

Atmospheric Model

August 24, 2020

fnc.py - Atmospheric Model block

The goal of this file is explaining each step that was taken towards implementing the US Standard Atmosphere 1976 in Python. In addition, the script written for testing is shown and detailed.

```
[8]: %% Script information
# Name: fnc.py
# Authors: Trajectory Team (Matias Pellegrini, Pablo Lobo)
# Owner: LIA Aerospace
#
%% Script description
#
# The aim of this module is defining functions to be used in the simulation.
#
%% Packages
import numpy as np
import c as c

%% Atmospheric properties

# This block implements the US Standard Atmosphere 1976 model.
# As of 24Aug2020, only the 0-76km portion of the model is implemented.
```

Function #1: layer

The aim of this function is to define which layer the vehicle is currently flying through. The implementation of this is based on Table 4 of the original document.

```
[9]: def layer(Hz: float)->float:
    # === INPUTS ===
    # Hz [m'] - Geopotential height
    # === OUTPUTS ===
    # b [adim] - Subscript of the layer
    #
    # Input control
    try:
        int(Hz)
    except ValueError:
        try:
```

```

        float(Hz)
    except ValueError:
        print("Fn: layer. Input must be a number.")
        return

    # Cases
    if Hz>=0 and Hz<11000:
        b = 0
    elif Hz>=11000 and Hz<20000:
        b = 1
    elif Hz>=20000 and Hz<32000:
        b = 2
    elif Hz>=32000 and Hz<47000:
        b = 3
    elif Hz>=47000 and Hz<51000:
        b = 4
    elif Hz>=51000 and Hz<71000:
        b = 5
    elif Hz>=71000 and Hz<84852:
        b = 6
    elif Hz==84852:
        b = 7
    else:
        print('Fn: Layer. Hz must be a value between 0 and 84852m')
        b = 'Error - Check Hz'
        return
    return b

```

Function #2: table4

The aim of this function is to define the constants provided by Table 4 given the current geometrical height at which the vehicle is.

In this case, some steps of the calculation process need to be detailed due to not being explicit in the original document. This is the case for the calculation of the multiple constant values attributed to T_{mb} and P_b for each layer of the atmosphere.

Calculation of T_{mb} for layers 0 to 7

The value for layer 0 is expressed right under eq. (23) in the original document, and is:

$$T_{M,0} = T_o = 288.15K$$

In order to calculate the T_{mb} values for layers 1 to 7, continuity conditions need to be applied at the boundary layers. This, expressed in an equation format, means the values can be calculated in the following way:

$$T_{M,b} = T_{M,b-1} + L_{M,b-1} * (H_b - H_{b-1})$$

Calculation of P_b for layers 0 to 7

The value for layer 0 is expressed right under eq. (33) in the original document, and is:

$$P_b = P_0 = 101325 N/m^2$$

In order to calculate the P_b values for layers 1 to 7, continuity conditions need to be applied at the boundary layers. This is slightly more complicated than in the case of T_{mb} , since the expression for P varies for the different layers.

For layers 1, 3, 4, 6 and 7 the expression is:

$$P_b = P_{b-1} * \left[\frac{T_{M,b-1}}{T_{M,b-1} + L_{M,b-1} * (H_b - H_{b-1})} \right] \left[\frac{g'_0 * M_0}{R^* * L_{M,b-1} * 1000} \right]$$

For layers 2 and 5 the expression is:

$$P_b = P_{b-1} * \left[\frac{-g'_0 * M_0 * (H_b - H_{b-1})}{R^* * T_{M,b-1}} \right]$$

The different constants used are:

$$g'_0 = 9.80665 \left[\frac{m^2}{s^2 m} \right]$$

- Defined at page 3 of the document.

$$M_0 = 28.9644 \left[\frac{kg}{kmol} \right]$$

- Defined at page 9 of the document, under eq. (21).

$$R^* = 0.00831432 \left[\frac{Nm}{kmol.K} \right]$$

- Defined at page 2 of the document, Table 2.

The two instances where a 1000 is added is due to the different constant values used in the equations being originally in different units (mostly km' and m').

```
[10]: def table4(z: float):
    # === INPUTS ===
    # z [m] - Geometric height
    # === OUTPUTS ===
    # b [adim]      Subscript of the layer
    # Lmb [K/km']   Molecular-scale temperature gradient (Table 4)
    # Tmb [K]       Temperature constant
    # Hb [km']      Geopotential Height of the layer (Table 4)
    # Hz [km']      Geopotential Height of the vehicle
    # Pb [N/m^2]    Pressure constant
```

```

# Input control
try:
    int(z)
except ValueError:
    try:
        float(z)
    except ValueError:
        print("Fn: table4. Input must be a number.")
        return

# The layer is defined.
ro = 6356.766 * 10**3          # [m] - Earth's radius - (Page 4)
Hz = (z*ro) / (ro + z)        # [m'] - Geopotential height of the vehicle
b = layer(Hz)                  # [adim] - Subscript of the layer
# Verifying b
if b==None:
    print('Fn: table4. Z must be a value between 0m and 85999m.')
    return
Hz = Hz*0.001                  # [km'] - Geopotential height of the vehicle
Hb_vec = np.array([0, 11, 20, 32, 47, 51, 71, 84.852])
Lmb_vec = np.array([-6.5, 0, 1, 2.8, 0, -2.8, -2, 0])
Tmb_vec = np.array([288.15, 216.65, 216.65, 228.65, 270.65, 270.65, 214.65,
↪186.946])
pb_vec = np.array([101325, 22632.06, 5474.88, 868.01, 110.90, 66.93, 3.95])
Hb = Hb_vec[b]
Lmb = Lmb_vec[b]
Tmb = Tmb_vec[b]
Pb = pb_vec[b]
return b, Lmb, Tmb, Hb, Hz, Pb

