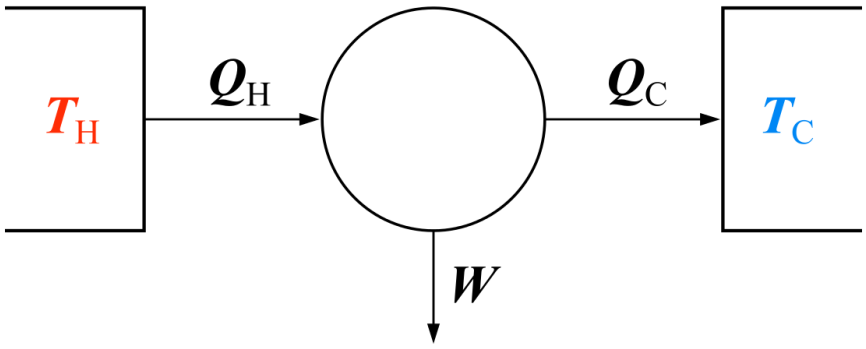
Cairo University

Faculty of Engineering

2nd year Mechanical Engineering

Thermodynamics cycles solver



submitted to: Dr. Maha Hassanein

submitted by: Blizzard team

Abstract

This document is dedicated from Blizzard team to clarify the purpose of the project. Thermodynamics cycle solver (TCS) – clearly from its name- calculates the outputs of a defined thermodynamic steam cycle type using some inputs defined by the user. The document discusses the purpose and the motive of the (TCS), definition of a thermodynamic cycle and its main components, the challenges of attempting to solve a cycle, mathematics, equations, laws and logic used to solve the problem, solving procedure, method testing and result analysis.

Table of contents

List of figures

1. Introduction

Generating power is considered as one of the most vital problems cared of in the modern age. Many methods are innovated to solve this matter but thermodynamic power cycles are the most used ones due to its heritage and simplicity. Steam power cycle in specific are very common. To design or to solve a steam power cycle is to determine the outputs of the cycle and the steam states at various points using defined inputs. The solving process implies comprehension and application of many laws, equations and logic in a logical algorithm. Such a process generates many numbers and parameters that can easily confuse the researcher and the human error is very obvious. Nowadays, computer calculation programs are utilized in the field of problems where the human error is obvious. Thermodynamic cycles solver (TCS) allows the users to solve steam cycles easily, fast and with very small error. Only a few inputs are required to completely solve steam cycle using (TCS). The outputs of the cycles allow us to estimate the advantage extracted from the cycle and evaluate the performance and the efficiency.

2. What is the problem you are solving?

The problem chosen to solve is the problem of thermodynamic cycles (specifically steam), mainly what happens in solving thermodynamic power cycles.

3. Basic components:

A. **Turbine**: uses the energy of the steam to turn a turbine that produces electricity.

B. **Condenser**: condenses the steam back to water so that it can be returned to the heat source to be heated again.

C. **Pump**: Increases pressure and pumps water through the cycle.

D. **Boiler:** Provides heat to generate steam.

4. Vapor power cycles:

I.Carnot cycle

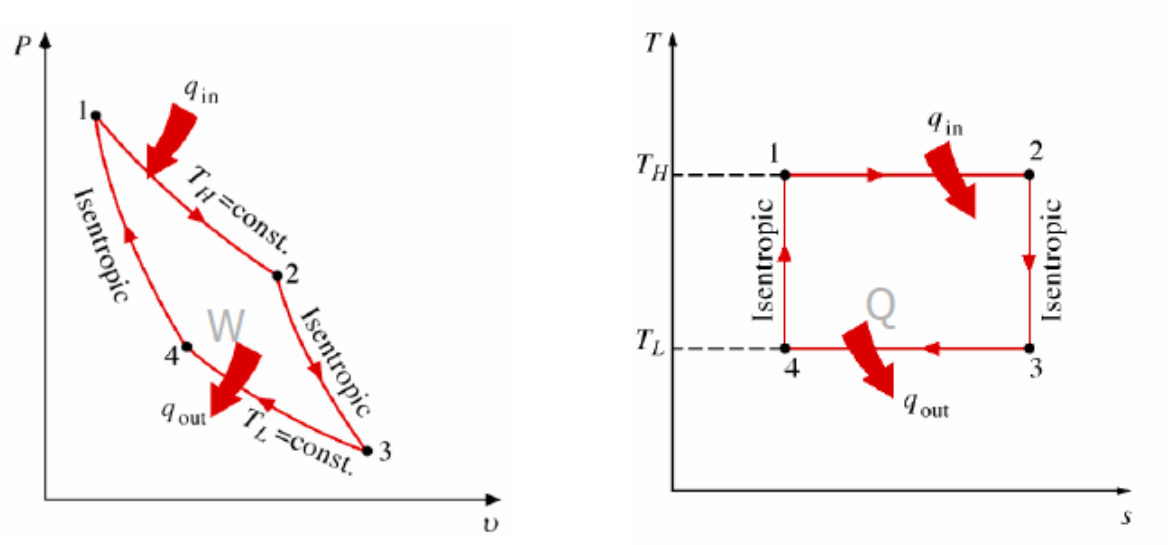


Figure 1-Carnot cycle

Process 1-2: Reversible isothermal heat addition process at constant Temperature TH.

Process 2-3: Reversible adiabatic expansion process from P2 to P3.

Process 3-4: Reversible isothermal heat rejection process at constant Temperature TL

Process 4-1: Reversible adiabatic compression process from P4 to P1.

The efficiency for a Carnot cycle is:$\eta_{th,carnot=1-\frac{T_L}{T_H}}$

Practically, it is very difficult to add or reject heat to or from the working fluid at constant temperature.
But, it is comparatively easy to add or reject heat to or from the working Fluid at constant pressure as in figure (1). Therefore, Carnot cycle is not used as an idealized cycle for steam power plants.

II.Rankine cycle

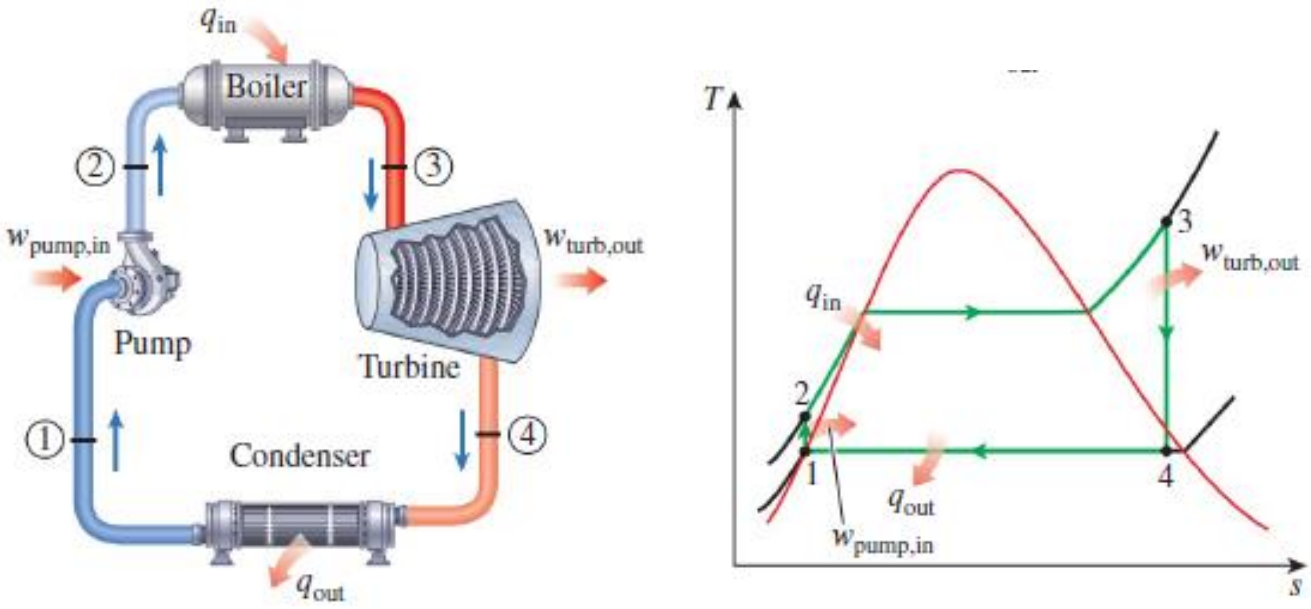


Figure 2-Rankine cycle

Process 1-2: Water from the condenser at low pressure is pumped into the boiler at high pressure. This process is adiabatic.
Process 2-3: Water is converted into steam at constant pressure by the addition of heat in the boiler.
Process 3-4: Adiabatic expansion of steam in the steam turbine.
Process 4-1: Constant pressure heat rejection in a condenser
Thermal Efficiency of Rankine Cycle: $\eta_{th,=1-\frac{q_{out}}{q_{in}}}$

Steams power cycles are essential for generating mechanical power generally and generating electricity in power stations in specific. The idea of a cycle is transform

a type of energy from one form (heat energy) to another (mechanical energy) through a working fluid (steam). Liquid is pumped in range where specific volume is relatively small, so the work exerted by the pump on steam is also relatively small and can be ignored. Temperature is raised with boiler to move steam to a range where specific volume is relatively large and pressure lines spacing are wider. That's the range where the mechanical power extraction occurs using turbine. The condenser is used to return steam to its initial conditions as in figure (2).

