITIONING

Aim of Experiment

Summer cooling and air conditioning processes will be explained through the measurements. In an air conditioning unit, cooling and demoisturizing and final heating processes wil be shown, and related calculations will be carried out. The process will be shown in psyhrometric chart.

Experimental set-up

Basic air conditioning experimental set-up unit in the lab will be used for this experiment. The experimental system is shown in the figure below.

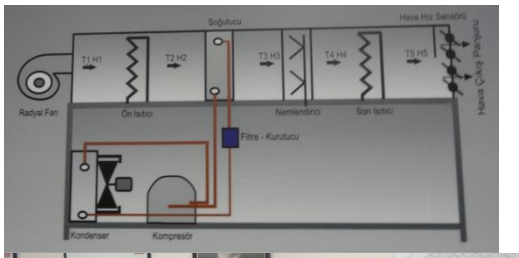


Figure 1. Basic air conditioning units

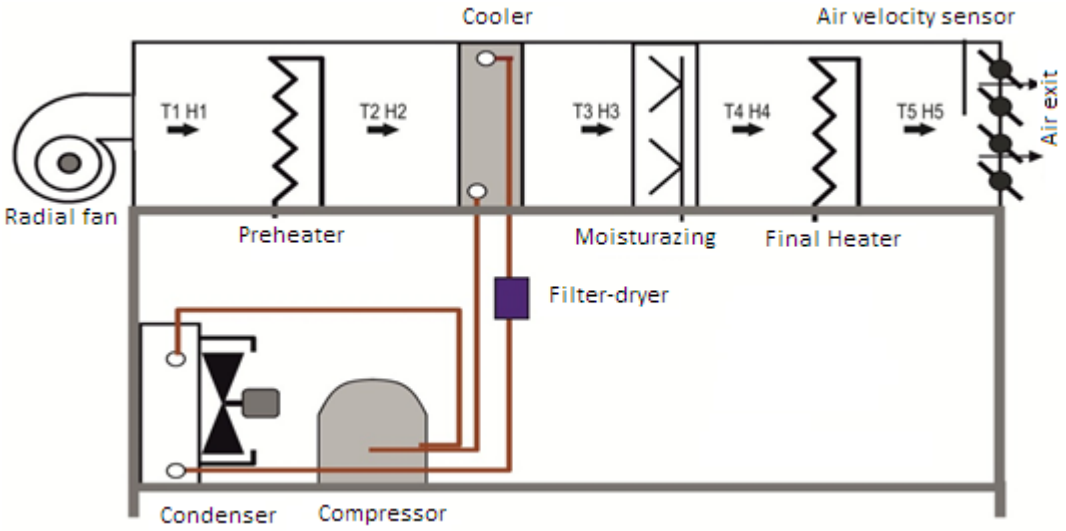


Figure2. Basic air conditioning units flow diagram

Devices to be used in the experiment

1. Basic air conditioning experiment set
2. Anemometer
3. Atmospheric air moisture reading

Experiment

5) Startup the fan.
6) Startup cooler-demoisterizer
7) Starup final heater.
8) When the system reached to equilibrium, record the system parameters and outside air parameters.

Evaluation of results

Summer air air conditioning process is measured when the system come into equilibrium condition. The basic parameters to be used in calculations are listed below:

Volumetric air flow rate: $\dot{V}_h = A.v$ $[m^3/s]$

A: cross sectional area (m^2)
v: velocity(m/s)

Mas flow rate of air $\dot{m}_h = \frac{V_h}{v_g}$

v_g : specific volume of inlet air (m^3/kg)

Cooler heat transfer: $\dot{Q}_{cooler} = \dot{m}_{dry\,air}\,(h_2 - h_3) - m_w h_w = \dot{m}_{dry\,air}\,(h_2 - h_3) - \dot{m}_{dry\,air}\,w\,h_w$

Last heater heat transfer: $\dot{Q}_{last\,heater} = \dot{m}_{dry\,air}\,(h_5 - h_4)$

Summer air conditioning process should be shown in psyhchrometric diagram.

Requirements of the lab report

- Definition of the experiment and
- Knowledge and explanation of summer air conditioning process.
- Heat transfer characteristics and detailed calculations of the process
- Show the process in psychrometric diagram.

You can utilize the related codes given in the appendix and SCO1 library

Experimental measurements and their results

Number of measurements	**1**	**2**	**3**
T_1 [^{0}C]			
ϕ_1 [%]			
T_2 [^{0}C]			
ϕ_2 [%]			
T_3 [^{0}C]			
ϕ_3 [%]			
T_4 [^{0}C]			
ϕ_4 [%]			
T_5 [^{0}C]			
ϕ_5 [%]			

Ege University, School of Engineering Department of Mechanical Engineering	
Name of the experiment	**SHELL & TUBE HEAT EXCHANGER**

Aim of Experiment

Measuring and calculating of heat transfer characteristics of Shell & tube heat exchanger

Experimental set-up

Heat exchanger experimental set up in the lab will be used to carry out this experiment. The system Picture is given below.

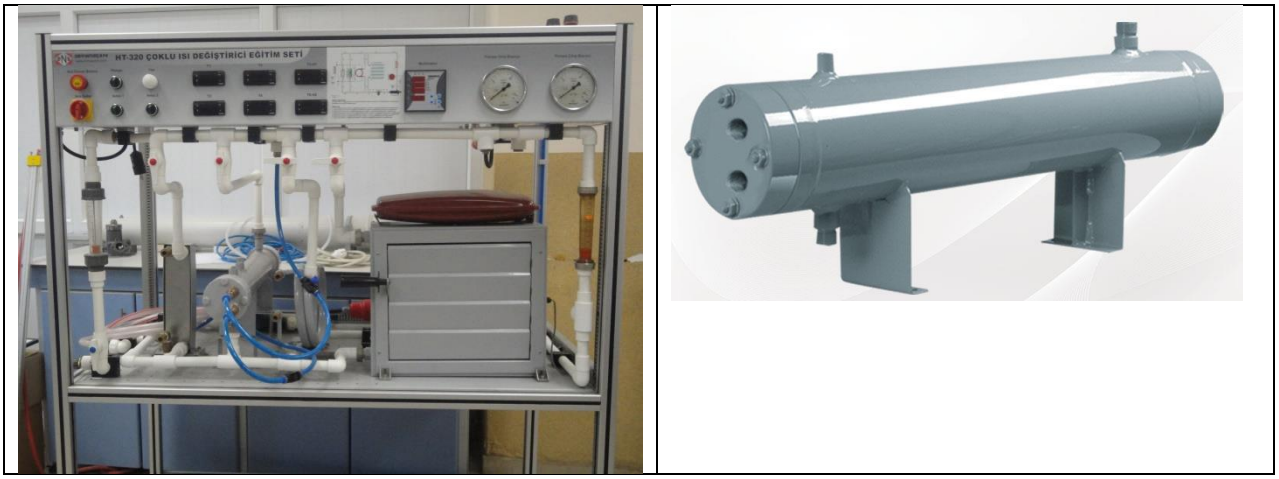


Figure 1. Heat exchanger experimental setup

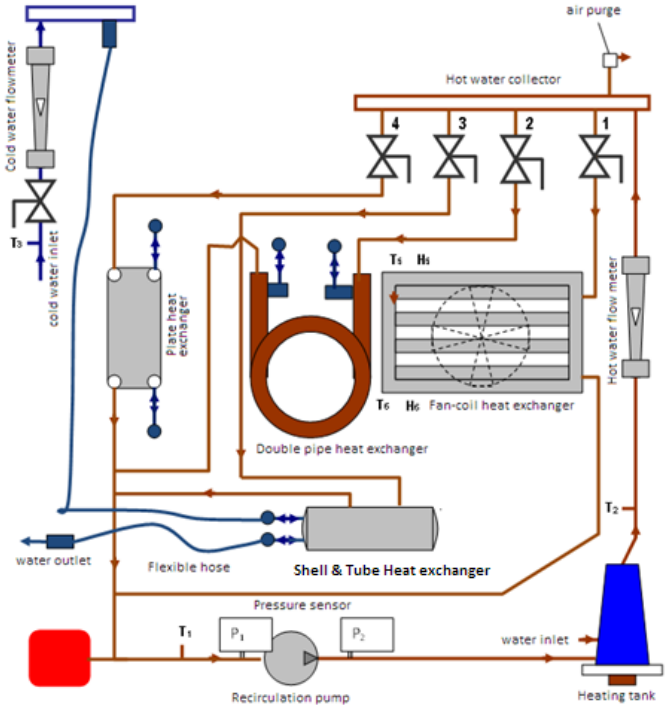
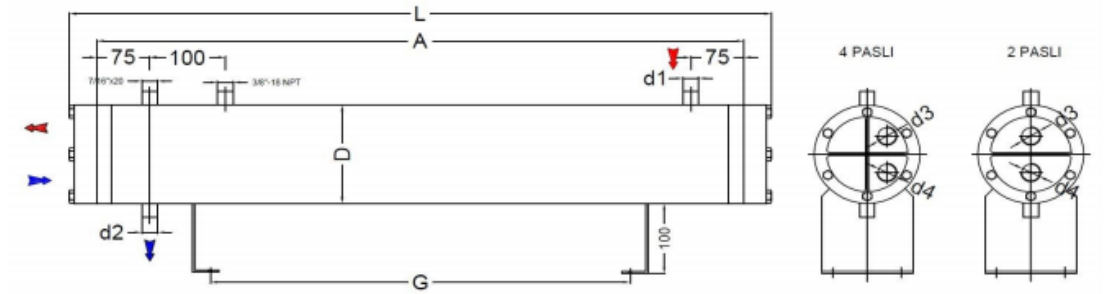


Figure 2. Heat exchanger setup flow diagram

Instruments to be used in the experiment

1. Experimental heat exchanger setup. Shell & Tube heat exchanger in the system: YTO-K-6 Flow rate 0.65 m^3/h capacity approximately 4 kW

Su Soğutmalı Shell And Tube Kondenserler
Water cooling shell & tube condenser



YTO-K | 5/8" Finli Bakır Borulu/5/8" finned copper tube | 1 Temmuz 2017

Kondenzasyon : 40 oC, Su Giriş/Çıkış : 30/35 oC Şartlarında/condensation 40 oC water inlet/outpu 30/35 oC

Tip Model YTO-K	Kapasite Capacity KW	Alan Arae m2	su water m3/h	Basınç Pressure Bar	R407C dm3	su water dm3	gaz/gas giris/in d1	gaz/gas çıkış/ouy d2	su/water çıkış/ouy d3	su/water giris/in d4	ölçüler dimension A	ölçüler dimension L	ölçüler dimension G	ölçüler dimension D
YTO-K-3	2,0	0,28	0,3	0,02	0,9	0,4	16	16	3/4 "	3/4 "	300	400	150	90
YTO-K-8	6,6	0,66	1,1	0,16	1,9	0,6	16	16	3/4 "	3/4 "	600	700	300	90
YTO-K-6	3,9	0,55	0,7	0,02	1,4	0,7	19	16	3/4 "	3/4 "	300	400	150	114
YTO-K-15	12,8	1,32	2,2	0,14	3,1	1,3	19	16	3/4 "	3/4 "	600	700	300	114

Experiment

1) Turn on the power switch.
2) Operate the pump. Open the valve #3 in the hot water collector. Adjust the water flow rates of 500 L/h then 400 L/h and then 200 L/h.
3) Turn on the switch of heater.
4) When the system equilibrium is established, record the system parameters.

Evaluation of results

By using experimental data and equations given below calculate heating capacities of the heat exchanger. Calculate also heat capacities by using heat transfer theories and compare the results.

Hot water heat transfer: $\dot{Q}_1 = \dot{m}_{hot\_water}\, C_{p\_hot\_water} (T_2 - T_1)$

$\dot{Q}_1 = U.I/1000$ [kW]

Cold water heat transfer: $\dot{Q}_2 = \dot{m}_{cold\_water}\, C_{p\_cold\_water} (T_4 - T_3)$

Experimental Heat transfer coefficient: $U = \dfrac{Q_1}{A.\Delta T_{ln}}$ [W/m^2K]

Logarithmic temperature difference: $\Delta T_{ln} = \dfrac{\Delta T_1 - \Delta T_2}{\ln\left(\dfrac{\Delta T_1}{\Delta T_2}\right)}$ $\Delta T_1 = T_2 - T_3$

$\Delta T_2 = T_1 - T_4$

Double pipe heat exhanger properties: Di=5/8"- Do=7/8"

$A=\pi DL=3.14x0{,}016x2.26=0{,}1135\ m^2$

The experimental results can be cheched out by using theoretical calculations as well

$$UA=\frac{1}{R_t}=\frac{1}{\frac{1}{h_i A}+\frac{t}{kA}+\frac{1}{h_0 A}}$$

$$U_0 A_0=U_i A_i=\frac{1}{R_t}=\frac{1}{\frac{1}{h_i A_i}+\frac{\ln(r_0/r_i)}{2\pi kL}+\frac{1}{h_0 A_0}}$$

Heat transfer equation in a pipe:

Laminar region (Re<2100) $Nu_D=3.66$

Transitional turbulent region (2100<Re<3300) Abraham, Sparrow Tong [55] equations can be used.

$$Nu_D=2.2407\left(\frac{\mathrm{Re}}{1000}\right)^4-29.499\left(\frac{\mathrm{Re}}{1000}\right)^3+142.32\left(\frac{\mathrm{Re}}{1000}\right)^2-292.51\left(\frac{\mathrm{Re}}{1000}\right)+219.88$$

Turbulant region (Re>3300) Gnielinski[56] equation

$$Nu_D=\frac{(f/8)(\mathrm{Re}_D-1000)\,\mathrm{Pr}}{1+12.7(f/8)(\mathrm{Pr}^{2/3}-1)}$$

Goudar- Sonnad equation for friction factor

$$a=\frac{2}{\ln(10)}$$

$$b=\frac{(\varepsilon/D)}{3.7}$$

$$d=\frac{\ln(10)}{5.02}\mathrm{Re}$$

$$s=bd+\ln(d)$$

$$q=s^{(s/(s+1))}$$

$$\delta_{LA}=\frac{g}{g+1}z$$

$$\delta_{CFA}=\delta_{LA}\left(1+\frac{z/2}{(g+1)^2+(z/3)(2g-1)}\right)$$

$$\frac{1}{\sqrt{f}}=a\left[\ln\left(\frac{d}{q}\right)+\delta_{CFA}\right] \qquad \textbf{(3.2-3C)}$$

Requirements in the lab experiment

- Description of the experiment.
- Measurement values
- Experimental calculations of heat transfer characterisitcs of the Shell & tube heat exchanger
- Calculations of heat transfer characterisitcs of the Shell & tube heat exchanger from heat transfer formulations
- Comparison of theoretical calculations with experimental results.

Experimental values

Number of measurements		**1**	**2**	**3**
U	**[Volt]**			
I	**[Amper]**			
cos(ϕ)				
T_1	**[˚C]**			
T_2	**[˚C]**			
T_3	**[˚C]**			
T_4	**[˚C]**			
$\dot{V}_{hot\_water}$	**[L/h]**			
$\dot{V}_{cold,\ su}$	**[L/h]**			

Appendix

(This section is taken from Numerical Analysis with java examples , Dr. M. Turhan ÇOBAN)

LINEAR CURVE FITTING : LEAST SQUARE METHODS

Least square methods are one of the most used curve fitting methods. Polinomial equation version is widely used. Assuming having a linear function $f(a_j,x)$ where a_j are m linear coefficient and x are independent variable and $\phi_j(x)$ are m sub-funtions linearly multiplied with the coefficients

$$f(a_j,x)=\sum_{j=0}^{m} a_j^{(m)}\phi_j(x)$$

It should be noted again that linearity is only for the coefficients, $\phi_j(x)$ functions does not have to be linear. It is desired to fit data x_i, f_i, i=0..n into the equation so that the difference between data and fitted function dependent values for all points will be minimum. In order to establish this, the following function H will be minimized with respect to a_j

$$H(a_0^{(m)},....,a_m^{(m)})=\sum_{i=1}^{n} w(x_i)\left[\, f_i-\sum_{j=0}^{m} a_j^{(m)}\phi_j(x_i)\,\right]^2$$

Where $w(x_i)$ values in the equation are called weight function. In order to minimiz the function root of the derivative of the function can be calculated.

$$\frac{\partial H(a_0^{(m)},....,a_m^{(m)})}{\partial a_k^{(m)}}=2\sum_{i=1}^{n} w(x_i)\left[\, f_i-\sum_{j=0}^{m} a_j^{(m)}\phi_j(x_i)\,\right]\phi_k(x_i)=0 \qquad k=0,.....,m$$

$$\left\{\sum_{j=0}^{m} w(x_i)\phi_j(x_i)\phi_k(x_i)\right\}\left[a_j^{(m)}\right]=\left[\sum_{i=1}^{n} w(x_i)\phi_k(x_i)f_i\right] \qquad k=0,.....,m$$

For weight function to be taken equal to unity , $w(x_i)$=1, equation in the open form can be written in the following form.

$$\begin{Bmatrix} \sum_{i=1}^{n}\phi_0^2(x_i) & \sum_{i=1}^{n}\phi_0(x_i)\phi_1(x_i) & \sum_{i=1}^{n}\phi_0(x_i)\phi_2(x_i) & .... & \sum_{i=1}^{n}\phi_0(x_i)\phi_m(x_i) \\ \sum_{i=1}^{n}\phi_0(x_i)\phi_1(x_i) & \sum_{i=1}^{n}\phi_1^2(x_i) & \sum_{i=1}^{n}\phi_1(x_i)\phi_2(x_i) & .... & \sum_{i=1}^{n}\phi_1(x_i)\phi_m(x_i) \\ \sum_{i=1}^{n}\phi_0(x_i)\phi_2(x_i) & \sum_{i=1}^{n}\phi_1(x_i)\phi_2(x_i) & \sum_{i=1}^{n}\phi_2^2(x_i) & .... & \sum_{i=1}^{n}\phi_2(x_i)\phi_m(x_i) \\ .... & .... & .... & .... & .... \\ \sum_{i=1}^{n}\phi_0(x_i)\phi_m(x_i) & \sum_{i=1}^{n}\phi_1(x_i)\phi_m(x_i) & \sum_{i=1}^{n}\phi_2(x_i)\phi_m(x_i) & .... & \sum_{i=1}^{n}\phi_m^2(x_i) \end{Bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ .... \\ a_m \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n}\phi_0(x_i)f(x_i) \\ \sum_{i=1}^{n}\phi_1(x_i)f(x_i) \\ \sum_{i=1}^{n}\phi_2(x_i)f(x_i) \\ .... \\ \sum_{i=1}^{n}\phi_m(x_i)f(x_i) \end{bmatrix}$$

This equation is an m+1 linear system of equation. It can easily be solved by using a system of equation solving method. Aa a special form of the above equation a form of a polinomial can be assumed. If $\phi_j(x)=x^j$, equation becomes

$$f(x)=\sum_{j=0}^{m} a_j^{(m)}x^j$$

In this case the minimisation equation becomes

$$\frac{\partial H(a_0^{(m)},....,a_m^{(m)})}{\partial a_k^{(m)}}=2\sum_{i=1}^{n}\left[\, f_i-\sum_{j=0}^{m} a_j^{(m)}x_i^j\,\right]x_i^k=0 \qquad k=0,.....,m$$

$$\sum_{k=0}^{m} a_j^{(m)}\left[\,\sum_{i=1}^{n} x_i^{j+k}\,\right]=\sum_{i=1}^{n} f_i\,x_i^k \qquad k=0,.....,m$$

This equation can be written in the open format as:

Program 6.1-1 Polynomial least square curve fitting

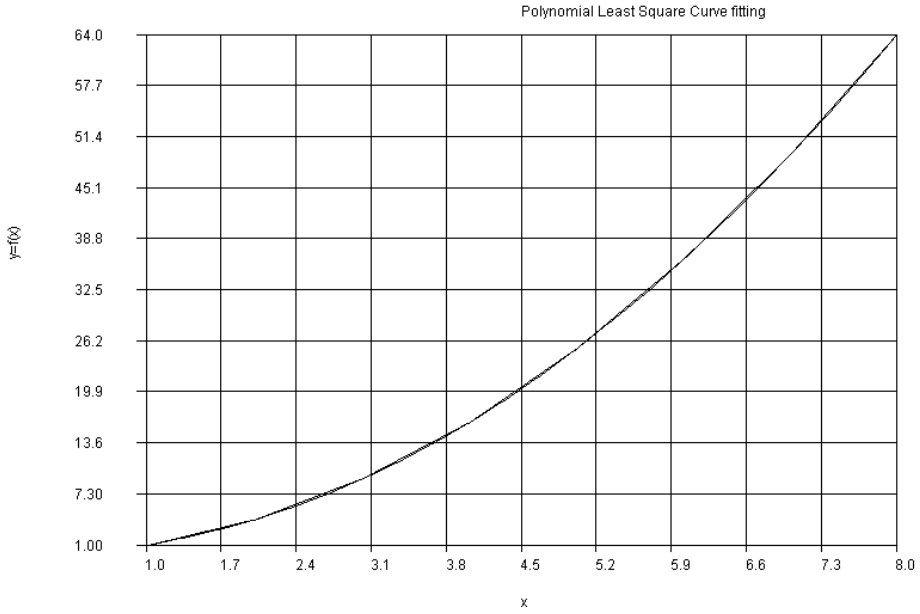
```
import java.io.*;
import javax.swing.*;

class NA44
```

```
{
// Polynomial least square
  public static double[] gausswithpartialpivot(double a[][],double b[])
  { //Gauss elimination with partial pivoting
  int n=b.length;
  double x[]=new double[n];
  double carpan=0;
  double toplam=0;
  double buyuk;
  double dummy=0;
  //gauss elimination
  int i,j,k,p,ii,jj;
  for(k=0;k<(n-1);k++)
  { //partial pivoting
              p=k;
              buyuk=Math.abs(a[k][k]);
              for(ii=k+1;ii<n;ii++)
              {  dummy=Math.abs(a[ii][k]);
                 if(dummy > buyuk) {buyuk=dummy;p=ii;}
              }
              if(p!=k)
              { for(jj=k;jj<n;jj++)
                { dummy=a[p][jj];
                  a[p][jj]=a[k][jj];
                  a[k][jj]=dummy;
                }
                dummy=b[p];
                b[p]=b[k];
                b[k]=dummy;
              }
              //
    for(i=k+1;i<n;i++)
    {  carpan=a[i][k]/a[k][k];
       a[i][k]=0;
       for(j=k+1;j<n;j++)
       {   a[i][j]-=carpan*a[k][j]; }
          b[i]   =b[i]   -carpan*b[k];
              }
  }
  //backward substitution
  x[n-1]=b[n-1]/a[n-1][n-1];
  for(i=n-2;i>=0;i--)
  {
    toplam=0;
    for(j=i+1;j<n;j++)
    {  toplam+=a[i][j]*x[j];}
    x[i]=(b[i]-toplam)/a[i][i];
  }
  return x;
}

public static double[] PolynomialLSQ(double xi[],double yi[],int n)
{ //Polynomial least square
int l=xi.length;
int i,j,k;
int np1=n+1;
double A[][];
A=new double[np1][np1];
double B[];
B=new double[np1];
double X[];
X=new double[np1];
for(i=0;i<n+1;i++)
  {  for(j=0;j<n+1;j++)
     {if(i==0 && j==0) A[i][j]=l;
     else  for(k=0;k<l;k++) A[i][j] += Math.pow(xi[k],(i+j));
     }
```

```
    for(k=0;k<l;k++) { if(i==0)  B[i]+= yi[k];
                  else      B[i] += Math.pow(xi[k],i)*yi[k];}
  }
X=gausswithpartialpivot(A,B);
double max=0;
for(i=0;i<n+1;i++)
 if(Math.abs(X[i]) > max) max = Math.abs(X[i]);
for(i=0;i<n+1;i++)
 if((Math.abs(X[i]/max) > 0) && (Math.abs(X[i]/max) < 1.0e-100)) X[i]=0;
return X;
}

public static double funcPolynomialLSQ(double e[],double x)
{
// this function calculates the value of
// least square curve fitting function
int n=e.length;
double ff;
if(n!=0.0)
  { ff=e[n-1];
  for(int i=n-2;i>=0;i--)
   { ff=ff*x+e[i]; }
  }
else
  ff=0;
return ff;
}

public static double error(double x[],double y[],double e[])
{
//calculates absolute square root error of a least square approach
double n=x.length;
 int k;
 double total=0;
 for(k=0;k<n;k++)
 {
 total+=(y[k]-funcPolynomialLSQ(e,x[k]))*(y[k]-funcPolynomialLSQ(e,x[k]));
 }
total=Math.sqrt(total);
return total;
}

public static double[][]  funcPolynomialLSQ(double xi[],double yi[],int polinomkatsayisi,int aradegersayisi)
 {
             //aradegersayisi:  x--o--o--x--o--o--x zincirinde x deneysel noktalar ise
             // ara değer sayisi 2 dir
              int n=xi.length;
             int nn=(n-1)*(aradegersayisi+1)+1;
             double z[][]=new double[2][nn];
   double E[]=PolynomialLSQ(xi,yi,polinomkatsayisi);
   System.out.println("katsayilar :\n"+Matrix.toStringT(E));
   double dx=0;
   int k=0;
   int i;
   for(i=0;i<(n-1);i++)
     {z[0][k]=xi[i];z[1][k]=funcPolynomialLSQ(E,z[0][k]);k++;
     for(int j=0;j<aradegersayisi;j++)
     {dx=(xi[i+1]-xi[i])/((double)aradegersayisi+1.0);
     z[0][k]=z[0][k-1]+dx;z[1][k]=funcPolynomialLSQ(E,z[0][k]);k++;}
     }
     z[0][k]=xi[i];z[1][k]=funcPolynomialLSQ(E,z[0][k]);
   return z;
 }


 public static void main(String args[]) throws IOException
  {
```

```
  double x[];
  double y[];
  String s1=JOptionPane.showInputDialog("Data file name (input1.txt)");
  // JFileChooser fc=new JFileChooser();
  // if (fc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {File file =
fc.getSelectedFile();s1=file.getName(); }
  double a[][]=Text.readDoubleT(s1);
  x=a[0];
  y=a[1];
  double z[][]=funcPolynomialLSQ(x,y,4,2);
  System.out.println("Polynomial Least Square Curve fitting\n"+Matrix.toStringT(z));
  Plot pp=new Plot(a[0],a[1]);
  pp.setPlabel("Polynomial Least Square Curve fitting");
  pp.setXlabel("x");
  pp.setYlabel("y=f(x)");
  pp.setPlotType(0,0);
  pp.addData(z[0],z[1]);
  pp.setPlotType(6,22);
  pp.setGrid(1,1);
  pp.plot();
  }
}
```

```
---------- Capture Output ----------
> "C:\turhan\java\bin\javaw.exe" NA44
katsayilar :
       0.000000000000000
      -0.000000000000000
       1.000000000000000
      -0.000000000000000
      -0.000000000000000

Polynomial Least Square Curve fitting
   1.0000000000   1.0000000000
   1.3333333333   1.7777777778
   1.6666666667   2.7777777778
   2.0000000000   4.0000000000
   2.3333333333   5.4444444444
   2.6666666667   7.1111111111
   3.0000000000   9.0000000000
   3.3333333333  11.1111111111
   3.6666666667  13.4444444444
   4.0000000000  16.0000000000
   4.3333333333  18.7777777778
   4.6666666667  21.7777777778
   5.0000000000  25.0000000000
   5.3333333333  28.4444444444
   5.6666666667  32.1111111111
   6.0000000000  36.0000000000
   6.3333333333  40.1111111111
   6.6666666667  44.4444444444
   7.0000000000  49.0000000000
   7.3333333333  53.7777777778
   7.6666666667  58.7777777778
   8.0000000000  64.0000000000
```



