Gasturbine

December 3, 2017

1 Alternatieve Toets TDY2

Calculationsheet for a gasturbine
note: The cell below is needed to import libraries and set global settings.

```
In [1]: %matplotlib inline
        %precision 3
        from IPython.display import Latex
        from math import nan
        import matplotlib.pyplot as plt
        from GasTurbine import *
```

2 Decription of the system

2.1 States

The following states are identified:
0. Air at atmospheric conditions $P_0 = P_{atm}$, $T_0 = T_{atm}$ and $v_0 = v_{atm}$ 1. Compressed air $P_1 >> P_0$, $T_1 > T_0$ and $v_1 << v_0$ 2. Heated air $P_2 = P_1$, $T_2 >> T_1$ and $v_2 > v_1$ 3. Heated air at atmospheric pressure $P_3 = P_0 = P_{atm}$, $T_3 < T_2$

2.2 Transitions

The system transitions are identified as follows:
0 -> 1 **Isentropic compression** were air is sucked into a compressor.
1 -> 2 **Isobaric heating** where air is heated in a combustion chamber.
2 -> 3 **Isentropic expansion** where air is expanded till atmospheric pressure.
3 -> 0 **Isobaric cooling** A fictive step where the air is cooled to atmorspheric conditions.

3 Know values

```
In [2]: P_atm = 1.013 * u.bar
        T_atm = (15. * u.degC).to(u.K)
        c = Cycle()
        c.c_p = 1005. * u.J / (u.kg * u.K)
        c.c_v = 716. * u.J / (u.kg * u.K)
        c.Q_in = 2000 * u.kW
```

```
In [3]: s = States(4)
        s[0].P = P_atm
        s[0].T = T_atm
        s[0].A = 1. * u.m ** 2
        s[0].V_flux = 6000. * u.m ** 3 / u.hr
        s[1].P = 10. * u.bar
        s[1].A = 0.5 * u.m ** 2
        s[2].A = 0.1 * u.m ** 2
        s[2].P = s[1].P
        s[3].A = 1. * u.m ** 2
        s[3].P = s[0].P
```

```
In [4]: c.t = s
        c.t[0].processtype = ProcessType.ISENTROPIC
        c.t[1].processtype = ProcessType.ISOBARIC
        c.t[2].processtype = ProcessType.ISENTROPIC
        c.t[3].processtype = ProcessType.ISOBARIC
```

4 Calculate of system constants

$$\frac{P_0 \dot{V}_0}{T_0} = \dot{m}R \rightarrow \dot{m} = \frac{P_0 \dot{V}_0}{RT_0} \tag{1}$$

$$R = c_p - c_v \tag{2}$$

$$k = \frac{c_p}{c_p} \tag{3}$$

```
In [5]: latex(c.R)
```

Out[5]:
$289.0000 \; \frac{\mathrm{J}}{(\mathrm{K} \cdot \mathrm{kg})}$

```
In [6]: latex(c.k)
```

Out[6]:
1.4036

```
In [7]: def mass_flux(P, V_flux, R, T):
            return (P * V_flux / (R * T)).to('kg/s')

        c.m_flux = mass_flux(s[0].P, s[0].V_flux, c.R, s[0].T)
        latex(c.m_flux)
```

Out[7]:
$2.0274 \; \frac{\mathrm{kg}}{\mathrm{s}}$

5 Calculate Transitions and States

5.1 State 0

$$\rho_0 = \frac{P_0}{T_0 R} \begin{cases} \frac{P_0 v_0}{T_0} = \dot{m} R \\ \rho_0 = \frac{\dot{m}}{v_0} \end{cases} \tag{4}$$

$$v_0 = \frac{\dot{V}_0}{\dot{m}} \tag{5}$$

$$c_0 A_0 \rho_0 = \dot{m} \rightarrow c_0 = \frac{\dot{m}}{\rho_0 A_0} \tag{6}$$

```
In [8]: def density(P, R, T):
            return P / (R * T)


        def specific_volume(m_flux, V_flux):
            return V_flux / m_flux


        def speed(m_flux, rho, A):
            return m_flux / (rho * A)
```

```
In [9]: s[0].rho = density(s[0].P, c.R, s[0].T)
        s[0].v = specific_volume(c.m_flux, s[0].V_flux)
        s[0].c = speed(c.m_flux, s[0].rho, s[0].A)
        s[0].print()
```

