Adiabatic Flame Temperature calculation using Python and Cantera. : Skill-Lync

Skill-Lync

Adiabatic flame temperature:

The temperature of the products in adiabatic combustion of fuel without applying for any shaft work is defined as the “Adiabatic Flame Temperature”.

In a combustion process, the heat produced during the exothermic chemical reaction is released to their product and the temperature of the products is raised. There is no possibility for dissipation of the heat to the surrounding and the process will be adiabatic as there is no heat loss to the surrounding. As a result, the temperature of the products suddenly increases and it produces a flame. This will heat up the product gases in flame region and the temperature rise will be maximum. This highest temperature is known as the adiabatic flame temperature.

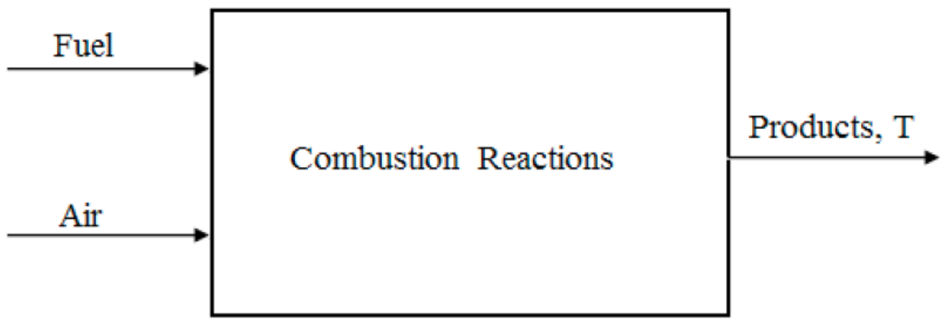
The temperature rise depends on the amount of excess air used or the air-fuel ratio. The flame temperature has the highest value for using pure oxygen gas and it decreases by using air. So, the exact stoichiometric air is to be supplied for better result. With a too large amount of excess air, the flame temperature will be reduced. When the heat loss to the environment or diluted by the inert gases and there is incomplete combustion. So, the temperature of the products will be less. The flame temperature is determined from the energy balance of the reaction at equilibrium. There are two types of adiabatic flame temperature:

1. Constant pressure adiabatic flame temperature

2. Constant volume adiabatic flame temperature.

Adiabatic flame temperature Calculation.

From the first law of thermodynamics at a control volume,



At Constant Pressure; **$\Delta H=\Delta U+p\Delta V+V\Delta p$**

Under Adiabatic conditions, $Q_p = 0$, $\Delta H = 0$

For equilibrium conditions,

$$\sum H\_reactants=\sum H\_Products$$

At Constant Pressure; **$Q_v=\Delta U$**

Under Adiabatic conditions, $Q_v = 0 = \Delta H - nRT$

For equilibrium conditions,

$$\sum H\_reactants - \sum H\_Products - R*\sum(nreactants\cdot Tstd - nProducts\cdot Tad)$$

Combustion Stoichiometry:

A stoichiometric mixture contains the exact amount of fuel and oxidizer such that after combustion is completed, all the fuel and oxidizer are consumed to form products. This ideal mixture approximately yields the maximum flame temperature, as all the energy released from combustion is used to heat the products.

For stoichiometric methane combustion with air, the balanced reaction equation reads:

$$CH_4+2\cdot(O_2+3.76\cdot N_2)=1\cdot CO_2+2\cdot H_2O+7.52\cdot N_2$$

or In general, we can express the stoichiometric equation as

$$C_xH_y + (x + y/4)(O_2 + 3.76N_2) = xCO_2 + (y/2)H_2O + (x + y/4)N_2$$

The amount of air required for combusting a stoichiometric mixture is called stoichiometric or theoretical air. The above formula is for a single-component fuel and cannot be applied to a fuel consisting of multiple components.

When the fuel is not getting completely burned, the product contains other gases also like O_2, H_2, CO, OH, etc. In practice, fuels are often combusted with an amount of air different from the stoichiometric ratio. If less air than the stoichiometric amount is used, the mixture is described as fuel-rich. If excess air is used, the mixture is described as fuel lean. The common method to describe the mixture is equivalence ratio(phi,ϕ).

The equivalence ratio is the ratio of actual fuel-air ratio to the stoichiometric fuel-air ratio. So, if ϕ<1, the mixture is considered lean and if ϕ>1, it's rich.

For Lean Mixture(ϕ<1)

$$CH4 + \frac{2}{\phi}(O2 + 3.76N2) = CO2 + 2H2O + \left(\frac{2}{\phi} - 2\right)O2 + \frac{7.52}{\phi} \cdot N2$$

For Rich Mixture(ϕ>1)

$$CH4 + \frac{2}{\phi}(O2 + 3.76N2) = \left(\frac{4}{\phi} - 3\right)CO2 + 2H2O + \left(4 - \frac{4}{\phi}\right)CO + 7.$$

In the challenge, it has been asked to perform the two types of AFT analysis with few variations. In the first case, constant volume analysis of Adiabatic flame temperature is being asked and the effect of equivalence ratio on AFT also to be performed.