LINEAR CURVE FITTING : LEAST SQUARE METHOD WITH MULTI VARIABLES

Curve fitting of a multivariable function basically the same of general curve fitting of the single variable function. The only basic difference is definition of the function to be used. Assume that

$f(x_0, x_1,..., x_{n-1}, x_n) = \sum_{j=0}^{m} a_j^{(m)} \phi_j(x_0, x_1,..., x_{n-1}, x_n)$ j th degree function is given. It is desired to fit x_0, x_1, x_2,…, x_n ,f_i, i=0...n data set into this function. The best fitted $a_j^{(m)}$ values to be found. For this purpose Minimum value of

$H(a_0^{(m)},...., a_m^{(m)}) = \sum_{i=1}^{n} w(x_0, x_1,..., x_{n-1}, x_n)\left[f_i - \sum_{j=0}^{m} a_j^{(m)} \phi_j(x_0, x_1,..., x_{n-1}, x_n) \right]^2$ function should be found. w W

$(x_0, x_1,..., x_{n-1}, x_n)$ is called weight function and it should satisfy the condition

$w(x_0, x_1,..., x_{n-1}, x_n) \geq 0 \quad i = 1,...., n$. The minimum of the function is the root of the derivative of the function.

$$\frac{\partial H(a_0^{(m)},...., a_m^{(m)})}{\partial a_k^{(m)}} = 2\sum_{i=1}^{n} w(x_0, x_1,..., x_{n-1}, x_n)\left[f_i - \sum_{j=0}^{m} a_j^{(m)} \phi_j(x_0, x_1,..., x_{n-1}, x_n) \right]\phi_k(x_0, x_1,..., x_{n-1}, x_n) = 0 \quad k = 0,....., m$$

$$\left\{ \sum_{j=0}^{m} w(x_0, x_1,..., x_{n-1}, x_n)\phi_j(x_0, x_1,..., x_{n-1}, x_n)\phi_k(x_0, x_1,..., x_{n-1}, x_n) \right\}\left[a_j^{(m)}\right] = \left[\sum_{i=1}^{n} w(x_0, x_1,..., x_{n-1}, x_n)\phi_k(x_0, x_1,..., x_{n-1}, x_n) f_i \right] \quad k = 0,.., m$$

If unit value is taken as weight function, equation becomes:

$$\left\{ \sum_{j=0}^{m} \phi_j(x_0, x_1,..., x_{n-1}, x_n)\phi_k(x_0, x_1,..., x_{n-1}, x_n) \right\}\left[a_j^{(m)}\right] = \left[\sum_{i=1}^{n} \phi_k(x_0, x_1,..., x_{n-1}, x_n) f_i \right] \quad k = 0,.., m$$

As an example a surface polinomial can be defined as

$f(x,y) = a_0 + a_1x + a_2y + a_3x^2 + a_4y^2 + a_5xy + a_6x^3 + a_7y^3 + a_8x^2y + a_9xy^2 + a_{10}x^4 + a_{11}y^4 + a_{12}x^2y^2 + a_{13}x^3y + a_{14}xy^2$

This polynomial is defined in the example problem

Problem 6.3-1 Multidimensional general least square curve fitting

```
İmport java.io.*;
import java.util.*;
import javax.swing.*;
import java.awt.event.*;

// surface curve fitting
//f(x,y)=a0+a1*x+a2*y+a3*x^2+a4*y^2+a5*xy+a6*x^3+a7*y^3+a8*x^2y
//      +a9*xy^2+a10*x^4+a11*y^4+a12*x^2y^2+a13x^3y+a14*xy^3
abstract class f_xir
{
  abstract double func(double x[],int equation_ref);
}

class fa extends f_xir
{
```

```
            double func(double x[],int i)
            {//a general surface curve fitting model
   double xx=0.0;
   //density function
   if(i==0)      {xx=1.0;          }
   else if (i==1) {xx=x[0];         }
   else if (i==2) {xx=x[1];         }
   else if (i==3) {xx=x[0]*x[1];    }
   else if (i==4) {xx=x[1]*x[1];    }
   else if (i==5) {xx=x[0]*x[1];    }
   else if (i==6) {xx=x[0]*x[0]*x[0];}
   else if (i==7) {xx=x[1]*x[1]*x[1];}
   else if (i==8) {xx=x[0]*x[0]*x[1];}
   else if (i==9) {xx=x[0]*x[1]*x[1];}
   else if (i==10){xx=x[0]*x[0]*x[0]*x[0];}
   else if (i==11){xx=x[1]*x[1]*x[1]*x[1];}
   else if (i==12){xx=x[0]*x[0]*x[1]*x[1];}
   else if (i==13){xx=x[0]*x[0]*x[0]*x[1];}
   else if (i==14){xx=x[0]*x[1]*x[1]*x[1];}
   return xx;
            }
}
class NA46
{
//  General least square curve fitting with multivariable
            public static double[][] Transpose(double [][] left)
   { //transpose matrix (if A=a(i,j) Transpose(A)=a(j,i)
   int i,j;
   int n=left.length;
   int m=left[0].length;
   double b[][];
   b=new double[m][n];
   for(i=0;i<n;i++)
    {for(j=0;j<m;j++)  {b[j][i]=left[i][j];} }
   return b;
   }
  public static double[] gausswithpartialpivot(double a[][],double b[])
  { //Gauss elimination with partial pivoting
  int n=b.length;
  double x[]=new double[n];
  double carpan=0;
  double toplam=0;
  double buyuk;
  double dummy=0;
  //gauss elimination
  int i,j,k,p,ii,jj;
  for(k=0;k<(n-1);k++)
  { //partial pivoting
            p=k;
            buyuk=Math.abs(a[k][k]);
            for(ii=k+1;ii<n;ii++)
            {  dummy=Math.abs(a[ii][k]);
              if(dummy > buyuk) {buyuk=dummy;p=ii;}
            }
            if(p!=k)
            { for(jj=k;jj<n;jj++)
              { dummy=a[p][jj];
                a[p][jj]=a[k][jj];
                a[k][jj]=dummy;
              }
              dummy=b[p];
              b[p]=b[k];
              b[k]=dummy;
            }
            //
   for(i=k+1;i<n;i++)
   {  carpan=a[i][k]/a[k][k];
```

```
      a[i][k]=0;
      for(j=k+1;j<n;j++)
      {   a[i][j]-=carpan*a[k][j]; }
         b[i]   =b[i]   -carpan*b[k];
             }
  }
  //backward substitution
  x[n-1]=b[n-1]/a[n-1][n-1];
  for(i=n-2;i>=0;i--)
  {
    toplam=0;
    for(j=i+1;j<n;j++)
    {  toplam+=a[i][j]*x[j];}
    x[i]=(b[i]-toplam)/a[i][i];
  }
  return x;
}

public static double[] GeneralLeastSquare(double c[][],int n)
{
int n1=c.length;
  int n2=c[0].length-1;
  System.out.println("n1="+n1+"n2="+n2);
  double xi[][]=new double[n1][n2];
  double yi[]=new double[n1];
  for(int i=0;i<n1;i++){for(int j=0;j<n2;j++){xi[i][j]=c[i][j];};yi[i]=c[i][n2];}
  return GeneralLeastSquare(xi,yi,n);
}

public static double[] GeneralLeastSquare(double xi[][],double yi[],int n)
{ // n dimensional surface general least square curve fitting
fa f=new fa();
int l=xi.length;
int i,j,k;
int np1=n+1;
double A[][];
A=new double[np1][np1];
double B[];
B=new double[np1];
double X[];
X=new double[np1];
for(i=0;i<n+1;i++)
  {  B[i]=0;
             for(j=0;j<np1;j++)
    {
             A[i][j]=0.0;
    for(k=0;k<l;k++) A[i][j]+=f.func(xi[k],i)*f.func(xi[k],j);
    }
    for(k=0;k<l;k++) B[i]+= f.func(xi[k],i)*yi[k];
  }
X=gausswithpartialpivot(A,B);
//X=B/A;
double max=0;
for(i=0;i<n+1;i++)
 if(Math.abs(X[i]) > max) max = Math.abs(X[i]);
for(i=0;i<n+1;i++)
 if((Math.abs(X[i]/max) > 0) && (Math.abs(X[i]/max) < 1.0e-100)) X[i]=0;
Text.printT(X);
return X;
}

public static double funcGeneralLeastSquare(double e[],double x[])
{
// multidimensional function calculation
fa f=new fa();
int n=e.length;
double ff=0;
```

```
if(n!=0.0)
  {
  for(int i=n-1;i>=0;i--)
   {ff+=e[i]*f.func(x,i);}
  }
return ff;
}

public static double[] funcGeneralLeastSquare(double e[],double xi[][])
{
// multidimensional function calculation
fa f=new fa();
int n=e.length;
double ff[]=new double[xi.length];
for(int k=0;k<xi.length;k++)
{
if(n!=0.0)
  {
  for(int i=n-1;i>=0;i--)
   {ff[k]+=e[i]*f.func(xi[k],i);}
  }
}
return ff;
}

public static double[][] cikti(double c[][],int polinomkatsayisi,int aradegersayisi)
{ int n1=c.length;
  int n2=c[0].length-1;
  System.out.println("n1="+n1+"n2="+n2);
  double xi[][]=new double[n1][n2];
  double yi[]=new double[n1];
  for(int i=0;i<n1;i++){for(int j=0;j<n2;j++){xi[i][j]=c[i][j];};yi[i]=c[i][n2];
  }
  return cikti(xi,yi,polinomkatsayisi,aradegersayisi);
}


public static double[][] cikti(double xi[][],double yi[],int polinomkatsayisi,int aradegersayisi)
{
              int n=xi.length;
              int nk=xi[0].length;
              int nn=(n-1)*(aradegersayisi+1)+1;
    double E[]=GeneralLeastSquare(xi,yi,polinomkatsayisi);
    double x[][]=new double[nn][nk];
    double c[][]=new double[nn][nk+1];
    double yy[]=new double[nn];
    double dx[]=new double[nk];
    int k=0;
    int i,j,w;
    for(i=0;i<(n-1);i++)
      {   for(w=0;w<nk;w++){x[k][w]=xi[i][w];dx[w]=(xi[i+1][w]-xi[i][w])/((double)aradegersayisi+1.0);};k++;
         for(j=0;j<aradegersayisi;j++)
         { for(w=0;w<nk;w++){x[k][w]=x[k-1][w]+dx[w];};k++;}
      }
    for(w=0;w<nk;w++){x[k][w]=xi[i][w];}
    yy=funcGeneralLeastSquare(E,x);
    for(i=0;i<x.length;i++)
     {for(w=0;w<nk;w++) c[i][w]=x[i][w];
      c[i][nk]=yy[i];
     }
    return c;
}

public static double funcGeneralLeastSquare(double e[],double x,double y)
{
// multidimensionaş surface fitting
double xx[]=new double[2];
```

```
xx[0]=x;
xx[1]=y;
return funcGeneralLeastSquare(e,xx);
}

public static double hata(double c[][],double e[])
{int n1=c.length;
  int n2=c[0].length-1;
  System.out.println("n1="+n1+"n2="+n2);
  double xi[][]=new double[n1][n2];
  double yi[]=new double[n1];
  for(int i=0;i<n1;i++){for(int j=0;j<n2;j++){xi[i][j]=c[i][j];};yi[i]=c[i][n2];
  }
return hata(xi,yi,e);
}

public static double hata(double x[][],double y[],double e[])
{
//calculates absolute square root error of a least square approach
double n=x.length;
 int k;
 double total=0;
 for(k=0;k<n;k++)
 {
 total+=(y[k]-funcGeneralLeastSquare(e,x[k]))*(y[k]-funcGeneralLeastSquare(e,x[k]));
 }
total=Math.sqrt(total);
return total;
}

  public static void main(String[] args)
  {
  String s1=JOptionPane.showInputDialog("file name : ");
  double c[][]=Text.readDouble(s1);
  double b[]=GeneralLeastSquare(c,4);
  double xi[][]=cikti(c,4,5);
  System.out.println("output \n"+Matrix.toString(xi));
  System.out.println("error \n"+Matrix.toString(hata(c,b)));

  }
}
```

AIR CONDITIONING (This section is taken from Numerical Thermodynamics , Dr. M. Turhan ÇOBAN)

5.1 PROPERTIES OF WET AIR

Equation of states for dry air and water was investigated previously.The two fluid can be mixed to give properties of wet air. Details of wet air properties are investigated in every thermodynamics book. Mixing rulle usually assumes ideal gas for each gas, but water properties can be obtained by using actual eqaution of state. Wet air properties used in air conditioning and refrigeration applications. The basic mixing rule is based on adding up partial pressures of each component to obtain total pressure of the system.

$$P = P_{dry\ air} + P_{water\ \ vapor} \qquad (5.1.1)$$

Total mass of the gas is found by summing the mass of the component gases

$m = m_{dry\ air} + m_{water\ \ vapor}$ (5.1.2), The ratio of water vapor to dry air gives us specific moisture

The specific moisture can be calculated by using ideal gas law

$P_{dry\ air}V = m_{dry\ air}(R/M_{dry\ air})T$ (5.1.3),

$P_{water\ \ vapor}V = m_{water\ \ vapor}(R/M_{water\ \ vapor})T$ (5.1.4),

In these terms, P is pressure, m is mass, T is temperature in degree Kelvin. From these equations:

$$w = \left(\frac{M_{water\ \ vapor}}{M_{dry\ air}}\right)\frac{P_{water\ \ vapor}}{P - P_{water\ \ vapor}} \qquad (5.1.5),$$

where ratio of molar masses are equal to:

$$\left(\frac{M_{water\ vapor}}{M_{dry\ air}}\right)=\frac{18.016}{28.964197}=0.6220093$$

water vapor molar mass to ratio of saturated water vapor molar mass at same temperature.

$$\phi=\frac{x_{water\ vopor}}{x_{saturated\ water\ vopor}}=\frac{P_{water\ vopor}}{P_{saturated\ water\ vopor}} \quad (5.1.6),$$

Degree of saturation $DOS=\frac{m_{water\ vapor}}{m_{saturated\ water\ vapor}}=\frac{w_{water\ vapor}}{w_{saturated\ water\ vapor}}$ (5.1.7),

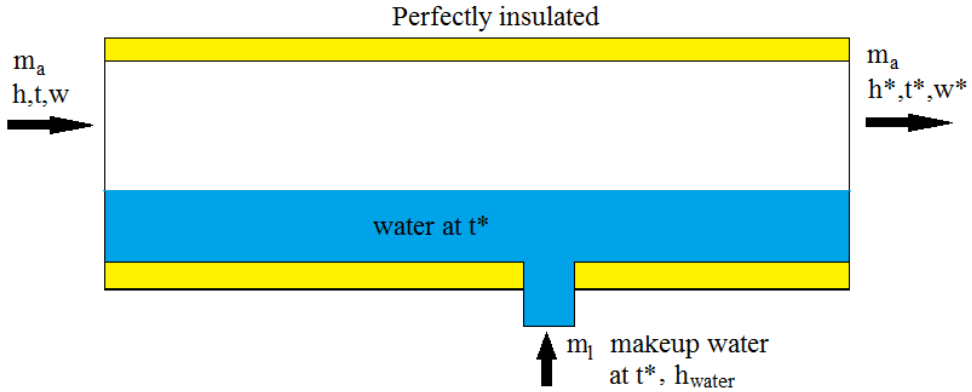
The enthalpy of wet air can be calculated summing partial enthalpies

$$h=\frac{H}{m_{dry\ air}}=h_{dry\ air}+\frac{m_{water\ vapor}}{m_{dry\ air}}h_{water\ vapor}=h_{dry\ air}+wh_{water\ vapor} \quad (5.1.8),$$

Entropy is also calculated the same way

$$s=\frac{s}{m_{dry\ air}}=s_{dry\ air}+\frac{m_{water\ vopor}}{m_{dry\ air}}s_{water\ vapor}=s_{dry\ air}+ws_{water\ vapor} \quad (5.1.9),$$

Another concept used for wet air is adyabatic saturation temperature. If air flow through an infinite length channel filled with water at the bottom and all walss are insulated, it will absorb water and will be reached to adyabatic saturation point. The temparature of adyabatic saturation point is also called wet air temperature, it is an idealised thermodynamic concept and can be calculated from the energy balance of the infinitely long channel. Basic energy equation:
Energy of the air entering the channel = enegy of the air leaving the channel + energy of evaporated water,



So $m_a h+m_l h^*_{water}=m_a h^*$

$m_l=m_a(w^*-w)$

$h+(w^*-w)h^*_{water}=h^*$ (5.1.10),

Wet air temperature is close to the wet bulb temperature which defines similar process as isothermal process rather than adiabatic process. In an isothermal process temperature is depends on the heat transfer coefficient. The last property should be defined is dew point temperature. If air is cooled without changing the pressure, in some temperature condensatiıon point is reached. It is called dew point temperature. In order to calculate properties of wet air a program, wetair.java is developed and codes are given in Appendix 7.

By using this program, thermodynamic properties of humidified air can be calculated by giving a pair (plus air pressure) of thermodynamic properties. Following properties can be defined in the program:

"tdb_twb" dry air - wet air temperature
"tdb_rh" dry air temperature - relative humidity
"tdb_w" dry air temperature - specifice humidity
"tdb_tdew" dry air temperature - çiğ noktası temperature
"tdb_pv" dry air temperature - su buharı basıncı
"tdb_dos" dry air temperature – degree of saturation
"tdb_h" dry air temperature - enthalpy
"tdb_s" dry air temperature - entropy
"w_rh" specific humidity - relative humidity
"w_h" specific humidity - enthalpy

After specifying known thermodynamic couple that properties are given to the program and results are obtained. For example to obtain properties 25 C dry bulb temperature and 20 degree C wet bulb temperature and 1.01325 bar pressure, the following code should be called

```
wetair y=new wetair();
double a[]=y.property("tdb_twb",25.0,20.0,1.0132);
```

A is a vector that carries out all thermodynamic properties. The parameters of a is as follows:

a[0]= P, pressure	a[8]= h enthalpy
a[1]= T, dry bulb temperature	a[9]= s, entropy
a[2]= v, specific volume	a[10]= dew point temperature
a[3]= Pv partial pressure of water vapor	a[11]= ha, enthalpy of dry air
a[4]= Pa partial pressure of dry air	a[12]= hv, enthalpy of water vapor
a[5]= w, specific humidity	a[13]= hv*w, latent enthalpy
a[6]= rh, relative humidity	a[14]= T wet air temperature
a[7]= dos, degree of saturation	

A simple program is given to show this call

Program 5.1.1 Properties of wetair

```
public class wetairtest
{
public static String output(double b[])
{
String s="";
for(int i=0;i<b.length;i++)
{s+="a"+i+": "+b[i]+"\n";}
return s;
}
public static void main(String arg[])
{
wetair wa=new wetair();
double a[]=wa.property("tdb_twb",25.0,20.0,1.01325);
System.out.println(output(a));
}
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" wetairtest
a0: 1.01325
a1: 25.0
a2: 0.844681235079535
a3: 0.020117894070933378
a4: 0.9931321059290666
a5: 0.012599267065824954
a6: 0.6348526592806563
a7: 0.6274558633488576
a8: 57.1893891270598
a9: 0.1983765100615907
a10: 17.59459840312558
a11: 25.092773476196555
a12: 2547.498634894733
a13: 32.09661565086324
a14: 20.0
> Terminated with exit code 0.
```

If only result set is required, and the values will not be used, a simpler call will be:

```
wetair y=new wetair();
System.out.println(y.toString("tdb_twb",25.0,20.0,1.0132));
```

This will give direct output with unit definitions.

Program 5.1.2 calling the properties of wetair

```
public class wetairtest1
{
public static void main(String arg[])
{
```

```
wetair wa=new wetair();
String s=wa.toString("tdb_twb",25.0,20.0,1.01325);
System.out.println(s);
}}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" wetairtest1
0 P,_pressure_____ 1.01325000000000_bar_____
1 T,_dry_bulb_temperature_____ 25.00000000000000_deg_C_____
2 v,_dry_air_specific_vol_____ 0.84468123507953_m^3/kg_____
3 Pv_water_vapor_pressure_____ 0.02011789407093_bar_____
4 Pa_air_partial_pressure_____ 0.99313210592907_bar_____
5 w,_humidity_ratio_____ 0.01259926706582_kg_vapor/kg_dry_air_____
6 rh,relative_humidity_____ 0.63485265928066_____
7 dos,degree_of_saturation_____ 0.62745586334886_____
8 h_enthalpy_____ 57.18938912705980_KJ/kg_dry_air_____
9 s,_entropy_____ 0.19837651006159_KJ/kg_dry_air_K_____
10 tdew,dew_point_temperature___ 17.59459840312558_deg_C_____
11 ha,_partial_air_enthalpy_____ 25.09277347619656_KJ/kg_dry_air_____
12 hv,partial_water_vap_enthalpy 2547.49863489473300_KJ/kg_dry_air_____
13 hv*w,_latent_enthalpy_____ 32.09661565086324_KJ/kg_dry_air_____
14 T_wet_bulb_temperature_____ 20.00000000000000_deg_C_____


> Terminated with exit code 0.
```

If only results are required from a Graphic user interface program, psTEN.java is prepared for you. Program will open with a pressure input window. When the desired pressure is given a psychrometric diagram will be drawn for us to data entry. Data can be inputted from here by using Mouse, or from the third page manual entry page for more accurate specifications. In both cases all the selected points will be entered to the psychrometric chart and all thermodynamic properties will be listed as a separate list. Values in this list can directly be copied by using ctr-C and past into any programs (such as MS Word, Open Office write or an editör program)

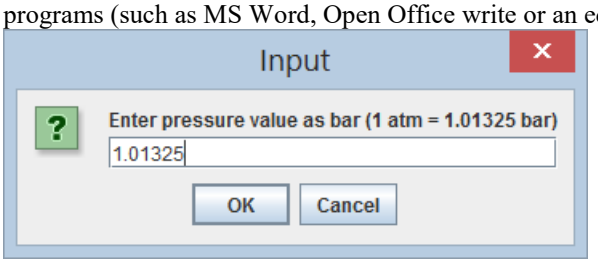


Figure 5.1.1 psTEN.java program entry point

Single zone winter air conditioning

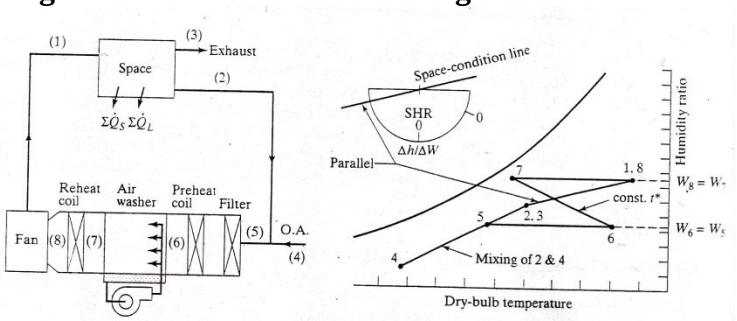


Figure 5.2.7.1 a single space winter air conditioning process

Figure 5.2.7.1 shows winter air conditioning process of a single zone. While some of the air is exhausted to replace fresh air, the remaining air circulates through a series of air conditioning processes, first, it is preheated, and then moisturised in an air washer. After a last heating process ir is send to the room. In the room, the warm air inlet is mixed up with the air in the room to create the comfort state required. The details of the individual processes are defined earlier as single processes so, we are only given here process diagrams and program codes to create winter air conditioning. It should be note that process can be mad efor a multizone system with a much complicated details.

winter air conditioning process thermodynamic properties

Pressure bar	Temperature degre...	v dry air specific vol...	Pv_water partial pre...	Pa_dry air partial pr...	w_specific humidit...	ϕ relative humidity	dos,degree of satur...	h enthalpy KJ/kgkh	s, entrophy KJ/kgkh	dp ,dew point temp...	ha,enthalpy of dry ai...	hv,enthalpy of stea...	hv*w,sensible heat ...	tw wet air temperat...
1.01325	45.0	0.9160093464586...	0.0162235780831...	0.9970264219168...	0.0101206816091...	0.1691227470398...	0.1556027422590...	71.36553936308134	0.2441649536341...	14.231228501611...	45.20358010634567	2584.9997329281...	26.1619592567356...	24.026031836406...
1.01325	21.0	0.8437043636001...	0.0124363904771...	1.0008136095228 24	0.0077287797944...	0.5	0.4937868598314...	40.710030617425...	0.1436348884100...	10.194691590905...	21.07703782841 16...	2540.244813693389	19.6329927890142...	14.601445973338...
1.01325	21.0	0.8437043636001...	0.0124363904771...	1.0008136095228 24	0.0077287797944...	0.5	0.4937868598314...	40.710030617425...	0.1436348884100...	10.194691590905...	21.07703782841 16...	2540.244813693389	19.6329927890142...	14.601445973338...
1.01325	0.0	0.7785480338037...	0.0061086255735...	1.0071413744264...	0.0037724444638...	1.0	1.0	9.436178591904245	0.0345424850799...	-4.9213543245329...	0.0	2501.3432755135...	9.436178591904245	-4.0139720462921...
1.01325	10.5	0.8110315879174...	0.0092824785199...	1.0039675214800 48	0.0057506121291...	0.7313201391474...	0.7288359800648...	25.033374795759 24	0.0898012868009...	5.898261128237889	10.537381746463 8...	2520.773914801 2...	14.4959930492954...	8.137276279613163
1.01325	25.0	0.8524909851492 12	0.0092824785199...	1.0039675214800 48	0.0057506121291...	0.2929236108078...	0.2863861270205...	39.744136644719...	0.1403815157146...	5.898261128237889	25.09277347619 65...	2547.7919288383...	14.6513631685227...	14.285027995258 6
1.01325	14.113304573013...	0.8269786570776...	0.0160999948195...	0.9971500051804...	0.0100423424519...	1.0	1.0	39.544109034421...	0.1389539233728...	14.113304573013...	14.16406893065 70...	2527.3027906901...	25.380040103764 62	14.113304572218...
1.01325	14.113304573013...	0.8269786570776...	0.0160999948195...	0.9971500051804...	0.0100423424519...	1.0	1.0	39.544109034421...	0.1389539233728...	14.113304573013...	14.16406893065 70...	2527.3027906901...	25.380040103764 62	14.113304572218...