```

Function #3: tm

The aim of this function is to estimate the Tm value according to equation (23) of the US Standard Atmosphere 1976. This function gives the temperature for the range 0-76km.

```

[11]: def tm(Tmb,Lmb,HZ,Hb):
    # === INPUTS ===
    # Tmb [K]          Temperature constant
    # Lmb [K/km']      Molecular-scale temperature gradient
    # Hz [km']         Geopotential height of interest
    # Hb [km']         Geopotential Height for the particular layer (Table 4)
    # === OUTPUTS ===
    # Tm [K]           Temperature at given geopotential height Hz
    Tm = Tmb + Lmb*(Hz-Hb)    # [K] - Temperature at given geopotential
↪height Hz
    return Tm

```

Function #4: p

The aim of this function is to estimate the P value according to equation (33a 33b) of the US Standard Atmosphere 1976. This function gives the pressure for the range 0-76km.

```
[12]: def p(Tmb,Lmb,HZ,Hb,Pb):
    # === INPUTS ===
    # Tmb [K]          Temperature constant
    # Lmb [K/km']      Molecular-scale temperature gradient
    # Hz [km']         Geopotential height of interest
    # Hb [km']         Geopotential Height for the particular layer (Table 4)
    # Pb [N/m^2]       Pressure constant
    # === OUTPUTS ===
    # P [N/m^2]        Pressure at given geopotential height Hz
    # === CONSTANTS ===
    go = 9.80665          # [m^2/s^2.m] - Gravity @ SL (Page 2)
    R = 8.31432 * 10**-3  # [Nm / (kmol.K)] - Gas constant (Page 2)
    Mo = 28.9644          # [kg/kmol] - Mean Molecular Weight - (Page 9)
    if Lmb!=0:
        P = Pb*(Tmb / (Tmb + (Lmb*(Hz-Hb))))*((go*Mo)/(R*Lmb*1000))
    elif Lmb==0:
        P = Pb*np.exp((-go*Mo*(Hz-Hb))/(R*Tmb*1000))
    return P
```

Function #5: rho

The aim of this function is to estimate the density value according to equation (42) of the US Standard Atmosphere 1976. This function provides the density for the range 0-86km.

```
[13]: def rho(P,Tm):
    # === INPUTS ===
    # Tm [K]          Temperature at given geopotential height Hz
    # P [N/m^2]        Pressure at given geopotential height Hz
    # === OUTPUTS ===
    # rho [kg/m^3]     Density at given geopotential height Hz
    # === CONSTANTS ===
    R = 8.31432 * 10**-3  # [Nm / (kmol.K)] - Gas constant (Page 2)
    Mo = 28.9644          # [kg/kmol] - Mean Molecular Weight - (Page 9)
    rho = (P*Mo)/(R*Tm)   # [kg/m^3] - Density (Eq 42)
    return rho
```

Function #6: Vs

The aim of this function is to estimate the speed of sound value according to equation (50) of the US Standard Atmosphere 1976. This function provides the speed of sound for the range 0-86km. It applies only when the sound wave is a small perturbation on the ambient condition.

```
[14]: def Vs(Tm):
    # === INPUTS ===
    # Tm [K]          Temperature at given geopotential height Hz
    # === OUTPUTS ===
```

```

# Vs [m/s]      Speed of sound at given temperature Tm(Hz)
# === CONSTANTS ===
R = 8.31432 * 10**-3      # [Nm / (kmol.K)] - Gas constant (Page 2)
Mo = 28.9644             # [kg/kmol] - Mean Molecular Weight - (Page 9)
gamma = 1.4              # [adim] - Ratio of Cp/Cv
Vs = ((gamma*R*Tm)/Mo)**0.5 # [m/s] - Local speed of sound
return Vs

```

Function #7: visc

The aim of this function is to estimate the dynamic and kinematic viscosity according to equations (51) and (52) of the US Standard Atmosphere 1976. This function provides the speed of sound for the range 0-86km.

According to p10 of this standard, $T_m = T$ for the 0-80km range. From 80 to 86 the difference is very small.

```

[15]: def visc(Tm,rho):
# === INPUTS ===
# Tm [K]      Temperature at given geopotential height Hz
# rho [kg/m^3] Density at given geopotential height Hz
# === OUTPUTS ===
# dvisc [N.s/m^2]      Dynamic Viscosity
# kvisc [m^2/s]      Kinematic Viscosity
# === CONSTANTS ===
beta = 1.458*10**-6      # [kg/s.m.K^0.5] - "Constant"
S = 110.4                # [K] - Sutherland's constant
dvisc = (beta*Tm**1.5)/(Tm+S) # [N.s/m^2] - Dynamic viscosity
kvisc = dvisc/rho        # [m^2/s] - Kinematic Viscosity
return dvisc, kvisc

```