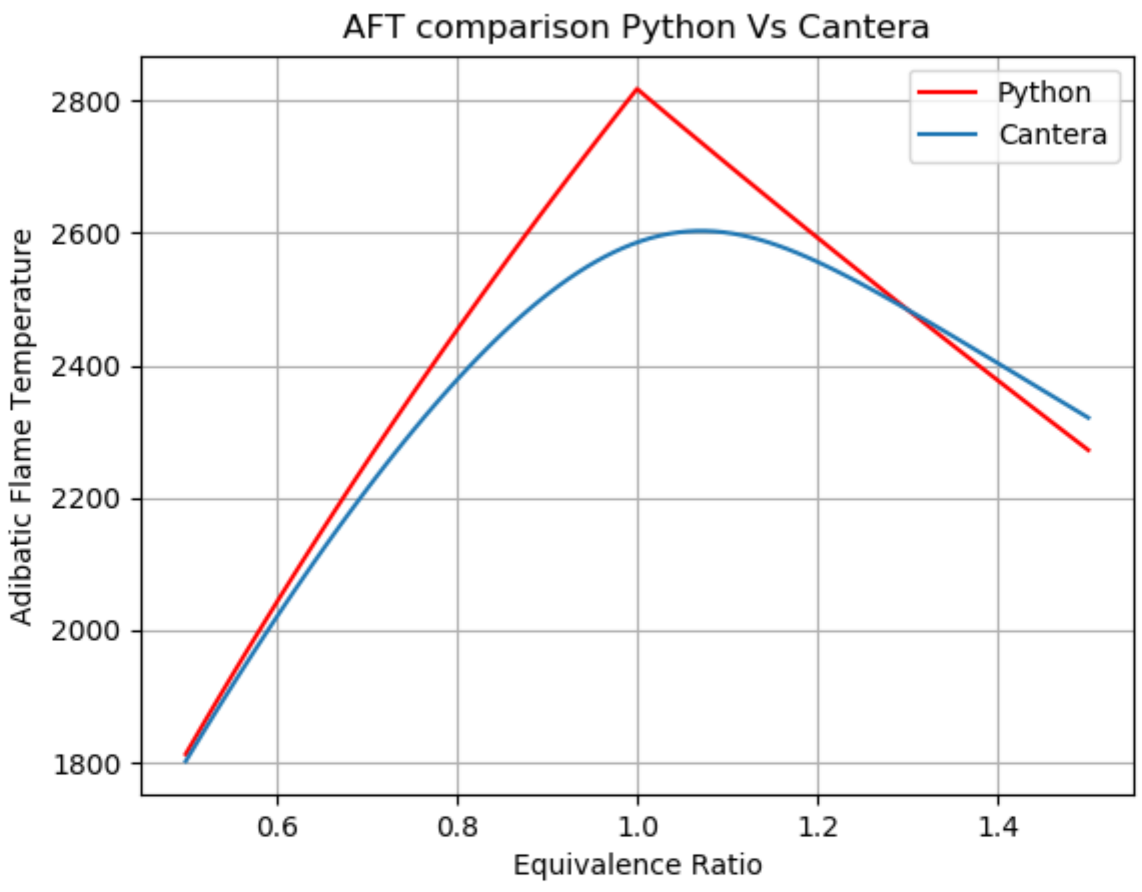
Case 1: Effect of equivalence ratio on AFT