winter air conditioning process calculated properties

pre heater heat transfer kW	Reheater heat transfer kW	water added in moisturising unit kg/s
59.83682475224401	129.53369895791252	0.017823632374112972

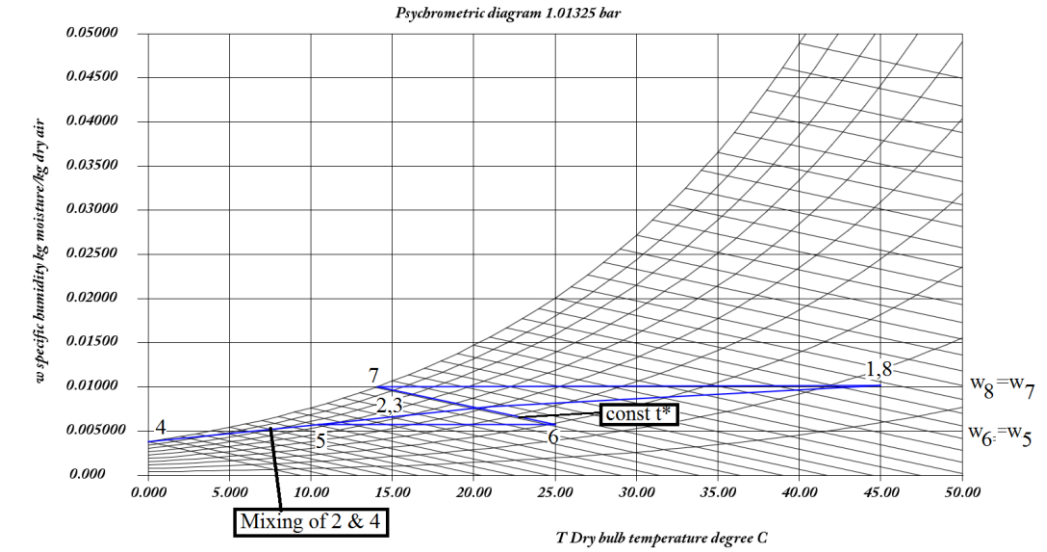


Figure 5.2.7.2 a single space winter air conditioning process

Program 5.2.7.1 single zone winter air conditioning process

```
import javax.swing.*;
public class wetairtest10
{ // cooling-demoisturizing and adiabatic winter air conditioning process
Plot pi;
double P;
wetair yh;
double tmin=0;
double tmax=50.0;
String heading[]={"Pressure bar",
"Temperature degree C",
"v dry air specific volume m^3/kgkh",
"Pv_water partial pressure bar",
"Pa_dry air partial pressure bar",
"w,_specific humidity kg moist./kg dry air",
"\u03D5'+" relative humidity",
"dos,degree of saturation",
"h enthalpy KJ/kgkh",
"s, entrophy KJ/kgkh",
"dp ,dew point temperature degree C",
"ha,enthalpy of dry air KJ/kgkh",
"hv,enthalpy of steam KJ/kgkh",
"hv*w,sensible heat KJ/kgkh",
"tw wet air temperature C"};
public double[][] winter_air_conditioning(double Pi,double T1,double T2,double f2,double T4,double f4,double T6,double Qsensible,double Qlatent)
{   P=Pi;
    yh=new wetair();
    ps_plot_hazirla(tmin,tmax,P);
                // room air conditions
                double a2[]=yh.property("tdb_rh",T2,f2,P);
                double h2=a2[8];
                double w2=a2[5];
                double hv2=a2[12];
                // outside air inlet
                double a4[]=yh.property("tdb_rh",T4,f4,P);
                double h4=a4[8];
                double w4=a4[5];
                // pre heating inlet
```

```
            double h5=h2*0.5+h4*0.5;
            double w5=w2*0.5+w4*0.5;
            double t5=0.5*T2+0.5*T4;
            // pre heating exit
            double w6=w5;
            double a6[]=yh.property("tdb_w",T6,w6,P);
            double h6=a6[8];
            // reheating entry

            double  m_supply_air=Qsensible/(1.0216*(T1-T2));
            //first guess w1
            double w1;
            double t7;
            double a1new[];
            double a7[];
            double a7new[];
            double h7new;
            double a6guess[];
            double t6guess;
            double titer=0.001;
            double  w1new=w2+titer;
             a1new=yh.property("tdb_w",T1,w1new,P);
            double hv1=a1new[12];
            double hort=(hv1+hv2)*0.5;
            w1=w2+(Qlatent/(m_supply_air*hort));
            a7new=yh.property("w_h",w1,h6,P);
            t7=a7new[1];
            double w7=w1;
            double mw=m_supply_air*(w7-w6);
            double a1[]=yh.property("tdb_w",T1,w1,P);
            double h1=a1[8];
            double Q56=m_supply_air*(h6-h5); // pre heater heating kW
            double Q78=(m_supply_air+mw)*(h1-h6);// re_heater heating kW
            // last evaluation
            double a1yeni[]=yh.property("tdb_w",T1,w1,P);
            double a2yeni[]=yh.property("tdb_rh",T2,f2,P);
            double a3yeni[]=a2yeni;
            double a4yeni[]=yh.property("tdb_rh",T4,f4,P);
            double a5yeni[]=yh.property("tdb_w",t5,w5,P);
            double a6yeni[]=yh.property("tdb_w",T6,w6,P);
            double a7yeni[]=yh.property("tdb_w",t7,w7,P);
            double a[][]=new double[8][a1yeni.length];
   a[0]=a1yeni;
   a[1]=a2yeni;
   a[2]=a[1];
   a[3]=a4yeni;
   a[4]=a5yeni;
   a[5]=a6yeni;
   a[6]=a7yeni;
   a[7]=a[6];
            addData(a4yeni,a5yeni,5,0,0,255);
   addData(a5yeni,a2yeni,5,0,0,255);
            addData(a2yeni,a1yeni,5,0,0,255);
            addData(a1yeni,a7yeni,5,0,0,255);
            addData(a7yeni,a6yeni,5,0,0,255);
            addData(a6yeni,a5yeni,5,0,0,255);
   ps_plot(0,50);
            print(a,heading,"winter air conditioning process thermodynamic properties");
            String baslik2[]={"pre heater heat transfer kW","Reheater heat transfer kW","water added in moisturising unit kg/s"};
            double b[][]={{Q56,Q78,mw}};
            print(b,baslik2,"winter air conditioning process calculated properties");

            return a;
}
public void print(double a[][],String s)
{//String heading[]={"winter air conditioning"};
 Text.print(a,heading,"winter air conditioning data");}

public void print(double a[][],String heading2[],String s)
{String heading[];Text.print(a,heading2,s);}
public void ps_plot_hazirla(double tmin,double tmax,double P1)
    {
            double x1[][]=new double[62][51];
            double y1[][]=new double[62][51];
            double A[][][]=new double[2][62][51];
```

```
            A=ps_verisi(tmin,tmax,P1);
            for(int i=0;i<A[0].length;i++)
             for(int j=0;j<A[0][0].length;j++)
                {x1[i][j]=A[0][i][j];y1[i][j]=A[1][i][j];}
 pi=new Plot(x1,y1);
 String ss2="Psychrometric diagram "+P+" bar";
 pi.setPlabel(ss2);
 pi.setXlabel("T Dry bulb temperature degree C");
 pi.setYlabel("w specific humidity kg moisture/kg dry air");
 pi.setMinMax(tmin,tmax,0.0,0.05);
 pi.setXgrid(1);
 pi.setYgrid(1);
 }
public static double[][][] ps_verisi(double tmin,double tmax,double P)
{
        //prepare pyschrometric chart data for pressure P
wetair w=new wetair();
        double aa[]=new double[14];
double bb[]=new double[14];
        //f2 ff=new f2();
        //double x1[][]=new double[62][51];
        //double y1[][]=new double[62][51];
        double A[][][]=new double[2][62][51];
//initilise as a function plot
int i,j;
i=0;
double h;
for(double rh=0.1;rh<=1.0;rh+=0.1)
{   j=0;
    for(double t=tmin;t<=tmax;t+=1.0)
    {
            aa=w.property1("tdb_rh",t,rh,P);
            h=aa[8];
              if(rh>0.99)
      {
              h=aa[8];
      A[0][11+j][0]=t;A[1][11+j][0]=aa[5];
      bb=w.property1("w_h",0.0,h,P);
      A[0][11+j][1]=bb[10];A[1][11+j][1]=0.0;
      }
              A[0][i][j]=t;
      A[1][i][j]=aa[5];
              j++;
      }
    i++;
}
return A;
}
 public void addData(double a1[][],int plottype,int red,int green,int blue)
 { double x1[]=new double[2];
   double y1[]=new double[2];
   x1[0]=a1[0][1];
   y1[0]=a1[0][5];
   x1[1]=a1[1][1];
   y1[1]=a1[1][5];
   pi.addData(x1,y1,plottype,red,green,blue);
 }

 public void addData(double a1[],double a2[],int plottype,int red,int green,int blue)
 { double x1[]=new double[2];
   double y1[]=new double[2];
   x1[0]=a1[1];
   y1[0]=a1[5];
   x1[1]=a2[1];
   y1[1]=a2[5];
   pi.addData(x1,y1,plottype,red,green,blue);
 }

 public void addData(double a1[],int plottype,int red,int green,int blue)
 { double x1[]=new double[1];
   double y1[]=new double[1];
   x1[0]=a1[1];
   y1[0]=a1[5];
   pi.addData(x1,y1,plottype,red,green,blue);
 }
```

```
    public void ps_plot(double tmin,double tmax)
    {   pi.setMinMax(tmin,tmax,0,0.05);
                pi.plot();
            }

public static void main(String arg[])
{ wetairtest10 wat=new wetairtest10();
double p=1.01325; //  bar
double T1=45.0;   //  room inlet temperature degree C
double T2=21.0;   //  room exit temperature
double f2=0.5;    //   room exit relative humidity
double T4=0.0;    //  outside inlet temperature
double f4=1.0;    //   outside  relative humidity
double T6=25.0;   //   preheater exit temperature
double Qsensible=100.0; //  sensible heat load
double Qlatent=25.0;   //  latent heat load
double a[][]=wat.winter_air_conditioning(p,T1,T2,f2,T4,f4,T6,Qsensible,Qlatent);
}
}
```

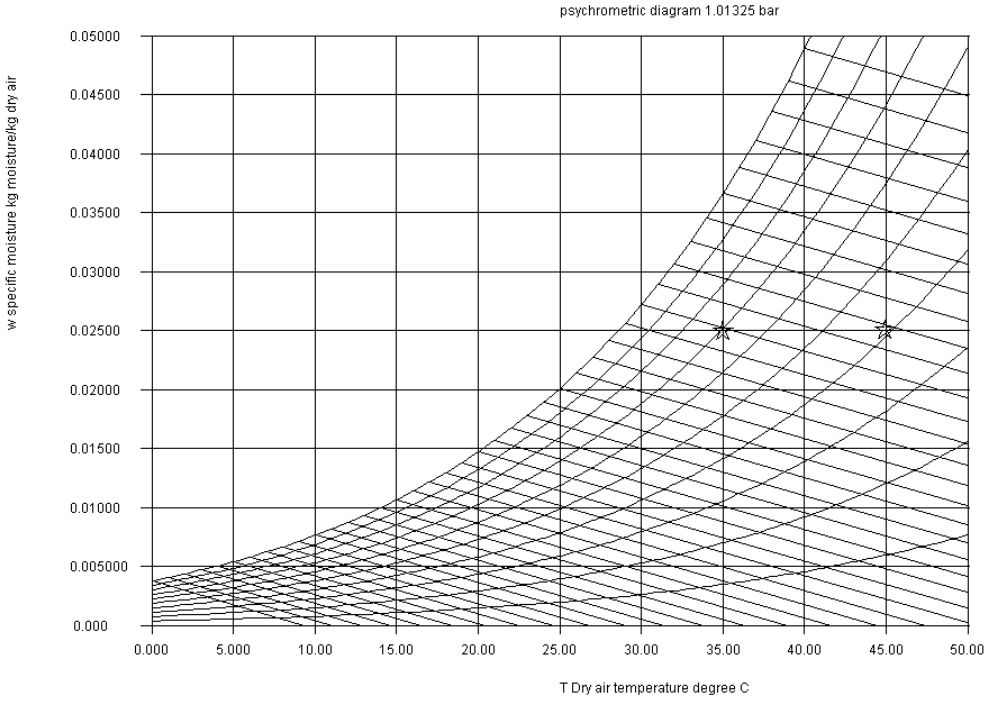


Figure 5.1.2 psTEN.java program dynamic graphic data entry psychrometric diagram

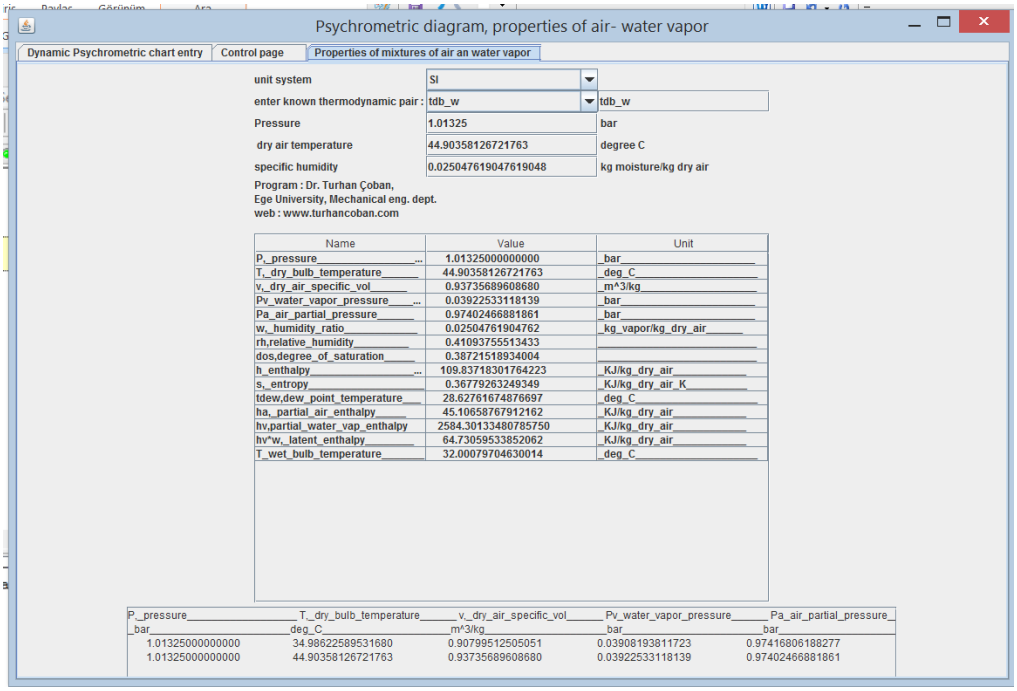


Figure 5.1.3 psTEN.java program data entry and output

Table Program psTEN outpus as copied to MS Word program.

P,_pressure	T,_dry_bulb_temperature	v,_dry_air_specific_vol	Pv_water_vapor_pressure	Pa_air_partial_pressure	w,_humidity_ratio	rh,relative_humidity	dos,degree_of_saturation	hv*w,_latent_enthalpy	T_wet_bulb_temperature
_bar	deg_C	m^3/kg	bar	bar	kg_vapor/kg_dry_air			KJ/kg_dry_air	KJ/
1.01325000000000	34.98622589531680	0.90799512505051	0.03908193811723	0.97416806188277	0.02495238095238	0.69498646380791	0.68274984816344	99.15201142589950	
1.01325000000000	44.90358126721763	0.93735689608680	0.03922533118139	0.97402466881861	0.02504761904762	0.41093755513433	0.38721518934004	109.83718301764223	

Of course, if a certain psychrometric processes to be defined, a basic form of the equation should be utilised rather than a A GUI program.

5.2.8 Single zone summer air conditioning process

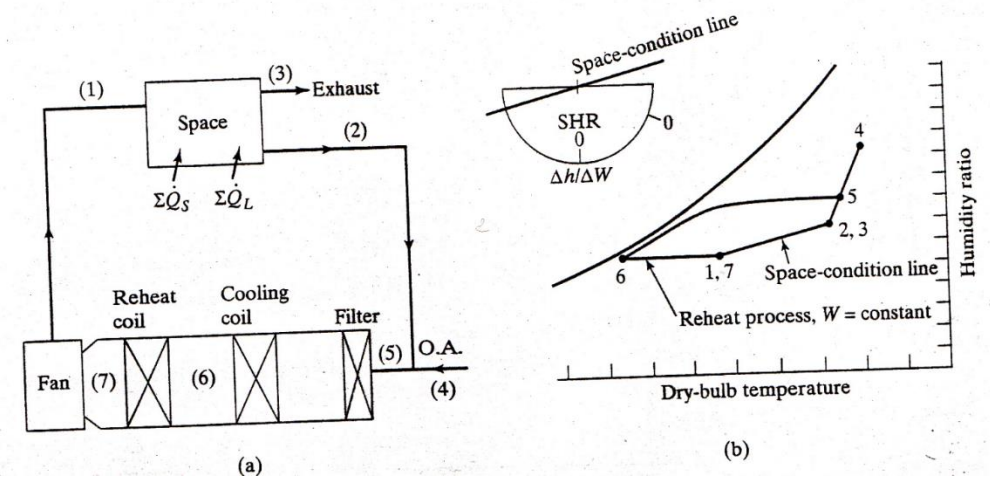
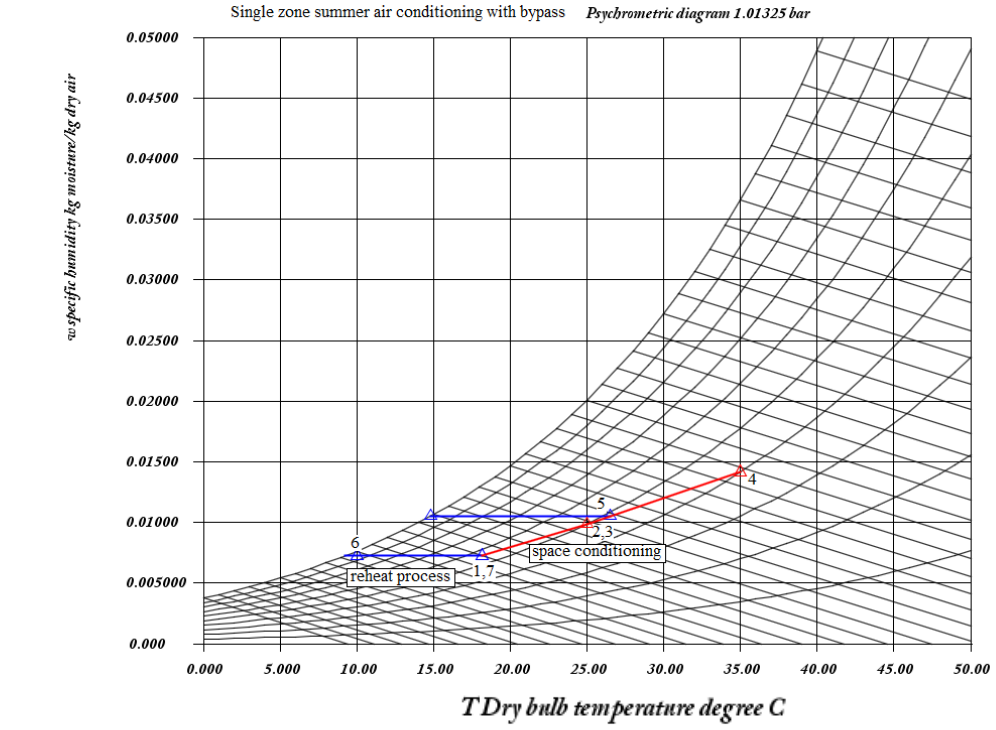


Figure 5.2.81 a single space summer air conditioning process with a bypass

Summer comfort air conditioning process include cooling and dehumidification processes. Dehumidification usually applied by cooling, but since dehumidified air has very high relative humidity, it is dropped either by mixing this air with the air bypassed from the room. Air can be mixed in the room directly, but cleaning of the air required also some filtering process. Or it can be reheat by using a heating coil. In the given process reheating is used.



summer air conditioning data

Pressure bar	Temperature ...	v dry air specif...	Pv_water parti...	Pa_dry air par...	w,_specific hu...	ϕ relative hum...	dos,degree of...	h enthalpy KJ/...	s, entrophy KJ...	dp ,dew point ...	ha,enthalpy of...	hv,enthalpy of ...	hv*w,sensible...	tw wet air tem...
1.01325	18.18907891...	0.834995376...	0.011661528...	1.001588471...	0.007241623...	0.558356798...	0.553214731...	36.61297328...	0.129713841...	9.236995232...	18.25526938...	2535.026050...	18.35770389...	13.04789836...
1.01325	25.0	0.858099633...	0.015844537...	0.997405462...	0.009880470...	0.5	0.492057122...	50.26440154...	0.175628665...	13.86700289...	25.09277347...	2547.614331...	25.17162806...	17.88662108...
1.01325	25.0	0.858099633...	0.015844537...	0.997405462...	0.009880470...	0.5	0.492057122...	50.26440154...	0.175628665...	13.86700289...	25.09277347...	2547.614331...	25.17162806...	17.88662108...
1.01325	35.0	0.892847862...	0.022510784...	0.990739215...	0.014131916...	0.4	0.386367279...	71.41004813...	0.245116228...	19.38725809...	35.14569674...	2566.131204...	36.26435138...	23.93488029...
1.01325	26.50924955...	0.863330438...	0.016850188...	0.996399811...	0.010518187...	0.486223230...	0.477534715...	53.43624852...	0.186179408...	14.81726776...	26.61384280...	2550.405168...	26.82563928...	18.89184789...
1.01325	10.0	0.811524982...	0.011661528...	1.001588471...	0.007241623...	0.95	0.949417848...	28.28279788...	0.100711597...	9.236995232...	10.03555124...	2519.772943...	18.24724664...	9.581246535...
1.01325	9.236995232...	0.809338164...	0.011661528...	1.001588471...	0.007241623...	1.0	1.0	27.50672583...	0.097967047...	9.236995232...	9.269763010...	2518.352845...	18.23696282...	9.236995231...
1.01325	14.81726776...	0.829629399...	0.016850188...	0.996399811...	0.010518187...	1.0	1.0	41.46688420...	0.145425452...	14.81726776...	14.87067124...	2528.592824...	26.59621295...	14.81726776...

single zone summer air conditioni...

evaporator heat transfer k...	heater heat transfer kW	condansation mass flow ...
754.6035194079752	249.90526195869305	0.09829691816952062

Program 5.28.1 single zone summer air conditioning process

```
import javax.swing.*;
public class wetairtest11
{ // cooling-demoisturizing and adiabatic summer air conditioning process
Plot pi;
double P;
wetair yh;
double tmin=0;
double tmax=50.0;
String heading[]={"Pressure bar",
"Temperature degree C",
"v dry air specific volume m^3/kgkh",
"Pv_water partial pressure bar",
"Pa_dry air partial pressure bar",
"w,_specific humidity kg moist./kg dry air",
'\u03D5'+" relative humidity",
"dos,degree of saturation",
"h enthalpy KJ/kgkh",
"s, entrophy KJ/kgkh",
"dp ,dew point temperature degree C",
"ha,enthalpy of dry air KJ/kgkh",
"hv,enthalpy of steam KJ/kgkh",
"hv*w,sensible heat KJ/kgkh",
"tw wet air temperature C"};
public double[][] summer_air_conditioning(double Pi,double m1, double m3,double T2, double f2,
                    double T4, double f4, double T6,double f6,double dQsensible,double dQlatent)
{ // m1 mass flow rate,    room girişi kg/s
  // m3 mass flow rate,    eksoz havası kg/s
```