III.Reheat Rankine cycle

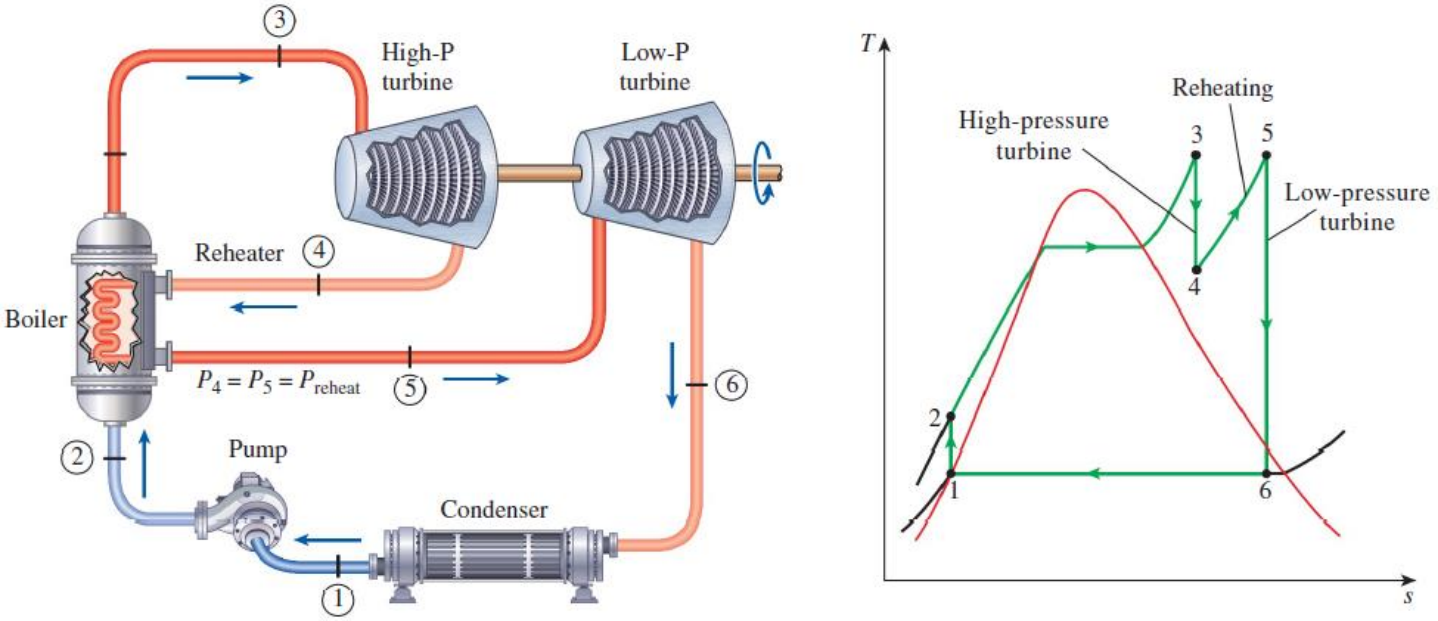


Figure 3-Reheat Rankine cycle

To take advantage of the increased efficiencies at higher boiler pressure without facing the excessive moisture at the final stages of the turbine, reheating is used. In the ideal reheating cycle, the expansion process takes place in two stages, i.e., the high-pressure and low-pressure turbines as in figure (3).

Process 1-2: pump – isentropic compression
Process 2-3: Boiler – heat added at constant pressure
Process 3-4: high pressure turbine (HP) – isentropic expansion
Process 4-5: Boiler – heat added at constant pressure

Process 5-6: low pressure turbine (LP) – isentropic expansion

Process 6-1: condenser – heat rejected at constant pressure

IV.Regenerative Rankine cycle

- Open Feed-water Heaters

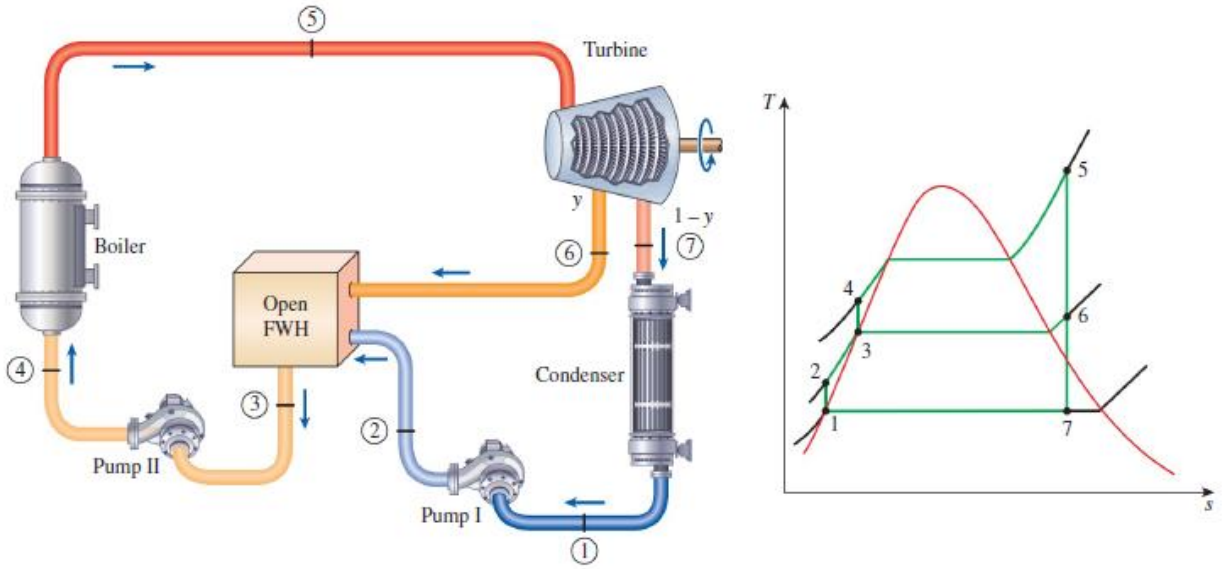


Figure 4-Open feed water heater

As in figure (4), it's observed that:

Process 1-2: Pump #1 - isentropic compression.

Process 2-3: Feed-water Heater (FWH) - temperature increases mixing

Process 6-3: Feed-water Heater (FWH) - temperature decreases by mixing.

Process 3-4: pump #2 – isentropic compression.

Process 4-5: Boiler - heat added at constant pressure.

Process 5-6: High-Pressure Turbine (HP) - isentropic expansion.

Process 6-7: Low-Pressure Turbine (LP)- isentropic expansion.

Process 7-1: Condenser - heat rejected at constant pressure.

- Closed Feed-water Heaters

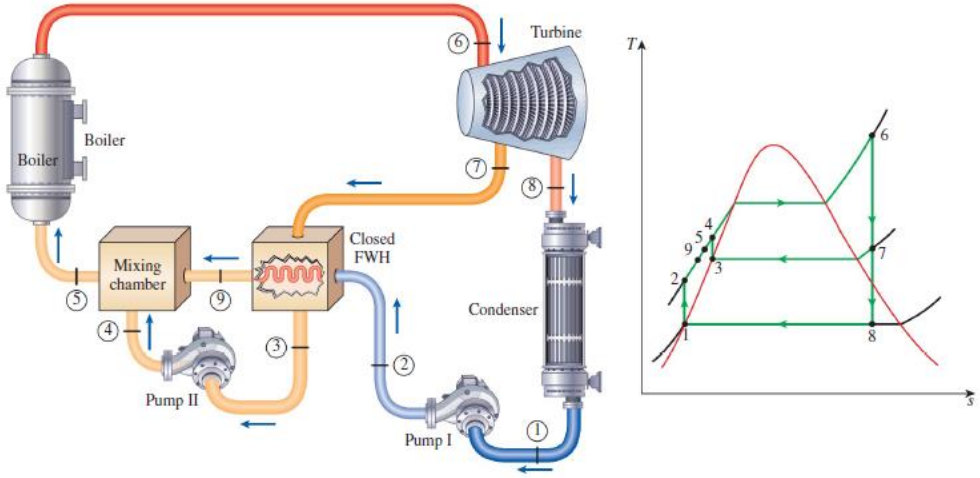


Figure 5-Closed feed water heater

As observed in figure (5), a closed feed-water heater is just a heat exchanger. The streams exchanging heat do not mix. In the closed feed-water heater shown here, superheated vapor in stream 7 is cooled and condensed when it transfers heat to stream 2. Stream 9 is at the same pressure, but a higher temperature than stream 2. This reduces the irreversibility in the boiler and improves the efficiency of the cycle. In the diagram shown here, stream 3 is pumped up to the pressure of the boiler and then mixed with stream 9 to make up the boiler feed, stream 5. This further reduces the irreversibility of the boiler, but stream 3 is sometimes returned to the condenser.

5. Gas power cycle

Gas power cycles are cycles that operate using the combustion of many fuels, they are mainly used in automotive applications, but they are also used in power generation.