Out[9]:

$$State_0 = \begin{pmatrix} P_0 & 101300.0000 \text{ Pa} \\ T_0 & 288.1500 \text{ K} \\ v_0 & 0.8221 \frac{\text{m}^3}{\text{kg}} \\ \dot{V}_0 & 1.6667 \frac{\text{m}^3}{\text{s}} \\ \rho_0 & 1.2164 \frac{\text{kg}}{\text{m}^3} \\ c_0 & 1.6667 \frac{\text{m}}{\text{s}} \end{pmatrix} \tag{7}$$

5.2 State 1

Where ρ_1, c_1 and v_1 are determined with the previous given formula.

$$\frac{T_0^k}{P_0^{k-1}} = \frac{T_1^k}{P_1^{k-1}} \rightarrow T_1 = \left(\left(\frac{P_1}{P_0} \right)^{k-1} T_0^k \right)^{\frac{1}{k}} \tag{8}$$

$$\dot{V}_1 = c_1 A_1 \tag{9}$$

```
In [10]: @u.wraps(ret='K', args=('K', 'Pa', 'Pa', ''))
         def isentropische_dT(T_s, P_s, P_e, k):
             return ((P_e / P_s) ** (k - 1) * T_s ** k) ** (1 / k)
```

```
        def V_flux(c, A):
            return c * A
```

```
In [11]: s[1].T = isentropische_dT(s[0].T, s[0].P, s[1].P, c.k)
         s[1].rho = density(s[1].P, c.R, s[1].T)
         s[1].c = speed(c.m_flux, s[1].rho, s[1].A)
         s[1].V_flux = V_flux(s[1].c, s[1].A)
         s[1].v = specific_volume(c.m_flux, s[1].V_flux)
         s[1].print()
```

Out[11]:

$$State_1 = \begin{pmatrix} P_1 & 1000000.0000 \text{ Pa} \\ T_1 & 556.6315 \text{ K} \\ v_1 & 0.1609 \frac{\text{m}^3}{\text{kg}} \\ \dot{V}_1 & 0.3261 \frac{\text{m}^3}{\text{s}} \\ \rho_1 & 6.2163 \frac{\text{kg}}{\text{m}^3} \\ c_1 & 0.6523 \frac{\text{m}}{\text{s}} \end{pmatrix} \tag{10}$$

5.3 Transition 0 to 1

The transition from **state 0** to **state 1** is an isentropic process where work needs to be put into the system and no heat is transfered.

$$w_{0-1} = \frac{-1}{k-1}(P_1 v_1 - P_0 v_0) \tag{11}$$

$$q_{0-1} = 0. \tag{12}$$

$$\Delta u = q_{0-1} - w_{0-1} \tag{13}$$

$$w_{t,0-1} = q_{0-1} - \Delta h - \Delta e_{kin} - \Delta e_{pot} \tag{14}$$

$$\Delta e_{kin} = \frac{1}{2}(c_1^2 - c_0^2) \tag{15}$$

$$\Delta h_{0-1} = q_{0-1} - w_{t,0-1} - \Delta e_{kin} \tag{16}$$

```
In [12]: def isentropisch_work(P_s, V_s, P_e, V_e, k):
             return -1. / (k - 1) * (P_e * V_e - P_s * V_s)


         def delta_u(q, w):
             return q - w


         def delta_e_kin(c_s, c_e):
             return 0.5 * (c_e ** 2 - c_s ** 2)
```

```
def delta_h(T_s, T_e, c_p):
    return c_p * (T_e - T_s)


def technical_work(q, dh, de_kin, de_pot):
    return q - dh - de_kin - de_pot
```

In [13]:
```
c.t[0].q = 0. * u.kJ / u.kg
c.t[0].w = isentropisch_work(s[0].P, s[0].v, s[1].P, s[1].v, c.k)
c.t[0].du = delta_u(c.t[0].q, c.t[0].w)
c.t[0].de_kin = delta_e_kin(s[0].c, s[1].c)
c.t[0].dh = delta_h(s[0].T, s[1].T, c.c_p)
c.t[0].w_t = technical_work(c.t[0].q, c.t[0].dh, c.t[0].de_kin, c.t[0].de_pot)
c.t[0].print()
```