For this case, both the python and Cantera analysis is done and presented on the same plot for comparison. The combined code for AFT for python and Cantera is:

```
"""
Code for finding out the constant volume AFT of methane for different equivalen
Equivalence ration is to be varied from 0.5 to 1.5

CH4 + 2(O2 + 3.76N2) = CO2 + 2H2O + 7.52N2
"""

import matplotlib.pyplot as plt
import math
import numpy as np
import cantera as ct

# Define the fluid: CH4
phi = np.linspace(0.5,1.5,101)


def h(T, co_effs):
        """
        Function to evaluate enthalpy
        """
        R = 8.314 #J/mol-K
        a1 = co_effs[0]
        a2 = co_effs[1]
        a3 = co_effs[2]
        a4 = co_effs[3]
        a5 = co_effs[4]
        a6 = co_effs[5]
        return (a1 + a2*T/2 + a3*pow(T,2)/3 + a4*pow(T,3)/4 + a5*pow(T,4)/5 + a


ch4_coeffs_l = [5.14987613E+00,-1.36709788E-02,4.91800599E-05,-4.84743026E-08,1
o2_coeffs_l  = [3.78245636E+00,-2.99673416E-03,9.84730201E-06,-9.68129509E-09,3
n2_coeffs_l = [0.03298677E+02,0.14082404E-02,-0.03963222E-04,0.05641515E-07,-0.
n2_coeffs_h = [0.02926640E+02,0.14879768E-02,-0.05684760E-05,0.10097038E-09,-0.
co2_coeffs_h = [3.85746029E+00,4.41437026E-03,-2.21481404E-06,5.23490188E-10,-4
h2o_coeffs_h = [3.03399249E+00,2.17691804E-03,-1.64072518E-07,-9.70419870E-11,1
o2_coeffs_h = [3.28253784E+00,1.48308754E-03,-7.57966669E-07,2.09470555E-10,-2.
co_coeffs_h = [2.71518561E+00,2.06252743E-03,-9.98825771E-07,2.30053008E-10,-2.
h2_coeffs_h = [3.33727920E+00,-4.94024731E-05,4.99456778E-07,-1.79566394E-10,2.


def F(T,phi):

        R = 8.314 #J/mol-K
        x = 1
        y = 4

        if phi<=1:


                        h_co2_p = (h(T,co2_coeffs_h))
                        h_h2o_p = (h(T,h2o_coeffs_h))
                        h_n2_p = (h(T,n2_coeffs_h))
                        h_o2_p = (h(T,o2_coeffs_h))

                        H_product = (h_co2_p + 2*h_h2o_p + ((2/phi)-2)*h_o2_p +
                        N_prod = 3 + (7.52/phi) + (2/phi-2)

                        Tstd = 298.15
                        h_ch4_r = h(Tstd,ch4_coeffs_l)
                        h_o2_r = h(Tstd,o2_coeffs_l)
                        h_n2_r = h(Tstd,n2_coeffs_l)

                        H_reactants = (h_ch4_r + ((2/phi)*(h_o2_r + 3.76*h_n2_r
                        N_reactants = 1 + 4.76*(2/phi)

        else:


                        h_co2_p = (h(T,co2_coeffs_h))
                        h_h2o_p = (h(T,h2o_coeffs_h))
                        h_n2_p = (h(T,n2_coeffs_h))
                        h_o2_p = (h(T,o2_coeffs_h))
                        h_co_p = (h(T,co_coeffs_h))


                        H_product = (((4/phi)-3)*h_co2_p + (4-(4/phi))*h_co_p +
                        N_prod = (4/phi-3) +2 + (4-4/phi) + (7.52/phi)

                        Tstd = 298.15
                        h_ch4_r = h(Tstd,ch4_coeffs_l)
                        h_o2_r = h(Tstd,o2_coeffs_l)
                        h_n2_r = h(Tstd,n2_coeffs_l)

                        H_reactants = (h_ch4_r + ((2/phi)*(h_o2_r + 3.76*h_n2_r
                        N_reactants = 1 + 4.76*(2/phi)
        return (H_reactants - H_product - R*((N_reactants*Tstd) - (N_prod*T)))

# Newton Rapson Method to solve for AFT iteratively

def fprime(T,phi):
        return(F(T+1e-6,phi) - F(T,phi))/1e-6


# Method solves for AFT till satisfying the criteria of tolerence
tol = 1e-3
alpha = 0.9
Temp=[]


for i in phi:

        T_guess = 1500
        R = 8.314 #J/mol-K

        while(abs(F(T_guess,i))> tol):
                T_guess = T_guess - alpha*((F(T_guess,i)/fprime(T_guess,i)))

        Temp.append(T_guess)

print(max(Temp))
print(Temp)

plt.figure()
plt.plot(phi,Temp,color='red')
plt.xlabel('Equivalence Ratio')
plt.ylabel('Adibatic Flame Temperature')
plt.title('AFT comparison Python Vs Cantera')

################################################################################
# Solution using Cantera

gas = ct.Solution('gri30.xml')

npoints = 100
phi = np.linspace(0.5,1.5,npoints)
tad = np.zeros(npoints)
fuel_species = 'CH4'

for i in range(npoints):

        gas.set_equivalence_ratio(phi[i], 'CH4' , 'O2:2.0, N2:7.52')
        gas.TP = 298.15,101325

        gas.equilibrate('UV','auto')

        tad[i] = gas.T

        print('At phi = {0:10.4g},    Tad = {1:10.6g}'.format(phi[i], tad[i]))


plt.plot(phi, tad)
plt.legend(('Python','Cantera'))
plt.grid()
plt.show()
```

The equivalence ratio has been varied from 0.5 to 1.5 because this is the practical range in which combustion can take place. Theoretically, it has been seen that stoichiometric mixture(ϕ = 1) can give maximum Adiabatic temperature, but practically it can change slightly.

The Plot to represent the comparative analysis of AFT is:



Observations:

1. The python code gives maximum value of AFT at ϕ = 1, it is because it does not consider all the species in the mixture.

2. The Cantera considers all the species, so the variation can be said to be practical than python code.

3. Cantera gives the maximum AFT at slightly more value of ϕ. (ϕ = 1.07).

Case 2: Constant pressure reactor with heat loss:

Case 2 contains three variations, which is to be presented separately.

Case 2.1 Effect of loss of heat on Adiabatic Flame Temperature:

The adiabatic chamber is practically impossible to maintain if the temperature gradient present is huge. So consideration of loss of heat is a must for practical calculation of AFT. So, the factor (H_loss) is considered here and its effect on AFT is performed.

$$-Q^0_{rxn,p} = \sum_i N_{i,R} \Delta \hat{h}^o_{i,R} - \sum_i N_{i,P} \Delta \hat{h}^o_{i,P} \cdot$$

$$-Q^0_{rxn,p} = \mathrm{LHV} \cdot N_{fuel} \cdot M_{fuel} = \mathrm{LHV} \cdot m_f$$

Where,

hi,R = Enthalpy of reactants.

hi,P = Enthalpy of products.

Ni,R = No. of moles of reactants.

Ni,P = No. of moles of products.

Qrnx,p = Heat loss.

LHV = Lower heating value of a fuel.

Nfuel = No. of moles of fuel.

Mfuel = Molecular weight of the fuel.

mfuel = mass of fuel.

The heat loss coefficient(h_loss) is varied from 0 to 1. It indicates that heat loss is maximum when h_loss is 1, and heat loss is minimum when h_loss is 0.