I. Ericsson Cycle

Ericsson cycle, shown in figure (6), is one of the main gas power cycles that are used in power stations, it uses regeneration to increase cycle efficiency, it consists of four processes:

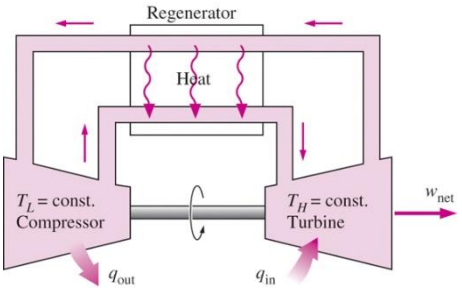


Figure 6-Ericsson cycle

1. Constant temperature expansion
2. Constant pressure regeneration (from working fluid to regenerator)
3. Constant temperature compression
4. Constant pressure regeneration (from regenerator to working fluid)

II. Brayton Cycle

Brayton cycle is called the basic gas turbine engine cycle, it has two types open-cycle – shown in figure (7)- and closed-cycle – shown in figure (8)-, they are pretty much the same except for the heat addition and heat rejection since the open type uses combustion directly for heat addition to fresh air and the heat rejection consists of ejecting the exhaust gases, while the closed type keeps the air contained within the cycle, it consists of these processes:

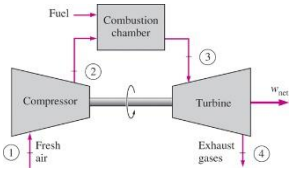


Figure 7-open Brayton cycle

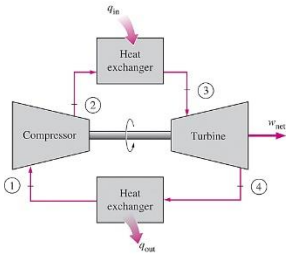


Figure 8-closed Brayton cycle

1. Isentropic compression
2. Constant pressure heat addition
3. Isentropic expansion
4. Constant pressure heat rejection

Some modifications can be made to Brayton cycle in order to make it more efficient such as reheating to increase power output, adding intercooling between two compression stages which lowers energy consumed by compressors or adding regeneration before combustion to lower the heat input needed, or adding both together in the same cycle, all these additions can increase efficiency.

6. Why is this interesting and important?

An important application of thermodynamics is the analysis of power cycles through which the energy absorbed as heat can be continuously converted into mechanical work. A thermodynamic analysis of the heat engine cycles provides valuable information regarding the design of new cycles or a combined cycle for improving the existing cycles, or pushing efficiency gas turbine output with Brayton or combined with topping or bottoming Rankine cycle via heat exchangers/recuperates respectively. Application of combined cycle driven power plants, either steam or nuclear, plays a very important role in industry these days in order to make existence of these power plants more cost effective.

7. What makes it challenging to solve?

Steam stables provides the comprehension of what happens in a specific component to get an output, sometimes this process is slow, time consuming and error could exist due to human error.

8. Solving algorithm

As indicated before, the solving of our problem, for a steam cycle, means to have:

- A fully defined fluid properties (pressure, temperature, entropy, enthalpy) at every state in the cycle.
- Work extracted from or supplied to each component in the cycle.
- Net work of the cycle.
- Heat input or output from each component in the cycle.
- Net heat extracted from or supplied to the cycle.
- Thermal efficiency of the cycle.

9. Mathematics and laws:

1) **The first law of thermodynamics**: "*Energy can neither be created nor destroyed, it can only be transformed from one form to another*" [1].

To apply a thermodynamics law, attention is dedicated to a matter "system" –shown in figure (9)- that is defined by a quantity of mass and certain system boundary that is surrounded by surroundings.

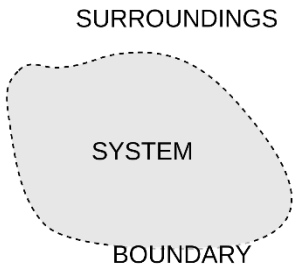


Figure 9-thermodynamic system

Systems can be classified into three types:

- Isolated system: prevents energy and mass transfer across the system boundary (fixed mass and energy).
- Closed system: A quantity of a matter of fixed mass and identity upon which attention is focused for study. It allows energy transfer (such as heat and mechanical work) but for a fixed mass.
- Open system: it's a control volume upon which attention is focused for study. The system is surrounded by a boundary called control surface. Mass crosses the system boundary (either in, out or both together). Everything outside the boundary is called surroundings.

The first law of thermodynamics concerns about this system. It shows that the overall energy of the system must resort to equality. There's validity in saying that Energy IN can be equal to Energy OUT plus the energy change.

- **for a closed system:** assuming that the system takes in heat and transfers it to work, we shall express this with:

$$dE_{IN} - dE_{out} = dE_{change} \quad (1) \qquad [1]$$

$$\delta Q = dE_{IN}, \quad \delta W = dE_{out},$$

$$dE_{change} = internal\ energy + kinetic\ energy + potential\ energy \quad (2)$$

Note that there are types of energy that our study doesn't address them such as chemical energy and electromagnetic energy. Also, the mass is constant for a closed system.

change in internal energy= dU

change in kinetic energy = $d\left(\frac{mv^2}{2}\right) = md\left(\frac{v^2}{2}\right) = mvdv$

change in potential energy = $d(mgz) = mgdz$

the first law of thermodynamics for a closed system is formed from eq. (1) and eq. (2)

$$\delta Q - \delta W = dU + mvdv + mgdz \quad (3)$$

$$Q - W = \Delta U + \frac{m\Delta v^2}{2} + mg\Delta z \quad (4)$$

- **for an open system:** assuming that the system takes in heat and transfers it to work, we shall express this with:

$$mass\ inlet = m_i\ (kg), \qquad mass\ exit = m_e\ (kg),$$

$$initial\ state\ mass = m_1\ (kg),$$

$$final\ state\ mass = m_2\ (kg)$$

$$u = specific\ internal\ energy\ \left(\frac{kJ}{kg}\right),$$

$$h = enthalpy\ enthalpy = u + Pv\ \left(\frac{kJ}{kg}\right)$$

$$P = fluid\ pressure\ (kPa), v = specific\ volume\ \left(\frac{m^3}{kg}\right), flow\ work = Pv\ (kJ)$$

Subscript (e) means exit, subscript (i) means inlet.

$$sum\ of\ input\ energy = \delta Q + \delta m_i \left(u + PV + \frac{v^2}{2} + gz\right)_i$$

$$= \delta Q + \delta m_i \left(h + \frac{v^2}{2} + gz\right)_i \quad (5)$$

$$sum\ of\ output\ energy = \delta W + \delta m_e \left(u + PV + \frac{v^2}{2} + gz\right)_e$$

$$= \delta W + \delta m_e \left(h + \frac{v^2}{2} + gz\right)_e \quad (6)$$

$$change\ of\ stored\ energy = m_2 \left(u + \frac{v^2}{2} + gz\right)_2 - m_1 \left(u + \frac{v^2}{2} + gz\right)_1$$

$$= dE_{CV} \quad (7)$$

Substitute eq. (2), eq. (5), eq. (6) and eq. (7) in eq. (1)

$$\delta Q + \delta m_i \left(h + \frac{v^2}{2} + gz\right)_i = \delta W + \delta m_e \left(h + \frac{v^2}{2} + gz\right)_e + m_2 \left(u + \frac{v^2}{2} + gz\right)_2 - m_1 \left(u + \frac{v^2}{2} + gz\right)_1$$

$$\delta Q - \delta W = \delta m_e \left(h + \frac{v^2}{2} + gz\right)_e - \delta m_i \left(h + \frac{v^2}{2} + gz\right)_i + m_2 \left(u + \frac{v^2}{2} + gz\right)_2 - m_1 \left(u + \frac{v^2}{2} + gz\right)_1 \quad (8)$$

For a time rate

$$\dot{Q} - \dot{W} = \dot{m}_e \left(h + \frac{v^2}{2} + gz\right)_e - \dot{m}_i \left(h + \frac{v^2}{2} + gz\right)_i + \frac{dE_{CV}}{dt} \quad (9) \quad [2]$$

For most of applications in the steam cycle:

- steady state: $\frac{dE_{CV}}{dt} = 0$
- mostly one inlet and one exit: $\dot{m}_e = \dot{m_i} = \dot{m}$
- change in potential energy and kinetic energy are negligible

the final form that will be used for the algorithm solving is

$$\dot{Q} - \dot{W} = \dot{m}(h_e - h_i) \quad (10)$$

to obtain the equation for a unit mass: $\dot{q} - \dot{w} = (h_e - h_i)$ (11) [2]

$$where\ \dot{q} = \frac{\dot{Q}}{m}\left(\frac{kJ}{kg}\right), \qquad \dot{w} = \frac{\dot{W}}{m}\left(\frac{kJ}{kg}\right)$$

2) **the efficiency of the second law of the thermodynamics**: this principle declares to what extent the actual characteristics of the cycle differs from the ideal characteristics of the cycle.