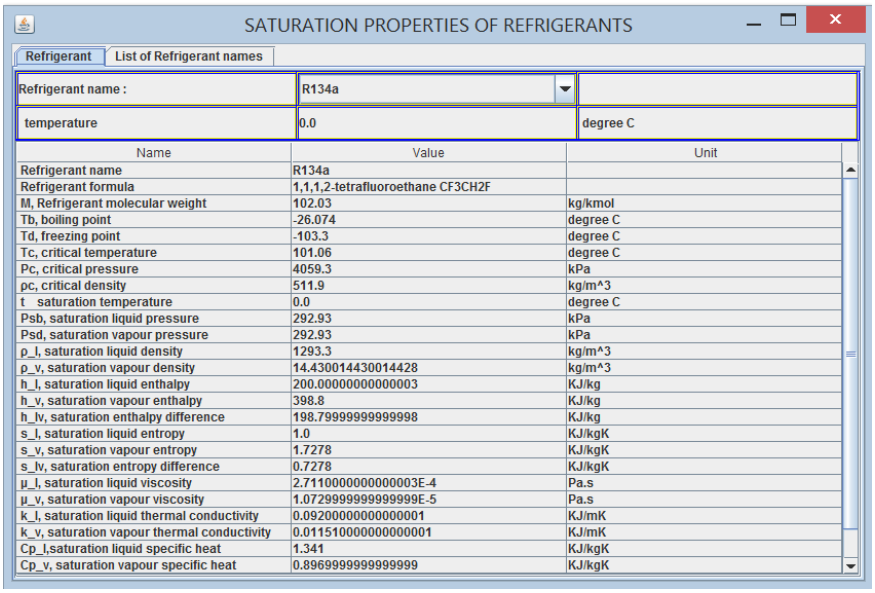
```
// T2 room temperature,    degree C
// f2 room relative humidity,   0-1
// T4 outside air temperature, degree C
// f4 outside air relative humidity 0-1
// T6 evaporator exit temperature
// f6 evaporator exit relative humidity
// f2 giriş relative humidity, ikinci kanal 0-1
// dQsensible sensible heat input kW
// dQlatent   latent heat input kW

//2 nd point room exit
P=Pi;
ps_plot_hazirla(tmin,tmax,P);
yh=new wetair();
double a2[]=yh.property("tdb_rh",T2,f2,P);
double h2=a2[8];
double w2=a2[5];
double ha2=a2[11];
//6 th point evaporator exit
double a6[]=yh.property("tdb_rh",T6,f6,P);
double h6=a6[8];
double w6=a6[5];
double w1=w6;
double ha1=ha2-dQsensible/m1;
//15 degree room temperature evaluation
// 1A noktası
double a1A[]=yh.property("tdb_w",15.0,w1,P);
double ha1A=a1A[11];
double Cp1=(ha2-ha1A)/(T2-15.0);
double T1=ha1/Cp1;
double a1[]=yh.property("tdb_w",T1,w1,P);
double h1=a1[8];
double m2=m1-m3;
double m4=m3;
double m5=m1;
// outside air entry
double a4[]=yh.property("tdb_rh",T4,f4,P);
double h4=a4[8];
double w4=a4[5];

double h5=(m4*h4+m2*h2)/m5;
double w5=(m4*w4+m2*w2)/m5;
double a5[]=yh.property("w_h",w5,h5,P);
double T5=a5[1];
double Q56=m5*(h5-h6);//KW
double Q67=m5*(h1-h6);//KW
double dm=m5*(w5-w6);
double a8[]=yh.property("w_rh",w6,1.0,P);
double a9[]=yh.property("w_rh",w5,1.0,P);
double a[][]=new double[8][a1.length];
a[0]=a1;
a[1]=a2;
a[2]=a2;
a[3]=a4;
a[4]=a5;
a[5]=a6;
a[6]=a8;
a[7]=a9;
// Plot
// thick blue line
addData(a9,a5,5,0,0,255);
//triangle
addData(a9,a5,24,0,0,255);
addData(a8,a1,5,0,0,255);
addData(a6,a1,24,0,0,255);
addData(a1,a2,5,255,0,0);
addData(a2,a4,5,255,0,0);
```

```
  addData(a2,a4,24,255,0,0);
  ps_plot();
  print(a,"single zone summer air conditioning process thermodynamic properties");
  String baslik2[]={"evaporator heat transfer kW","heater heat transfer kW","condansation mass flow
rate kg/s"};
  double b[][]={{Q56,Q67,dm}};
  print(b,baslik2,"single zone summer air conditioning process exit properties");
  return a;
}
public void print(double a[][],String s)
{//String heading[]={"summer air conditioning"};
 Text.print(a,heading,"summer air conditioning data");}

public void print(double a[][],String heading2[],String s)
{String heading[];Text.print(a,heading2,s);}
public void ps_plot_hazirla(double tmin,double tmax,double P1)
     {
               double x1[][]=new double[62][51];
               double y1[][]=new double[62][51];
               double A[][][]=new double[2][62][51];
               A=ps_verisi(tmin,tmax,P1);
               for(int i=0;i<A[0].length;i++)
                for(int j=0;j<A[0][0].length;j++)
                   {x1[i][j]=A[0][i][j];y1[i][j]=A[1][i][j];}
     pi=new Plot(x1,y1);
     String ss2="Psychrometric diagram "+P+" bar";
     pi.setPlabel(ss2);
     pi.setXlabel("T Dry bulb temperature degree C");
     pi.setYlabel("w specific humidity kg moisture/kg dry air");
     pi.setMinMax(tmin,tmax,0.0,0.05);
     pi.setXgrid(1);
     pi.setYgrid(1);
     }
    public static double[][][] ps_verisi(double tmin,double tmax,double P)
    {
              //prepare pyschrometric chart data for pressure P
    wetair w=new wetair();
              double aa[]=new double[14];
    double bb[]=new double[14];
              //f2 ff=new f2();
              //double x1[][]=new double[62][51];
              //double y1[][]=new double[62][51];
              double A[][][]=new double[2][62][51];
    //initilise as a function plot
    int i,j;
    i=0;
    double h;
    for(double rh=0.1;rh<=1.0;rh+=0.1)
    {   j=0;
       for(double t=tmin;t<=tmax;t+=1.0)
       {
                 aa=w.property1("tdb_rh",t,rh,P);
                 h=aa[8];
                   if(rh>0.99)
         {
                    h=aa[8];
         A[0][11+j][0]=t;A[1][11+j][0]=aa[5];
         bb=w.property1("w_h",0.0,h,P);
         A[0][11+j][1]=bb[10];A[1][11+j][1]=0.0;
         }
                    A[0][i][j]=t;
         A[1][i][j]=aa[5];
                    j++;
         }
       i++;
    }
    return A;
```

```
    }
    public void addData(double a1[][],int plottype,int red,int green,int blue)
    { double x1[]=new double[2];
      double y1[]=new double[2];
      x1[0]=a1[0][1];
      y1[0]=a1[0][5];
      x1[1]=a1[1][1];
      y1[1]=a1[1][5];
      pi.addData(x1,y1,plottype,red,green,blue);
    }

    public void addData(double a1[],double a2[],int plottype,int red,int green,int blue)
    { double x1[]=new double[2];
      double y1[]=new double[2];
      x1[0]=a1[1];
      y1[0]=a1[5];
      x1[1]=a2[1];
      y1[1]=a2[5];
      pi.addData(x1,y1,plottype,red,green,blue);
    }

    public void addData(double a1[],int plottype,int red,int green,int blue)
    { double x1[]=new double[1];
      double y1[]=new double[1];
      x1[0]=a1[1];
      y1[0]=a1[5];
      pi.addData(x1,y1,plottype,red,green,blue);
    }

    public void ps_plot()
    {   pi.setMinMax(tmin,tmax,0,0.05);
                pi.plot();
              }

public static void main(String arg[])
{ wetairtest11 wat=new wetairtest11();
double p=1.01325; //  bar
double T1=30.0;   //  room inlet temperature degree C
double m1=30.0;//kg/s
double m3=4.5; //kg/s
double T2=25.0;   //  room exit temperature
double f2=0.5;    //   room exit relative humidity
double T4=35.0;    //  outside inlet temperature
double f4=0.4;    //   outside  relative humidity
double T6=10.0;   //   preheater exit temperature
double f6=0.95;
double Qsensible=205.0; //  sensible heat load
double Qlatent=88.0;   //  latent heat load
double a[][]=wat.summer_air_conditioning(p,m1,m3,T2,f2,T4,f4,T6,f6,Qsensible,Qlatent);
}
}
```

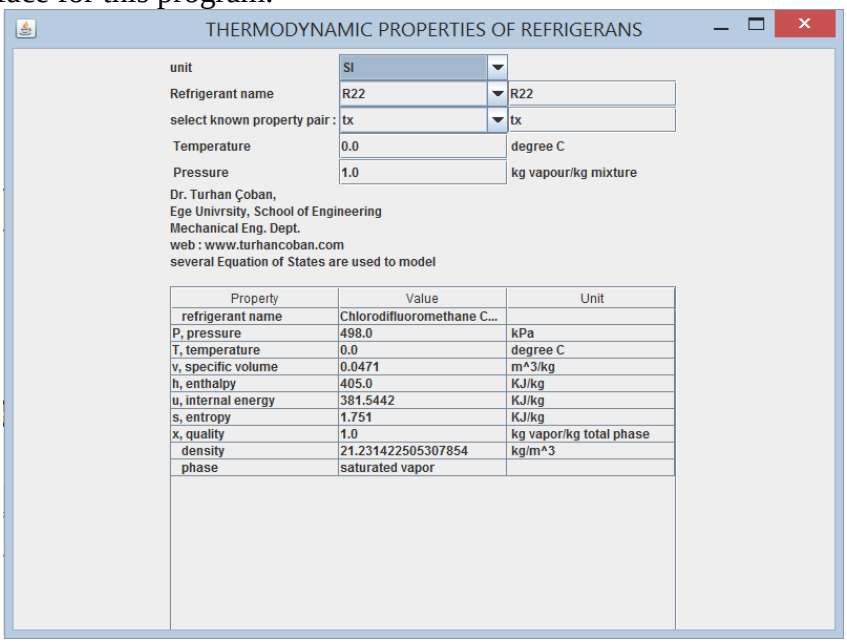
REFRIGERATION & HEAT PUMP CYCLES

Refrigeration & Heat pump cycles are basic cycles to use for heating and cooling spaces (heat transfer). Ususally work supply for the process and heat energy output is taken. A large range of refrigerants might use in refrigeration cycle. Several EOS for refrigerants are collected in one class, refrigerant.java. the class code is listed in A5. In addition to that, saturation properties of the refrigerants are evaluated in **ref_CS3EN.java** class. This program is basically a cubic spline curve fitting of the saturation $P_s(T_s)$ data. One advantage of such a curve fitting is simple creation of inverse curve fittin so function $T_s(P_s)$ can be evaluated as easily. Ref_CS3EN is also have thermophysical properties of refrigerants such as viscosity, thermal conductivity, surface tension. These packages are utilized in heat transfer applications as well as thermodynamic applications. Graphic user output of **ref_CS3EN** class is as follows:



Name	Value	Unit
Refrigerant name	R134a	
Refrigerant formula	1,1,1,2-tetrafluoroethane CF3CH2F	
M, Refrigerant molecular weight	102.03	kg/kmol
Tb, boiling point	-26.074	degree C
Td, freezing point	-103.3	degree C
Tc, critical temperature	101.06	degree C
Pc, critical pressure	4059.3	kPa
ρc, critical density	511.9	kg/m^3
t saturation temperature	0.0	degree C
Psb, saturation liquid pressure	292.93	kPa
Psd, saturation vapour pressure	292.93	kPa
ρ_l, saturation liquid density	1293.3	kg/m^3
ρ_v, saturation vapour density	14.430014430014428	kg/m^3
h_l, saturation liquid enthalpy	200.00000000000003	KJ/kg
h_v, saturation vapour enthalpy	398.8	KJ/kg
h_lv, saturation enthalpy difference	198.79999999999998	KJ/kg
s_l, saturation liquid entropy	1.0	KJ/kgK
s_v, saturation vapour entropy	1.7278	KJ/kgK
s_lv, saturation entropy difference	0.7278	KJ/kgK
μ_l, saturation liquid viscosity	2.7110000000000003E-4	Pa.s
μ_v, saturation vapour viscosity	1.0729999999999999E-5	Pa.s
k_l, saturation liquid thermal conductivity	0.09200000000000001	KJ/mK
k_v, saturation vapour thermal conductivity	0.011510000000000001	KJ/mK
Cp_l,saturation liquid specific heat	1.341	KJ/kgK
Cp_v, saturation vapour specific heat	0.8969999999999999	KJ/kgK

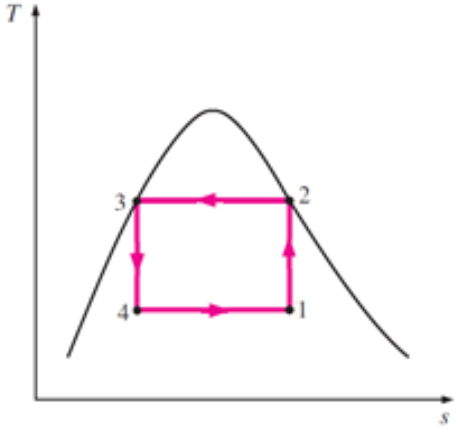
refrigerantEN is the equation of states for thermodynamic property calculations. **refTableEN** is graphic user interface for this program.

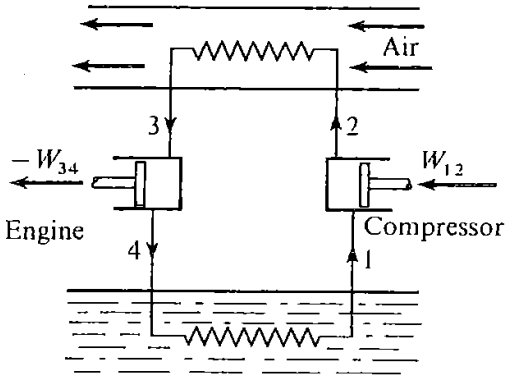


Property	Value	Unit
refrigerant name	Chlorodifluoromethane C...	
P, pressure	498.0	kPa
T, temperature	0.0	degree C
v, specific volume	0.0471	m^3/kg
h, enthalpy	405.0	KJ/kg
u, internal energy	381.5442	KJ/kg
s, entropy	1.751	KJ/kg
x, quality	1.0	kg vapor/kg total phase
density	21.231422505307854	kg/m^3
phase	saturated vapor	

By using this class we can able to program refrigerant cycles.

4.1 CARNOT CYCLE





Carnot cycle is an ideal cycle and it is important as a reference cycle . It consists of two isothermal and two isentropic processes. If a carnot cycle constructed by using an isentropic compressor , a condenser , an isentropic turbine and a boiler (evaporator), and, the cycle can be calculated by the following equations:

$W_{isentropic\,compressor} = m(h_2 - h_1) \qquad s_2 = s_1$ (4.1.1)

$Q_{condenser} = m(h_2 - h_3) = mT_2(s_2 - s_3) = mT_2(s_1 - s_4)$ (4.1.2)

$W_{isentropic\ turbine} = m(h_3 - h_4) \qquad s_3 = s_4$ (4.1.3)

$Q_{evaporator} = m(h_1 - h_4) = mT_1(s_1 - s_4)$ (4.1.4)

$W_{net\ in} = W_{isentropic\,compressor} - W_{isentropic\ turbine} = Q_{condenser} - Q_{boiler}$ (4.1.5)

$W_{net\ in} = mT_2(s_1 - s_4) - mT_1(s_1 - s_4) = m(T_2 - T_1)(s_1 - s_4)$ (4.1.6)

$$Coefficient\ of\ \ Performance\ \ for\ evaporator = \frac{Q_{evaporator}}{W_{compressor} - W_{turbine}} = \frac{mT_1(s_1 - s_4)}{m(T_2 - T_1)(s_1 - s_4)} = \frac{T_1}{(T_2 - T_1)} \quad (4.1.7)$$

$$Coefficient\ of\ \ Performance\ \ for\ condenser = \frac{Q_{condenser}}{W_{compressor} - W_{turbine}} = \frac{mT_2(s_1 - s_4)}{m(T_2 - T_1)(s_1 - s_4)} = \frac{T_2}{(T_2 - T_1)} \quad (4.1.8)$$

A computer model is developed to analyse this cycle in details

Program 4.1.1 ideal refrigeration Carnot cycle

```
 public class carnot1
{
// a R134 ideal carnot refrigeration cycle
// refrigerantEN b=new refrigerantEN("R134a");
// double a[]=b.property("tp",30.0,1.01325);
///---- String s values and meanings --------
//    tv=temperature-specific volume
//    tp=temperature-pressure
//    th=temperature-enthalpy
//    tu=temperature-internal energy
//    ts=temperature-entropy
//    tx=temperature quality
//    pv=pressure-specific volume
//    pt=pressure-temperature
//    ph=pressure-enthapy
//    pu=pressure-internal energy
//    ps=pressure-entropy
//    px=pressure-quality
//    vp=specific volume-pressure
//    vt=specific volume-temperature
//---- output values (method property) ---------
// r[0] P Pressure kPa
// r[1] t temperature degree C
// r[2] v specific volume m^3/kg
```

```
// r[3] h enthapy KJ/kg
// r[4] u internal energy KJ/kg
// r[5] s entropy KJ/kgK
// r[6] x quality kg vapour/kg total
// r[7] ro density kg/m^3

double T[];
double P[];
double h[];
double s[];
double x[];
double COP_condenser,COP_evaporator,COP_evap_carnot,COP_condenser_carnot;
double m;//kg/s
double Wt,Wp,W;//work
double Qevaporator,Qcondenser;
refrigerantEN b;
double c1[][];
double c3[][];
public carnot1(double mi,double P1, double P2)
{
m=mi;
b=new refrigerantEN("R134a");
T=new double[9];
P=new double[9];
h=new double[9];
s=new double[9];
x=new double[9];

P[1]=P1;//compressor inlet
P[2]=P2;//compressor outlet
c1=new double[2][20];
c3=new double[2][20];
}
public void cycle()
{
// isentropic compressor output
double a1[]=b.property("px",P[2],1.0);
h[2]=a1[3];
s[2]=a1[5];
T[2]=a1[1];
// isentropik compressor input
s[1]=s[2];
a1=b.property("ps",P[1],s[1]);
T[1]=a1[1];
h[1]=a1[3];
x[1]=a1[6];

//isentropic turbine input
P[3]=P[2];
a1=b.property("px",P[3],0.0);
h[3]=a1[3];
s[3]=a1[5];
T[3]=a1[1];
// //isentropic turbine output
P[4]=P[1];
s[4]=s[3];
a1=b.property("ps",P[4],s[4]);
h[4]=a1[3];
s[4]=a1[5];
T[4]=a1[1];
x[4]=a1[6];


// isentropik compressor
Wp=m*(h[2]-h[1]);
Wt=m*(h[3]-h[4]);
W=Wp-Wt;
```

```
Qevaporator=m*(h[1]-h[4]);
Qcondenser=m*(h[2]-h[3]);
COP_evaporator=Qevaporator/W;
COP_condenser_carnot=(T[2]+273.15)/(T[2]-T[1]);
COP_condenser=Qcondenser/W;
COP_evap_carnot=(T[1]+273.15)/(T[2]-T[1]);
}

public String toString()
{
cycle();
String ss="R134a carnot refrigeration cycle\n";
ss+=" compressor work = "+Wp+" kW\n";
ss+=" evaporator work = "+Wt+" kW\n";
ss+=" net work input= "+W+" kW\n";
ss+=" condenser heat output = "+Qcondenser+" kW\n";
ss+=" evaporator heat input = "+Qevaporator+" kW\n";
ss+=" COP evaporator    = " + COP_evaporator+" kW\n";
ss+=" COP condenser     = " + COP_condenser+" kW\n";
ss+=" COP evaporator  carnot = " + COP_evap_carnot+" kW\n";
ss+=" COP condenser   carnot = " + COP_condenser_carnot+" kW\n";
ss+=" h1 compressor input = "+h[1] +" kJ/kg\n";
ss+=" T1 compressor input = "+T[1] +" derece C\n";
ss+=" P1 compressor input = "+P[1] +" kPa \n";
ss+=" s1 compressor input = "+s[1] +" kJ/kgK\n";
ss+=" x1 compressor input = "+x[1] +" kgvapor/kgtotal"+"\n";
ss+=" T2 compressor output = "+T[3] +" derece C\n";
ss+=" P2 compressor output = "+P[3] +" kPa \n";
ss+=" s2 compressor output = "+s[2] +" kJ/kgK\n";
ss+=" x2 compressor output = "+x[2] +" kgvapor/kgtotal"+"\n";
ss+=" h3 turbine input = "+h[3] +" kJ/kg\n";
ss+=" T3 turbine input = "+T[3] +" derece C\n";
ss+=" P3 turbine input = "+P[3] +" kPa \n";
ss+=" s3 turbine input = "+s[3] +" kJ/kgK\n";
ss+=" x3 turbine input = "+x[3] +" kgvapor/kgtotal"+"\n";
ss+=" h4 turbine output = "+h[4] +" kJ/kg\n";
ss+=" T4 turbine output = "+T[4] +" derece C\n";
ss+=" s4 turbine output = "+s[4] +" kJ/kgK\n";
ss+=" x4 turbine output = "+x[4] +" kgvapor/kgtotal"+"\n";
return ss;
}
public double[][] TS1()
{
double a[][]=Text.readDoubleT("R134a_Ps.txt");
return a;
}

public double[][] TS()
{
double a[][]=new double[2][421];
double tc=b.r.Tc;
System.out.println("tc="+tc);
double dt=(tc-10.0)/100.0;
int i=0;
double t;
for(t=-90.0;t<=tc;t+=dt)
{double a1[]=b.property("tx",t,0.0);
 a[1][i]=t;
 a[0][i]=a1[5];
 i++;
}
double a3[]=b.property("tx",tc,1.0);
 a[1][i]=t;
 a[0][i]=a3[5];
 i++;
for(t=tc;t>=-90;t-=dt)
{double a2[]=b.property("tx",t,1.0);
```

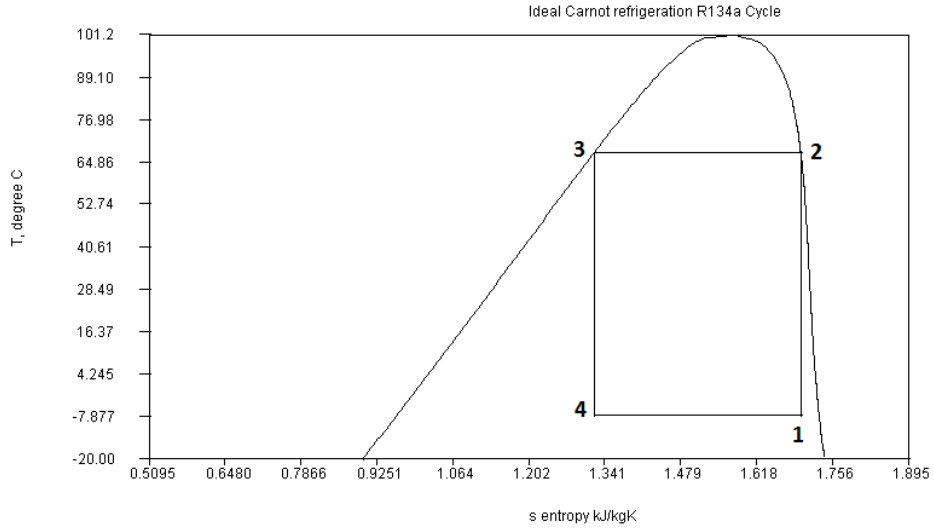
```
 a[1][i]=t;
 a[0][i]=a2[5];
 i++;
 }
return a;
 }

public void plot()
{
 Plot pp=new Plot(TS());
 double t1[]={T[1],T[2],T[3],T[4],T[1]};
 double s1[]={s[1],s[2],s[3],s[4],s[1]};
 pp.setPlabel("Ideal Carnot refrigeration R134a Cycle");
 pp.setYlabel("T, degree C");
 pp.setXlabel("s entropy kJ/kgK");
 pp.addData(s1,t1);
 pp.plot();
 }
 }
```

Program 4.2.1 ideal refrigeration Carnot cycle

```
import javax.swing.*;

public class carnottest
{
public static void main(String arg[])
{
double m=50/745.25928;  // kg/s çevrim R134a debisi
double P2=2000.0;         //  bar turbin giriş basıncı
double P1=220.0 ;         //  bar turbin çıkış basıncı
carnot1 r=new carnot1(m,P1,P2);
System.out.println(r.toString());
r.plot();
 }
 }
```

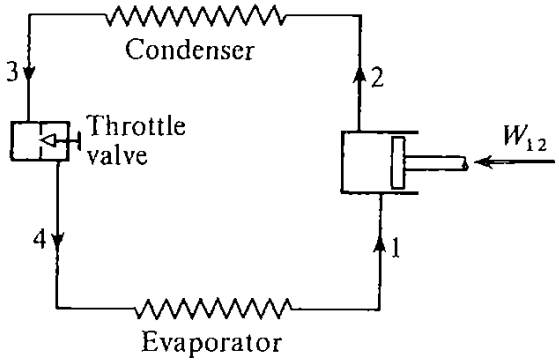
```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" carnottest
R134a carnot refrigeration cycle
 compressor work = 2.9194665890306633 kW
 evaporator work = 1.019143155011926 kW
 net work input= 1.9003234340187372 kW
 condenser heat output = 8.611038292271756 kW
 evaporator heat input = 6.710714858253018 kW
 COP evaporator    = 3.5313540516949975 kW
 COP condenser     = 4.531354051694998 kW
 COP evaporator  carnot = 3.5351212616738557 kW
 COP condenser   carnot = 4.535121261673855 kW
 h1 compressor input = 385.2301064454953 kJ/kg
 T1 compressor input = -7.63976989535311 derece C
 P1 compressor input = 220.0 kPa
 s1 compressor input = 1.6991352552692158 kJ/kgK
 x1 compressor input = 0.9566957691465375 kgvapor/kgtotal
 T2 compressor output = 67.46662970207672 derece C
 P2 compressor output = 2000.0 kPa
 s2 compressor output = 1.6991352552692158 kJ/kgK
 x2 compressor output = 0.0 kgvapor/kgtotal
 h3 turbine input = 300.39617385297873 kJ/kg
 T3 turbine input = 67.46662970207672 derece C
 P3 turbine input = 2000.0 kPa
 s3 turbine input = 1.3222852352261514 kJ/kgK
 x3 turbine input = 0.0 kgvapor/kgtotal
 h4 turbine output = 285.2056559745564 kJ/kg
 T4 turbine output = -7.63976989535311 derece C
 s4 turbine output = 1.3222852352261514 kJ/kgK
 x4 turbine output = 0.46733390915649975 kgvapor/kgtotal
```

```
tc=101.06
```



4.2 STANDARD REFRIGERATION CYCLE

Carnot cycle is not a cycle to realise as a real engine. Using an turbine or expansion engine is expensive, instead of these, using of a simple throttle valve will be simplify the process. Of course COP value will be less due to this change, but the cost is also more affordable.



