THE EVALUATION OF THE ADIABATIC FLAME TEMPERATURE USING PYTHON AND CANTERA : Skill-Lync

Skill-Lync

THE EVALUATION OF THE ADIABATIC FLAME TEMPERATURE USING PYTHON AND CANTERA

I. OBJECTIVES

1. Write a program to analyze the effect of equivalence ratio on the final adiabatic temperature of methane contained in a constant volume chamber. Also, compare the results with those obtained using Cantera.

2. Write a program to analyze the effect of fractional heat loss on the final adiabatic temperature of methane contained in a constant pressure chamber.

3. Write a program to calculate the AFT of the combustion process for a given equivalence ratio and heat loss fraction.

4. Write a program to analyze the variation of Adiabatic Flame Temperature for different hydrocarbons (Ethane, Ethene & Ethyne) contained in a constant pressure chamber.

5. Write a program to analyze the variation of Adiabatic Flame Temperature for different alkanes (Methane, Ethane & Propane) contained in a constant pressure chamber.

II. INTRODUCTION

A. Adiabatic Flame Temperature (AFT)

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 the product gases in the flame region and the temperature rise will be maximum. This highest temperature is known as the adiabatic flame temperature.

B. Processes Involved In Combustion

There are two processes which can determine the final AFT of the system -

1. Constant Volume Process

The Internal Energy of reactants and products are equal in a constant volume process.

$U_R = U_P$

$H_R - (PV)_R = H_P - (PV)_P$

$H_R - (nRT)_R = H_P - (nRT)_P$

$(H_P - H_R) - \left((nRT)_P - (nRT)_R\right) = 0$

2. Constant Pressure Process

The enthalpy of reactants and products are equal in a constant pressure process.

$H_R = H_P$

$H_P - H_R = 0$

C. Stoichiometric Equation

The general stoichiometric equation for the combustion of a hydrocarbon can be represented as follows -

$$C_xH_y + ar(O_2 + 3.76N_2) \rightarrow xCO_2 + \left(\frac{y}{2}\right)H_2O + (3.76ar)N_2$$

The term stoichiometric refers to the presence of just enough oxygen to completely burn all the fuel. The AFT obtained at the stoichiometric ratio is maximum.

However, such ideal conditions are rarely observed in a combustion process and therefore there is always some additional species leftover at the end of the process.

D. Fuel - Air Equivalence Ratio (ϕ)

The fuel - air equivalence ratio is defined as the ratio of the Fuel-Air ratio to the corresponding stoichiometric Fuel-Air ratio of the given combustion reaction.

$$\phi = \frac{Fuel/Air}{(Fuel/Air)_{st}}$$

E. General Equation of Combustion Process

The general equation for the combustion of a hydrocarbon can be represented as follows -

$$C_xH_y + g(O_2 + 3.76N_2) \rightarrow aCO_2 + bH_2O + cN_2 + dO_2 + eCO$$

1. For Stoichiometric Mixture $(\phi = 1)$

- $ar = x + \frac{y}{4}; g = ar; a = x; b = \frac{y}{2}; c = 3.76ar; d = 0; e = 0$

2. For Lean Mixture $(\phi < 1)$

- $ar = x + \frac{y}{4}; g = \left(\frac{ar}{\phi}\right); a = x; b = \frac{y}{2}; c = 3.76\left(\frac{ar}{\phi}\right); d = ar \cdot \left(\left(\frac{1}{\phi}\right) - 1\right); e = 0$

3. For Rich Mixture $(\phi > 1)$

- $ar = x + \frac{y}{4}; g = \left(\frac{ar}{\phi}\right); a = \left(\frac{2ar}{\phi}\right) - x - \frac{y}{2}; b = \frac{y}{2}; c = 3.76 \cdot ar \cdot \left(\frac{1}{\phi}\right); d = 0; e = 2 \cdot x$

F. Enthalpy of the System

The enthalpy of the system depends on the species involved in the process and the temperature of the system. The enthalpy of the system can be calculated using the formula -

$$H = \left(a_1 + a_2\frac{T}{2} + a_3\frac{T^2}{3} + a_4\frac{T^3}{4} + a_5\frac{T^4}{5} + \frac{a_6}{T}\right) \cdot R \cdot T$$

where $a_1, a_2, a_3, a_4, a_5, a_6 \rightarrow$ The numerical coefficients supplied in **Nasa Thermodynamic Data Files**

G. Newton - Raphson Method

It is a root-finding algorithm which produces successively better approximations to the roots of a real-valued function.