This difference is due to friction losses, pipeline losses and losses related to the components efficiency itself. The entropy of the state indicates these losses, the more losses you have, the more entropy value is. Hence, we can say that an ideal component turbine or pump (isentropic) keeps a constant entropy. η_{is}(isentropic efficiency) for a turbine is the ratio between the actual work extracted and the ideal work extracted. η_{is}(isentropic efficiency) for a pump is the ratio between the ideal work supplied and the actual work supplied.

$$\eta_{is(turbine)} = \frac{h_{in} - h_{out\ actual}}{h_{in} - h_{out\ ideal}}\ (12), \qquad \eta_{is(pump)} = \frac{h_{out\ ideal} - h_{in}}{h_{out\ actual} - h_{in}}\ (13)$$

3) **the thermal efficiency of the whole cycle**: It indicates to what extent the cycle succeeded to extract net energy (work) from the input energy (heat). It's the ratio between the net work to the heat input.

$$\eta_{thermal} = \frac{w_{net}}{Q_{total\ input}} = 1 - \frac{Q_{total\ output}}{Q_{total\ input}}\ (14)$$

4) **interpolation**: we can get the steam characteristics from steam table for any state. the algorithm is to have two independent properties through which we can get any other property for the same state. the steam table provides discrete points for the steam properties graphs. Actually, in-between points presence is very possible. To solve this problem, we approximate the non-linear curve to a linear relationship through this equation using the data one step before and one step after:

$$\frac{known\ property - same\ prop._{\cdot 1\ step\ before}}{same\ prop._{\cdot 1\ step\ after} - same\ prop._{\cdot 1\ step\ before}} = \frac{required\ property - same\ prop._{\cdot 1\ step\ before}}{same\ prop._{\cdot 1\ step\ after} - same\ prop._{\cdot 1\ step\ before}}\ (15)$$

5) **wet region Equation**: in the wet region, the pressure and temperature remain constant during heating or cooling. But the vapor content changes. The vapor content increases with heating and decreases with cooling. The dryness fraction (steam quality) expresses the vapor mass content in the steam. x=0 means completely liquid water steam (fluid). x=1 means completely vapor steam (gas). the properties (u, h, s, v) changes with quality. For p means property:

$$p_{fg} = p_{gas} - p_{fluid} \ (16)$$
$$p_{required} = p_{fluid} + x * p_{fg} \ (17)$$

10. Solving procedure

For a simple Rankine cycle shown in figure (10):

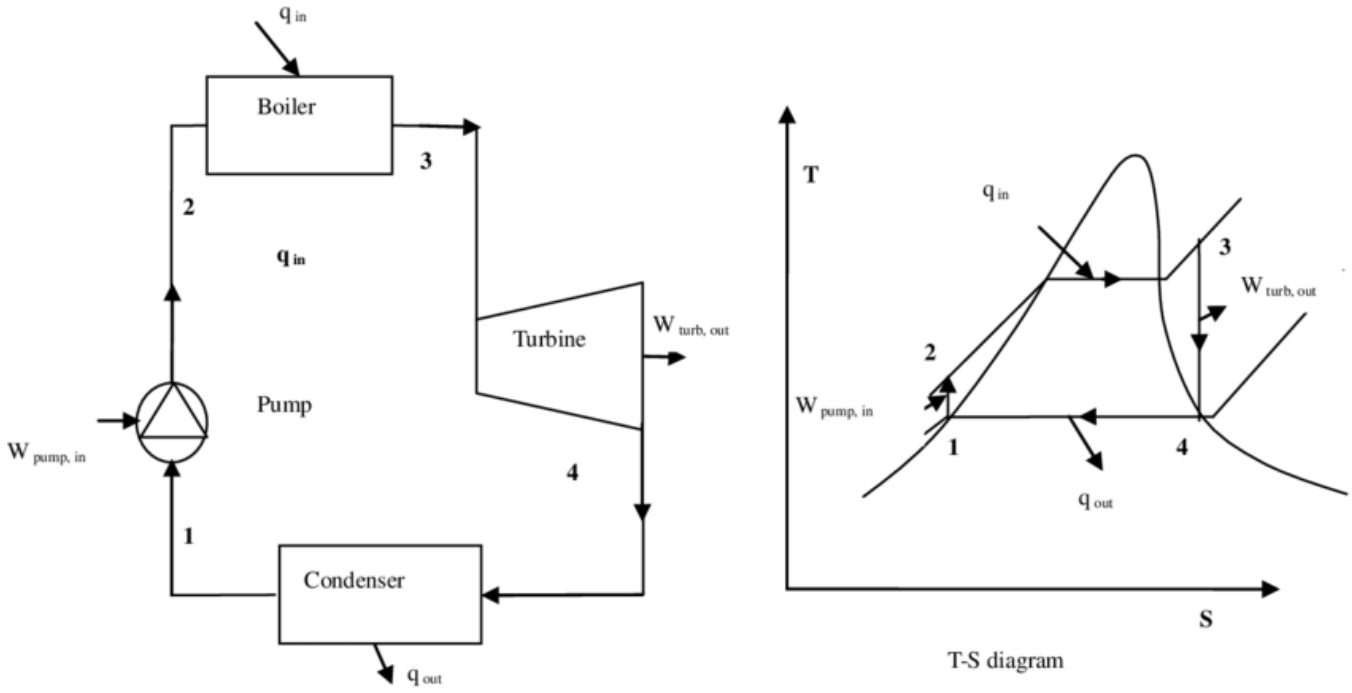


Figure 10-Simple Rankine cycle

1) pump: from the flow chart provided in figure (11), for a given water steam state (quality=0) with given temperature, entropy, enthalpy, the pump increases the water pressure up to the known boiling pressure. Also, the output entropy can be deduced through the efficiency of the second law of the thermodynamics. The output enthalpy is deduced through interpolation. For an adiabatic pump, it doesn't imply any heat transfer so it's said that: $\dot{q} = 0$. From eq. (11):

$$-\dot{w_{pump}} = \dot{h_e} - \dot{h_i}$$
$$\dot{w_{pump}} = -\dot{h_e} + \dot{h_i} \ \ (-ve\ sign)$$
$$\dot{w_{pump}} = \dot{h_e} - \dot{h_i} \ \ (\ abs. + ve\ sign) \ (18)$$

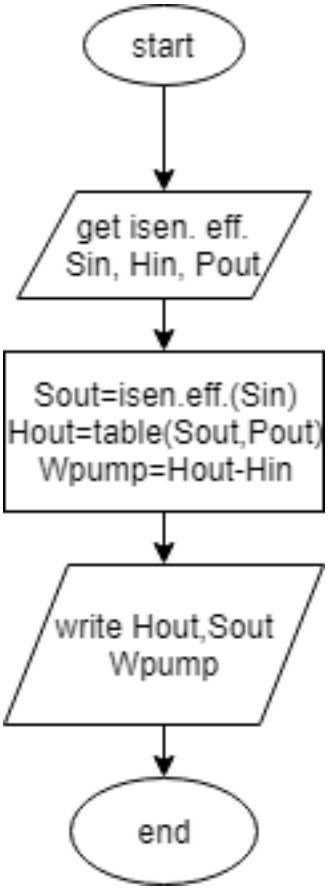


Figure 11- pump flow chart

2) boiler: for a given pump output pressure as a boiler input pressure. The output temperature must be decided by the user. The process is constant pressure. Through interpolation, the output entropy and enthalpy is calculated. For a boiler, it doesn't imply any type of mechanical work so it's said that $\dot{w} = 0$. From eq. (11)

$$\dot{q}_{boiler} = h_e - h_i \ (19)$$

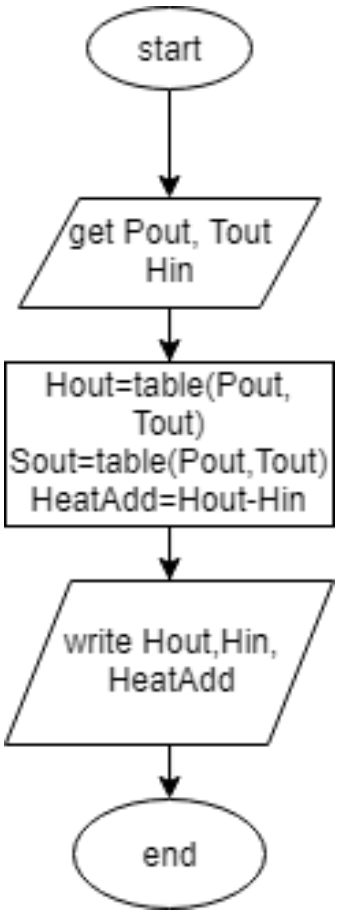


Figure 12-boiler flow chart

3) turbine: for a given input enthalpy and entropy from the boiler output, and a given coolant water temperature. Thus the turbine output pressure is determined. The ideal output enthalpy is calculated by interpolation. The actual output enthalpy is calculated by 2nd eff. Law. The output entropy and steam quality are determined by interpolation. For an adiabatic turbine, it doesn't imply any heat transfer so it's said that: $\dot{q} = 0$. From eq. (11):

$$-\dot{w}_{turbine} = h_e - h_i$$
$$\dot{w}_{turbine} = h_i - h_e \ (20)$$

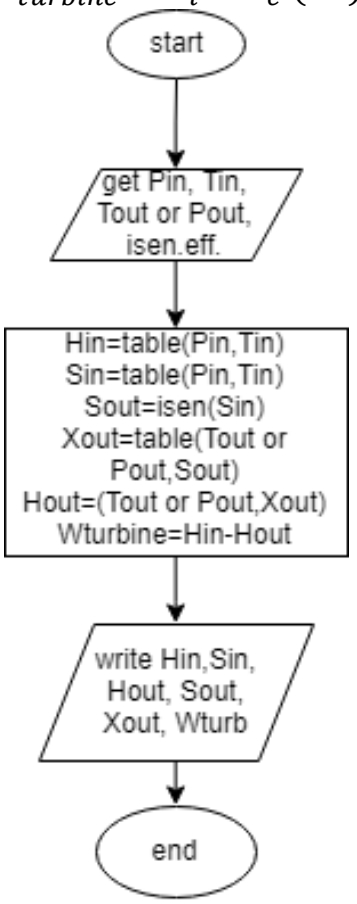


Figure 13-turbine flow chart

4) condenser: for a given input temperature, entropy and enthalpy from the turbine output, and for a known output steam quality (x=0). The output entropy and enthalpy is dedicated by interpolation and given for the pump as input where the cycle is repeated. For a condenser, it doesn't imply any mechanical work so it's said that: $\dot{w} = 0$. From eq. (11):