Now, if this change is adjusted to carnot cycle given in the previously

$W_{isentropic\,compressor} = m(h_2 - h_1)$ $s_2 = s_1$ (4.2.1)

$Q_{condenser} = m(h_2 - h_3) = mT_2(s_2 - s_3) = mT_2(s_1 - s_4)$ (4.2.2)

Throttle valve: $h_3 = h_4$ (4.2.3)

$Q_{evaporator} = m(h_1 - h_4) = mT_1(s_1 - s_4)$ (4.2.4)

$W_{net\,in} = W_{isentropic\,compressor} = Q_{condenser} - Q_{evaporator}$ (4.2.5)

$$Coefficient\ of\ \ Performance\ \ for\ evaporator = \frac{Q_{evaporator}}{W_{compressor}} \quad (4.2.6)$$

$$Coefficient\ of\ \ Performance\ \ for\ condenser = \frac{Q_{condenser}}{W_{compressor}} \quad (4.2.7)$$

Program 4.2.1 refrigeration cycle (turbine in carnot cycle is replaced with throttling valve)

```
 public class ref_cycle1
{
// a R134 ideal carnot refrigeration cycle
// refrigerantEN b=new refrigerantEN("R134a");
// double a[]=b.property("tp",30.0,1.01325);
///---- String s values and meanings --------
```

```
//    tv=temperature-specific volume
//    tp=temperature-pressure
//    th=temperature-enthalpy
//    tu=temperature-internal energy
//    ts=temperature-entropy
//    tx=temperature quality
//    pv=pressure-specific volume
//    pt=pressure-temperature
//    ph=pressure-enthapy
//    pu=pressure-internal energy
//    ps=pressure-entropy
//    px=pressure-quality
//    vp=specific volume-pressure
//    vt=specific volume-temperature
//---- output values (method property) ---------
// r[0] P Pressure kPa
// r[1] t temperature degree C
// r[2] v specific volume m^3/kg
// r[3] h enthapy KJ/kg
// r[4] u internal energy KJ/kg
// r[5] s entropy KJ/kgK
// r[6] x quality kg vapour/kg total
// r[7] ro density kg/m^3

double T[];
double P[];
double h[];
double s[];
double x[];
double COP_condenser,COP_evaporator,COP_evap_carnot,COP_condenser_carnot;
double m;//kg/s
double Wp,W;//work
double Qevaporator,Qcondenser;
refrigerantEN b;
double c1[][];
double c3[][];
public ref_cycle1(double mi,double P1, double P2)
{
m=mi;
b=new refrigerantEN("R134a");
T=new double[9];
P=new double[9];
h=new double[9];
s=new double[9];
x=new double[9];

P[1]=P1;//compressor inlet
P[2]=P2;//compressor outlet
c1=new double[2][20];
c3=new double[2][20];
}
public double[][] line(double P1,double P2,double hi,int n)
{
double TT1[]=new double[n];
double ss1[]=new double[n];
double aa[];
double p;
for(int i=0;i<n;i++)
{p=P1+i*(P2-P1)/(n-1);
 aa=b.property("ph",p,hi);
 TT1[i]=aa[1];
 ss1[i]=aa[5];
}
double a[][]={TT1,ss1};
return a;
}
```

```
public void cycle()
{
// isentropic compressor output
double a1[]=b.property("px",P[2],1.0);
h[2]=a1[3];
s[2]=a1[5];
T[2]=a1[1];
// isentropik compressor input
s[1]=s[2];
a1=b.property("ps",P[1],s[1]);
T[1]=a1[1];
h[1]=a1[3];
x[1]=a1[6];

//expansion valve input-condenser output
P[3]=P[2];
a1=b.property("px",P[3],0.0);
h[3]=a1[3];
s[3]=a1[5];
T[3]=a1[1];
//expansion valve output
P[4]=P[1];
h[4]=h[3];
a1=b.property("ph",P[4],h[4]);
s[4]=a1[5];
T[4]=a1[1];
x[4]=a1[6];


// isentropik compressor
Wp=m*(h[2]-h[1]);
W=Wp;
Qevaporator=m*(h[1]-h[4]);
Qcondenser=m*(h[2]-h[3]);
COP_evaporator=Qevaporator/W;
COP_condenser_carnot=(T[2]+273.15)/(T[2]-T[1]);
COP_condenser=Qcondenser/W;
COP_evap_carnot=(T[1]+273.15)/(T[2]-T[1]);
}

public String toString()
{
cycle();
String ss="R134a ideal refrigeration cycle 1\n";
ss+=" compressor work = "+Wp+" kW\n";
ss+=" net work input= "+W+" kW\n";
ss+=" condenser heat output = "+Qcondenser+" kW\n";
ss+=" evaporator heat input = "+Qevaporator+" kW\n";
ss+=" COP evaporator    = " + COP_evaporator+" kW\n";
ss+=" COP condenser     = " + COP_condenser+" kW\n";
ss+=" COP evaporator  carnot = " + COP_evap_carnot+" kW\n";
ss+=" COP condenser   carnot = " + COP_condenser_carnot+" kW\n";
ss+=" h1 compressor input = "+h[1] +" kJ/kg\n";
ss+=" T1 compressor input = "+T[1] +" derece C\n";
ss+=" P1 compressor input = "+P[1] +" kPa \n";
ss+=" s1 compressor input = "+s[1] +" kJ/kgK\n";
ss+=" x1 compressor input = "+x[1] +" kgvapor/kgtotal"+"\n";
ss+=" T2 compressor output = "+T[3] +" derece C\n";
ss+=" P2 compressor output = "+P[3] +" kPa \n";
ss+=" s2 compressor output = "+s[2] +" kJ/kgK\n";
ss+=" x2 compressor output = "+x[2] +" kgvapor/kgtotal"+"\n";
ss+=" h3 turbine input = "+h[3] +" kJ/kg\n";
ss+=" T3 turbine input = "+T[3] +" derece C\n";
ss+=" P3 turbine input = "+P[3] +" kPa \n";
ss+=" s3 turbine input = "+s[3] +" kJ/kgK\n";
ss+=" x3 turbine input = "+x[3] +" kgvapor/kgtotal"+"\n";
ss+=" h4 turbine output = "+h[4] +" kJ/kg\n";
```

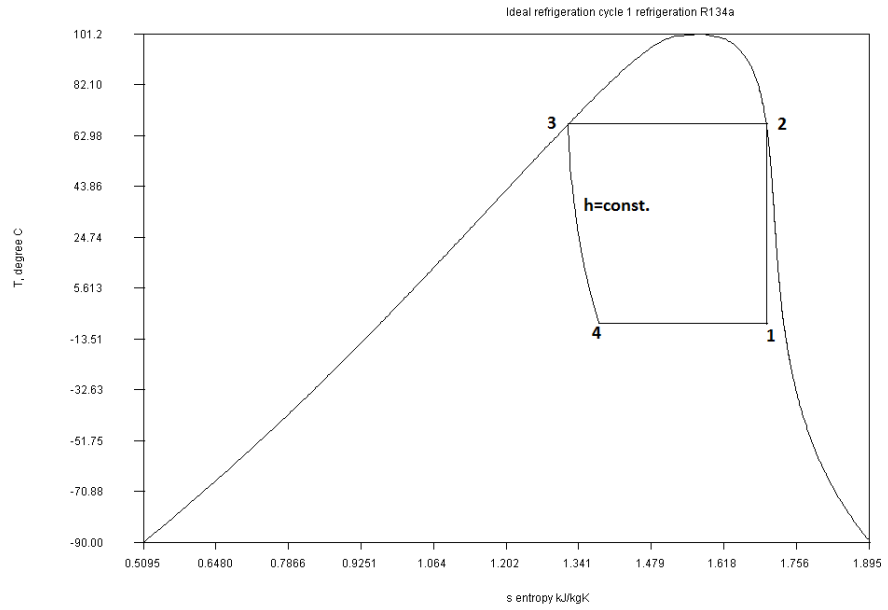
```
ss+=" T4 turbine output = "+T[4] +" derece C\n";
ss+=" s4 turbine output = "+s[4] +" kJ/kgK\n";
ss+=" x4 turbine output = "+x[4] +" kgvapor/kgtotal"+"\n";
return ss;
}

public double[][] TS()
{
double a[][]=new double[2][421];
double tc=b.r.Tc;
System.out.println("tc="+tc);
double dt=(tc-10.0)/100.0;
int i=0;
double t;
for(t=-90.0;t<=tc;t+=dt)
{double a1[]=b.property("tx",t,0.0);
 a[1][i]=t;
 a[0][i]=a1[5];
 i++;
}
double a3[]=b.property("tx",tc,1.0);
 a[1][i]=t;
 a[0][i]=a3[5];
 i++;
for(t=tc;t>=-90;t-=dt)
{double a2[]=b.property("tx",t,1.0);
 a[1][i]=t;
 a[0][i]=a2[5];
 i++;
}
return a;
}

public void plot()
{
 Plot pp=new Plot(TS());
 double a[][]=line(P[3],P[4],h[3],15);
 double tt[]=a[0];
 double ss[]=a[1];
 System.out.println("n="+tt.length);
 double t1[]=new double[19];
 double s1[]=new double[19];
 t1[0]=T[1];t1[1]=T[2];t1[2]=T[3];
 s1[0]=s[1];s1[1]=s[2];s1[2]=s[3];
 for(int i=0;i<tt.length;i++)
 {t1[i+3]=tt[i];s1[i+3]=ss[i];}
 t1[18]=T[1];
 s1[18]=s[1];
 pp.setPlabel("Ideal refrigeration cycle 1 refrigeration R134a ");
 pp.setYlabel("T, degree C");
 pp.setXlabel("s entropy kJ/kgK");
 pp.addData(s1,t1);
 pp.plot();
}
}
```

Program 4.2.2 refrigeration cycle (turbine in carnot cycle is replaced with throttling valve) test program

```
public class ref_cycle1test
{
public static void main(String arg[])
{
double m=50/745.25928;  // kg/s çevrim R134a debisi
double P2=2000.0;         //  bar turbin giriş basıncı
double P1=220.0 ;         //  bar turbin çıkış basıncı
ref_cycle1 r=new ref_cycle1(m,P1,P2);
System.out.println(r.toString());
```

```
r.plot();
 }
 }
```

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" ref_cycle1test
R134a ideal refrigeration cycle 1
 compressor work = 2.9194665890306633 kW
 net work input= 2.9194665890306633 kW
 condenser heat output = 8.611038292271756 kW
 evaporator heat input = 5.691571703241092 kW
 COP evaporator    = 1.9495245208923035 kW
 COP condenser     = 2.9495245208923038 kW
 COP evaporator  carnot = 3.5351212616738557 kW
 COP condenser   carnot = 4.535121261673855 kW
 h1 compressor input = 385.2301064454953 kJ/kg
 T1 compressor input = -7.63976989535311 derece C
 P1 compressor input = 220.0 kPa
 s1 compressor input = 1.6991352552692158 kJ/kgK
 x1 compressor input = 0.9566957691465375 kgvapor/kgtotal
 T2 compressor output = 67.46662970207672 derece C
 P2 compressor output = 2000.0 kPa
 s2 compressor output = 1.6991352552692158 kJ/kgK
 x2 compressor output = 0.0 kgvapor/kgtotal
 h3 turbine input = 300.39617385297873 kJ/kg
 T3 turbine input = 67.46662970207672 derece C
 P3 turbine input = 2000.0 kPa
 s3 turbine input = 1.3222852352261514 kJ/kgK
 x3 turbine input = 0.0 kgvapor/kgtotal
 h4 turbine output = 300.39617385297873 kJ/kg
 T4 turbine output = -7.63976989535311 derece C
 s4 turbine output = 1.3788234773837613 kJ/kgK
 x4 turbine output = 0.540763706713116 kgvapor/kgtotal
```



Compressor inlet point of the previous cycle is not very prefferable as well. It will be better if the compressor inlet is on saturation line or in superheated region.

Program 4.2.3 ideal refrigeration cycle

```
 public class ref_cycle2
{
// a R134 ideal refrigeration  cycle
// refrigerantEN b=new refrigerantEN("R134a");
// double a[]=b.property("tp",30.0,1.01325);
///---- String s values and meanings --------
```

```
//    tv=temperature-specific volume
//    tp=temperature-pressure
//    th=temperature-enthalpy
//    tu=temperature-internal energy
//    ts=temperature-entropy
//    tx=temperature quality
//    pv=pressure-specific volume
//    pt=pressure-temperature
//    ph=pressure-enthapy
//    pu=pressure-internal energy
//    ps=pressure-entropy
//    px=pressure-quality
//    vp=specific volume-pressure
//    vt=specific volume-temperature
//---- output values (method property) ---------
// r[0] P Pressure kPa
// r[1] t temperature degree C
// r[2] v specific volume m^3/kg
// r[3] h enthapy KJ/kg
// r[4] u internal energy KJ/kg
// r[5] s entropy KJ/kgK
// r[6] x quality kg vapour/kg total
// r[7] ro density kg/m^3

double T[];
double P[];
double h[];
double s[];
double x[];
double COP_condenser,COP_evaporator,COP_evap_carnot,COP_condenser_carnot;
double m;//kg/s
double Wp,W;//work
double Qevaporator,Qcondenser;
refrigerantEN b;
double c1[][];
double c3[][];
public ref_cycle2(double mi,double P1, double P2)
{
m=mi;
b=new refrigerantEN("R134a");
T=new double[9];
P=new double[9];
h=new double[9];
s=new double[9];
x=new double[9];

P[1]=P1;//compressor inlet
P[2]=P2;//compressor outlet
c1=new double[2][20];
c3=new double[2][20];
}
public double[][] line(double P1,double P2,double hi,int n)
{
double TT1[]=new double[n];
double ss1[]=new double[n];
double aa[];
double p;
for(int i=0;i<n;i++)
{p=P1+i*(P2-P1)/(n-1);
 aa=b.property("ph",p,hi);
 TT1[i]=aa[1];
 ss1[i]=aa[5];
}
double a[][]={TT1,ss1};
return a;
}
public double[][] line1(double T1,double T2,double Pi,int n)
```

```
{
double TT1[]=new double[n];
double ss1[]=new double[n];
double aa[];
for(int i=0;i<n;i++)
{TT1[i]=T1+i*(T2-T1)/(n-1);
 aa=b.property("tp",TT1[i],Pi);
 ss1[i]=aa[5];
}
double a[][]={TT1,ss1};
return a;
}

public void cycle()
{

// isentropik compressor input
double a1[]=b.property("px",P[1],1.0);
T[1]=a1[1];
h[1]=a1[3];
s[1]=a1[5];
x[1]=a1[6];
// isentropic compressor output
s[2]=s[1];
a1=b.property("ps",P[2],s[2]);
h[2]=a1[3];
T[2]=a1[1];
//saturation point
P[5]=P[2];
a1=b.property("px",P[5],1.0);
h[5]=a1[3];
T[5]=a1[1];
s[5]=a1[5];
//expansion valve input-condenser output
P[3]=P[2];
a1=b.property("px",P[3],0.0);
h[3]=a1[3];
s[3]=a1[5];
T[3]=a1[1];
//expansion valve output
P[4]=P[1];
h[4]=h[3];
a1=b.property("ph",P[4],h[4]);
s[4]=a1[5];
T[4]=a1[1];
x[4]=a1[6];


// isentropik compressor
Wp=m*(h[2]-h[1]);
W=Wp;
Qevaporator=m*(h[1]-h[4]);
Qcondenser=m*(h[2]-h[3]);
COP_evaporator=Qevaporator/W;
COP_condenser_carnot=(T[2]+273.15)/(T[2]-T[1]);
COP_condenser=Qcondenser/W;
COP_evap_carnot=(T[1]+273.15)/(T[2]-T[1]);
}

public String toString()
{
cycle();
String ss="R134a ideal refrigeration cycle 2\n";
ss+=" compressor work = "+Wp+" kW\n";
ss+=" net work input= "+W+" kW\n";
ss+=" condenser heat output = "+Qcondenser+" kW\n";
ss+=" evaporator heat input = "+Qevaporator+" kW\n";
```

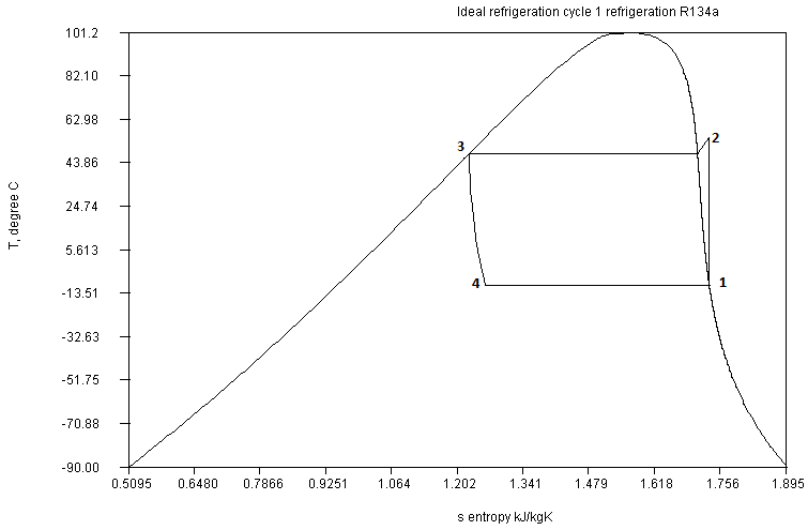
```
ss+=" COP evaporator    = " + COP_evaporator+" kW\n";
ss+=" COP condenser     = " + COP_condenser+" kW\n";
ss+=" COP evaporator  carnot = " + COP_evap_carnot+" kW\n";
ss+=" COP condenser   carnot = " + COP_condenser_carnot+" kW\n";
ss+=" h1 compressor input = "+h[1] +" kJ/kg\n";
ss+=" T1 compressor input = "+T[1] +" derece C\n";
ss+=" P1 compressor input = "+P[1] +" kPa \n";
ss+=" s1 compressor input = "+s[1] +" kJ/kgK\n";
ss+=" x1 compressor input = "+x[1] +" kgvapor/kgtotal"+"\n";
ss+=" T2 compressor output = "+T[2] +" derece C\n";
ss+=" P2 compressor output = "+P[2] +" kPa \n";
ss+=" s2 compressor output = "+s[2] +" kJ/kgK\n";
ss+=" x2 compressor output = "+x[2] +" kgvapor/kgtotal"+"\n";
ss+=" T5 condenser saturation = "+T[5] +" derece C\n";
ss+=" P5 condenser saturation = "+P[5] +" kPa \n";
ss+=" s5 condenser saturation = "+s[5] +" kJ/kgK\n";
ss+=" x5 condenser saturation = "+x[5] +" kgvapor/kgtotal"+"\n";

ss+=" h3 turbine input = "+h[3] +" kJ/kg\n";
ss+=" T3 turbine input = "+T[3] +" derece C\n";
ss+=" P3 turbine input = "+P[3] +" kPa \n";
ss+=" s3 turbine input = "+s[3] +" kJ/kgK\n";
ss+=" x3 turbine input = "+x[3] +" kgvapor/kgtotal"+"\n";
ss+=" h4 turbine output = "+h[4] +" kJ/kg\n";
ss+=" T4 turbine output = "+T[4] +" derece C\n";
ss+=" s4 turbine output = "+s[4] +" kJ/kgK\n";
ss+=" x4 turbine output = "+x[4] +" kgvapor/kgtotal"+"\n";
return ss;
}

public double[][] TS()
{
double a[][]=new double[2][421];
double tc=b.r.Tc;
System.out.println("tc="+tc);
double dt=(tc-10.0)/100.0;
int i=0;
double t;
for(t=-90.0;t<=tc;t+=dt)
{double a1[]=b.property("tx",t,0.0);
 a[1][i]=t;
 a[0][i]=a1[5];
 i++;
}
double a3[]=b.property("tx",tc,1.0);
 a[1][i]=t;
 a[0][i]=a3[5];
 i++;
for(t=tc;t>=-90;t-=dt)
{double a2[]=b.property("tx",t,1.0);
 a[1][i]=t;
 a[0][i]=a2[5];
 i++;
}
return a;
}

public void plot()
{
 Plot pp=new Plot(TS());
 double a[][]=line(P[3],P[4],h[3],15);
 double tt[]=a[0];
 double ss[]=a[1];
 System.out.println("n="+tt.length);
 double t1[]=new double[20];
 double s1[]=new double[20];
 t1[0]=T[1];t1[1]=T[2];t1[2]=T[5];t1[3]=T[3];
```

```
 s1[0]=s[1];s1[1]=s[2];s1[2]=s[5];s1[3]=s[3];
 for(int i=0;i<tt.length;i++)
 {t1[i+4]=tt[i];s1[i+4]=ss[i];}
 t1[19]=T[1];
 s1[19]=s[1];
 pp.setPlabel("Ideal refrigeration cycle 1 refrigeration R134a ");
 pp.setYlabel("T, degree C");
 pp.setXlabel("s entropy kJ/kgK");
 pp.addData(s1,t1);
 pp.plot();
 }
 }
```

Program 4.2.4 ideal refrigeration cycle test program

```
import javax.swing.*;

public class ref_cycle2test
{
public static void main(String arg[])
{
double m=50/745.25928;  // kg/s çevrim R134a debisi
double P2=1250.0;         //  bar turbin giriş basıncı
double P1=200.0 ;         //  bar turbin çıkış basıncı
ref_cycle2 r=new ref_cycle2(m,P1,P2);
System.out.println(r.toString());
r.plot();
}}
```

```
 ---------- Capture Output ----------
> "C:\java\bin\javaw.exe" ref_cycle2test
R134a ideal refrigeration cycle 2
 compressor work = 2.5667447396341565 kW
 net work input= 2.5667447396341565 kW
 condenser heat output = 10.902434167277017 kW
 evaporator heat input = 8.33568942764286 kW
 COP evaporator    = 3.2475724207896746
 COP condenser     = 4.247572420789674
 COP evaporator  carnot = 4.040569717710718
 COP condenser   carnot = 5.040569717710718
 h1 compressor input = 392.85464093227904 kJ/kg
 T1 compressor input = -10.07576059284282 derece C
 P1 compressor input = 200.0 kPa
 s1 compressor input = 1.7341511989113898 kJ/kgK
 x1 compressor input = 1.0 kgvapor/kgtotal
 T2 compressor output = 55.032443889818154 derece C
 P2 compressor output = 1250.0 kPa
 s2 compressor output = 1.7341511989113898 kJ/kgK
 x2 compressor output = 0.0 kgvapor/kgtotal
 T5 condenser saturation = 47.87587075762405 derece C
 P5 condenser saturation = 1250.0 kPa
 s5 condenser saturation = 1.7093434574756143 kJ/kgK
 x5 condenser saturation = 0.0 kgvapor/kgtotal
 h3 turbine input = 268.60964290930445 kJ/kg
 T3 turbine input = 47.87587075762405 derece C
 P3 turbine input = 1250.0 kPa
 s3 turbine input = 1.2284066394805506 kJ/kgK
 x3 turbine input = 0.0 kgvapor/kgtotal
 h4 turbine output = 268.60964290930445 kJ/kg
 T4 turbine output = -10.07576059284282 derece C
 s4 turbine output = 1.2622418532709556 kJ/kgK
 x4 turbine output = 0.3976100345359028 kgvapor/kgtotal
```



Now a more realistic cycle can be investigated. In this cycle, compresor input will be superheated, and condenser output will be in liquid region. Pressure drop in condenser and evaporator will be taken into account. Throttling process in expansion valve will have a small entalphy change due to heat transfer to the surrounding.

Program 4.2.5 standard refrigeration cycle

```
 public class ref_cycle3
{
// a R134 ideal refrigeration  cycle
// refrigerantEN b=new refrigerantEN("R134a");
// double a[]=b.property("tp",30.0,1.01325);
///---- String s values and meanings --------
//    tv=temperature-specific volume
//    tp=temperature-pressure
//    th=temperature-enthalpy
//    tu=temperature-internal energy
//    ts=temperature-entropy
//    tx=temperature quality
//    pv=pressure-specific volume
//    pt=pressure-temperature
//    ph=pressure-enthapy
//    pu=pressure-internal energy
//    ps=pressure-entropy
//    px=pressure-quality
//    vp=specific volume-pressure
//    vt=specific volume-temperature
//---- output values (method property) ---------
// r[0] P Pressure kPa
// r[1] t temperature degree C
// r[2] v specific volume m^3/kg
// r[3] h enthapy KJ/kg
// r[4] u internal energy KJ/kg
// r[5] s entropy KJ/kgK
// r[6] x quality kg vapour/kg total
// r[7] ro density kg/m^3

double T[];
double P[];
double h[];
double s[];
double x[];
boolean supercritical=false;
String refname; //refrigerant name
double Tc,Pc;
double COP_condenser,COP_evaporator,COP_evap_carnot,COP_condenser_carnot;
double m;//kg/s mass flow rate of refrigerant
```

```
double Wp;//compressor work
double Wpi;//isentropic compressor work
double dPevap1;// evaporator pressure drop saturated region
double dPevap2;// evaporator pressure drop superheated region
double dPcond1;// condenser pressure drop superheated region
double dPcond2;// condenser pressure drop saturated region
double dPcond3;// condenser pressure drop liquid region
double dh;     // expansion valve enthalpy chenge (heat loss)
double dT1,dT6;
double eta_isent; // isentropic efficiency of compressor
double Qevaporator,Qcondenser; //condenser and evaporator heat transfer
refrigerantEN b; //refrigerantEN class object for properties
double c1[][];
double c3[][];
public ref_cycle3(String refnamei,double mi,double dT1i,double P1,double eta_isent1i, double
P2,double dT6i,double dPevap1i,double dPevap2i,double dPcond1i,double dPcond2i,double
dPcond3i,double dhi)
{
m=mi;
refname=refnamei;
dPevap1=dPevap1i;
dPevap2=dPevap2i;
dPcond1=dPcond1i;
dPcond2=dPcond2i;
dPcond3=dPcond3i;
dh=dhi;
dT1=dT1i;
dT6=dT6i;
eta_isent=eta_isent1i;
b=new refrigerantEN(refname);
T=new double[9];
P=new double[9];
h=new double[9];
s=new double[9];
x=new double[9];
Tc=b.r.Tc;//critical temperature degree C
Pc=b.r.Pc;//critical pressure kPa
P[1]=P1;//compressor inlet
P[2]=P2;//compressor outlet, evaporator superheated outlet
P[0]=P1-dPevap1; //evaporator saturation exit(x=1) pressure
P[4]=P[0]-dPevap2;//evaporator saturation inlet pressure
P[5]=P2-dPcond1;//condenser saturation inlet (x=1)
P[6]=P[5]-dPcond2;//condenser saturation exit (x=0)
P[3]=P[6]-dPcond3;//condenser  exit liquid state)
c1=new double[2][20];
c3=new double[2][20];
}

public void cycle()
{
// compressor saturation point 0
double a1[]=b.property("px",P[0],1.0);
T[0]=a1[1];
h[0]=a1[3];
s[0]=a1[5];
x[0]=a1[6];
// compressor inlet, evaporator exit point 1
T[1]=T[0]+dT1;
a1=b.property("tp",T[1],P[1]);
h[1]=a1[3];
s[1]=a1[5];
x[1]=a1[6];
// isentropic compressor exit point 7
s[7]=s[1];
P[7]=P[2];
a1=b.property("ps",P[7],s[7]);
h[7]=a1[3];
```

