Prof. C. Naaktgeboren, PhD
https://github.com/CNThermSci/ApplThermSci

B04 – Ciclos de Refrigeração

01 – Ciclos de Refrigeração por Compressão de Vapor de Simples Estágio

Exemplo `B0401-01` – Ciclo de Refrigeração por Compressão de Vapor de Simples Estágio

Original.

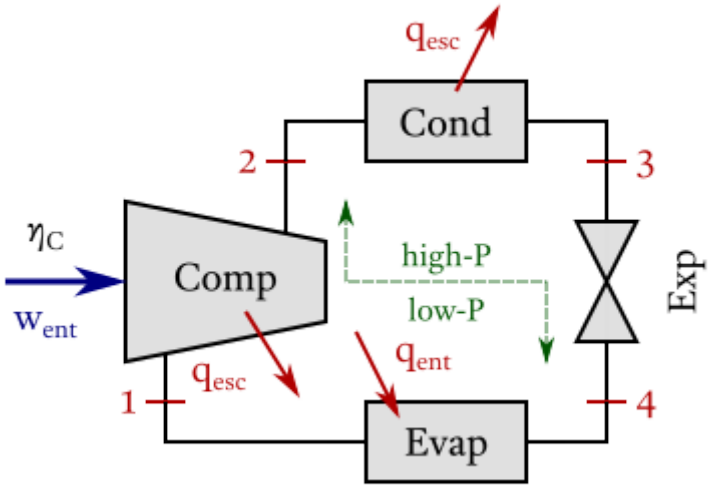
```
prob = Dict(
    :WC =>    1.0:  0.25:    3.0,   # Taxa de trabalho, kW
    :ηC =>   75.0:  5.00:   90.0,   # Eficiência isentrópica, %
    :IC =>   50.0:  5.00:   75.0,   # Irrev. perdida no compressor, %
    :Tc =>   60.0:  5.00:   80.0,   # Temperatura de condensação, °C
    :Te => -20.0:  5.00:   -5.0,   # Temperatura de evaporação, °C
);
```

Recompute

```
Dict(
  :IC ⇒ 70.0
  :Te ⇒ -15.0
  :WC ⇒ 2.75
  :ηC ⇒ 80.0
  :Tc ⇒ 70.0
)
```

Enunciado:

Um ciclo de refrigeração por compressão de vapor, ilustrado abaixo, opera com entrada de potência de **2.75kW** no compressor, o qual possui eficiência isentrópica de **80.0%** e perde **70.0%** da taxa de irreversibilidade na forma de calor para o meio, conforme indicado. A temperatura de condensação é de **70.0°C** e a de evaporação é de **-15.0°C**. Determine, considerando o emprego do **R134a**:



(a) A vazão mássica de refrigerante, em kg/s

(b) A taxa de rejeição de calor (no condensador), em kW

(c) A capacidade de refrigeração, em ton

(d) O COP do refrigerador, em %

Resolução

Escreve-se uma função que resolve o ciclo, utilizando **CoolProp** via **Pycall.jl** para propriedades termofísicas.

```
solve (generic function with 1 method)
```

```
function solve(
        WC,          # Potência de compresssão, (kW)
        ηC,          # Eficiência isentrópica, (norm)
        IC,          # Fração de perda de irrev. por calor, (norm)
        Tc,          # Temp. de condensação (K)
        Te;          # Temp. de evaporação (K)
        FL="R134a"   # Fluido refrigerante (CoolProp name)
    )
    # Cycle States
    St1 = CP.State(FL, Dict("T" => Te, "Q" => 1)) # All T's in K
    St3 = CP.State(FL, Dict("T" => Tc, "Q" => 0))
    S2s = CP.State(FL, Dict("P" => St3.p, "S" => St1.s))
    wCs = S2s.h - St1.h # Isentropic compressor work
    wCr = wCs / ηC      # ηC normalized
    IrC = wCr - wCs     # Irreversibility, normalized
    qCs = IC * IrC      # Compressor heat loss
    h_2 = St1.h + wCr - qCs     # Energy balance
    St2 = CP.State(FL, Dict("P" => St3.p, "H" => h_2))
    St4 = CP.State(FL, Dict("P" => St1.p, "H" => St3.h))
    # Quantities of interest
    md  = WC / wCr
    q23 = St2.h - St3.h
    q41 = St1.h - St4.h
    COP = q41 / wCr
    return (md, md * q23, md * q41, COP * 1.0e+2)
end
```

(a) A vazão mássica de refrigerante é de **0.04115 kg/s**

(b) A taxa de rejeição de calor (no condensador) é de **5.877 kW**

(c) A capacidade de refrigeração é de **0.9987 ton** (= 3.512 kW)

(d) O COP do refrigerador é de **127.7%**

```
begin
    A, B, C, D = solve(
        the[:WC],
        the[:ηC] / 1.0e+2,
        the[:IC] / 1.0e+2,
        the[:Tc] + 273.15,
        the[:Te] + 273.15,
        FL = "R134a"
    )
    Markdown.parse(
        @sprintf """
**(a)** A vazão mássica de refrigerante é de **%.4g kg/s**

**(b)** A taxa de rejeição de calor (no condensador) é de **%.4g kW**

**(c)** A capacidade de refrigeração é de **%.4g ton** (= %.4g kW)

**(d)** O COP do refrigerador é de **%.4g%%**
    """ A B C/3.517 C D
    )
end
```

Bibliotecas e Demais Recursos

Bibliotecas

```
begin
    using PlutoUI
    using PyCall
    CP = pyimport("CoolProp.CoolProp")
    using Printf
end
```