$$\dot{q}_{condenser} = h_e - h_i \ (-ve\ sign)(21)$$

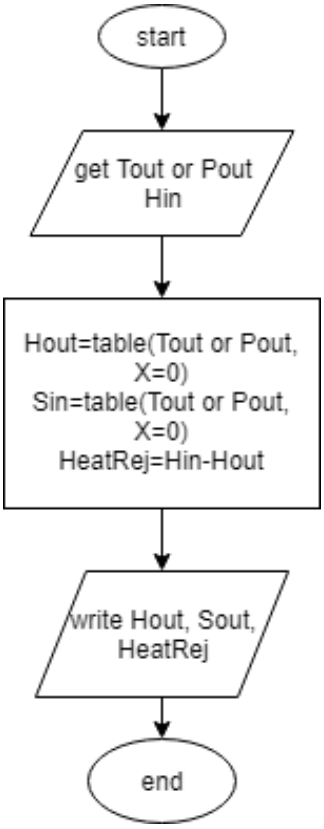


Figure 14-condenser flow chart

5) calculations for the whole cycle:

$$\dot{w}_{net} = \dot{w}_{turbine} - \left|\dot{w}_{pump}\right| \quad (22)$$

From eq. (14)

$$\eta_{thermal} = \frac{\dot{w}_{net}}{\dot{q}_{boiler}}$$

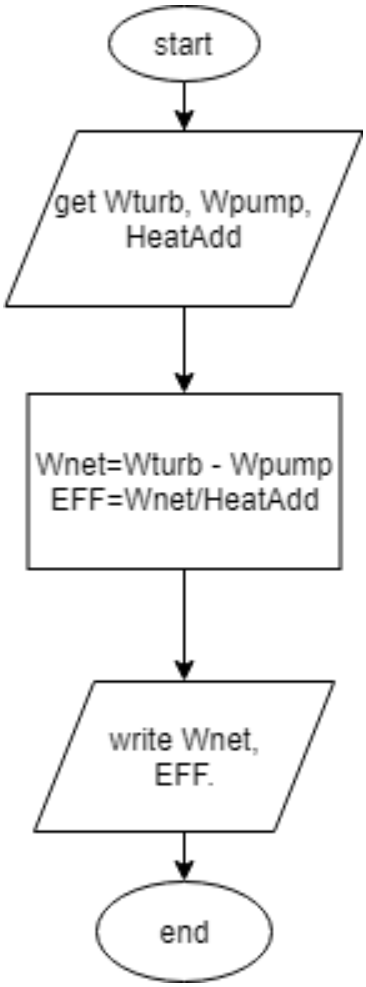


Figure 15-cycle calculation flow chart

11. Data description

- Simple Rankine cycle (figure (16))

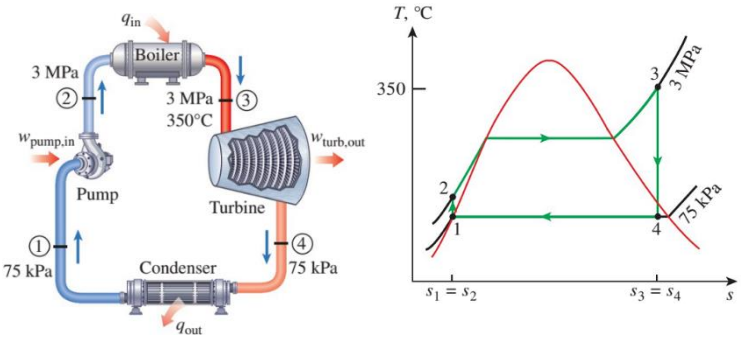


Figure 16-simple Rankine cycle example

	TURBINE		CONDENSER		PUMP		BOILER	
	IN	OUT	IN	OUT	IN	OUT	IN	OUT
P(bar)	30	0.75	0.75	0.75	0.75	30	30	30
T(C)	350	-	-	-	-	91.96	91.96	350
h(kJ/kg)	3115	2402	2402	384.3	384.3	387.4	387.4	3115
s(kJ/kgK)	6.742	6.724	6.724	1.213	1.213	1.213	1.213	6.742
Quality	-	-	0.8856	0	0	-	-	-
$\dot{w}$(kW)	713		-		-3.1		-	
$\dot{q}$(kW)	-		-2,017.7		-		2,727.6	
$\eta_{ther.}$	26.14%							

The inputs in black, the outputs in red

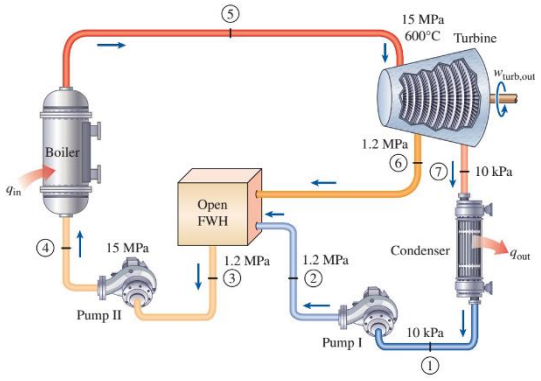
Manual calculation time is 9 minutes.

The next table will conclude what the code actually generated.

Code	TURBINE		CONDENSER		PUMP		BOILER	
	IN	OUT	IN	OUT	IN	OUT	IN	OUT
P(bar)	30	0.75	0.75	0.75	0.75	30	30	30
T(C)	350	-	-	-	-	91.96	92	350
h(kJ/kg)	3115	2401.8	2401.8	384.3	384.3	387.4	387.6	3115
s(kJ/kgK)	6.742	6.742	6.742	1.213	1.213	1.213	1.213	6.742
Quality	-	-	0.8856	0	0	-	-	-
$\dot{w}$(kW)	713.215		-		-3.3		-	
$\dot{q}$(kW)	-		-2,017.5		-		2,727.4	
$\eta_{ther.}$	26.03%							

The performance error is approximately 0.4226%.

- Regenerative Rankine cycle (figure (17))



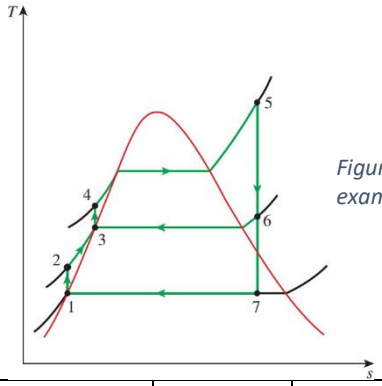


Figure 17-regenerative Rankine cycle example

		P(bar)	T(C)	h(kJ/kg)	s(kJ/kgK)	Quality	$\dot{w}$(kW)	$\dot{q}$(kW)	$\eta_{ther.}$
Turbine	IN	150	600	3581	6.677	-	1,297.9	-	46.32 %
	OUT	0.1	-	2114	6.677	0.8038			
Condenser	IN	0.1	-	2114	6.677	0.8038	-	-1,485.4	
	OUT	0.1	-	191.7	0.6489	0			
Pump1	IN	0.1	-	191.7	0.6489	0	-0.927	-	
	OUT	12	45.83	192.9	0.6489	-			y(bleed mass fraction) (kg)
OFW	IN1 Bleed	12	-	2858	6.677	-	-	-	
	IN2 Pump1	12	45.83	192.9	0.6489	-			0.2273
	OUT	12	-	798.6	2.216	0			
Pump2	IN	12	-	798.6	2.216	0	-15.4	-	
	OUT	150	190	814	2.216	-			
boiler	IN	150	190	814	2.216	-	-	2,767	
	OUT	150	600	3581	6.677	-			

Manual calculation time is 21 minutes.

The next table will conclude what the code actually generated.

code		P(bar)	T(C)	h(kJ/kg)	s(kJ/kgK)	Quality	$\dot{w}$(kW)	$\dot{q}$(kW)	$\eta_{ther.}$
Turbine	IN	150	600	3581.5	6.6775	-	1,298	-	46.32 %
	OUT	0.1	-	2114.6	6.6775	0.8038			
Condenser	IN	0.1	-	2114.6	6.6775	0.8038	-	-1,486	
	OUT	0.1	-	191.7	0.6489	0			
Pump1	IN	0.1	-	191.7	0.6489	0	-1.012	-	
	OUT	12	45.8676	193.13	0.6489	-			y(bleed mass fraction) (kg)
OFW	IN1 Bleed	12	-	2858	6.6775	-	-	-	
	IN2 Pump1	12	45.8676	193.13	0.6489	-			0.2272
	OUT	12	-	798.6	2.216	0			
Pump2	IN	12	-	798.6	2.216	0	-15.27	-	
	OUT	150	189.947	813.87	2.216	-			
boiler	IN	150	189.947	813.87	2.216	-	-	2,767.6	
	OUT	150	600	3581.5	6.6775	-			

the performance error is approximately zero percent.