The general formula for Newton Raphson method can be given as -

$$x_{n+1} = x_n - \alpha\frac{f(x_n)}{f'(x_n)}$$

where $f(x) \rightarrow$ Real-valued function; $f'(x) \rightarrow$ Derivative of the function; $\alpha \rightarrow$ Multiplicity of the root

III. PYTHON AND CANTERA PROGRAMS AND OUTPUTS

In this project, we shall be considering the following cases -

1. Effect of equivalence ratio on the adiabatic flame temperature.

2. Effect of heat loss on the adiabatic flame temperature.

3. Calculation of AFT for a given equivalence ratio and heat loss fraction.

4. Variation of adiabatic flame temperature for different hydrocarbons.

5. Variation of adiabatic flame temperature for different alkanes.

CASE 1 - EFFECT OF EQUIVALENCE RATIO ON ADIABATIC FLAME TEMPERATURE

1. PROCESS: Combustion of hydrocarbon in a Constant Volume Chamber

2. HYDROCARBON:

- **Methane:** x = 1; y = 4

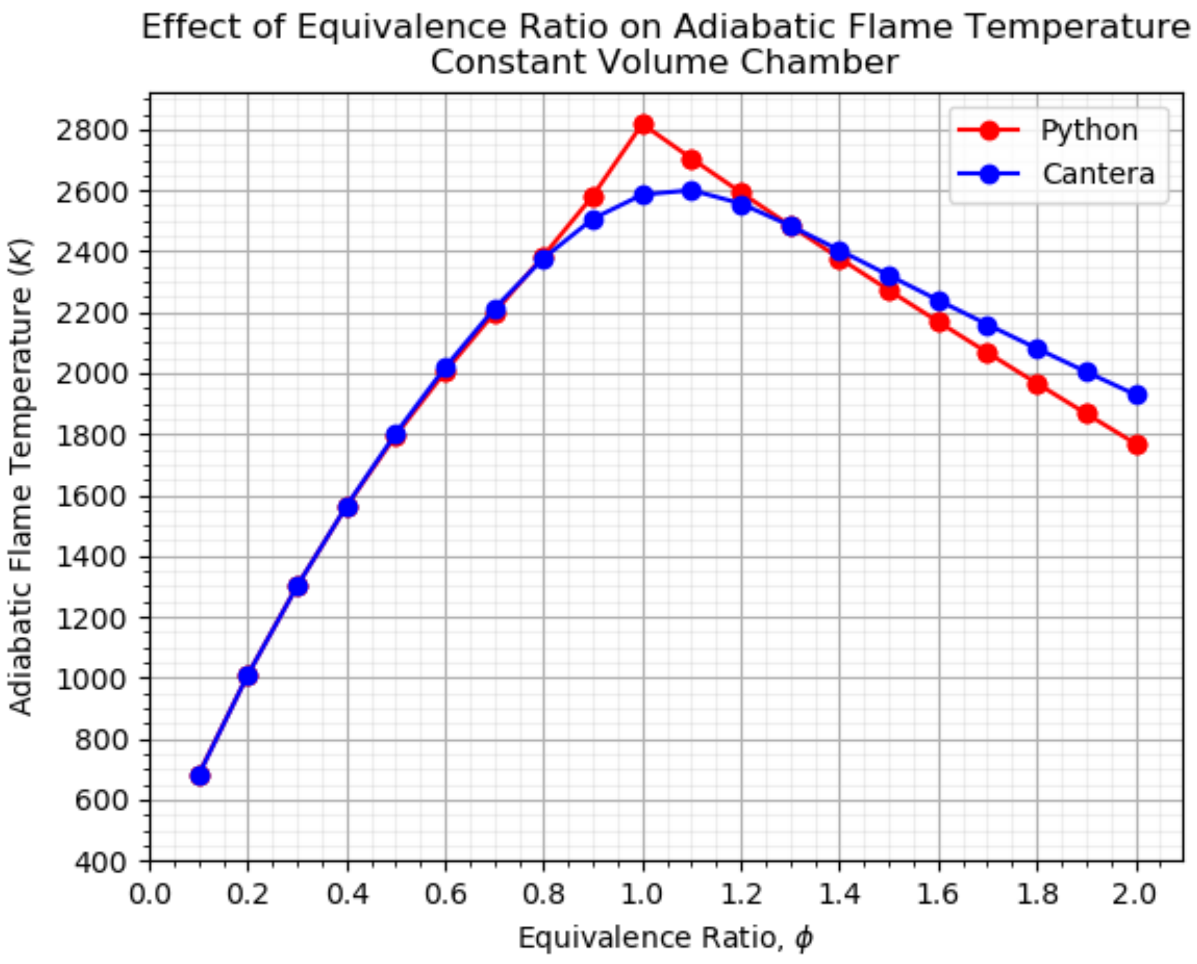
3. CODE:

```
"""
        In this program we are going to calculate the AFT of Methane at differe

        General Equation:
                CxHy + g(O2 + 3.76N2) = aCO2 + bH2O + cN2 + dO2 + eCO
                x = 1; y = 4; ar = x + y/4; g = ar * (1/phi)

        Stoichiometric Equation (phi = 1):
                CxHy + ar(O2 + 3.76N2) = xCO2 + (y/2)H2O + 3.76(x + y/4)N2
        Lean Mixture Equation (phi < 1):
                CxHy + ar(1/phi)(O2 + 3.76N2) = xCO2 + (y/2)H2O + 3.76(ar/phi)N
        Rich Mixture Equation (phi > 1):
                CxHy + ar(1/phi)(O2 + 3.76N2) = ((ar/2phi) - x - y/2)CO2 + (y/2

"""

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


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

# NASA Polynomial Constants

ch4_coeffs_l = [5.14987613E+00, -1.36709788E-02, 4.91800599E-05, -4.84743026E-0
o2_coeffs_l = [3.78245636E+00, -2.99673416E-03, 9.84730201E-06, -9.68129509E-09
n2_coeffs_l = [0.03298677E+02, 0.14082404E-02, -0.03963222E-04, 0.05641515E-07,

n2_coeffs_h = [0.02926640E+02, 0.14879768E-02, -0.05684760E-05, 0.10097038E-09,
co2_coeffs_h = [3.85746029E+00, 4.41437026E-03, -2.21481404E-06, 5.23490188E-10
h2o_coeffs_h = [3.03399249E+00, 2.17691804E-03, -1.64072518E-07, -9.70419870E-1
o2_coeffs_h =  [3.78245636E+00, -2.99673416E-03, 9.84730201E-06, -9.68129509E-0
co_coeffs_h = [2.71518561E+00, 2.06252743E-03, -9.98825771E-07, 2.30053008E-10,

# Function to evaluate Enthalpy
```