```
T[7]=a1[1];
//compressor exit point 2
h[2]=h[1]+(h[7]-h[1])/eta_isent;
a1=b.property("ph",P[2],h[2]);
T[2]=a1[1];
h[2]=a1[3];
s[2]=a1[5];
x[2]=a1[6];
// check if cycle is supercritical
if(P[3]>Pc)
{System.out.println("condenser is supercritical");
supercritical=true;
}
else supercritical=false;
//expansion valve input
if(!supercritical)
{
//saturation point
a1=b.property("px",P[5],1.0);
h[5]=a1[3];
T[5]=a1[1];
s[5]=a1[5];
a1=b.property("px",P[6],0.0);
h[6]=a1[3];
T[6]=a1[1];
s[6]=a1[5];
T[3]=T[6]-dT6;
a1=b.property("tx",T[3],0.0);
//a1=b.property("tp",T[3],P[3]);
}
else
{
T[3]=Tc-dT6;
a1=b.property("tp",T[3],P[3]);
}
h[3]=a1[3];
T[3]=a1[1];
s[3]=a1[5];
// point 4 evaporator inlet, expansion valve output
h[4]=h[3]+dh;
a1=b.property("ph",P[4],h[4]);
T[4]=a1[1];
s[4]=a1[5];
// isentropik compressor
Wpi=m*(h[7]-h[1]);
//compressor
Wp=m*(h[2]-h[1]);
Qevaporator=m*(h[1]-h[4]);
Qcondenser=m*(h[2]-h[3]);
COP_evaporator=Qevaporator/Wp;
COP_condenser_carnot=(T[2]+273.15)/(T[2]-T[1]);
COP_condenser=Qcondenser/Wp;
COP_evap_carnot=(T[1]+273.15)/(T[2]-T[1]);
}

public String toString()
{
cycle();
String ss="R134a ideal refrigeration cycle 2\n";
ss+=" compressor work = "+Wp+" kW\n";
ss+=" isentropic compressor work = "+Wpi+" kW\n";
ss+=" condenser heat output = "+Qcondenser+" kW\n";
ss+=" evaporator heat input = "+Qevaporator+" kW\n";
ss+=" COP evaporator    = " + COP_evaporator+" kW\n";
ss+=" COP condenser     = " + COP_condenser+" kW\n";
ss+=" COP evaporator  carnot = " + COP_evap_carnot+" kW\n";
ss+=" COP condenser   carnot = " + COP_condenser_carnot+" kW\n";
```

```
ss+=" h1 compressor input = "+h[1] +" kJ/kg\n";
ss+=" T1 compressor input = "+T[1] +" derece C\n";
ss+=" P1 compressor input = "+P[1] +" kPa \n";
ss+=" s1 compressor input = "+s[1] +" kJ/kgK\n";
ss+=" x1 compressor input = "+x[1] +" kgvapor/kgtotal"+"\n";
ss+=" h2 compressor input = "+h[2] +" kJ/kg\n";
ss+=" T2 compressor output = "+T[2] +" derece C\n";
ss+=" P2 compressor output = "+P[2] +" kPa \n";
ss+=" s2 compressor output = "+s[2] +" kJ/kgK\n";
ss+=" x2 compressor output = "+x[2] +" kgvapor/kgtotal"+"\n";
if(!supercritical)
{
ss+=" h5 compressor input = "+h[5] +" kJ/kg\n";
ss+=" T5 condenser saturation = "+T[5] +" derece C\n";
ss+=" P5 condenser saturation = "+P[5] +" kPa \n";
ss+=" s5 condenser saturation = "+s[5] +" kJ/kgK\n";
ss+=" x5 condenser saturation = "+x[5] +" kgvapor/kgtotal"+"\n";
ss+=" h6 compressor input = "+h[6] +" kJ/kg\n";
ss+=" T6 condenser saturation = "+T[6] +" derece C\n";
ss+=" P6 condenser saturation = "+P[6] +" kPa \n";
ss+=" s6 condenser saturation = "+s[6] +" kJ/kgK\n";
ss+=" x6 condenser saturation = "+x[6] +" kgvapor/kgtotal"+"\n";
}
ss+=" h3 turbine input = "+h[3] +" kJ/kg\n";
ss+=" T3 turbine input = "+T[3] +" derece C\n";
ss+=" P3 turbine input = "+P[3] +" kPa \n";
ss+=" s3 turbine input = "+s[3] +" kJ/kgK\n";
ss+=" x3 turbine input = "+x[3] +" kgvapor/kgtotal"+"\n";
ss+=" h4 turbine output = "+h[4] +" kJ/kg\n";
ss+=" T4 turbine output = "+T[4] +" derece C\n";
ss+=" s4 turbine output = "+s[4] +" kJ/kgK\n";
ss+=" x4 turbine output = "+x[4] +" kgvapor/kgtotal"+"\n";
return ss;
}

public double[][] TS(double ti)
{
double a[][]=new double[2][10000];
double tc=b.r.Tc;
double dt=(tc-1.0)/100.0;
int i=0;
double t;
for(t=ti;t<=tc;t+=dt)
{double a1[]=b.property("tx",t,0.0);
 a[1][i]=t;
 a[0][i]=a1[5];
 i++;
}
double a3[]=b.property("tx",tc,1.0);
 a[1][i]=t;
 a[0][i]=a3[5];
 i++;
for(t=tc;t>=ti;t-=dt)
{double a2[]=b.property("tx",t,1.0);
 a[1][i]=t;
 a[0][i]=a2[5];
 i++;
}
int n=i;
double b[][]=new double[2][n];
for(i=0;i<n;i++) {b[0][i]=a[0][i];b[1][i]=a[1][i];}
return b;
}
public double[][] line(double P1,double P2,double hi,double ho,int n)
{
double TT1[]=new double[n];
double ss1[]=new double[n];
```

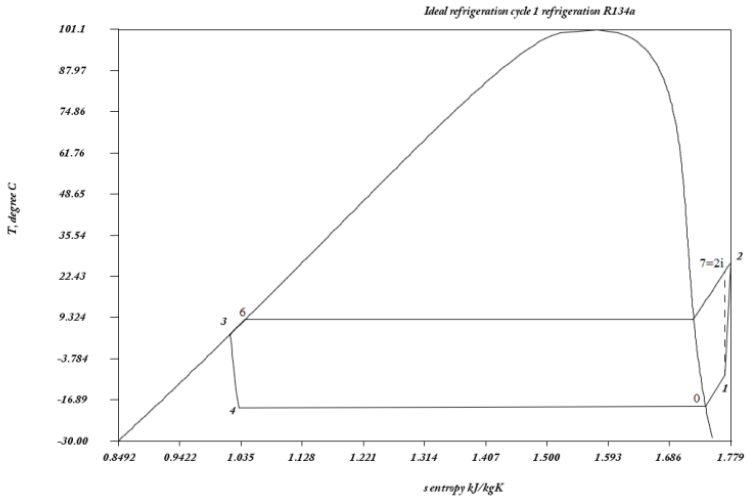
```
double aa[];
double p,h;
for(int i=0;i<n;i++)
{p=P1+i*(P2-P1)/(n-1);
 h=hi+i*(ho-hi)/(n-1);
 aa=b.property("ph",p,h);
 TT1[i]=aa[1];
 ss1[i]=aa[5];
}
double a[][]={TT1,ss1};
return a;
}
public double[][] line1(double T1,double T2,double Pi,double Po,int n)
{
double TT1[]=new double[n];
double ss1[]=new double[n];
double aa[];
double P;
for(int i=0;i<n;i++)
{TT1[i]=T1+i*(T2-T1)/(n-1);
 P=Pi+i*(Po-Pi)/(n-1);
 aa=b.property("tp",TT1[i],P);
 ss1[i]=aa[5];
}
double a[][]={TT1,ss1};
return a;
}
public Plot plot1(Plot pp)
{double a[][]={{T[3],T[4]},{s[3],s[4]}};
 double tt[]=a[0];
 double ss[]=a[1];
 pp.addData(ss,tt);
 double b[][]=new double[2][15];
 double c[][];
 if(!supercritical)
 { c=line1(T[2],T[5],P[2],P[5],13);
   for(int i=0;i<c[0].length;i++)
   {b[0][i]=c[0][i];b[1][i]=c[1][i];}
    b[0][13]=T[6];b[1][13]=s[6];
    b[0][14]=T[3];b[1][14]=s[3];
 }
 else
 {c=line1(T[2],T[3],P[2],P[3],15);b=c;}
 tt=b[0];
 ss=b[1];
 pp.addData(ss,tt);
 double t1[]=new double[4];
 double s1[]=new double[4];
 t1[0]=T[4];t1[1]=T[0];t1[2]=T[1];t1[3]=T[2];
 s1[0]=s[4];s1[1]=s[0];s1[2]=s[1];s1[3]=s[2];
 pp.addData(s1,t1);
 double t2[]=new double[2];
 double s2[]=new double[2];
 t2[0]=T[1];t2[1]=T[7];
 s2[0]=s[1];s2[1]=s[7];
 pp.addData(s2,t2,1);

 return pp;
}
public void plot(double ti)
{   Plot pp=new Plot(TS(ti));
   pp= plot1(pp);
   pp.setPlabel("Ideal refrigeration cycle 1 refrigeration ");
   pp.setYlabel("T, degree C");
   pp.setXlabel("s entropy kJ/kgK");
            pp.plot();
}}
```

Program 4.2.5 standard refrigeration cycle test program

```
import javax.swing.*;

public class ref_cycle3test
{
public static void main(String arg[])
{
double m=0.05;  // kg/s çevrim R134a debisi
double P2=400.0;         //  kPa turbin giriş basıncı
double P1=140.0 ;         //  kPa turbin çıkış basıncı
double dT1i=10.0; //degree C
double eta_isent1i=0.9;
double dT6i=5.0; //degree C
double dPevap1i=2.0; //kPa
double dPevap2i=2.0; //kPa
double dPcond1i=2.0; //kPa
double dPcond2i=2.0; //kPa
double dPcond3i=2.0; //kPa
double dhi=2.0; //kJ/kgK
ref_cycle3 r=new
ref_cycle3("R22",m,dT1i,P1,eta_isent1i,P2,dT6i,dPevap1i,dPevap2i,dPcond1i,dPcond2i,dPco
nd3i,dhi);
System.out.println(r.toString());
r.plot(-50.0);
}
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" ref_cycle3test
R134a ideal refrigeration cycle 2
 compressor work = 2.9946164677482345 kW
 isentropic compressor work = 2.695154820973411 kW
 condenser heat output = 12.412532218649785 kW
 evaporator heat input = 9.283734261742502 kW
 COP evaporator    = 3.100141324182089 kW
 COP condenser     = 4.144948894902471 kW
 COP evaporator  carnot = 3.990344898354031 kW
 COP condenser   carnot = 4.990344898354031 kW
 h1 compressor input = 400.58952866027346 kJ/kg
 T1 compressor input = -0.7139216789336409 derece C
 P1 compressor input = 200.0 kPa
 s1 compressor input = 1.7631762691950033 kJ/kgK
 x1 compressor input = 2.0 kgvapor/kgtotal
 h2 compressor input = 445.2248429128773 kJ/kg
 T2 compressor output = 67.55989556263937 derece C
 P2 compressor output = 1250.0 kPa
 s2 compressor output = 1.7763586400487221 kJ/kgK
 x2 compressor output = 1.1435914281628234 kgvapor/kgtotal
 h5 compressor input = 422.98465471180526 kJ/kg
 T5 condenser saturation = 47.71824296880384 derece C
 P5 condenser saturation = 1245.0 kPa
 s5 condenser saturation = 1.709402795937088 kJ/kgK
 x5 condenser saturation = 0.0 kgvapor/kgtotal
 h6 compressor input = 268.13803320986443 kJ/kg
 T6 condenser saturation = 47.56008650804041 derece C
 P6 condenser saturation = 1240.0 kPa
 s6 condenser saturation = 1.2268885030779424 kJ/kgK
```

```
x6 condenser saturation = 0.0 kgvapor/kgtotal
h3 turbine input = 260.2137464279225 kJ/kg
T3 turbine input = 42.56008650804041 derece C
P3 turbine input = 1235.0 kPa
s3 turbine input = 1.201962034810739 kJ/kgK
x3 turbine input = 0.0 kgvapor/kgtotal
h4 turbine output = 262.2137464279225 kJ/kg
T4 turbine output = -11.364829272039977 derece C
s4 turbine output = 1.2393954627972106 kJ/kgK
x4 turbine output = 0.0 kgvapor/kgtotal
```



Refrigeration cycles can be supercritical as well, this is specially the case for refrigerants such as R744(carbondioxide). Considering that as a natural refrigerant CO2 usage in refrigeration will be increased in the future.

```
public class refISO_cycle3
{
// a R134 ideal refrigeration  cycle
// refISO b=new refISO("R744");
// double a[]=b.property("tp",30.0,1.01325);
///---- String s values and meanings --------
//    tv=temperature-specific volume
//    tp=temperature-pressure
//    th=temperature-enthalpy
//    tu=temperature-internal energy
//    ts=temperature-entropy
//    tx=temperature quality
//    pv=pressure-specific volume
//    pt=pressure-temperature
//    ph=pressure-enthapy
//    pu=pressure-internal energy
//    ps=pressure-entropy
//    px=pressure-quality
//    vp=specific volume-pressure
//    vt=specific volume-temperature
//---- output values (method property) ---------
// r[0] P Pressure kPa
// r[1] t temperature degree C
// r[2] v specific volume m^3/kg
// r[3] h enthapy KJ/kg
// r[4] u internal energy KJ/kg
// r[5] s entropy KJ/kgK
// r[6] x quality kg vapour/kg total
// r[7] ro density kg/m^3

public double T[];
public double P[];
public double h[];
public double s[];
```

```
public double x[];
public boolean supercritical=false;
public String refname; //refrigerant name
public double Tc,Pc;
public double COP_condenser,COP_evaporator,COP_evap_carnot,COP_condenser_carnot;
public double m;//kg/s mass flow rate of refrigerant
public double N;//kmol/s molar flow rate;
public double Wp;//compressor work
public double Wpi;//isentropic compressor work
public double dPevap1;// evaporator pressure drop saturated region
public double dPevap2;// evaporator pressure drop superheated region
public double dPcond1;// condenser pressure drop superheated region
public double dPcond2;// condenser pressure drop saturated region
public double dPcond3;// condenser pressure drop liquid region
public double dh;     // expansion valve enthalpy chenge (heat loss)
public double dT1,dT6;
public double eta_isent; // isentropic efficiency of compressor
public double Qevaporator,Qcondenser; //condenser and evaporator heat transfer
refISO b; //refrigerantEN class object for properties
double c1[][];
double c3[][];
public refISO_cycle3(String refnamei,double mi,double dT1i,double P1,double eta_isent1i, double
P2,double dT6i,double dPevap1i,double dPevap2i,double dPcond1i,double dPcond2i,double
dPcond3i,double dhi)
{
m=mi;

refname=refnamei;
dPevap1=dPevap1i;
dPevap2=dPevap2i;
dPcond1=dPcond1i;
dPcond2=dPcond2i;
dPcond3=dPcond3i;
dh=dhi;
dT1=dT1i;
dT6=dT6i;
eta_isent=eta_isent1i;
b=new refISO(refname);
N=m/b.M;
T=new double[9];
P=new double[9];
h=new double[9];
s=new double[9];
x=new double[9];
Tc=b.Tc;//critical temperature degree C
Pc=b.Pc;//critical pressure kPa
P[1]=P1;//compressor inlet
P[2]=P2;//compressor outlet, evaporator superheated outlet
P[0]=P1+dPevap1; //evaporator saturation exit(x=1) pressure
P[4]=P[0]+dPevap2;//evaporator saturation inlet pressure
P[5]=P2-dPcond1;//condenser saturation inlet (x=1)
P[6]=P[5]-dPcond2;//condenser saturation exit (x=0)
P[3]=P[6]-dPcond3;//condenser  exit liquid state)
c1=new double[2][20];
c3=new double[2][20];
}

public void print(int i)
{System.out.println("T["+i+"] = "+T[i]+" P["+i+"] = "+P[i]+" h["+i+"] = "+h[i]+" s["+i+"] = "+s[i]+"
x["+i+"] = "+x[i]);}

public void cycle()
{
// compressor saturation point 0
double a1[]=b.property("px",P[0],1.0);
T[0]=a1[1];
h[0]=a1[3];
```

```
s[0]=a1[5];
x[0]=a1[6];
//print(0);
// compressor inlet, evaporator exit point 1
T[1]=T[0]+dT1;
a1=b.property("tp",T[1],P[1]);
h[1]=a1[3];
s[1]=a1[5];
x[1]=a1[6];
//print(1);
// isentropic compressor exit point 7
s[7]=s[1];
P[7]=P[2];
a1=b.property("ps",P[7],s[7]);
h[7]=a1[3];
T[7]=a1[1];
x[7]=a1[6];
//print(7);
//compressor exit point 2
h[2]=h[1]+(h[7]-h[1])/eta_isent;
a1=b.property("ph",P[2],h[2]);
T[2]=a1[1];
h[2]=a1[3];
s[2]=a1[5];
x[2]=a1[6];
//print(2);
// check if cycle is supercritical
if(P[3]>Pc)
{System.out.println("heat exchanger is supercritical");
supercritical=true;
}
else supercritical=false;
//expansion valve input

if(!supercritical)
{
//saturation point
a1=b.property("px",P[5],1.0);
h[5]=a1[3];
T[5]=a1[1];
s[5]=a1[5];
x[5]=a1[6];
//print(5);
a1=b.property("px",P[6],0.0);
h[6]=a1[3];
T[6]=a1[1];
s[6]=a1[5];
x[6]=a1[6];
//print(6);
T[3]=T[6]-dT6;
a1=b.property("tp",T[3],P[3]);
//a1=b.property("tx",T[3],0.0);
h[3]=a1[3];
s[3]=a1[5];
x[3]=a1[6];
//print(3);
}
else
{
T[3]=Tc-dT6;
a1=b.property("tp",T[3],P[3]);
h[3]=a1[3];
T[3]=a1[1];
s[3]=a1[5];
x[3]=a1[6];
//print(3);
}
```

```
// point 4 evaporator inlet, expansion valve output
h[4]=h[3]+dh;
double x4=b.x_Ph(P[4],h[4]);
a1=b.property("px",P[4],x4);
T[4]=a1[1];
s[4]=a1[5];
x[4]=a1[4];
x[4]=a1[6];
//print(4);
// isentropik compressor
Wpi=N*(h[7]-h[1]);
//compressor
Wp=N*(h[2]-h[1]);
Qevaporator=N*(h[1]-h[4]);
Qcondenser=N*(h[2]-h[3]);
COP_evaporator=Qevaporator/Wp;
COP_condenser_carnot=(T[2]+273.15)/(T[2]-T[1]);
COP_condenser=Qcondenser/Wp;
COP_evap_carnot=(T[1]+273.15)/(T[2]-T[1]);
}

public String toString()
{
cycle();
String ss=refname+" ideal refrigeration cycle \n";
ss+=" compressor work = "+Wp+" kW\n";
ss+=" isentropic compressor work = "+Wpi+" kW\n";
ss+=" condenser heat output = "+Qcondenser+" kW\n";
ss+=" evaporator heat input = "+Qevaporator+" kW\n";
ss+=" COP evaporator    = " + COP_evaporator+" kW\n";
if(!supercritical)
{ss+=" COP condenser     = " + COP_condenser+" kW\n";}
else
{ss+=" COP heat exchanger     = " + COP_condenser+" kW\n";}
ss+=" COP evaporator  carnot = " + COP_evap_carnot+" kW\n";
if(!supercritical)
ss+=" COP condenser   carnot = " + COP_condenser_carnot+" kW\n";
else
ss+=" COP heat exchanger   carnot = " + COP_condenser_carnot+" kW\n";
ss+=" h1 compressor input = "+h[1]/b.M +" kJ/kg\n";
ss+=" T1 compressor input = "+(T[1]-273.15) +" degree C\n";
ss+=" P1 compressor input = "+P[1] +" kPa \n";
ss+=" s1 compressor input = "+s[1]/b.M +" kJ/kgK\n";
ss+=" x1 compressor input = "+x[1] +" kgvapor/kgtotal"+"\n";
ss+=" h2 compressor input = "+h[2]/b.M +" kJ/kg\n";
ss+=" T2 compressor output = "+(T[2]-273.15) +" degree C\n";
ss+=" P2 compressor output = "+P[2] +" kPa \n";
ss+=" s2 compressor output = "+s[2]/b.M +" kJ/kgK\n";
ss+=" x2 compressor output = "+x[2] +" kgvapor/kgtotal"+"\n";
if(!supercritical)
{
ss+=" h5 condenser saturation = "+h[5]/b.M +" kJ/kg\n";
ss+=" T5 condenser saturation = "+(T[5]-273.15) +" degree C\n";
ss+=" P5 condenser saturation = "+P[5] +" kPa \n";
ss+=" s5 condenser saturation = "+s[5] +" kJ/kgK\n";
ss+=" x5 condenser saturation = "+x[5]/b.M +" kgvapor/kgtotal"+"\n";
ss+=" h6 condenser saturation = "+h[6]/b.M +" kJ/kg\n";
ss+=" T6 condenser saturation = "+(T[6]-273.15) +" degree C\n";
ss+=" P6 condenser saturation = "+P[6] +" kPa \n";
ss+=" s6 condenser saturation = "+s[6]/b.M +" kJ/kgK\n";
ss+=" x6 condenser saturation = "+x[6] +" kgvapor/kgtotal"+"\n";
}
ss+=" h3  expansion valve input = "+h[3]/b.M +" kJ/kg\n";
ss+=" T3  expansion valve input = "+(T[3]-273.15) +" degree C\n";
ss+=" P3  expansion valve input = "+P[3] +" kPa \n";
ss+=" s3  expansion valve input = "+s[3]/b.M +" kJ/kgK\n";
ss+=" x3  epansion valve input = "+x[3] +" kgvapor/kgtotal"+"\n";
```

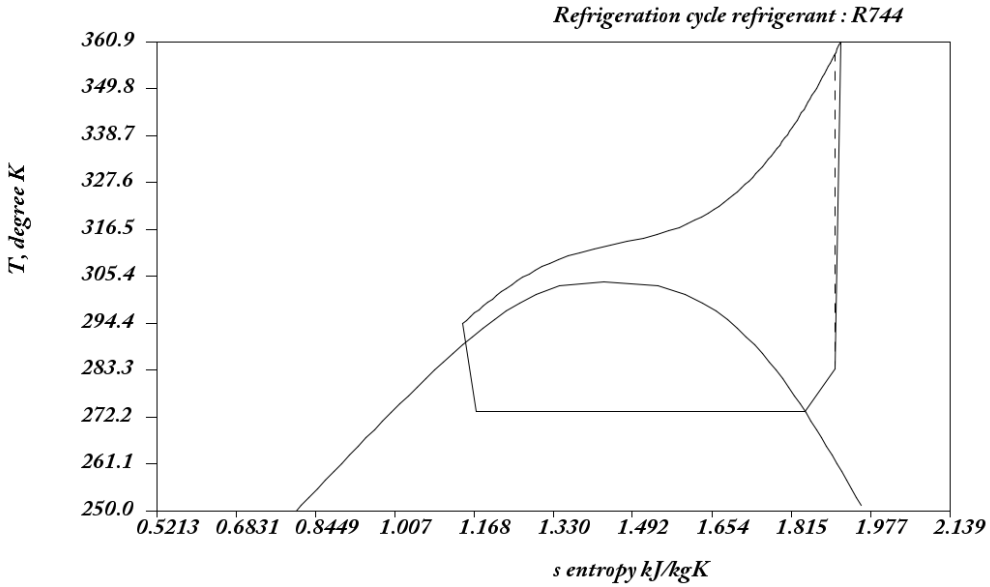
```
ss+=" h4 expansion valve output = "+h[4]/b.M +" kJ/kg\n";
ss+=" T4 expansion valve output = "+(T[4]-273.15) +" degree C\n";
ss+=" s4 expansion valve output = "+s[4]/b.M +" kJ/kgK\n";
ss+=" x4 expansion valve output = "+x[4] +" kgvapor/kgtotal"+"\n";
return ss;
}


public double[][] line(double P1,double P2,double hi,double ho,int n)
{
double TT1[]=new double[n];
double ss1[]=new double[n];
double aa[];
double p,h;
for(int i=0;i<n;i++)
{p=P1+i*(P2-P1)/(n-1);
 h=hi+i*(ho-hi)/(n-1);
 aa=b.property("ph",p,h);
 TT1[i]=aa[1];
 ss1[i]=aa[5]/b.M;
}
double a[][]={TT1,ss1};
return a;
}
public double[][] line1(double T1,double T2,double Pi,double Po,int n)
{
double TT1[]=new double[n];
double ss1[]=new double[n];
double aa[];
double P;
for(int i=0;i<n;i++)
{TT1[i]=T1+i*(T2-T1)/(n-1);
 P=Pi+i*(Po-Pi)/(n-1);
 aa=b.property("tp",TT1[i],P);
 ss1[i]=aa[5]/b.M;
}
double a[][]={TT1,ss1};
return a;
}
public Plot plot1(Plot pp)
{double a[][]={{T[3],T[4]},{s[3]/b.M,s[4]/b.M}};
 double tt[]=a[0];
 double ss[]=a[1];
 pp.addData(ss,tt);
 double bb[][]=new double[2][15];
 double cc[][];
 if(supercritical)
 {cc=line1(T[2],T[3],P[2],P[3],80);bb=cc;}
 else //subcritical cycle
 { cc=line1(T[2],T[5],P[2],P[5],13);
   for(int i=0;i<cc[0].length;i++)
   {bb[0][i]=cc[0][i];bb[1][i]=cc[1][i];}
    bb[0][13]=T[6];bb[1][13]=s[6]/b.M;
    bb[0][14]=T[3];bb[1][14]=s[3]/b.M;
 }
 tt=bb[0];
 ss=bb[1];
 pp.addData(ss,tt);
 double t1[]=new double[4];
 double s1[]=new double[4];
 t1[0]=T[4];t1[1]=T[0];t1[2]=T[1];t1[3]=T[2];
 s1[0]=s[4]/b.M;s1[1]=s[0]/b.M;s1[2]=s[1]/b.M;s1[3]=s[2]/b.M;
 pp.addData(s1,t1);
 double t2[]=new double[2];
 double s2[]=new double[2];
 t2[0]=T[1];t2[1]=T[7];
 s2[0]=s[1]/b.M;s2[1]=s[7]/b.M;
```

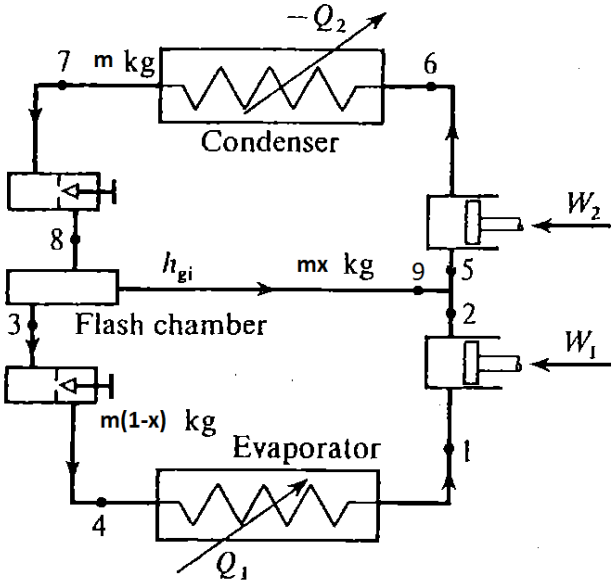
```
 pp.addData(s2,t2,1);
 return pp;
}
public void plot(double ti)
{  double a[][]=b.TS_saturation();
   for(int i=0;i<a[0].length;i++) {a[1][i]=a[1][i]/b.M;}
   Plot pp=new Plot(a[1],a[0]);
   pp= plot1(pp);
   pp.setPlabel("Refrigeration cycle refrigerant : "+b.name);
   pp.setYlabel("T, degree K");
   pp.setXlabel("s entropy kJ/kgK");
            pp.plot();
}
}
```