12. Analysis of the results

Manual Calculations requires very long time to use steam tables every single time a state properties are needed. That's beside the application of numerous laws and equations where a human errors are of high probability to occur whether while using steam tables or applying equations. Every type of steam power cycle differs from another in some steps and that's makes manual solving harder.

Meanwhile, solving thermodynamic with the assist of Thermodynamic cycle solver (TCS) enables the user to identify sufficient input for solving and (TCS) does the rest. The only time consumed is the time needed to identify the inputs. There is no room for human error as all the processes are done automatically and engineered to avoid the common errors. The errors in parameters' values are very small and can be ignored. (TCS) has undergone many example tests to assess the performance. The obtained result that the errors are negligible and sometimes there is no error at all for some parameters. (TCS) is available for different types of steam power cycle.

13. Conclusion

The application of MATLAB and all coding programs in the real life issues is essential, time and effort saving and more accurate. (TCS) allows the users to study the feasibility, practicality, achievability and reasonableness of steam power cycles. That's indispensable for an application that we cannot do without such as steam power cycles. One more advantage of programming the application is that you can try change different inputs and noticing the change of performance and net work. To operate (TCS), the user needs to identify the type of steam power cycle (simple Rankine cycle, reheat Rankine cycle and regenerative Rankine cycle) and some specific inputs. (TCS) then will calculate the properties of each state in the cycle, heat transfer, mechanical work and the thermal efficiency related to the cycle. To sum up, for time and effort saving and accurate performance, (TCS generated with MATLAB) is a perfect choice.

14. Appendix (MATLAB code)

- Functions

Enthalpy calculations

```
function [ H ] = Enthalpy(P,T,h,x)
%Enthalpy takes in steam table values of enthalpy and an input of T and P
%and outputs the value of enthalpy
%%This line finds the value of S in the known values of the table
    z = zeros(1,2);
    y = zeros(1,2);
    H = h(find(T==x(2)),find(P==x(1)));
    % This if condition checks if the value was found in the table or not
    % If it is found then we can proceed if not then there has to be
    % interpolation
    if isempty(H);
        [~,po] = min(abs(P-x(1)));
        [~,To] = min(abs(T-x(2)));
        z(1) = P(po);%rounded P1
        y(1) = T(To);%rounded T1
        if (x(1)-z(1) > 0);
            z(2) = P(po+1);
        else if (x(1) - z(1) < 0);
                 z(1) = P(po+1);
                 z(2) = P(po);
            else z(2) = z(1);
            end
        end
        if (x(2)- y(1) > 0);
            y(2) = T(To+1);
        else if ((x(2) - y(1) < 0));
                y(1) = T(To+1);
                y(2) = T(To);
        else y(2) = y(1);
            end
        end
        %Mathematical expression
            %H = (((H2-H1)*(Tx-T1))/(T2-T1))+H1
            %Tx = x(1), H1 = H(y(1),z(1))
            %H2 = H(y(2),z(2))
        %Interpolation step
        if z(2) == z(1)
            a = find(P == z(1));
            b = find(T == y(2));
            c = find(T == y(1));
            H = (((h(b,a)-h(c,a))*(x(2)-T(c)))/(T(b)-T(c)))+h(c,a);
            H = abs(H);
        end
        if y(2) == y(1)
            a = find (T == y(1));
            b = find (P == z(1));
            c = find (P == z(2));
            H = (((h(a,c)-h(a,b))*(x(1)-P(b)))/(P(c)-P(b)))+ h(a,b);
            H = abs(H);
        end
```

```
        if z(2)~= z(1) && y(2) ~= y(1)
            a = find (P == z(1));
            b = find (T == y(2));
            c = find (T == y(1));
            d = find (P == z(2));
            H(1) = (((h(b,a)-h(c,a))*(x(2)-T(c)))/(T(b)-T(c)))+h(c,a);
            H(2) = (((h(b,d)-h(c,d))*(x(2)-T(c)))/(T(b)-T(c)))+h(c,d);
            H(3) = (((H(2)-H(1))*(x(1)-z(1)))/(z(2)-z(1)))+ H(1);
            H(1) = [];
            H(1) = [];
            H = abs(H);
        end
    end
end
```

Entropy calculation

```
function  S = Entropy( P,T,s,x)
    %This line finds the value of S in the known values of the table
    z = zeros(1,2);
    y = zeros(1,2);
    S = s(find(T==x(2)),find(P==x(1))); %#ok<FNDSB>
    % This if condition checks if the value was found in the table or not
    % If it is found then we can proceed if not then there has to be
    % interpolation
    if isempty(S);
        [~,po] = min(abs(P-x(1)));
        [~,To] = min(abs(T-x(2)));
        z(1) = P(po);%rounded P1
        y(1) = T(To);%rounded T1
        if (x(1)-z(1) > 0);
            z(2) = P(po+1);
        else if (x(1) - z(1) < 0);
                 z(1) = P(po-1);
                 z(2) = P(po);
            else z(2) = z(1);
            end
        end
        if (x(2)- y(1) > 0);
            y(2) = T(To+1);
        else if ((x(2) - y(1) < 0));
                y(1) = T(To-1);
                y(2) = T(To);
        else y(2) = y(1);
            end
        end
        %Mathematical expression
            %S = (((S2-S1)*(Tx-T1))/(T2-T1))+S1
            %Tx = x(1), S1 = S(y(1),z(1))
            %S2 = S(y(2),z(2))
        %Interpolation step
        if z(2) == z(1)
            a = find(P == z(1));
            b = find(T == y(2));
```

```
            c = find(T == y(1));
            S = (((s(b,a)-s(c,a))*(x(2)-T(c)))/(T(b)-T(c)))+s(c,a);
        end
        if y(2) == y(1)
            a = find (T == y(1));
            b = find (P == z(1));
            c = find (P == z(2));
            S = (((s(a,c)-s(a,b))*(x(1)-P(b)))/(P(c)-P(b)))+ s(a,b);
        end
        if z(2)~= z(1) && y(2) ~= y(1)
            a = find (P == z(1));
            b = find (T == y(2));
            c = find (T == y(1));
            d = find (P == z(2));
            S(1) = (((s(b,a)-s(c,a))*(x(2)-T(c)))/(T(b)-T(c)))+s(c,a);
            S(2) = (((s(b,d)-s(c,d))*(x(2)-T(c)))/(T(b)-T(c)))+s(c,d);
            S(3) = (((S(2)-S(1))*(x(1)-z(1)))/(z(2)-z(1)))+ S(1);
            S(1) = [];
            S(1) = [];
        end
    end
end
```

Wet region enthalpy calculation

```
function hwet = wetenthalpy( Type,TorP,Qsatout,hsat,Phsat,Psat)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
if isequal(Type,1)
Tsatout=TorP;
Tsat=0:1:100;
i=1;
while (Tsatout>Tsat(i))
    i=i+1;
end
m=(Tsatout-Tsat(i-1))/(Tsat(i)-Tsat(i-1));
hwet=(m*(hsat(i,1)-hsat(i-1,1))+hsat(i-1,1))+Qsatout*(m*(hsat(i,2)-hsat(i-
1,2))+hsat(i-1,2));
else
    Psatout=TorP;
    i=1;
while (Psatout>Psat(i))
    i=i+1;
end
m=(Psatout-Psat(i+1))/(Psat(i)-Psat(i+1));
hwet = (m*(Phsat(i,1)-Phsat(i-1,1))+Phsat(i-1,1))+Qsatout*(m*(Phsat(i,2)-
Phsat(i-1,2))+Phsat(i-1,2));
end
end
```

Wet region entropy calculation

```
function swet = wetentropy( Type,TorP,Qsatout,ssat,Pssat,Psat)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
if isequal(Type,1)
Tsatout=TorP;
Tsat=[0:1:100];
i=1;
while (Tsatout>Tsat(i)) i=i+1; end
m=(Tsatout-Tsat(i-1))/(Tsat(i)-Tsat(i-1));
swet=(m*(ssat(i,1)-ssat(i-1,1))+ssat(i-1,1))+Qsatout*(m*(ssat(i,2)-ssat(i-
1,2))+ssat(i-1,2));
else
    Psatout=TorP;
    i=1;
while (Psatout>Psat(i)) i=i+1; end
m=(Psatout-Psat(i-1))/(Psat(i)-Psat(i-1));
swet=(m*(Pssat(i,1)-Pssat(i-1,1))+Pssat(i-1,1))+Qsatout*(m*(Pssat(i,2)-
Pssat(i-1,2))+Pssat(i-1,2));
end
end
```

Wet region calculation

```
function Q = entq(Type, TorP,sout,ssat,Psat,Pssat )
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
if isequal(Type,1)
    Tsatout=TorP;
    Tsat=[0:1:100];
    i=1;
    while (Tsatout>Tsat(i)) i=i+1; end
    m=(Tsatout-Tsat(i-1))/(Tsat(i)-Tsat(i-1));
    Q=(sout-(m*(ssat(i,1)-ssat(i-1,1))+ssat(i-1,1)))/(m*(ssat(i,2)-ssat(i-
1,2))+ssat(i-1,2));
else
    Psatout=TorP;
    i=1;
    while (Psatout>Psat(i)) i=i+1; end
    m=(Psatout-Psat(i-1))/(Psat(i)-Psat(i-1));
    Q=(sout-(m*(Pssat(i,1)-Pssat(i-1,1))+Pssat(i-1,1)))/(m*(Pssat(i,2)-
Pssat(i-1,2))+Pssat(i-1,2));
end
end
```

