Aula 7 – Introdução à Programação Orientada a Objetos
Parte 2 - Aplicação na Engenharia e Ciência Térmicas

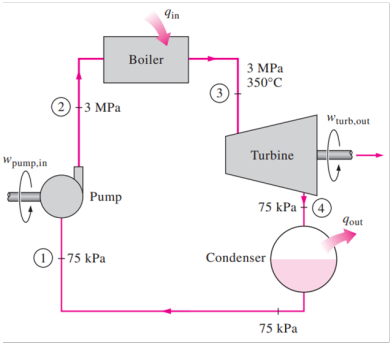
EMC410235 - Programação Científica para Engenharia e Ciência Térmicas

Prof. Rafael F. L. de Cerqueira

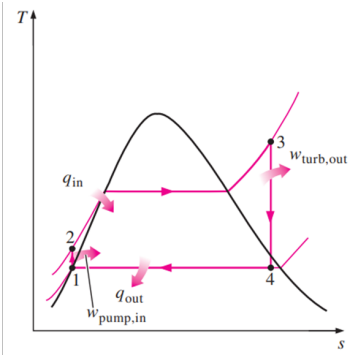
2025.2

Ciclo Rankine

- Ciclo termodinâmico ideal usado na geração de energia a partir de vapor.



- Fluido de trabalho: tipicamente água.
- Objetivo: converter calor em trabalho com o máximo de eficiência possível.



Equações fundamentais do ciclo

Conservação da massa (regime permanente):

$$\sum \dot{m}_{\text{entrada}} = \sum \dot{m}_{\text{saída}}$$

Para uma entrada e uma saída:

$$\dot{m}_{\text{in}} = \dot{m}_{\text{out}} = \dot{m}$$

Primeira Lei da Termodinâmica:

$$\dot{Q}_{\text{in}} - \dot{Q}_{\text{out}} + \dot{W}_{\text{in}} - \dot{W}_{\text{out}} + \sum_{\text{in}} \dot{m}h - \sum_{\text{out}} \dot{m}h = 0$$

$$\dot{Q}_{\text{liq}} - \dot{W}_{\text{liq}} = \dot{m}(h_{\text{out}} - h_{\text{in}})$$

Aplicações típicas:

- Turbina: $\dot{W}_{\text{turb}} = \dot{m}(h_{\text{entrada}} - h_{\text{saída}})$
- Bomba: $\dot{W}_{\text{bomba}} = \dot{m}(h_{\text{saída}} - h_{\text{entrada}})$

Eficiência dos componentes

Eficiência isentrópica da turbina:

$$\eta_{\text{turb}} = \frac{h_3 - h_4}{h_3 - h_{4s}}$$

Eficiência isentrópica da bomba:

$$\eta_{\text{bomba}} = \frac{h_{2s} - h_1}{h_2 - h_1}$$

Eficiência térmica do ciclo Rankine:

$$\eta_{\text{térmica}} = \frac{\dot{W}_{\text{liq}}}{\dot{Q}_{\text{in}}} = \frac{(h_3 - h_4) - (h_2 - h_1)}{h_3 - h_2}$$

Introdução ao CoolProp

- Biblioteca open-source para propriedades termodinâmicas de fluidos.
- Suporte a água, refrigerantes, gases industriais, entre outros (incluindo misturas!).
- Baseada em equações de estado precisas.
- Integração com Python via `from CoolProp.CoolProp import PropsSI`.
- `conda install conda-forge::coolprop`

Exemplo de uso:

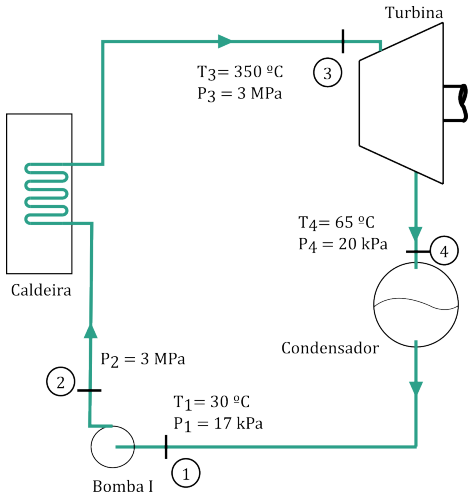
Calculando entalpia da água

```
from CoolProp.CoolProp import PropsSI

h = PropsSI('H', 'P', 1e6, 'T', 300, 'Water')
print(h) # entalpia em J/kg
```

Estados 1 e 2 – Bomba

Entrada da bomba (1) e saída (2)

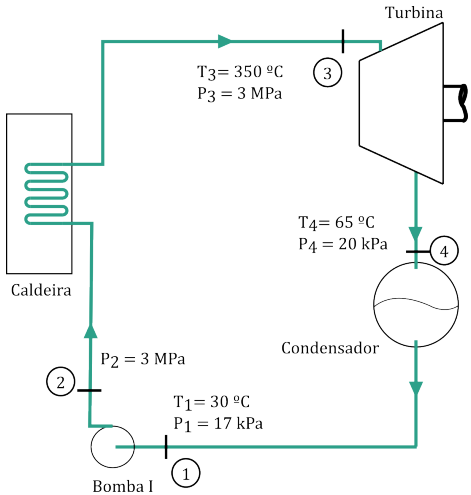
```
from CoolProp.CoolProp import PropsSI

# Estado 1 - entrada da bomba
P1 = 17e3           # Pa
T1 = 30 + 273.15    # K (30°C)
h1 = PropsSI('H', 'P', P1, 'T', T1, 'Water')
s1 = PropsSI('S', 'P', P1, 'T', T1, 'Water')

# Estado 2 - saída da bomba
P2 = 7e6            # Pa
T2 = T1
h2 = PropsSI('H', 'P', P2, 'T', T2, 'Water')
s2 = PropsSI('S', 'P', P2, 'T', T2, 'Water')
```



Estados 3 e 4 – Turbina

Entrada da turbina (3) e saída (4)

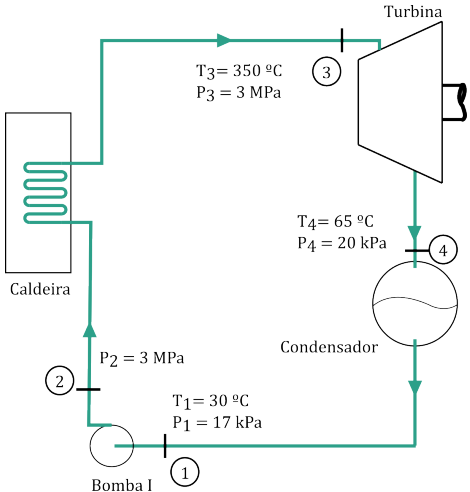
```
# Estado 3 - entrada da turbina
P3 = 3e6              # Pa
T3 = 350 + 273.15     # K (500°C)
h3 = PropsSI('H', 'P', P3, 'T', T3, 'Water')
s3 = PropsSI('S', 'P', P3, 'T', T3, 'Water')

# Estado 4 - saída da turbina
P4 = 20e3             # Pa
T4 = 65 + 273.15      # K (65°C)
h4 = PropsSI('H', 'P', P4, 'T', T4, 'Water')
s4 = PropsSI('S', 'P', P4, 'T', T4, 'Water')
```



Cálculos energéticos – Ciclo Rankine

Exemplo com 1 kg/s de água

```
mdot = 1.0  # kg/s

# Trabalho da bomba (ideal)
W_bomba = mdot * (h2 - h1)
W_bomba_alt = mdot * (1/999.0)(P2 - P1)

# Calor recebido na caldeira
Q_in = mdot * (h3 - h2)

# Trabalho útil da turbina (real)
W_turbina = mdot * (h3 - h4)

# Calor rejeitado no condensador
Q_out = mdot * (h4 - h1)

# Eficiência térmica
eta = (W_turbina - W_bomba) / Q_in
```