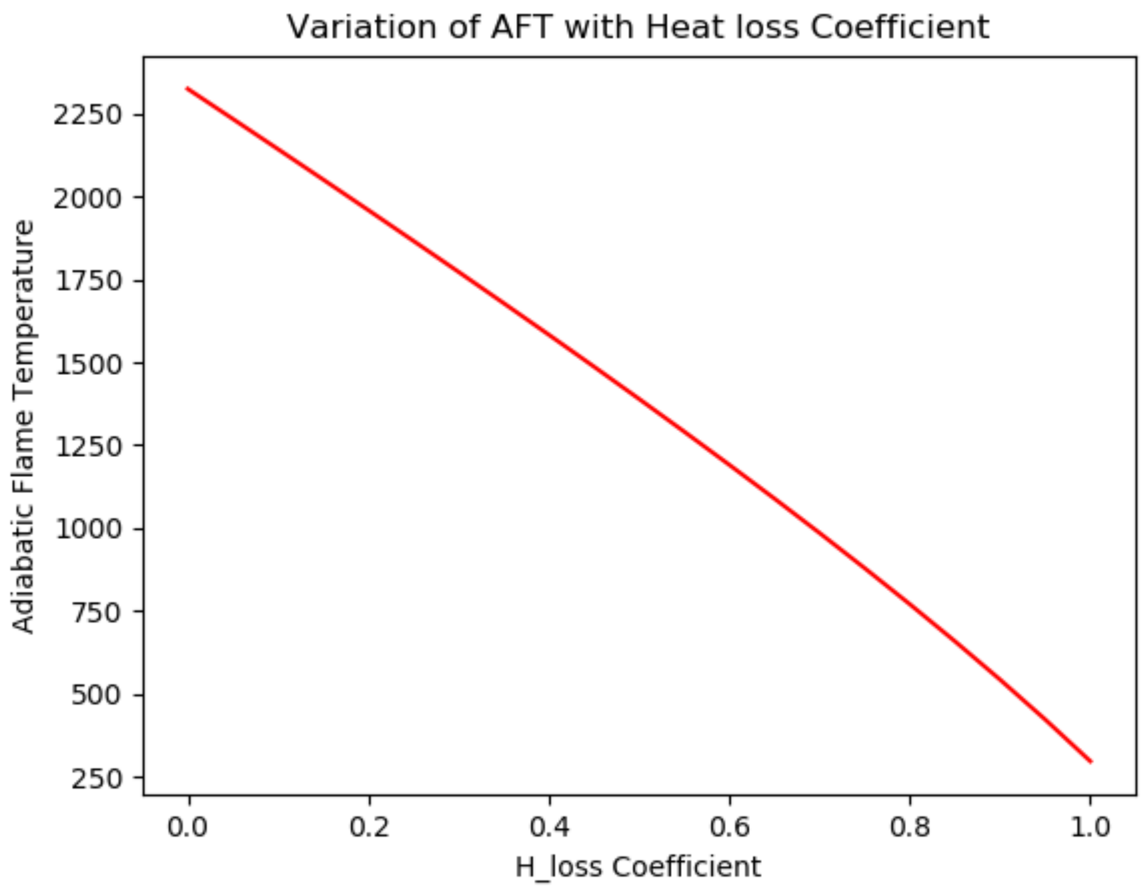
The code to see the effect of heat loss coefficient is:

```
import matplotlib.pyplot as plt
import math
import numpy as np
import cantera as ct

"""
# Inputs to the system
Fuel - CH4
Reactor - Constant Pressure
HHV - 55.5 MJ/kg
LHV - 50 MJ/kg
MW - 16.04 gram/mol
Nfuel - 1 mol
mfuel - 16.04 gram
phi = 1 Stoichiometric condition
```

```
"""
x = 1
y = 4
R = 8.314 #J/kmol-K
Qs = 802000              # As product are at gas phase Qs = LHV*mf = 50*e+6*16.
H_loss = np.linspace(0,1,21)
H_product = []
H_reactants = []

def h(T, co_effs):
        """
        Function to evaluate enthalpy
        """
        R = 8.314 #J/mol-K
        a1 = co_effs[0]
        a2 = co_effs[1]
        a3 = co_effs[2]
        a4 = co_effs[3]
        a5 = co_effs[4]
        a6 = co_effs[5]

        return (a1 + a2*T/2 + a3*pow(T,2)/3 + a4*pow(T,3)/4 + a5*pow(T,4)/5 + a

ch4_coeffs_l = [5.14987613E+00,-1.36709788E-02,4.91800599E-05,-4.84743026E-08,1
o2_coeffs_l  = [3.78245636E+00,-2.99673416E-03,9.84730201E-06,-9.68129509E-09,3
n2_coeffs_l = [0.03298677E+02,0.14082404E-02,-0.03963222E-04,0.05641515E-07,-0.
n2_coeffs_h = [0.02926640E+02,0.14879768E-02,-0.05684760E-05,0.10097038E-09,-0.
co2_coeffs_h = [3.85746029E+00,4.41437026E-03,-2.21481404E-06,5.23490188E-10,-4
h2o_coeffs_h = [3.03399249E+00,2.17691804E-03,-1.64072518E-07,-9.70419870E-11,1
o2_coeffs_h = [3.28253784E+00,1.48308754E-03,-7.57966669E-07,2.09470555E-10,-2.
co_coeffs_h = [2.71518561E+00,2.06252743E-03,-9.98825771E-07,2.30053008E-10,-2.

def f(T, H_loss):
        """
                Function to solve for Heat balance and AFT
        """

        h_co2_p = h(T,co2_coeffs_h)
        h_h2o_p = h(T,h2o_coeffs_h)
        h_n2_p  = h(T,n2_coeffs_h)

        H_product = h_co2_p + 2*h_h2o_p + 7.52*h_n2_p

        Tstd = 298.15
        h_ch4_r = h(Tstd,ch4_coeffs_l)
        h_o2_r = h(Tstd,o2_coeffs_l)
        h_n2_r = h(Tstd,n2_coeffs_l)

        H_reactants = h_ch4_r + 2*h_o2_r + 7.52*h_n2_r

        return H_product - H_reactants + (H_loss*Qs)

# Newton Rapson Method to solve for AFT iteratively

def fprime(T,H_loss):
        return(f(T+1e-6,H_loss) - f(T,H_loss))/1e-6


# Method solves for AFT till satisfying the criteria of tolerence
tol = 1e-3
```

```
alpha = 0.9
Temp=[]

# Newton Rapson Method
for i in H_loss:

        T_guess = 1500
        R = 8.314 #J/mol-K

        while(abs(f(T_guess,i))> tol):
                T_guess = T_guess - alpha*((f(T_guess,i)/fprime(T_guess,i)))
        Temp.append(T_guess)

print(Temp)
T_loss = (Temp[7])
print(T_loss)

plt.figure()
plt.plot(H_loss,Temp,color='red')
plt.xlabel('H_loss Coefficient')
plt.ylabel('Adiabatic Flame Temperature')
plt.title('Variation of AFT with Heat loss Coefficient')
plt.show()
```