Out[13]:

$$Trans_{0->1} = \begin{pmatrix} q_{0->1} & 0.0000\ \frac{\mathrm{kJ}}{\mathrm{kg}} \\ w_{0->1} & -192.2327\ \frac{\mathrm{kJ}}{\mathrm{kg}} \\ w_{t,0->1} & -269.8227\ \frac{\mathrm{kJ}}{\mathrm{kg}} \\ \Delta u_{0->1} & 192.2327\ \frac{\mathrm{kJ}}{\mathrm{kg}} \\ \Delta e_{0->1,kin} & -0.0012\ \frac{\mathrm{kJ}}{\mathrm{kg}} \\ \Delta e_{0->1,pot} & 0.0000\ \frac{\mathrm{kJ}}{\mathrm{kg}} \\ \Delta h_{0->1} & 269.8239\ \frac{\mathrm{kJ}}{\mathrm{kg}} \end{pmatrix} \tag{17}$$

5.4 Transition 1 to 2

State 2 can best be calculated by first determining the heat and work transfer in transition 1 -> 2. Since the amount of heating energy Q_{in} is known. The transition from **state 1** to **state 2** is an isobaric heating process. Where heat is put into the system which is kept at an constant pressure.

$$q_{1-2} = \frac{Q_{in}}{\dot{m}} \tag{18}$$

$$w_{1-2} = q_{1-2} - \frac{q_{1-2}}{k} \begin{cases} q_{1-2} = \frac{k}{k-1} P(v_2 - v_1) \\ w_{1-2} = P(v_2 - v_1) \end{cases} \tag{19}$$

In [14]:
```
def isobaar_heat_Qm(Q_in, m_flux):
    return Q_in / m_flux


def isobaar_work_kq(k, q):
    return q - (q / k)
```

In [15]:
```
c.t[1].q = isobaar_heat_Qm(c.Q_in, c.m_flux)
c.t[1].w = isobaar_work_kq(k=c.k, q=c.t[1].q)
c.t[1].du = delta_u(c.t[1].q, c.t[1].w)
```

5.5 State 2

From here **state 2** can be calculated, where ρ_2, v_2, c_2 and $\dot{V}_2$ are calculated using earlier provided formulas.

$$q_{1-2} = c_p(T_2 - T_1) \rightarrow T_2 = \frac{q_{1-2}}{c_p} + T_1 \tag{20}$$

```
In [16]: def isobaar_dT(q, c_p, T_s):
             return q / c_p + T_s
```

```
In [17]: s[2].T = isobaar_dT(c.t[1].q, c.c_p, s[1].T)
         s[2].rho = density(s[2].P, c.R, s[2].T)
         s[2].c = speed(c.m_flux, s[2].rho, s[2].A)
         s[2].V_flux = V_flux(s[2].c, s[2].A)
         s[2].v = specific_volume(c.m_flux, s[2].V_flux)
         s[2].print()
```

Out[17]:

$$State_2 = \begin{pmatrix} P_2 & 1000000.0000 \text{ Pa} \\ T_2 & 1538.2036 \text{ K} \\ v_2 & 0.4445 \frac{\text{m}^3}{\text{kg}} \\ \dot{V}_2 & 0.9013 \frac{\text{m}^3}{\text{s}} \\ \rho_2 & 2.2495 \frac{\text{kg}}{\text{m}^3} \\ c_2 & 9.0127 \frac{\text{m}}{\text{s}} \end{pmatrix} \tag{21}$$

5.6 Transition 1 to 2 (continued)

From here the transition can be calculated further, using earlier given formulas.

```
In [18]: c.t[1].de_kin = delta_e_kin(s[1].c, s[2].c)
         c.t[1].dh = delta_h(s[1].T, s[2].T, c.c_p)
         c.t[1].w_t = technical_work(c.t[1].q, c.t[1].dh, c.t[1].de_kin, c.t[1].de_pot)
         c.t[1].print()
```

Out[18]:

$$Trans_{1->2} = \begin{pmatrix} q_{1->2} & 986.4800 \frac{\text{kJ}}{\text{kg}} \\ w_{1->2} & 283.6743 \frac{\text{kJ}}{\text{kg}} \\ w_{t,1->2} & -0.0404 \frac{\text{kJ}}{\text{kg}} \\ \Delta u_{1->2} & 702.8056 \frac{\text{kJ}}{\text{kg}} \\ \Delta e_{1->2,kin} & 0.0404 \frac{\text{kJ}}{\text{kg}} \\ \Delta e_{1->2,pot} & 0.0000 \frac{\text{kJ}}{\text{kg}} \\ \Delta h_{1->2} & 986.4800 \frac{\text{kJ}}{\text{kg}} \end{pmatrix} \tag{22}$$