Turbine

```
function [hin,sin,hout,sout,Q] = Turbine(
Pturbin,Tturbin,Type,TorP,P,T,h,s,Pssat,hsat,Psat,Phsat,ssat)
%Turbine fuction simulates an isentropic turbine work
% the inputs needed are:
% turbine inlet pressure (Pturbin)...turbine inlet
% Temperatu(Tturbine)....Type of output info('P' for pressure,'T' for
% Temperature)....value of output info(TorP)
% the rest inputs are just tables
x=[Pturbin, Tturbin];
hin = Enthalpy(P,T,h,x);
sin = Entropy(P,T,s,x);
sout=sin;
[~,p1] = min(abs(P-TorP));
if isequal(Type,2) && (sout > Pssat(p1,3));
    si = ((x(1) - P(p1))/(P(p1+1) - P(p1)))*(s(1:200, p1+1)-s(1:200, p1)) +
s(1:200, p1); %interpolates between 2 pressure values and creates an array
    [~,s1] = min(abs(si-x1(2))); %gets nearest 2 entropy values
    T1 = (sout-si(s1))/(si(s1+1)-si(s1))*(T(s1+1)-T(s1)) + T(s1);
%interpolates for T
    z1 = [x1(1), T1]; %To use Enthalpy funciton
    hout = Enthalpy(P,T,h,z1);
else
    Q = entq(Type,TorP,sout,ssat,Psat,Pssat);
    hout = wetenthalpy( Type,TorP,Q,hsat,Phsat,Psat);
end
end
```

Condenser

```
function [hcout,scout]=condenser(Type,TorP,hsat,ssat,Psat,Pssat,Phsat)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
hcout=wetenthalpy( Type,TorP,0,hsat,Phsat,Psat);
scout= wetentropy( Type,TorP,0,ssat,Pssat,Psat);
end
```

Pump

```
function [hpout,spout,Tpumpout] = pump(scout,Pturbin,P,s,h,T)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
spout=scout;
i=1;
while(Pturbin>=P(i))
    i=i+1;
end
```

```
m=(Pturbin-P(i-1))/(P(i)-P(i-1));
S = zeros(1,200);
for counter=1:200
    S(counter)=m*(s(counter,i)-s(counter,i-1))+s(counter,i-1);
end
j=1;
while(scout >= S(j))
    j=j+1;
end
n=(scout-S(j-1))/(S(j)-S(j-1));
Tpumpout=10*(j-1+n);
hpout=Enthalpy(P,T,h,[Pturbin, Tpumpout]);

end
```

Boiler

```
function [hinturbine,sinturbine,Hgained] =
boiler(Pturbin,Tturbin,hpout,P,T,h,s)
x=[Pturbin, Tturbin];
hinturbine=Enthalpy(P,T,h,x);
sinturbine=Entropy(P,T,s,x);
Hgained=hinturbine-hpout;
end
```

Reheat function

```
function [Sout, Hout] = ReHeat(P,T,h,s,x)
    Hout = Enthalpy(P,T,h,x);
    Sout = Entropy(P,T,s,x);
end
```

Regenerative function

```
function [Hout,Sout,y] = OpenFWH(P,T,h,s,Phsat,Pssat,Psat,x1,x2,x3)
        x1(2) = x1(1); % These 4 lines are because I'm too lazy/scared to
change the variable names
        x1(1) = x3;    % They basically allow me to just enter entropy values
in x1 and x2
        x2(2) = x2(1); % and enter the regeneration pressure in x3
        x2(1) = x3;
        %To get H1
        [~,p1] = min(abs(P-x1(1))); %gets nearest 2 pressure values
```

```
        si = ((x3 - P(p1))/(P(p1+1) - P(p1)))*(s(1:200, p1+1)-s(1:200, p1)) +
s(1:200, p1); %interpolates between 2 pressure values and creates an array up
to 380C since it is always subcooled
        [~,s1] = min(abs(si-x1(2))); %gets nearest 2 entropy values
        T1 = ((x1(2)-si(s1))/(si(s1+1)-si(s1)))*(T(s1+1)-T(s1)) + T(s1);
%interpolates for T
        z1 = [x3, T1]; %To use Enthalpy funciton
        H1 = Enthalpy(P,T,h,z1) ;
        %To get H2
        [~,p2] = min(abs(P-x2(1))); %gets nearest 2 pressure values
        si = ((x3 - P(p2))/(P(p2+1) - P(p2)))*(s(1:200, p2+1)-s(1:200, p2)) +
s(1:200, p2); %interpolates between 2 pressure values and creates an array
        [~,s2] = min(abs(si-x2(2))); %gets nearest 2 entropy values
        T2 = ((x2(2)-si(s2))/(si(s2+1)-si(s2)))*(T(s2+1)-T(s2)) + T(s2);
%interpolates for T
        z2 = [x2(1), T2]; %To use Enthalpy funciton
        H2 = Enthalpy(P,T,h,z2);
        %To get H3
        [~,p3] = min(abs(P-x1(1)));
        Hout = wetenthalpy(2,P(p3),0,0,Phsat,Psat);
        Sout = wetentropy(2,P(p3),0,0,Pssat,Psat);
        %To get y
        y = (Hout - H1)/(H2 - H1);
end
```

Cycle calculations

```
function [Qin, Wout] = QnW(C,H, y)
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here
    if (C == 2)
        Qin = H(1) - H(4);
        Wout = H(1) - H(2); end
    if (C == 3)
        Qin = H(1) - H(6) + H(3) - H(2);
        Wout = H(1) - H(2) + H(3) - H(4);
    end
    if (C == 4)
        Qin = H(1) - H(6);
        Wout = y*(H(1)-H(7))+(1-y)*(H(1) - H(2));
    end
end
```