The plot to represent the effect of heat loss coefficient is:



Observation:

1. It is been seen that when heat loss is zero, AFT is maximum and minimum when heat loss is maximum.

2. The AFT varies linearly with the heat loss coefficient.

3. For the heat loss coefficient of 0.35, the AFT is 1678.8364K.

Case 2.2 Effect of Fuel type(alkanes, alkynes, and alkenes) on Adiabatic Flame Temperature:

The number of carbon atoms considered here is 2. Which gives Alkane (C2H6), Alkene(C2H4), Alkyne(C2H2). The results are plotted using python as well as Cantera. The variation results are same as we obtained in the first case.

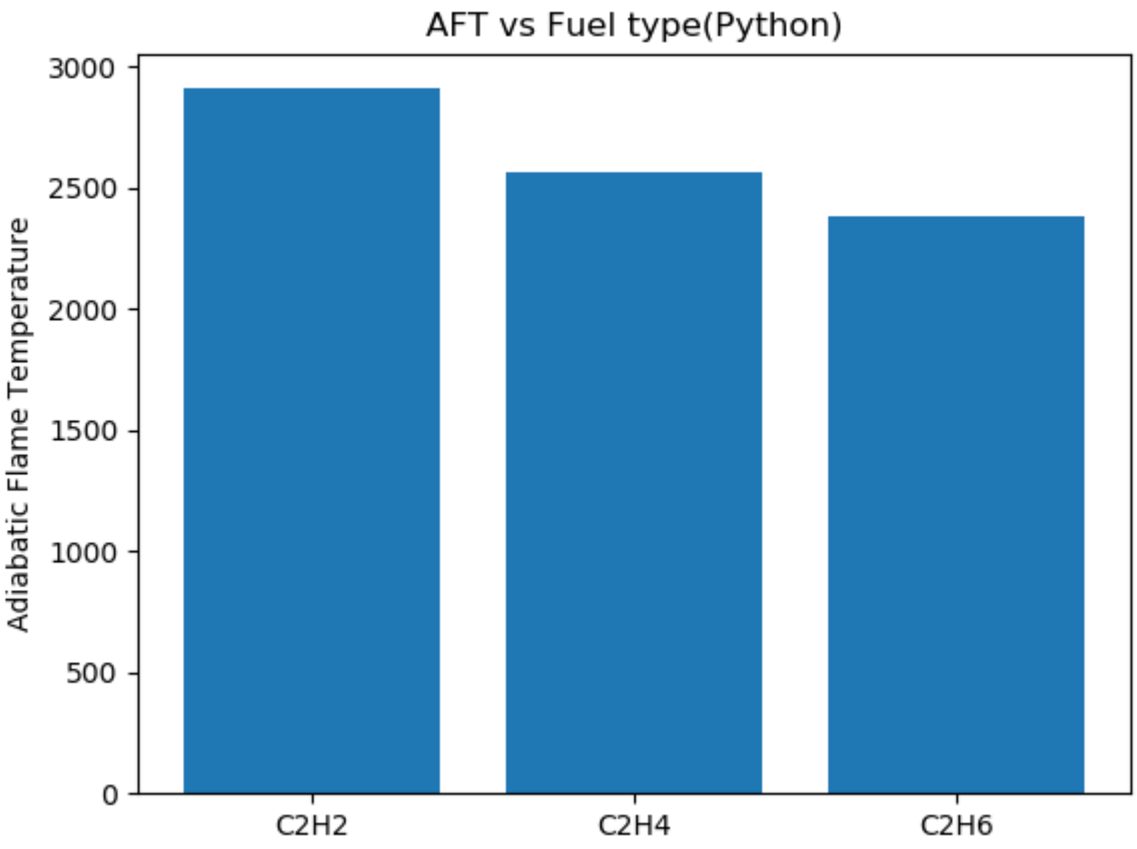
The code for effect of the fuel type on AFT is:

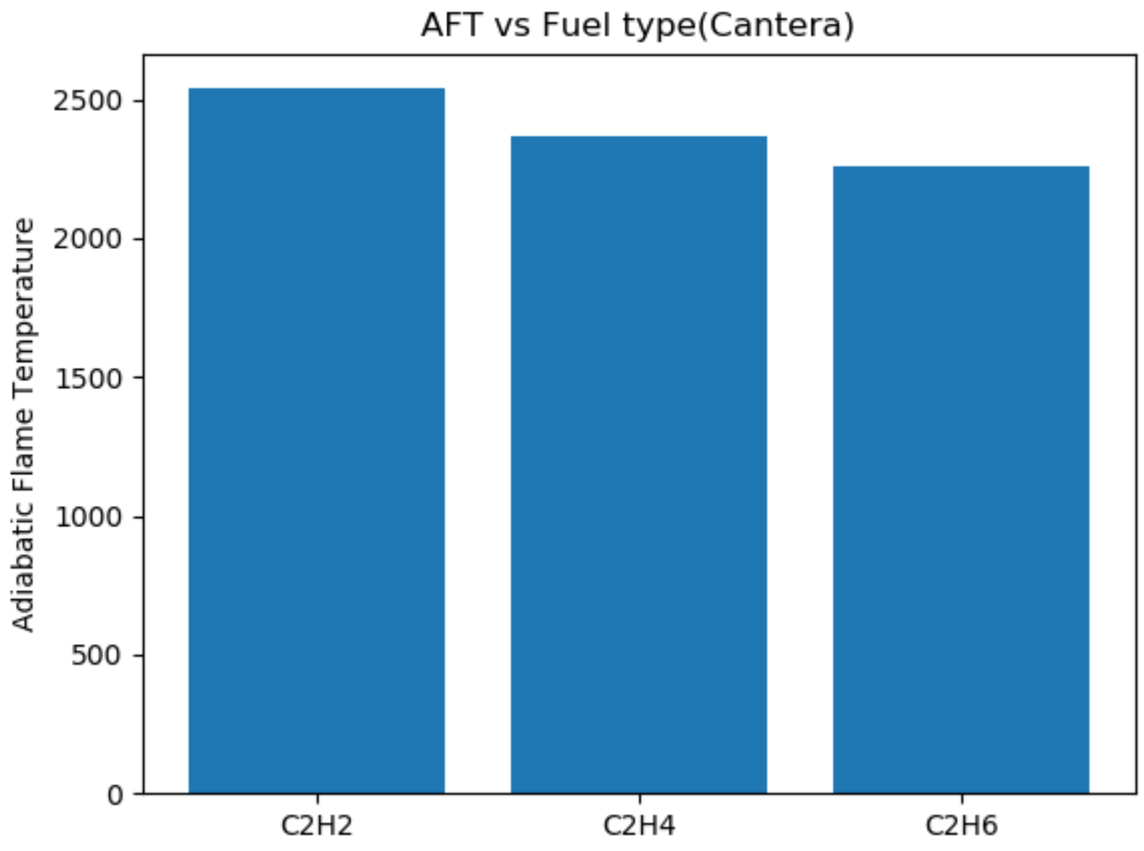
```
"""
Code for finding out the AFT of methane for different equivalence ratio

CH4 + 2(O2 + 3.76N2) = CO2 + 2H2O + 7.52N2
"""

import matplotlib.pyplot as plt
import math
import numpy as np

def h(T, co_effs):
        """
        Function to evaluate enthalpy
        """
        R = 8.314 #J/mol-K
        a1 = co_effs[0]
        a2 = co_effs[1]
        a3 = co_effs[2]
        a4 = co_effs[3]
        a5 = co_effs[4]
        a6 = co_effs[5]

        return (a1 + a2*T/2 + a3*pow(T,2)/3 + a4*pow(T,3)/4 + a5*pow(T,4)/5 + a

        # Coefficients for fuel: Alkynes(c2h2), Alkenes(c2h4), Alkane(c2h6)

c2h2_coeffs_l = [8.08681094E-01,2.33615629E-02,-3.55171815E-05,2.80152437E-08,-
c2h4_coeffs_l = [3.95920148E+00,-7.57052247E-03,5.70990292E-05,-6.91588753E-08,
c2h6_coeffs_l = [4.29142492E+00,-5.50154270E-03,5.99438288E-05,-7.08466285E-08,
        # Coefficients for Oxidiser: Oxygen(o2),Nitrogen(N2)

o2_coeffs_l  = [3.78245636E+00,-2.99673416E-03,9.84730201E-06,-9.68129509E-09,3
n2_coeffs_l = [0.03298677E+02,0.14082404E-02,-0.03963222E-04,0.05641515E-07,-0.

        # Coefficients for Products at high temperature: Co2,H2o,N2

n2_coeffs_h = [0.02926640E+02,0.14879768E-02,-0.05684760E-05,0.10097038E-09,-0.
co2_coeffs_h = [3.85746029E+00,4.41437026E-03,-2.21481404E-06,5.23490188E-10,-4
```

```
h2o_coeffs_h = [3.03399249E+00,2.17691804E-03,-1.64072518E-07,-9.70419870E-11,1


Tstd = 298.15
H1 = h(Tstd,c2h2_coeffs_l)
H2 = h(Tstd,c2h4_coeffs_l)
H3 = h(Tstd,c2h6_coeffs_l)
H  = [H1,H2,H3]
H = np.array(H)


def f(T,H):
        """
        #Function that represent the root finding problem
        """
        h_co2_p = h(T,co2_coeffs_h)
        h_h2o_p = h(T,h2o_coeffs_h)
        h_n2_p  = h(T,n2_coeffs_h)

        h_o2_r = h(Tstd,o2_coeffs_l)
        h_n2_r = h(Tstd,n2_coeffs_l)

        if H == H1:
                x = 2
                y = 2
                a = x + (y/4)
                H_product = x*h_co2_p + (y/2)*h_h2o_p + (3.76*a)*h_n2_p
                H_reactants = H1 + a*h_o2_r + (3.76*a)*h_n2_r
        elif H == H2:
                x = 2
                y = 4
                a = x + (y/4)
                H_product = x*h_co2_p + (y/2)*h_h2o_p + (3.76*a)*h_n2_p
                H_reactants = H2 + a*h_o2_r + (3.76*a)*h_n2_r
        elif H == H3:
                x = 2
                y = 6
                a = x + (y/4)
                H_product = x*h_co2_p + (y/2)*h_h2o_p + (3.76*a)*h_n2_p
                H_reactants = H3 + a*h_o2_r + (3.76*a)*h_n2_r

        return H_product - H_reactants

def fprime(T,H):
                return(f(T+1e-6,H) - f(T,H))/1e-6


T_guess = 1500
tol = 1e-3
ct = 0
alpha = 0.9
Temp =[]

for i in H:

        R = 8.314 #J/mol-K
        while(abs(f(T_guess,i))> tol):
                T_guess = T_guess - alpha*((f(T_guess,i)/fprime(T_guess,i)))

        Temp.append(T_guess)
```

```
print(Temp)
plt.figure()
plt.bar(['C2H2','C2H4','C2H6'],Temp)
plt.ylabel('Adiabatic Flame Temperature')
plt.title('AFT vs Fuel type(Python)')



################################################################################
# Solution using Cantera

import cantera as ct


gas = ct.Solution('gri30.xml')

gas.TPX = 298.15,101325,{'C2H2':1, 'O2':2.5, 'N2':9.4}

gas.equilibrate('HP','auto')
T_c2h2 = gas.T

gas.TPX = 298.15,101325,{'C2H4':1, 'O2':3, 'N2':11.28}

gas.equilibrate('HP','auto')
T_c2h4 = gas.T

gas.TPX = 298.15,101325,{'C2H6':1, 'O2':3.5, 'N2':13.16}

gas.equilibrate('HP','auto')
T_c2h6 = gas.T

temp = (T_c2h2,T_c2h4,T_c2h6)

temp= np.array(temp)
print(temp)

plt.figure()
plt.bar(['C2H2','C2H4','C2H6'],temp)
plt.ylabel('Adiabatic Flame Temperature')
plt.title('AFT vs Fuel type(Cantera)')
plt.show()
```