5.7 State 3

In [19]:
```
s[3].T = isentropische_dT(s[2].T, s[2].P, s[3].P, c.k)
s[3].rho = density(s[3].P, c.R, s[3].T)
s[3].c = speed(c.m_flux, s[3].rho, s[3].A)
s[3].V_flux = V_flux(s[3].c, s[3].A)
s[3].v = specific_volume(c.m_flux, s[3].V_flux)
s[3].print()
```

Out[19]:

$$State_3 = \begin{pmatrix} P_3 & 101300.0000\ \mathrm{Pa} \\ T_3 & 796.2779\ \mathrm{K} \\ v_3 & 2.2717\ \frac{\mathrm{m}^3}{\mathrm{kg}} \\ \dot{V}_3 & 4.6057\ \frac{\mathrm{m}^3}{\mathrm{s}} \\ \rho_3 & 0.4402\ \frac{\mathrm{kg}}{\mathrm{m}^3} \\ c_3 & 4.6057\ \frac{\mathrm{m}}{\mathrm{s}} \end{pmatrix} \tag{23}$$

5.8 Transition 2 to 3

In [20]:
```
c.t[2].q = 0. * u.kJ / u.kg
c.t[2].w = isentropisch_work(s[2].P, s[2].v, s[3].P, s[3].v, c.k)
c.t[2].du = delta_u(c.t[2].q, c.t[2].w)
c.t[2].de_kin = delta_e_kin(s[2].c, s[3].c)
c.t[2].dh = delta_h(s[2].T, s[3].T, c.c_p)
c.t[2].w_t = technical_work(c.t[2].q, c.t[2].dh, c.t[2].de_kin, c.t[2].de_pot)
c.t[2].print()
```

Out[20]:

$$Trans_{2->3} = \begin{pmatrix} q_{2->3} & 0.0000\ \frac{\mathrm{kJ}}{\mathrm{kg}} \\ w_{2->3} & 531.2187\ \frac{\mathrm{kJ}}{\mathrm{kg}} \\ w_{t,2->3} & 745.6653\ \frac{\mathrm{kJ}}{\mathrm{kg}} \\ \Delta u_{2->3} & -531.2187\ \frac{\mathrm{kJ}}{\mathrm{kg}} \\ \Delta e_{2->3,kin} & -0.0300\ \frac{\mathrm{kJ}}{\mathrm{kg}} \\ \Delta e_{2->3,pot} & 0.0000\ \frac{\mathrm{kJ}}{\mathrm{kg}} \\ \Delta h_{2->3} & -745.6352\ \frac{\mathrm{kJ}}{\mathrm{kg}} \end{pmatrix} \tag{24}$$