```
import javax.swing.*;

public class refISO_cycle3test
{
public static void main(String arg[])
{
double m=0.05;  // kg/s çevrim R134a debisi
double P2=9000.0;         //  kPa turbin giriş basıncı
double P1=3500.0 ;         //  kPa turbin çıkış basıncı
double dT1i=10.0; //degree C
double eta_isent1i=0.9;
double dT6i=10.0; //degree C
double dPevap1i=10.0; //kPa
double dPevap2i=10.0; //kPa
double dPcond1i=10.0; //kPa
double dPcond2i=10.0; //kPa
double dPcond3i=10.0; //kPa
double dhi=10.0; //kJ/kgK
refISO_cycle3 r=new
refISO_cycle3("R744",m,dT1i,P1,eta_isent1i,P2,dT6i,dPevap1i,dPevap2i,dPcond1i,dPcond2i,
dPcond3i,dhi);
System.out.println(r.toString());
r.plot(50.0);
}
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" refISO_cycle3test
heat exchanger is supercritical
R744 ideal refrigeration cycle
 compressor work = 2.3239323274386443 kW
 isentropic compressor work = 2.0915390946947783 kW
 condenser heat output = 12.328867048458902 kW
 evaporator heat input = 9.99357361508476 kW
 COP evaporator    = 4.300285983843308 kW
 COP heat exchanger     = 5.3051747259987305 kW
 COP evaporator  carnot = 7.185431145103652 kW
 COP heat exchanger   carnot = 8.185431145103653 kW
 h1 compressor input = 447.49431540176056 kJ/kg
 T1 compressor input = 10.26875975037035 degree C
 P1 compressor input = 3500.0 kPa
 s1 compressor input = 1.904419297963017 kJ/kgK
 h2 compressor input = 493.9729619505335 kJ/kg
 T2 compressor output = 87.7267088296864 degree C
 P2 compressor output = 9000.0 kPa
 s2 compressor output = 1.9173520752755946 kJ/kgK
h3  expansion valve input = 247.39562098135548 kJ/kg
 T3  expansion valve input = 20.980000000000018 degree C
 P3  expansion valve input = 8970.0 kPa
 s3  expansion valve input = 1.1451859384426248 kJ/kgK
 h4 expansion valve output = 247.6228431000654 kJ/kg
```

T4 expansion valve output = 0.3764727762339817 degree C
s4 expansion valve output = 1.1739723715818795 kJ/kgK





In order to improve COP of a refrigeration cycle, multistage refrigeration processes can be utilized. In the figure above, a two stage refrigeration cycle using flash chamber is shown. Flash chamber is a seperator seperating saturated liquid phase and saturated vapour phase of the the mixture. Energy balance in the flash chamber :

$mh_8 = mxh_9 + m(1-x)h_3$ (4.2.8)

Mixing process:

$mxh_9 + m(1-x)h_2 = mh_5$ (4.2.9)

$x = \frac{h_8 - h_3}{h_9 - h_3}$ (4.2.10)

Work output:

$W_1 = m(1-x)(h_2 - h_1)$ (4.2.11)

$W_2 = m(h_6 - h_5)$ (4.2.12)

$W = W_1 + W_2$ (4.2.13)

Evaporator heat transfer:

$Q_1 = m(1-x)(h_1 - h_4)$ (4.2.14)

Condenser heat transfer:

$Q_2 = m(h_6 - h_7)$ (4.2.15)

Program 4.2.6 standard refrigeration cycle program

```
public class ref_cycle4
{
// a R134 ideal refrigeration  cycle
// refrigerantEN b=new refrigerantEN("R134a");
// double a[]=b.property("tp",30.0,1.01325);
///---- String s values and meanings --------
//    tv=temperature-specific volume
//    tp=temperature-pressure
//    th=temperature-enthalpy
//    tu=temperature-internal energy
//    ts=temperature-entropy
//    tx=temperature quality
//    pv=pressure-specific volume
//    pt=pressure-temperature
//    ph=pressure-enthapy
//    pu=pressure-internal energy
//    ps=pressure-entropy
//    px=pressure-quality
//    vp=specific volume-pressure
//    vt=specific volume-temperature
//---- output values (method property) ---------
// r[0] P Pressure kPa
// r[1] t temperature degree C
// r[2] v specific volume m^3/kg
// r[3] h enthapy KJ/kg
// r[4] u internal energy KJ/kg
// r[5] s entropy KJ/kgK
// r[6] x quality kg vapour/kg total
// r[7] ro density kg/m^3

double T[];
double P[];
double h[];
double s[];
double x[];
boolean supercritical=false;
String refname; //refrigerant name
double Tc,Pc;
double COP_condenser,COP_evaporator,COP_evap_carnot,COP_condenser_carnot;
double m;//kg/s mass flow rate of refrigerant
double Wp1,Wp2;//compressor work
double Wp1i,Wp2i;//isentropic compressor work
double Wp;//net compressor work
double dPevap1;// evaporator pressure drop saturated region
double dPevap2;// evaporator pressure drop superheated region
double dPcomp1;// evaporator pressure drop saturated region
double dPcond1;// condenser pressure drop superheated region
double dPcond2;// condenser pressure drop saturated region
double dPcond3;// condenser pressure drop liquid region
double etaisent2,etaisent6; //isentropic efficiencies of turbines
double dh1,dh2;     // expansion valve enthalpy chenge (heat loss)
double dT1,dT7;
double eta_isent; // isentropic efficiency of compressor
double Qevaporator,Qcondenser; //condenser and evaporator heat transfer
refrigerantEN b; //refrigerantEN class object for properties
double c1[][];
double c3[][];
public ref_cycle4(String refnamei,double mi,double dT1i,double P1, double P2,double etaisent2i,double
```

```
P6,double etaisent6i,double dT7i,
double dPevap1i,double dPevap2i,double dPcomp1i,double dPcond1i,double dPcond2i,double
dPcond3i,double dh1i,double dh2i)
{
m=mi;
refname=refnamei;    // refrigerant name
dPevap1=dPevap1i;    // evaporator pressure drop saturated region
dPevap2=dPevap2i;    // evaporator pressure drop superheated region and compressor 1 inlet valve
dPcomp1=dPcomp1;     // compressor 1 exit valve
dPcond1=dPcond1i;    // condenser pressure drop superheated region compressor2 exit valve
dPcond2=dPcond2i;    // condenser pressure drop saturated region
dPcond3=dPcond3i;    // condenser pressure drop liquid region
dh1=dh1i; dh2=dh2i;  // enthalpy drop(heat gain) in expansion valves
dT1=dT1i;
dT7=dT7i;
etaisent2=etaisent2i;
etaisent6=etaisent6i;
b=new refrigerantEN(refname);
T=new double[14];
P=new double[14];
h=new double[14];
s=new double[14];
x=new double[14];
Tc=b.r.Tc;//critical temperature degree C
Pc=b.r.Pc;//critical pressure kPa
P[1]=P1;//compressor inlet
P[2]=P2;//compressor outlet, evaporator superheated outlet
P[0]=P1-dPevap1; //evaporator saturation exit(x=1) pressure
P[4]=P[0]-dPevap2;//evaporator saturation inlet pressure
P[5]=P[2]-dPcomp1;// mixing point pressure
P[9]=P[5];//flash tank saturated vapour exit
P[8]=P[5];//flash tank inlet
P[3]=P[5];//flash tank saturated liquid exit
P[6]=P6;//condenser inlet
P[12]=P[6];// 12=6i isentropic compressor exit point
P[10]=P[6]-dPcond1; // condenser  exit liquid state)
P[13]=P[10]-dPcond2;// condenser saturated liquid point
P[7]=P[13]-dPcond3; // condenser exit (liquid)
c1=new double[2][20];
c3=new double[2][20];
}

public void cycle()
{
// compressor saturation point 0
double a1[]=b.property("px",P[0],1.0);
T[0]=a1[1];
h[0]=a1[3];
s[0]=a1[5];
x[0]=a1[6];
// compressor inlet, evaporator exit point 1
T[1]=T[0]+dT1;
a1=b.property("tp",T[1],P[1]);
h[1]=a1[3];
s[1]=a1[5];
x[1]=a1[6];
// isentropic compressor exit point 11 (2i)
s[11]=s[1];
P[11]=P[2];
a1=b.property("ps",P[11],s[11]);
h[11]=a1[3];
T[11]=a1[1];
h[2]=h[1]+(h[11]-h[1])/etaisent2;
// compressor 1 exit point 2
a1=b.property("ph",P[2],h[2]);
s[2]=a1[5];
T[2]=a1[1];
```

```
// point 10 condenser saturated vapour point
x[10]=1.0;
a1=b.property("px",P[10],x[10]);
T[10]=a1[1];
h[10]=a1[3];
s[10]=a1[5];
// point 13 condenser saturated liquid point
x[13]=0.0;
a1=b.property("px",P[13],x[13]);
T[13]=a1[1];
h[13]=a1[3];
s[13]=a1[5];
T[7]=T[13]-dT7;
// point 7 condenser exit
if(T[7]==T[13] && P[7]==P[13])
{h[7]=h[13];T[7]=T[13];s[7]=s[13];x[7]=x[13];
}
else
{
 a1=b.property("tp",T[7],P[7]);
 h[7]=a1[3];
 s[7]=a1[5];
 x[7]=a1[6];
}
//point 3 flash chamber saturated liquid
x[3]=0.0;
a1=b.property("px",P[3],x[3]);
T[3]=a1[1];
h[3]=a1[3];
s[3]=a1[5];
x[3]=a1[6];
//point 4 flash chamber saturated liquid
h[4]=h[3]-dh2;
a1=b.property("ph",P[4],h[4]);
T[4]=a1[1];
h[4]=a1[3];
s[4]=a1[5];
x[4]=a1[6];
//point 9 flash chamber saturated vapour
x[9]=1.0;
a1=b.property("px",P[9],x[9]);
T[9]=a1[1];
h[9]=a1[3];
s[9]=a1[5];
// point 8 flash chamber input
h[8]=h[7]+dh2;
a1=b.property("ph",P[8],h[8]);
T[8]=a1[1];
s[8]=a1[5];
x[8]=a1[6]; //indicates seperation parameter as well
// point 5 compressor 2 inlet point
h[5]=x[8]*h[9]+(1.0-x[8])*h[2];
a1=b.property("ph",P[5],h[5]);
T[5]=a1[1];
s[5]=a1[5];
//point 12 isentropic compressor2 output
s[12]=s[5];
P[12]=P[6];
a1=b.property("ps",P[12],s[12]);
h[12]=a1[3];
T[12]=a1[1];
h[6]=h[5]+(h[12]-h[5])/etaisent6;
a1=b.property("ph",P[6],h[6]);
T[6]=a1[1];
s[6]=a1[5];
x[6]=a1[6];
 // isentropik compressor 1
```

```
Wp1i=m*(1.0-x[8])*(h[11]-h[1]);
//compressor 1
Wp1=m*(1.0-x[8])*(h[2]-h[1]);
Wp=Wp1+Wp2;
// isentropik compressor 2
Wp2i=m*(h[12]-h[5]);
//compressor 2
Wp2=m*(h[6]-h[5]);
Qevaporator=m*(1.0-x[8])*(h[1]-h[4]);
Qcondenser=m*(h[6]-h[7]);
COP_evaporator=Qevaporator/Wp;
COP_condenser_carnot=(T[6]+273.15)/(T[6]-T[0]);
COP_condenser=Qcondenser/Wp;
COP_evap_carnot=(T[0]+273.15)/(T[6]-T[0]);
}

public String toString()
{
cycle();
String ss="two stage flash point refrigeration cycle\n";
ss+="Flash ratio : "+x[8]+"\n";
ss+=" compressor1 work = "+Wp1+" kW\n";
ss+=" compressor2 work = "+Wp2+" kW\n";
ss+=" total compressor work = "+Wp+" kW\n";
ss+=" isentropic compressor 1 work = "+Wp1i+" kW\n";
ss+=" isentropic compressor 2 work = "+Wp2i+" kW\n";
ss+=" total isentropic compressor work = "+(Wp1i+Wp2i)+" kW\n";
ss+=" condenser heat output = "+Qcondenser+" kW\n";
ss+=" evaporator heat input = "+Qevaporator+" kW\n";
ss+=" COP evaporator    = " + COP_evaporator+" kW\n";
ss+=" COP condenser     = " + COP_condenser+" kW\n";
ss+=" COP evaporator  carnot = " + COP_evap_carnot+" kW\n";
ss+=" COP condenser   carnot = " + COP_condenser_carnot+" kW\n";

ss+=" h1 compressor 1 inlet = "+h[1] +" kJ/kg\n";
ss+=" T1 compressor 1 inlet = "+T[1] +" derece C\n";
ss+=" P1 compressor 1 inlet = "+P[1] +" kPa \n";
ss+=" s1 compressor 1 inlet = "+s[1] +" kJ/kgK\n";
ss+=" x1 compressor 1 inlet = "+x[1] +" kgvapor/kgtotal"+"\n";

ss+=" h2 compressor 1 exit = "+h[2] +" kJ/kg\n";
ss+=" T2 compressor 1 exit = "+T[2] +" derece C\n";
ss+=" P2 compressor 1 exit = "+P[2] +" kPa \n";
ss+=" s2 compressor 1 exit = "+s[2] +" kJ/kgK\n";
ss+=" x2 compressor 1 exit = "+x[2] +" kgvapor/kgtotal"+"\n";

ss+=" h5 compressor 2 inlet = "+h[5] +" kJ/kg\n";
ss+=" T5 compressor 2 inlet = "+T[5] +" derece C\n";
ss+=" P5 compressor 2 inlet = "+P[5] +" kPa \n";
ss+=" s5 compressor 2 inlet = "+s[5] +" kJ/kgK\n";
ss+=" x5 compressor 2 inlet = "+x[5] +" kgvapor/kgtotal"+"\n";

ss+=" h6 compressor 2 exit = "+h[6] +" kJ/kg\n";
ss+=" T6 compressor 2 exit = "+T[6] +" derece C\n";
ss+=" P6 compressor 2 exit = "+P[6] +" kPa \n";
ss+=" s6 compressor 2 exit = "+s[6] +" kJ/kgK\n";
ss+=" x6 compressor 2 exit = "+x[6] +" kgvapor/kgtotal"+"\n";

ss+=" h8 flash chamber input = "+h[3] +" kJ/kg\n";
ss+=" T8 flash chamber input = "+T[3] +" derece C\n";
ss+=" P8 flash chamber input = "+P[3] +" kPa \n";
ss+=" s8 flash chamber input = "+s[3] +" kJ/kgK\n";
ss+=" x8 flash chamber input = "+x[3] +" kgvapor/kgtotal"+"\n";

ss+=" h4 expansion valve exit-evaporator inlet = "+h[4] +" kJ/kg\n";
ss+=" T4 expansion valve exit-evaporator inlet = "+T[4] +" derece C\n";
ss+=" P4 expansion valve exit-evaporator inlet = "+P[4] +" kPa \n";
```

```
ss+=" s4 expansion valve exit-evaporator inlet = "+s[4] +" kJ/kgK\n";
ss+=" x4 expansion valve exit-evaporator inlet = "+x[4] +" kgvapor/kgtotal"+"\n";
return ss;
}

public double[][] TS(double ti)
{
double a[][]=new double[2][10000];
double tc=b.r.Tc;
double dt=(tc-1.0)/100.0;
int i=0;
double t;
for(t=ti;t<=tc;t+=dt)
{double a1[]=b.property("tx",t,0.0);
 a[1][i]=t;
 a[0][i]=a1[5];
 i++;
}
double a3[]=b.property("tx",tc,1.0);
 a[1][i]=t;
 a[0][i]=a3[5];
 i++;
for(t=tc;t>=ti;t-=dt)
{double a2[]=b.property("tx",t,1.0);
 a[1][i]=t;
 a[0][i]=a2[5];
 i++;
}
int n=i;
double b[][]=new double[2][n];
for(i=0;i<n;i++) {b[0][i]=a[0][i];b[1][i]=a[1][i];}
return b;
}
public double[][] line(double P1,double P2,double hi,double ho,int n)
{
double TT1[]=new double[n];
double ss1[]=new double[n];
double aa[];
double p,h;
for(int i=0;i<n;i++)
{p=P1+i*(P2-P1)/(n-1);
 h=hi+i*(ho-hi)/(n-1);
 aa=b.property("ph",p,h);
 TT1[i]=aa[1];
 ss1[i]=aa[5];
}
double a[][]={TT1,ss1};
return a;
}
public double[][] line1(double T1,double T2,double Pi,double Po,int n)
{
double TT1[]=new double[n];
double ss1[]=new double[n];
double aa[];
double P;
double x=1.0;
aa=b.property("tx",T1,x);
TT1[0]=T1;
ss1[0]=aa[5];
for(int i=1;i<n;i++)
{TT1[i]=T1+i*(T2-T1)/(n-1);
 P=Pi+i*(Po-Pi)/(n-1);
 aa=b.property("tp",TT1[i],P);
 ss1[i]=aa[5];
}
double a[][]={TT1,ss1};
Text.printT(a);
```

```
return a;
}

public void plot(double ti)
{
 Plot pp=new Plot(TS(ti));
 //enthalpy line
 double a34[][]=line(P[3],P[4],h[3],h[4],15);
 double a78[][]=line(P[7],P[8],h[7],h[7],15);
 double tt[]=a34[0];
 double ss[]=a34[1];
 pp.addData(ss,tt);

 tt=a78[0];
 ss=a78[1];
 pp.addData(ss,tt);

 double a10[][]=line1(T[0],T[1],P[0],P[1],15);
 tt=a10[0];
 ss=a10[1];
 pp.addData(ss,tt);

 double a92[][]=line1(T[9],T[2],P[9],P[2],16);
  tt=a92[0];
  ss=a92[1];
  pp.addData(ss,tt);
  double a10_6[][]=line1(T[10],T[6],P[10],P[6],16);
  tt=a10_6[0];
  ss=a10_6[1];
  pp.addData(ss,tt);
 double tt1[]={T[10],T[13],T[7]};
 double ss1[]={s[10],s[13],s[7]};
 pp.addData(ss1,tt1);
 double tt2[]={T[4],T[0]};
 double ss2[]={s[4],s[0]};
 pp.addData(ss2,tt2);
 double tt3[]={T[9],T[3]};
 double ss3[]={s[9],s[3]};
 pp.addData(ss3,tt3);
 double tt4[]={T[1],T[2]};
 double ss4[]={s[1],s[2]};
 pp.addData(ss4,tt4);
  double tt5[]={T[5],T[6]};
 double ss5[]={s[5],s[6]};
 pp.addData(ss5,tt5);
 pp.addData(ss4,tt4);
  double tt6[]={T[1],T[11]};
 double ss6[]={s[1],s[11]};
 pp.addData(ss6,tt6,1);
  double tt7[]={T[5],T[12]};
 double ss7[]={s[5],s[12]};
 pp.addData(ss7,tt7,1);
 pp.setPlabel("Flash chamber 2 stage refrigeration cycle refrigerant :"+refname);
 pp.setYlabel("T, degree C");
 pp.setXlabel("s entropy kJ/kgK");
 pp.plot();
}
}
```

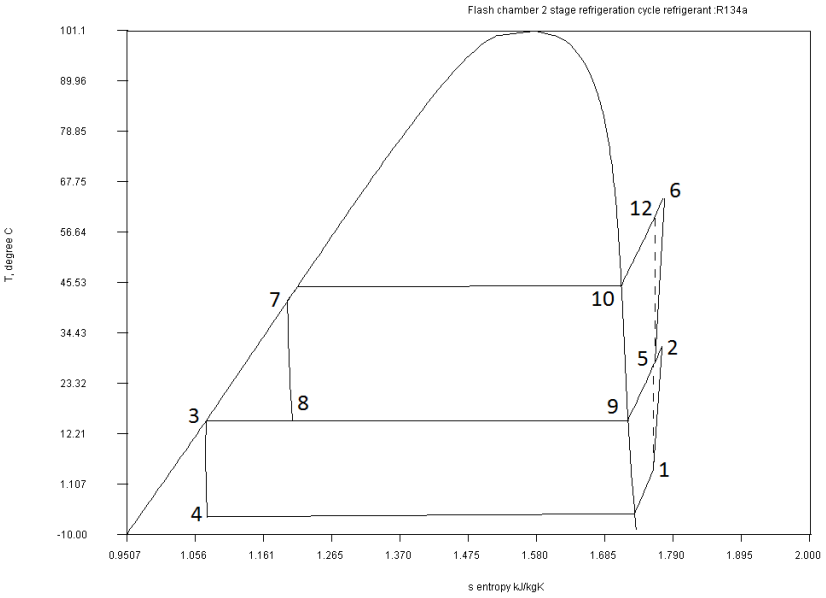
Program 4.2.7 standard refrigeration cycle test program

```
import javax.swing.*;
public class ref_cycle4test
{
public static double Pres(String refrigerant,double T)
{refrigerantEN r=new refrigerantEN(refrigerant);
double a[]=r.property("tx",T,0.0);
```

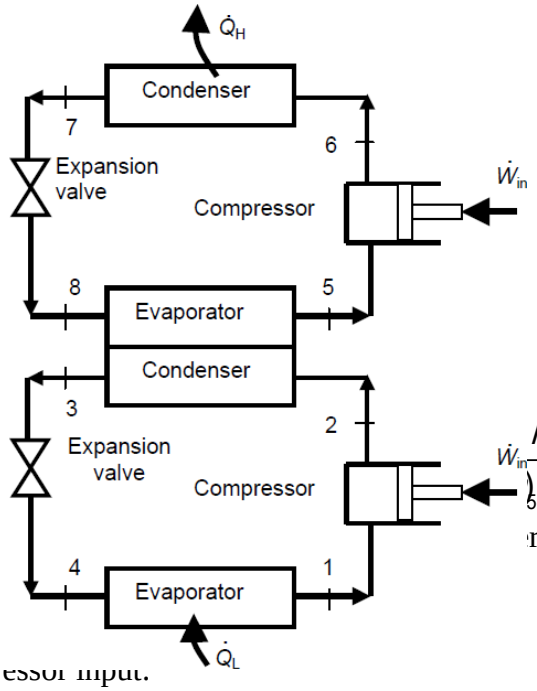
```
double P1=a[0];
return P1;
}
public static void main(String arg[])
{
double m=1.0;  // kg/s çevrim R134a debisi
String refrigerant="R134a";
refrigerantEN r=new refrigerantEN(refrigerant);
double P1=Pres(refrigerant,-5.0);
double P2=Pres(refrigerant,15.0);
double P6=Pres(refrigerant,45.0);
double dT1i=10.0; //degree C
double etaisent2i=0.8;
double etaisent6i=0.8;
double dT7i=3.0; //degree C
double dPevap1i=5.0;
double dPevap2i=5.0;
double dPcomp1i=5.0;
double dPcond1i=5.0;
double dPcond2i=5.0;
double dPcond3i=5.0;
double dh1i=1.0;
double dh2i=1.0;
ref_cycle4 re=new ref_cycle4("R134a",m,dT1i,P1,P2,etaisent2i,P6,etaisent6i,dT7i,
dPevap1i,dPevap2i,dPcomp1i,dPcond1i,dPcond2i,dPcond3i,dh1i,dh2i);
re.cycle();
System.out.println(re.toString());
re.plot(-10.0);
}
}
```

```
---------- Capture Output ----------
> "C:\java\bin\javaw.exe" ref_cycle4test
two stage flash point refrigeration cycle
Flash ratio : 0.21085196615233992
 compressor1 work = 14.826316597366656 kW
 compressor2 work = 24.06248085100617 kW
 total compressor work = 38.88879744837283 kW
 isentropic compressor 1 work = 11.861053277893324 kW
 isentropic compressor 2 work = 19.249984680804914 kW
 total isentropic compressor work = 31.11103795869824 kW
 condenser heat output = 184.61059903816306 kW
 evaporator heat input = 145.51094962363788 kW
 COP evaporator    = 3.7417189311860892 kW
 COP condenser     = 4.74714085163586 kW
 COP evaporator  carnot = 3.818621104617835 kW
 COP condenser   carnot = 4.818621104617835 kW
 h1 compressor 1 inlet = 403.8899286096781 kJ/kg
 T1 compressor 1 inlet = 4.452288435826651 derece C
 P1 compressor 1 inlet = 243.39 kPa
 s1 compressor 1 inlet = 1.7601850896021525 kJ/kgK
 x1 compressor 1 inlet = 2.0 kgvapor/kgtotal
 h2 compressor 1 exit = 422.67767940096854 kJ/kg
 T2 compressor 1 exit = 31.356987836072225 derece C
 P2 compressor 1 exit = 488.7799999999999 kPa
 s2 compressor 1 exit = 1.7726060681496862 kJ/kgK
 x2 compressor 1 exit = 0.0 kgvapor/kgtotal
 h5 compressor 2 inlet = 419.435265464414 kJ/kg
 T5 compressor 2 inlet = 27.93017810803139 derece C
 P5 compressor 2 inlet = 488.7799999999999 kPa
 s5 compressor 2 inlet = 1.7618975967688222 kJ/kgK
 x5 compressor 2 inlet = 0.0 kgvapor/kgtotal
 h6 compressor 2 exit = 443.49774631542016 kJ/kg
 T6 compressor 2 exit = 64.53053948624293 derece C
 P6 compressor 2 exit = 1161.01 kPa
 s6 compressor 2 exit = 1.7762464236009525 kJ/kgK
```

x6 compressor 2 exit = 1.1369546373837682 kgvapor/kgtotal
h8 flash chamber input = 220.5 kJ/kg
T8 flash chamber input = 14.999999999999995 derece C
P8 flash chamber input = 488.7799999999999 kPa
s8 flash chamber input = 1.0726 kJ/kgK
x8 flash chamber input = 0.0 kgvapor/kgtotal
h4 expansion valve exit-evaporator inlet = 219.5 kJ/kg
T4 expansion valve exit-evaporator inlet = -6.104109861511878 derece C
P4 expansion valve exit-evaporator inlet = 233.39 kPa
s4 expansion valve exit-evaporator inlet = 1.0741357660531674 kJ/kgK
x4 expansion valve exit-evaporator inlet = 0.135879209555772 kgvapor/kgtotal



Two single stage refrigeration cycles can be combined to make a cascade refrigeration cycle. In this cycle, upper cycle evaporator and lower cycle condenser is merged to form a single heat exchanger.



Comp

$\eta_{C_{upper}}$

Mass r cycle:

$\dot{m}_A$

Compressor input:

$$W_1 = \dot{m}_A (h_6 - h_5)$$
$$W_2 = \dot{m}_B (h_2 - h_1) \quad (4.2.17\text{-}18)$$

$$W_T = W_1 + W_2 \quad (4.2.19)$$

Evaporator heat transfer (Heat removal from refrigerated space):

$$Q_{evap} = \dot{m}_B (h_1 - h_4) \quad (4.2.20)$$

Condenser heat transfer:

$$Q_{cond} = \dot{m}_A (h_6 - h_7) \quad (4.2.21)$$

Coefficient of performance of the system (COPsys):

$$COP_{sys} = \frac{Q_{evap}}{W_T}$$

Program 4.2.8 cascade refrigeration cycle program