The Plot to represent the comparative analysis of AFT is:


AFT vs Fuel type(Python)
Adiabatic Flame Temperature
3000
2500
2000
1500
1000
500
0
C2H2
C2H4
C2H6



Observations:

1. The Alkyne gives maximum temperature because it contains less number of hydrogen atoms because less water will be produced.

2. The fuel which contains fewer hydrogen atoms gives maximum temperature.

3. The trend with the type of fuel is the same for python as well as Cantera, but the temperature value differs by at least 400K. Which is because Cantera solves for all the species.

Case 2.3 Effect of carbon atoms(1 to 3) with alkane fuel on Adiabatic Flame Temperature:

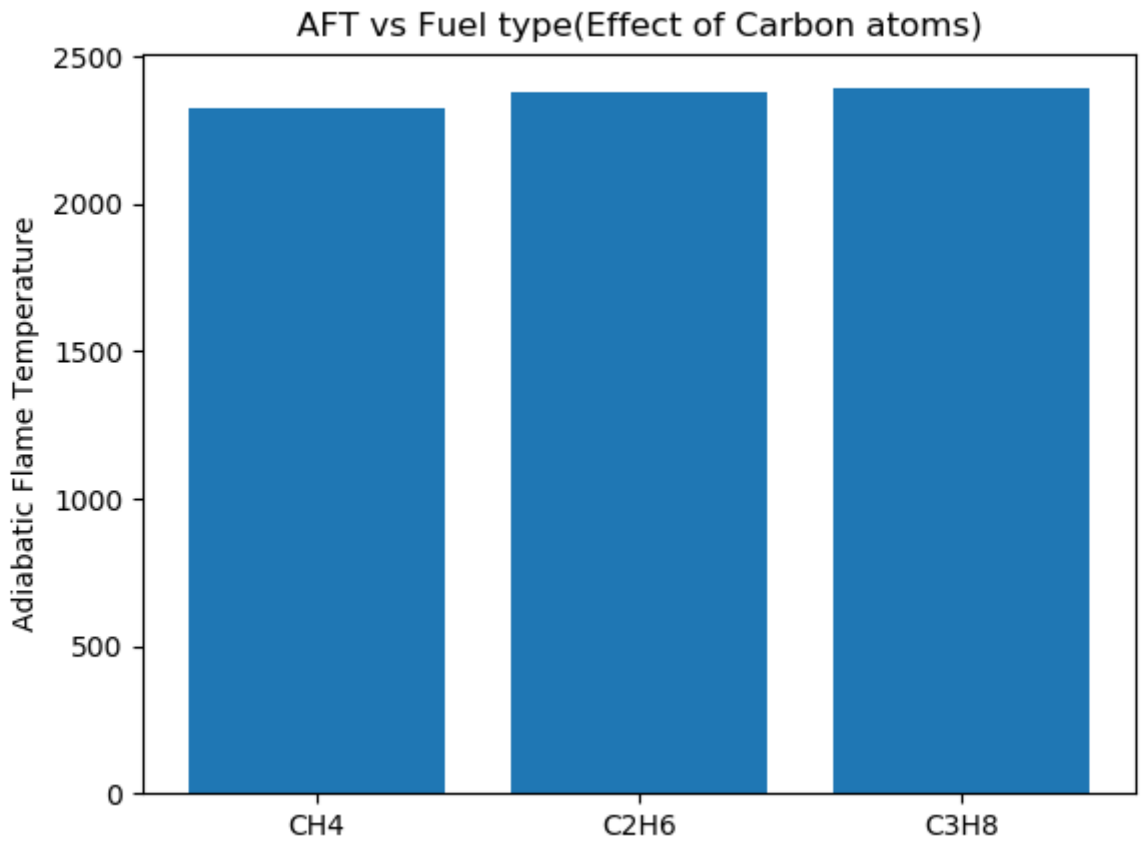
The carbon atoms of the alkane are varied from 1 to 3, and the effect on AFT is to be studied. The more carbon means more heat is to be produced and flame temperature will be increased with carbon atoms.