Arquitetura orientada a objetos da simulação

- O ciclo é composto por quatro tipos de componentes:
  - `Pump`, `Heater`, `Turbine`, `Condenser`
- Todos herdam da classe `Equipment`, que implementa a base comum.
- Os pontos de entrada e saída são objetos `ThermoProperty`, que encapsulam as propriedades termodinâmicas.
- Conexões entre equipamentos são feitas por objetos `Connector`.
- O ciclo completo é gerenciado por `ThermoCycle`, que organiza, inicializa e executa os cálculos.

Organização de Diretórios

- `app_aula.py` – script principal
- `src/` – diretório principal do código
  - `connector.py` – define conectores
  - `thermoProperty.py` – propriedades termodinâmicas
  - `thermoCycle.py` – gerenciamento do ciclo
  - `equipment/` – equipamentos do ciclo
    - `equipment.py` – classe base
    - `pump.py`, `turbine.py`, `heater.py`, `condenser.py`

Classe Connector

```
class Connector():
def __init__(self, name):
self.name = name
self.property_in = None
self.property_out = None
```

- Representa conexões entre equipamentos.
- Armazena propriedades termodinâmicas de entrada e saída.
- Método `calculate()` propaga o cálculo.

Exemplo: Turbine

```
class Turbine(Equipment):
  def calculate(self):
    self.work = - self.enthalpy_balance
```

- Calcula o trabalho gerado a partir do balanço de entalpia.

Classe ThermoProperty

```
    def calculate(self):
      if self.P and self.T:
        self.H = PropsSI('H', 'P', self.P, 'T', self.T, self.fluid)
        self.S = PropsSI('S', 'P', self.P, 'T', self.T, self.fluid)
```

- Calcula propriedades usando CoolProp.
- Armazena P, T, H, S, Q, `mass_flow_rate`.

Classe ThermoCycle

```
    def calculate_efficiency(self):
      W_liq = 0.0
      Q_in = 0.0
      for equipment in self.equipments:
        if isinstance(equipment, Turbine):
          W_liq += equipment.work
        elif isinstance(equipment, Heater):
          Q_in += equipment.heat
        self.efficiency = W_liq / Q_in
```

- Gerencia o ciclo, conecta, calcula e avalia eficiência.

Script Principal: app_aula.py

```
prop_1 = ThermoProperty("point_1", "Water")
prop_1.set_pressure(17e3)
prop_1.set_temperature(30)
...
pump = Pump("pump")
pump.add_connectors_in(pipe_3)
pump.add_connectors_out(pipe_4)
```

- Define pontos, conectores, equipamentos.
- Executa a simulação com `ThermoCycle()`.

Execução do Ciclo

```
rankine_power_cycle = ThermoCycle()
rankine_power_cycle.add_equipment(pump)
...
rankine_power_cycle.initialize()
rankine_power_cycle.calculate()
rankine_power_cycle.calculate_efficiency()
```

Ciclo: Saída Gráfica

- O método `draw` do `ThermoCycle` desenha e gera uma saída do ciclo montado usando a biblioteca `networkx`.
  `conda install networkx`

```
import networkx as nx
...

def draw(self, output_file):
G = nx.DiGraph()

for connector in self.connectors:
G.add_edge(connector.equipment_in.name,
connector.equipment_out.name,
label=connector.name)

plt.figure(figsize=(10, 8))
pos = nx.circular_layout(G)
```

```
nx.draw(G, pos, with_labels=True,
node_size=3000, node_color='lightblue',
arrows=True, font_size=10)

edge_labels = nx.get_edge_attributes(G, 'label
')
nx.draw_networkx_edge_labels(G, pos,
edge_labels=edge_labels,
font_color='red')

plt.savefig(output_file, dpi=300)
plt.close()
```