- GUI operation function

```
function varargout = untitled1(varargin)
% UNTITLED1 MATLAB code for untitled1.fig
%      UNTITLED1, by itself, creates a new UNTITLED1 or raises the existing
%      singleton*.
%
%      H = UNTITLED1 returns the handle to a new UNTITLED1 or the handle to
%      the existing singleton*.
%
%      UNTITLED1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in UNTITLED1.M with the given input arguments.
%
%      UNTITLED1('Property','Value',...) creates a new UNTITLED1 or raises
the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before untitled1_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to untitled1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help untitled1

% Last Modified by GUIDE v2.5 26-Apr-2019 16:44:48

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @untitled1_OpeningFcn, ...
                   'gui_OutputFcn',  @untitled1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before untitled1 is made visible.


function untitled1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to untitled1 (see VARARGIN)

% Choose default command line output for untitled1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
        set(handles.group1,'visible','off')
        set(handles.group2,'visible','off')
        set(handles.group3,'visible','off')
        set(handles.group4,'visible','off')
        image0(hObject , handles)
        set(handles.text9,'string','The instructions of each cyle and how to
input data will appear here')

% UIWAIT makes untitled1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = untitled1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function radio_off(handles)
set(handles.pradio,'value',0)
set(handles.tradio,'value',0)

function image0(hObject , handles)

handles.output = hObject;
 axes(handles.axes3)
    handles.gambar=imread('blizzard-01.png');
    image(handles.gambar)
    axis off


function image1(hObject , handles)

handles.output = hObject;
 axes(handles.axes3)
    handles.gambar=imread('Simple Rankine.png');
    image(handles.gambar)
    axis off

function image2(hObject , handles)
```

```
handles.output = hObject;
 axes(handles.axes3)
    handles.gambar=imread('Reheat Rankine.png');
    image(handles.gambar)
    axis off

function image3(hObject , handles)

handles.output = hObject;
 axes(handles.axes3)
    handles.gambar=imread('Regeneration Rankine.png');
    image(handles.gambar)
    axis off

% --- Executes on selection change in popup.
function popup_Callback(hObject, eventdata, handles)
% hObject    handle to popup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global a;
    a = get(hObject,'value');
    radio_off(handles)

switch a
    case 1
        set(handles.group1,'visible','off')
        set(handles.group2,'visible','off')
        set(handles.group3,'visible','off')
        set(handles.group4,'visible','off')
        image0(hObject , handles)
        set(handles.text9,'string','The instructions of each cyle and how to
input data will appear here')
    case 2
        set(handles.group1,'visible','on')
        set(handles.group4,'visible','on')
        set(handles.group2,'visible','off')
        set(handles.group3,'visible','off')
        image1(hObject , handles)
        set(handles.text9,'string','State (1) is the turbine Input,state (2)
is the condenser input')
    case 3
        set(handles.group1,'visible','on')
        set(handles.group2,'visible','on')
        set(handles.group4,'visible','on')
        set(handles.group3,'visible','off')
        image2(hObject , handles)
        set(handles.text9,'string','State (1) is the turbine Input,state (3)
is the Reheat Date,Stat (4)is the condenser input')
    case 4
        set(handles.group1,'visible','on')
        set(handles.group4,'visible','on')
        set(handles.group2,'visible','off')
        set(handles.group3,'visible','on')
        image3(hObject , handles)
```

```
        set(handles.text9,'string','State (1) is the turbine Input,State
(2)is the condenser input,State (7)is the Regeneration Data')
end




% Hints: contents = cellstr(get(hObject,'String')) returns popup contents as
cell array
%        contents{get(hObject,'Value')} returns selected item from popup


% --- Executes during object creation, after setting all properties.
function popup_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Regp_Callback(hObject, eventdata, handles)
% hObject    handle to Regp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Regp as text
%        str2double(get(hObject,'String')) returns contents of Regp as a
double


% --- Executes during object creation, after setting all properties.
function Regp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Regp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Rp_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to Rp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Rp as text
%        str2double(get(hObject,'String')) returns contents of Rp as a double


% --- Executes during object creation, after setting all properties.
function Rp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Rp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Rt_Callback(hObject, eventdata, handles)
% hObject    handle to Rt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Rt as text
%        str2double(get(hObject,'String')) returns contents of Rt as a double


% --- Executes during object creation, after setting all properties.
function Rt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Rt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Pt_Callback(hObject, eventdata, handles)
% hObject    handle to Pt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Pt as text
%        str2double(get(hObject,'String')) returns contents of Pt as a double
```

```
% --- Executes during object creation, after setting all properties.
function Pt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Pt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Tt_Callback(hObject, eventdata, handles)
% hObject    handle to Tt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Tt as text
%        str2double(get(hObject,'String')) returns contents of Tt as a double


% --- Executes during object creation, after setting all properties.
function Tt_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Tt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function pe_Callback(hObject, eventdata, handles)
% hObject    handle to pe (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of pe as text
%        str2double(get(hObject,'String')) returns contents of pe as a double


% --- Executes during object creation, after setting all properties.
function pe_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pe (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in solve.
function solve_Callback(hObject, eventdata, handles)
% hObject    handle to solve (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
h = evalin('base', 'h');
hsat = evalin('base', 'hsat');
P = evalin('base', 'P');
Phsat = evalin('base', 'Phsat');
Psat = evalin('base', 'Psat');
Pssat = evalin('base', 'Pssat');
s = evalin('base', 's');
ssat = evalin('base', 'ssat');
T = evalin('base', 'T');
Tsat = evalin('base', 'Tsat');
global Type;
global a;
if a == 2
    p1=str2double(get(handles.Pt,'string'));
    t1=str2double(get(handles.Tt,'string'));
    pt2=str2double(get(handles.pe,'string'));
    [h1,s1,h2,s2,x] =
Turbine(p1,t1,Type,pt2,P,T,h,s,Pssat,hsat,Psat,Phsat,ssat);
    [h3,s3]=condenser(Type,pt2,hsat,ssat,Psat,Pssat,Phsat);
    [h4,s4,t4] = pump(s3,p1,P,s,h,T);
    wt=abs(h1-h2);
    wp=abs(h4-h3);
    wnet=wt-wp;
    q=abs(h1-h4);
    eff=wnet/q;
    H=[h1,h2,h3,h4]';
    S=[s1,s2,s3,s4]';
    y =[H,S];
    uit = uitable('Data',y,'Position',[435 240 296 265]);
    uit.ColumnName = {'H','S'};
    set(handles.Work,'string',wnet);
    set(handles.Qin,'string',q);
    set(handles.Eff,'string',eff);
else if a==3
    p3=str2double(get(handles.Pt,'string'));
    t3=str2double(get(handles.Tt,'string'));
    pt6=str2double(get(handles.pe,'string'));
    p5=str2double(get(handles.Rp,'string'));
    t5=str2double(get(handles.Rt,'string'));
    [h5,s5,h6,s6] =
Turbine(p5,t5,Type,pt6,P,T,h,s,Pssat,hsat,Psat,Phsat,ssat);
```

```
    [h1,s1]=condenser(Type,pt6,hsat,ssat,Psat,Pssat,Phsat);
    Type = 2;
    [h3,s3,h4,s4,x] = Turbine(p3,t3,2,p5,P,T,h,s,Pssat,hsat,Psat,Phsat,ssat);
    [h2,s2,t2] = pump(s1,p3,P,s,h,T);
    wt=abs(h3-h4)+abs(h5-h6);
    wp=abs(h2-h1);
    wnet=wt-wp;
    q=abs(h5-h4)+ abs(h3-h2);
    eff=wnet/q;
    H=[h1,h2,h3,h4,h5,h6]';
    S=[s1,s2,s3,s4,s5,s6]';
    y =[H,S];
    uit = uitable('Data',y,'Position',[435 240 296 265]);
    uit.ColumnName = {'H','S'};
    set(handles.Work,'string',wnet);
    set(handles.Qin,'string',q);
    set(handles.Eff,'string',eff);
    else if a==4
    p1=str2double(get(handles.Pt,'string'));
    t1=str2double(get(handles.Tt,'string'));
    pt2=str2double(get(handles.pe,'string'));
    p7=str2double(get(handles.Regp,'string'));
    [~,~,h7,s7,x] = Turbine(p1,t1,2,p7,P,T,h,s,Pssat,hsat,Psat,Phsat,ssat);
    [h1,s1,h2,s2,x] =
Turbine(p1,t1,Type,pt2,P,T,h,s,Pssat,hsat,Psat,Phsat,ssat);
    [h3,s3]=condenser(Type,pt2,hsat,ssat,Psat,Pssat,Phsat);
    [h4,s4,t4] = pump(s3,p7,P,s,h,T);
    [h5,s5,y] = OpenFWH(P,T,h,s,Phsat,Pssat,Psat,s3,s2,p7);
    [h6,s6,t6] = pump(s5,p1,P,s,h,T);
    wnet = y*abs(h7-h1) + (1-y)*abs(h1-h2) - (1-y)*abs(h4-h3) - abs(h5-h6);
    q = abs(h1-h6);
    eff = wnet/q;
    set(handles.Work,'string',wnet);
    set(handles.Qin,'string',q);
    set(handles.Eff,'string',eff);
    H=[h1,h2,h3,h4,h5,h6,h7]';
    S=[s1,s2,s3,s4,s5,s6,s7]';
    z =[H,S];
    uit = uitable('Data',z,'Position',[435 240 296 265]);
    uit.ColumnName = {'H','S'};
        end
    end
end



function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pradio.
function pradio_Callback(hObject, eventdata, handles)
% hObject    handle to pradio (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Type;
Type = 2;
radio_off(handles)
set(handles.pradio,'value',1)




% Hint: get(hObject,'Value') returns toggle state of pradio


% --- Executes on button press in tradio.
function tradio_Callback(hObject, eventdata, handles)
% hObject    handle to tradio (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of tradio
global Type;
Type = 1;
radio_off(handles)
set(handles.tradio,'value',1)


% --- Executes when entered data in editable cell(s) in uitable1.
function uitable1_CellEditCallback(hObject, eventdata, handles)
% hObject    handle to uitable1 (see GCBO)
% eventdata  structure with the following fields (see
MATLAB.UI.CONTROL.TABLE)
%   Indices: row and column indices of the cell(s) edited
%   PreviousData: previous data for the cell(s) edited
%   EditData: string(s) entered by the user
%   NewData: EditData or its converted form set on the Data property. Empty
if Data was not changed
```

```
%   Error: error string when failed to convert EditData to appropriate value
for Data
% handles    structure with handles and user data (see GUIDATA)

function varargout = tu(varargin)
% TU MATLAB code for tu.fig
%      TU, by itself, creates a new TU or raises the existing
%      singleton*.
%
%      H = TU returns the handle to a new TU or the handle to
%      the existing singleton*.
%
%      TU('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in TU.M with the given input arguments.
%
%      TU('Property','Value',...) creates a new TU or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before tu_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to tu_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help tu

% Last Modified by GUIDE v2.5 26-Apr-2019 15:20:51

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @tu_OpeningFcn, ...
                   'gui_OutputFcn',  @tu_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before tu is made visible.
function tu_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% varargin   command line arguments to tu (see VARARGIN)

% Choose default command line output for tu
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes tu wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = tu_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
h = evalin('base', 'h');
hsat = evalin('base', 'hsat');
P = evalin('base', 'P');
Phsat = evalin('base', 'Psat');
Psat = evalin('base', 'Psat');
Pssat = evalin('base', 'Pssat');
s = evalin('base', 's');
ssat = evalin('base', 'ssat');
T = evalin('base', 'T');
Tsat = evalin('base', 'Tsat');
p=str2double(get(handles.edit1,'string'));
t=str2double(get(handles.edit2,'string'));
ty=str2double(get(handles.edit3,'string'));
t2=str2double(get(handles.edit4,'string'));
[a,b,c,d]=Turbine( p,t,ty,t2,P,T,h,s,Pssat,hsat,Psat,Phsat,ssat);
set(handles.edit5,'string',a);
set(handles.edit6,'string',b);
set(handles.edit7,'string',c);
set(handles.edit8,'string',d);



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a
double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a
double


% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a
double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a
double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

15. References

1. Thermodynamics: An engineering approach, 8th edition. Yunus Cengel & Michael A. Boles. Published by McGraw-Hill Education, 2 Penn Plaza, New York, NY 10121. Copyright © 2015 by McGraw-Hill Education.
2. Fundamentals of thermodynamics, 7th edition. Claus Borgnakke & Richard E. Sonntag, University of Michigan. Copyright © 2009 John Wiley & Sons, Inc.
3. www.wikipedia.com
4. www.mathworks.com
5. www.youtube.com/MATLAB
6. www.youtube.com/TutorialSchool