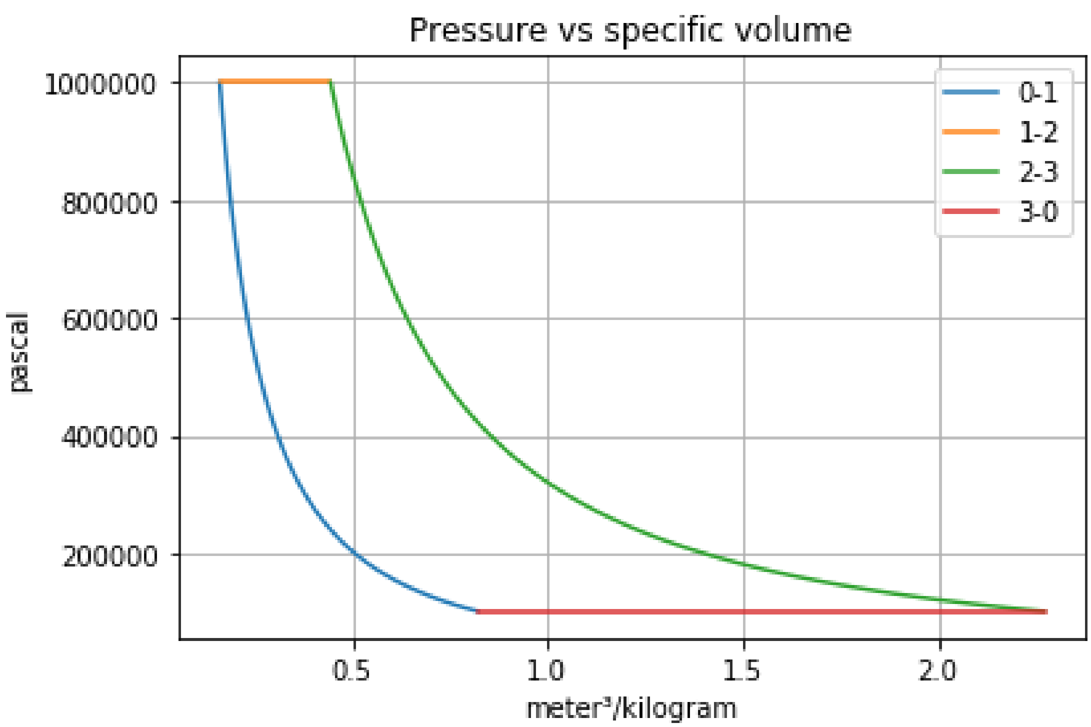
5.9 Transition 3 to 0

In [21]:
```
c.t[3].q = isobaar_heat(c_p=c.c_p, T_s=s[3].T, T_e=s[0].T)
c.t[3].w = isobaar_work(R=c.R, T_s=s[3].T, T_e=s[0].T)
c.t[3].du = delta_u(c.t[3].q, c.t[3].w)
c.t[3].de_kin = delta_e_kin(s[3].c, s[0].c)
c.t[3].dh = delta_h(s[3].T, s[0].T, c.c_p)
c.t[3].w_t = technical_work(c.t[3].q, c.t[3].dh, c.t[3].de_kin, c.t[3].de_pot)
c.t[3].print()
```

Out[21]:

$$Trans_{3->0} = \begin{pmatrix} q_{3->0} & -510.6686\ \frac{\mathrm{kJ}}{\mathrm{kg}} \\ w_{3->0} & -146.8490\ \frac{\mathrm{kJ}}{\mathrm{kg}} \\ w_{t,3->0} & 0.0092\ \frac{\mathrm{kJ}}{\mathrm{kg}} \\ \Delta u_{3->0} & -363.8196\ \frac{\mathrm{kJ}}{\mathrm{kg}} \\ \Delta e_{3->0,kin} & -0.0092\ \frac{\mathrm{kJ}}{\mathrm{kg}} \\ \Delta e_{3->0,pot} & 0.0000\ \frac{\mathrm{kJ}}{\mathrm{kg}} \\ \Delta h_{3->0} & -510.6686\ \frac{\mathrm{kJ}}{\mathrm{kg}} \end{pmatrix} \tag{25}$$

In [22]: c.plot_Pv()



In [23]: c.print_closed()

Out[23]:

	q	Delta u	w
0-1	0.000 kJ/kg	192.233 kJ/kg	-192.233 kJ/kg
1-2	986.480 kJ/kg	702.806 kJ/kg	283.674 kJ/kg
2-3	0.000 kJ/kg	-531.219 kJ/kg	531.219 kJ/kg
3-0	-510.669 kJ/kg	-363.820 kJ/kg	-146.849 kJ/kg
Sigma	475.811 kJ/kg	-0.000 kJ/kg	475.811 kJ/kg

In [24]: c.print_open()

Out[24]:

	q	Delta h	w_t	Delta e_{kin}	Delta e_{pot}
0-1	0.000 kJ/kg	269.824 kJ/kg	-269.823 kJ/kg	-0.001 kJ/kg	0.000 kJ/kg
1-2	986.480 kJ/kg	986.480 kJ/kg	-0.040 kJ/kg	0.040 kJ/kg	0.000 kJ/kg
2-3	0.000 kJ/kg	-745.635 kJ/kg	745.665 kJ/kg	-0.030 kJ/kg	0.000 kJ/kg
3-0	-510.669 kJ/kg	-510.669 kJ/kg	0.009 kJ/kg	-0.009 kJ/kg	0.000 kJ/kg
Sigma	475.811 kJ/kg	0.000 kJ/kg	475.811 kJ/kg	-0.000 kJ/kg	0.000 kJ/kg