```
 public class ref_cycle7
{ // Cascade cycle
//   two stage refrigeration  cycle
//   refrigerantEN b=new refrigerantEN("R134a");
//   double a[]=b.property("tp",30.0,1.01325);
///---- String s values and meanings --------
//    tv=temperature-specific volume
//    tp=temperature-pressure
//    th=temperature-enthalpy
//    tu=temperature-internal energy
//    ts=temperature-entropy
//    tx=temperature quality
//    pv=pressure-specific volume
//    pt=pressure-temperature
//    ph=pressure-enthapy
//    pu=pressure-internal energy
//    ps=pressure-entropy
//    px=pressure-quality
//    vp=specific volume-pressure
//    vt=specific volume-temperature
//---- output values (method property) ---------
// r[0] P Pressure kPa
// r[1] t temperature degree C
// r[2] v specific volume m^3/kg
// r[3] h enthapy KJ/kg
// r[4] u internal energy KJ/kg
// r[5] s entropy KJ/kgK
// r[6] x quality kg vapour/kg total
// r[7] ro density kg/m^3
ref_cycle3 rA,rB;
double mA,mB;
double Qheatexchanger;
double Wp;
double Qevaporator;
double Qcondenser;
double COP_evaporator;
double COP_condenser;
Plot pp;
public ref_cycle7(String refnameAi,double mAi,double dT1Ai,double P1A,double eta_isent1Ai, double
P2A,double dT6Ai,double dPevap1Ai,double dPevap2Ai,double dPcond1Ai,double dPcond2Ai,double
dPcond3Ai,double dhAi,
String refnameBi,double dT1Bi,double P1B,double eta_isent1Bi, double P2B,double dT6Bi,double
dPevap1Bi,double dPevap2Bi,double dPcond1Bi,double dPcond2Bi,double dPcond3Bi,double dhBi)
{
// top cycle
mA=mAi;
rA=new
ref_cycle3(refnameAi,mAi,dT1Ai,P1A,eta_isent1Ai,P2A,dT6Ai,dPevap1Ai,dPevap2Ai,dPcond1Ai,dP
cond2Ai,dPcond3Ai,dhAi);
rA.cycle();
double m=1.0;
```

```
// bottom cycle
rB=new
ref_cycle3(refnameBi,m,dT1Bi,P1B,eta_isent1Bi,P2B,dT6Bi,dPevap1Bi,dPevap2Bi,dPcond1Bi,dPcon
d2Bi,dPcond3Bi,dhBi);
rB.cycle();
mB=rA.Qevaporator/(rB.h[2]-rB.h[3]);
//rB=new
ref_cycle3(refnameBi,mB,dT1Bi,P1A,eta_isent1Bi,P2A,dT6Bi,dPevap1Bi,dPevap2Bi,dPcond1Bi,dPco
nd2Bi,dPcond3Bi,dhBi);
}

public void cycle()
{
// top cycle
rA=new
ref_cycle3(rA.refname,mA,rA.dT1,rA.P[1],rA.eta_isent,rA.P[2],rA.dT6,rA.dPevap1,rA.dPevap2,rA.dP
cond1,rA.dPcond2,rA.dPcond3,rA.dh);
rA.cycle();
rB=new
ref_cycle3(rB.refname,mB,rB.dT1,rB.P[1],rB.eta_isent,rB.P[2],rB.dT6,rB.dPevap1,rB.dPevap2,rB.dPc
ond1,rB.dPcond2,rB.dPcond3,rB.dh);
rB.cycle();
Wp=rA.Wp+rB.Wp;
Qevaporator=rB.Qevaporator;
Qcondenser=rA.Qcondenser;
Qheatexchanger=rB.Qcondenser;
COP_evaporator=Qevaporator/Wp;
COP_condenser=Qcondenser/Wp;
}

public String toString()
{
cycle();
String ss=" cascade refrigeration cycle \n";
ss+=" compressor work = "+Wp+" kW\n";
ss+=" condenser heat output = "+Qcondenser+" kW\n";
ss+=" evaporator heat input = "+Qevaporator+" kW\n";
ss+=" heat exchanger input/output = "+Qheatexchanger+" kW\n";
ss+=" COP evaporator    = " + COP_evaporator+" kW\n";
ss+=" COP condenser     = " + COP_condenser+" kW\n";
ss+=" Top cycle  \n";
ss+=" refrigerant = "+rA.refname+"\n";
ss+=" m = "+mA+"kg/s\n";
ss+=" h1 compressor input = "+rA.h[1] +" kJ/kg\n";
ss+=" T1 compressor input = "+rA.T[1] +" derece C\n";
ss+=" P1 compressor input = "+rA.P[1] +" kPa \n";
ss+=" s1 compressor input = "+rA.s[1] +" kJ/kgK\n";
ss+=" x1 compressor input = "+rA.x[1] +" kgvapor/kgtotal"+"\n";
ss+=" h2 compressor input = "+rA.h[2] +" kJ/kg\n";
ss+=" T2 compressor output = "+rA.T[2] +" derece C\n";
ss+=" P2 compressor output = "+rA.P[2] +" kPa \n";
ss+=" s2 compressor output = "+rA.s[2] +" kJ/kgK\n";
ss+=" x2 compressor output = "+rA.x[2] +" kgvapor/kgtotal"+"\n";
if(!rA.supercritical)
{
ss+=" h5 compressor input = "+rA.h[5] +" kJ/kg\n";
ss+=" T5 condenser saturation = "+rA.T[5] +" derece C\n";
ss+=" P5 condenser saturation = "+rA.P[5] +" kPa \n";
ss+=" s5 condenser saturation = "+rA.s[5] +" kJ/kgK\n";
ss+=" x5 condenser saturation = "+rA.x[5] +" kgvapor/kgtotal"+"\n";
ss+=" h6 compressor input = "+rA.h[6] +" kJ/kg\n";
ss+=" T6 condenser saturation = "+rA.T[6] +" derece C\n";
ss+=" P6 condenser saturation = "+rA.P[6] +" kPa \n";
ss+=" s6 condenser saturation = "+rA.s[6] +" kJ/kgK\n";
ss+=" x6 condenser saturation = "+rA.x[6] +" kgvapor/kgtotal"+"\n";
}
ss+=" h3 turbine input = "+rA.h[3] +" kJ/kg\n";
```

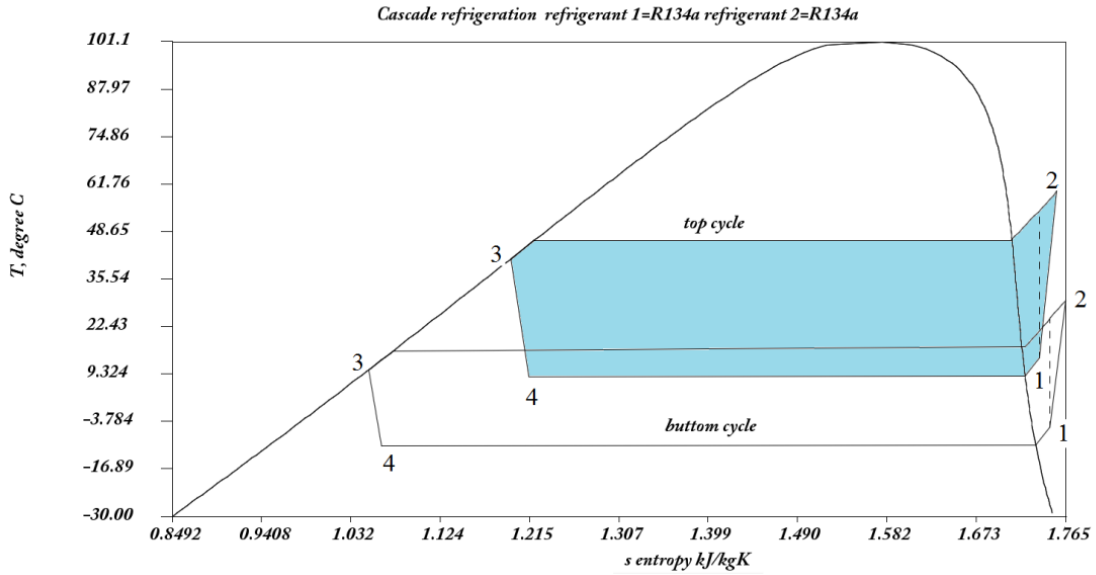
```
ss+=" T3 turbine input = "+rA.T[3] +" derece C\n";
ss+=" P3 turbine input = "+rA.P[3] +" kPa \n";
ss+=" s3 turbine input = "+rA.s[3] +" kJ/kgK\n";
ss+=" x3 turbine input = "+rA.x[3] +" kgvapor/kgtotal"+"\n";
ss+=" h4 turbine output = "+rA.h[4] +" kJ/kg\n";
ss+=" T4 turbine output = "+rA.T[4] +" derece C\n";
ss+=" s4 turbine output = "+rA.s[4] +" kJ/kgK\n";
ss+=" x4 turbine output = "+rA.x[4] +" kgvapor/kgtotal"+"\n";
ss+=" Bottom cycle  \n";
ss+=" refrigerant = "+rB.refname+"\n";
ss+=" m = "+mB+"kg/s\n";
ss+=" h1 compressor input = "+rB.h[1] +" kJ/kg\n";
ss+=" T1 compressor input = "+rB.T[1] +" derece C\n";
ss+=" P1 compressor input = "+rB.P[1] +" kPa \n";
ss+=" s1 compressor input = "+rB.s[1] +" kJ/kgK\n";
ss+=" x1 compressor input = "+rB.x[1] +" kgvapor/kgtotal"+"\n";
ss+=" h2 compressor input = "+rB.h[2] +" kJ/kg\n";
ss+=" T2 compressor output = "+rB.T[2] +" derece C\n";
ss+=" P2 compressor output = "+rB.P[2] +" kPa \n";
ss+=" s2 compressor output = "+rB.s[2] +" kJ/kgK\n";
ss+=" x2 compressor output = "+rB.x[2] +" kgvapor/kgtotal"+"\n";
if(!rA.supercritical)
{
ss+=" h5 compressor input = "+rB.h[5] +" kJ/kg\n";
ss+=" T5 condenser saturation = "+rB.T[5] +" derece C\n";
ss+=" P5 condenser saturation = "+rB.P[5] +" kPa \n";
ss+=" s5 condenser saturation = "+rB.s[5] +" kJ/kgK\n";
ss+=" x5 condenser saturation = "+rB.x[5] +" kgvapor/kgtotal"+"\n";
ss+=" h6 compressor input = "+rB.h[6] +" kJ/kg\n";
ss+=" T6 condenser saturation = "+rB.T[6] +" derece C\n";
ss+=" P6 condenser saturation = "+rB.P[6] +" kPa \n";
ss+=" s6 condenser saturation = "+rB.s[6] +" kJ/kgK\n";
ss+=" x6 condenser saturation = "+rB.x[6] +" kgvapor/kgtotal"+"\n";
}
ss+=" h3 turbine input = "+rB.h[3] +" kJ/kg\n";
ss+=" T3 turbine input = "+rB.T[3] +" derece C\n";
ss+=" P3 turbine input = "+rB.P[3] +" kPa \n";
ss+=" s3 turbine input = "+rB.s[3] +" kJ/kgK\n";
ss+=" x3 turbine input = "+rB.x[3] +" kgvapor/kgtotal"+"\n";
ss+=" h4 turbine output = "+rB.h[4] +" kJ/kg\n";
ss+=" T4 turbine output = "+rB.T[4] +" derece C\n";
ss+=" s4 turbine output = "+rB.s[4] +" kJ/kgK\n";
ss+=" x4 turbine output = "+rB.x[4] +" kgvapor/kgtotal"+"\n";
return ss;
}

public void plot(double ti)
{  double a[][]=rA.TS(ti);
            pp=new Plot(a[0],a[1]);
   pp=rA.plot1(pp);
   a=rB.TS(ti);
   pp.addData(a[0],a[1]);
   pp=rB.plot1(pp);
   pp.setPlabel("Cascade refrigeration  refrigerant 1="+rA.refname+" refrigerant 2="+rB.refname );
   pp.setYlabel("T, degree C");
   pp.setXlabel("s entropy kJ/kgK");
   pp.plot();
}
}
```

Since upper and lower cycles are completely independent, they can run with same refrigerant or two different refrigerants

Program 4.2.9 cascade refrigeration cycle test program

```
import javax.swing.*;

public class ref_cycle7testA
{
```

```
public static void main(String arg[])
{
double mA=1;  // kg/s çevrim R134a debisi
String refA="R134a";
double P2A=1200.0;         //  kPa turbin giriş basıncı
double P1A=400.0 ;         //  kPa turbin çıkış basıncı
double dT1Ai=5.0; //degree C
double eta_isent1Ai=0.8;
double dT6Ai=5.0; //degree C
double dPevap1Ai=2.0; //kPa
double dPevap2Ai=2.0; //kPa
double dPcond1Ai=2.0; //kPa
double dPcond2Ai=2.0; //kPa
double dPcond3Ai=2.0; //kPa
double dhAi=2.0; //kJ/kgK
String refB="R134a";
double P2B=500.0;         //  kPB turbin giriş bBsıncı
double P1B=200.0 ;         //  kPB turbin çıkış bBsıncı
double dT1Bi=5.0; //degree C
double etB_isent1Bi=0.8;
double dT6Bi=5.0; //degree C
double dPevap1Bi=2.0; //kPB
double dPevap2Bi=2.0; //kPB
double dPcond1Bi=2.0; //kPB
double dPcond2Bi=2.0; //kPB
double dPcond3Bi=2.0; //kPB
double dhBi=2.0; //kJ/kgK
ref_cycle7 r=new
ref_cycle7(refA,mA,dT1Ai,P1A,eta_isent1Ai,P2A,dT6Ai,dPevap1Ai,dPevap2Ai,dPcond1Ai,dPcond2Ai,
dPcond3Ai,dhAi,
refB,dT1Bi,P1B,etB_isent1Bi,P2B,dT6Bi,dPevap1Bi,dPevap2Bi,dPcond1Bi,dPcond2Bi,dPcond3Bi,dhBi);
System.out.println(r.toString());
r.cycle();
r.plot(-30.0);}}
```



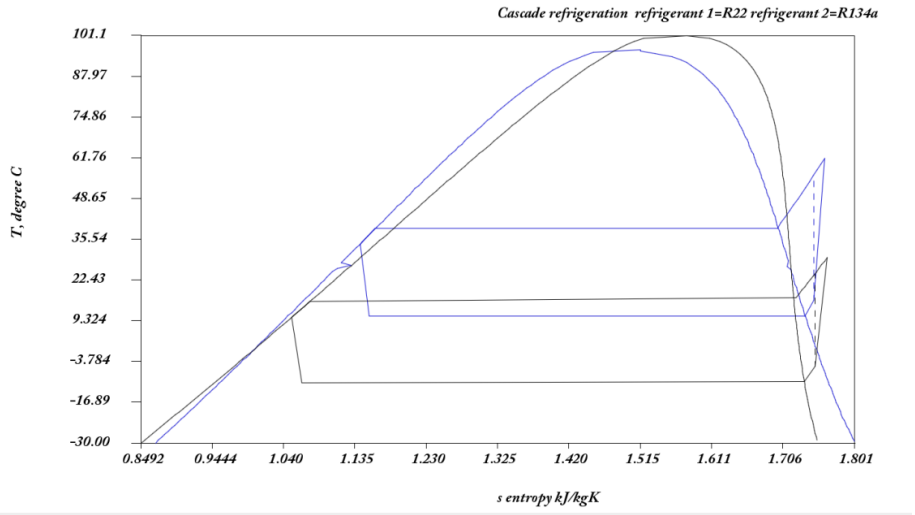
Program 4.2.10 cascade refrigeration cycle test program

```
---------- Capture Output ----------
> "C:\java\bin\java.exe" ref_cycle7testA
cascade refrigeration cycle
compressor work = 46.557689380580285 kW
condenser heat output = 179.1910789630847 kW
evaporator heat input = 129.2012201024127 kW
heat exchanger input/output = 147.90723649915645 kW
COP evaporator    = 2.7750780122757535 kW
COP condenser     = 3.8487966509314617 kW
Top cycle
```

refrigerant = R134a
m = 1.0kg/s
h1 compressor input = 408.2319432599678 kJ/kg
T1 compressor input = 13.760391726590042 derece C
P1 compressor input = 400.0 kPa
s1 compressor input = 1.7384553609550266 kJ/kgK
x1 compressor input = 2.0 kgvapor/kgtotal
h2 compressor input = 437.51578572389604 kJ/kg
T2 compressor output = 59.75600041799221 derece C
P2 compressor output = 1200.0 kPa
s2 compressor output = 1.7561889728460591 kJ/kgK
x2 compressor output = 1.0966552920873422 kgvapor/kgtotal
h5 compressor input = 422.3858140744719 kJ/kg
T5 condenser saturation = 46.21416207180537 derece C
P5 condenser saturation = 1198.0 kPa
s5 condenser saturation = 1.7100129797825283 kJ/kgK
x5 condenser saturation = 0.0 kgvapor/kgtotal
h6 compressor input = 265.93693684559605 kJ/kg
T6 condenser saturation = 46.14926677186208 derece C
P6 condenser saturation = 1196.0 kPa
s6 condenser saturation = 1.2201990973101455 kJ/kgK
x6 condenser saturation = 0.0 kgvapor/kgtotal
h3 turbine input = 258.32470676081135 kJ/kg
T3 turbine input = 41.14926677186208 derece C
P3 turbine input = 1194.0 kPa
s3 turbine input = 1.1966004528854404 kJ/kgK
x3 turbine input = 0.0 kgvapor/kgtotal
h4 turbine output = 260.32470676081135 kJ/kg
T4 turbine output = 8.611307337942751 derece C
s4 turbine output = 1.214757586220795 kJ/kgK
x4 turbine output = 0.0 kgvapor/kgtotal
Bottom cycle
refrigerant = R134a
m = 0.7160847400458633kg/s
h1 compressor input = 396.66384167379755 kJ/kg
T1 compressor input = -5.329546896665409 derece C
P1 compressor input = 200.0 kPa
s1 compressor input = 1.748643091085814 kJ/kgK
x1 compressor input = 2.0 kgvapor/kgtotal
h2 compressor input = 420.7864712323407 kJ/kg
T2 compressor output = 29.609873382254033 derece C
P2 compressor output = 500.0 kPa
s2 compressor output = 1.7647141124425834 kJ/kgK
x2 compressor output = 1.0706127884130356 kgvapor/kgtotal
h5 compressor input = 407.5782465453429 kJ/kg
T5 condenser saturBtion = 15.582823738854056 derece C
P5 condenser saturBtion = 498.0 kPa
s5 condenser saturBtion = 1.720567043376862 kJ/kgK
x5 condenser saturBtion = 0.0 kgvapor/kgtotal
h6 compressor input = 221.13917141892864 kJ/kg
T6 condenser saturBtion = 15.457077674540868 derece C
P6 condenser saturBtion = 496.0 kPa
s6 condenser saturBtion = 1.0747486638109192 kJ/kgK
x6 condenser saturBtion = 0.0 kgvapor/kgtotal
h3 turbine input = 214.23656417848645 kJ/kg
T3 turbine input = 10.457077674540868 derece C
P3 turbine input = 494.0 kPa
s3 turbine input = 1.050685283471824 kJ/kgK
x3 turbine input = 0.0 kgvapor/kgtotal
h4 turbine output = 216.23656417848645 kJ/kg
T4 turbine output = -10.585295785211429 derece C
s4 turbine output = 1.0638402933582916 kJ/kgK
x4 turbine output = 0.0 kgvapor/kgtotal

```
import javax.swing.*;
public class ref_cycle7test
```

```
{
public static void main(String arg[])
{
double mA=1;  // kg/s çevrim R134a debisi
String refA="R22";
double P2A=1500.0;         //  kPa turbin giriş basıncı
double P1A=700.0 ;         //  kPa turbin çıkış basıncı
double dT1Ai=5.0; //degree C
double eta_isent1Ai=0.8;
double dT6Ai=5.0; //degree C
double dPevap1Ai=2.0; //kPa
double dPevap2Ai=2.0; //kPa
double dPcond1Ai=2.0; //kPa
double dPcond2Ai=2.0; //kPa
double dPcond3Ai=2.0; //kPa
double dhAi=2.0; //kJ/kgK
String refB="R134a";
double P2B=500.0;         //  kPB turbin giriş bBsıncı
double P1B=200.0 ;         //  kPB turbin çıkış bBsıncı
double dT1Bi=5.0; //degree C
double etB_isent1Bi=0.8;
double dT6Bi=5.0; //degree C
double dPevap1Bi=2.0; //kPB
double dPevap2Bi=2.0; //kPB
double dPcond1Bi=2.0; //kPB
double dPcond2Bi=2.0; //kPB
double dPcond3Bi=2.0; //kPB
double dhBi=2.0; //kJ/kgK
ref_cycle7 r=new
ref_cycle7(refA,mA,dT1Ai,P1A,eta_isent1Ai,P2A,dT6Ai,dPevap1Ai,dPevap2Ai,dPcond1Ai,dPcond2Ai,dPcond3Ai,dhAi,ref
B,dT1Bi,P1B,etB_isent1Bi,P2B,dT6Bi,dPevap1Bi,dPevap2Bi,dPcond1Bi,dPcond2Bi,dPcond3Bi,dhBi);
System.out.println(r.toString());
r.cycle();
r.plot(-30.0);}}
```



Program 4.2.11 cascade refrigeration cycle test program

```
---------- Capture Output ----------
> "C:\java\bin\java.exe" ref_cycle7test
 cascade refrigeration cycle
 compressor work = 43.63978004024978 kW
 condenser heat output = 194.35609347721999 kW
 evaporator heat input = 147.0858959305664 kW
 heat exchanger input/output = 168.38129220334156 kW
 COP evaporator    = 3.3704545667944785 kW
 COP condenser     = 4.453645121445657 kW
 Top cycle
 refrigerant = R22
 m = 1.0kg/s
```

h1 compressor input = 412.1929202446289 kJ/kg
T1 compressor input = 15.82384668927146 derece C
P1 compressor input = 700.0 kPa
s1 compressor input = 1.7471638120291129 kJ/kgK
x1 compressor input = 2.0 kgvapor/kgtotal
h2 compressor input = 436.16772151850734 kJ/kg
T2 compressor output = 61.50679237571848 derece C
P2 compressor output = 1500.0 kPa
s2 compressor output = 1.7616085057473234 kJ/kgK
x2 compressor output = 1.1195970417376142 kgvapor/kgtotal
h5 compressor input = 416.1036313700978 kJ/kg
T5 condenser saturation = 39.026896625615855 derece C
P5 condenser saturation = 1498.0 kPa
s5 condenser saturation = 1.6999540073912813 kJ/kgK
x5 condenser saturation = 0.0 kgvapor/kgtotal
h6 compressor input = 248.2654432673927 kJ/kg
T6 condenser saturation = 38.97310626089193 derece C
P6 condenser saturation = 1496.0 kPa
s6 condenser saturation = 1.1618967780114848 kJ/kgK
x6 condenser saturation = 0.0 kgvapor/kgtotal
h3 turbine input = 241.81162804128735 kJ/kg
T3 turbine input = 33.97310626089193 derece C
P3 turbine input = 1494.0 kPa
s3 turbine input = 1.1418973251732016 kJ/kgK
x3 turbine input = 0.0 kgvapor/kgtotal
h4 turbine output = 243.81162804128735 kJ/kg
T4 turbine output = 10.728386486701817 derece C
s4 turbine output = 1.1541583870802548 kJ/kgK
x4 turbine output = 0.0 kgvapor/kgtotal
Bottom cycle
refrigerant = R134a
m = 0.8152087532019033kg/s
h1 compressor input = 396.66384167379755 kJ/kg
T1 compressor input = -5.329546896665409 derece C
P1 compressor input = 200.0 kPa
s1 compressor input = 1.748643091085814 kJ/kgK
x1 compressor input = 2.0 kgvapor/kgtotal
h2 compressor input = 420.7864712323407 kJ/kg
T2 compressor output = 29.609873382254033 derece C
P2 compressor output = 500.0 kPa
s2 compressor output = 1.7647141124425834 kJ/kgK
x2 compressor output = 1.0706127884130356 kgvapor/kgtotal
h5 compressor input = 407.5782465453429 kJ/kg
T5 condenser saturation = 15.582823738854056 derece C
P5 condenser saturation = 498.0 kPa
s5 condenser saturation = 1.720567043376862 kJ/kgK
x5 condenser saturation = 0.0 kgvapor/kgtotal
h6 compressor input = 221.13917141892864 kJ/kg
T6 condenser saturation = 15.457077674540868 derece C
P6 condenser saturation = 496.0 kPa
s6 condenser saturation = 1.0747486638109192 kJ/kgK
x6 condenser saturation = 0.0 kgvapor/kgtotal
h3 turbine input = 214.23656417848645 kJ/kg
T3 turbine input = 10.457077674540868 derece C
P3 turbine input = 494.0 kPa
s3 turbine input = 1.050685283471824 kJ/kgK
x3 turbine input = 0.0 kgvapor/kgtotal
h4 turbine output = 216.23656417848645 kJ/kg
T4 turbine output = -10.585295785211429 derece C
s4 turbine output = 1.0638402933582916 kJ/kgK
x4 turbine output = 0.0 kgvapor/kgtotal