The code to study the effect of carbon atom on AFT is:

```
"""
Code for finding out the AFT of methane for different equivalence ratio
```

```
CH4 + 2(O2 + 3.76N2) = CO2 + 2H2O + 7.52N2
"""

import matplotlib.pyplot as plt
import math
import numpy as np

def h(T, co_effs):
        """
        Function to evaluate enthalpy
        """
        R = 8.314 #J/mol-K
        a1 = co_effs[0]
        a2 = co_effs[1]
        a3 = co_effs[2]
        a4 = co_effs[3]
        a5 = co_effs[4]
        a6 = co_effs[5]

        return (a1 + a2*T/2 + a3*pow(T,2)/3 + a4*pow(T,3)/4 + a5*pow(T,4)/5 + a

# Coefficients for fuel: Alkane(ch4), Alkane(c2h6), Alkane(c3h8)

ch4_coeffs_l  = [5.14987613E+00,-1.36709788E-02,4.91800599E-05,-4.84743026E-08,
c2h6_coeffs_l = [4.29142492E+00,-5.50154270E-03,5.99438288E-05,-7.08466285E-08,
c3h8_coeffs_l = [0.93355381E+00,0.26424579E-01,0.61059727E-05,-0.21977499E-07,0

        # Coefficients for Oxidiser: Oxygen(o2),Nitrogen(N2)

o2_coeffs_l  = [3.78245636E+00,-2.99673416E-03,9.84730201E-06,-9.68129509E-09,3
n2_coeffs_l = [0.03298677E+02,0.14082404E-02,-0.03963222E-04,0.05641515E-07,-0.

        # Coefficients for Products at high temperature: Co2,H2o,N2

n2_coeffs_h = [0.02926640E+02,0.14879768E-02,-0.05684760E-05,0.10097038E-09,-0.
co2_coeffs_h = [3.85746029E+00,4.41437026E-03,-2.21481404E-06,5.23490188E-10,-4
h2o_coeffs_h = [3.03399249E+00,2.17691804E-03,-1.64072518E-07,-9.70419870E-11,1

Tstd = 298.15
H1 = h(Tstd,ch4_coeffs_l)
H2 = h(Tstd,c2h6_coeffs_l)
H3 = h(Tstd,c3h8_coeffs_l)
H  = [H1,H2,H3]
H = np.array(H)

def f(T,H):
        """
        #Function that represent the root finding problem
        """
        h_co2_p = h(T,co2_coeffs_h)
        h_h2o_p = h(T,h2o_coeffs_h)
        h_n2_p  = h(T,n2_coeffs_h)

        h_o2_r = h(Tstd,o2_coeffs_l)
        h_n2_r = h(Tstd,n2_coeffs_l)

        if H == H1:
                x = 1
                y = 4
                a = x + (y/4)
                H_product = x*h_co2_p + (y/2)*h_h2o_p + (3.76*a)*h_n2_p
                H_reactants = H1 + a*h_o2_r + (3.76*a)*h_n2_r
        elif H == H2:
                x = 2
                y = 6
                a = x + (y/4)
                H_product = x*h_co2_p + (y/2)*h_h2o_p + (3.76*a)*h_n2_p
                H_reactants = H2 + a*h_o2_r + (3.76*a)*h_n2_r
        elif H == H3:
                x = 3
                y = 8
                a = x + (y/4)
                H_product = x*h_co2_p + (y/2)*h_h2o_p + (3.76*a)*h_n2_p
                H_reactants = H3 + a*h_o2_r + (3.76*a)*h_n2_r

        return H_product - H_reactants

def fprime(T,H):
                return(f(T+1e-6,H) - f(T,H))/1e-6


T_guess = 1500
tol = 1e-3
ct = 0
alpha = 0.9
Temp =[]

for i in H:

        R = 8.314 #J/mol-K
        while(abs(f(T_guess,i))> tol):
                T_guess = T_guess - alpha*((f(T_guess,i)/fprime(T_guess,i)))

        Temp.append(T_guess)
print(Temp)
plt.figure()
plt.bar(['CH4','C2H6','C3H8'],Temp)
plt.ylabel('Adiabatic Flame Temperature')
plt.title('AFT vs Fuel type(Effect of Carbon atoms)')
plt.show()
```

The plot to show the effect of carbon atom on AFT is:



Observation:

1. The fuel which contains more carbon atoms gives maximum temperature.