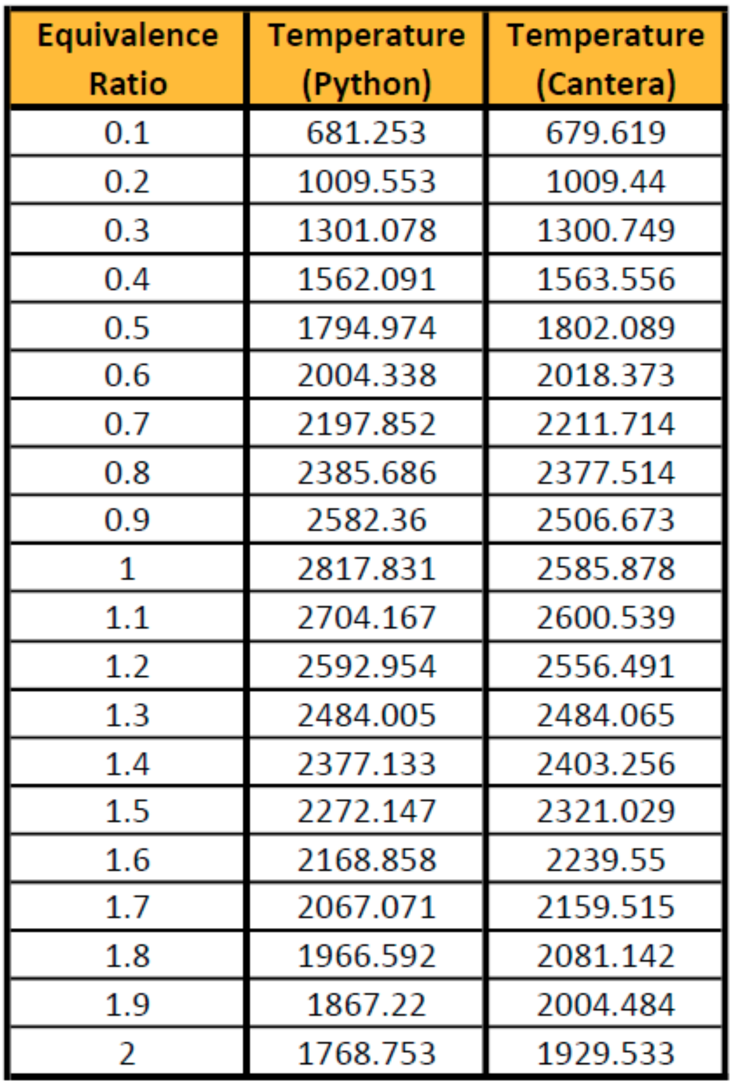
```
def h(T, co_effs):

        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


# Function that represents the root finding problem

def f(T):

        # Products At Temperature T

        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_products = a*h_co2_p + b*h_h2o_p + c*h_n2_p + d*h_o2_p + e*h_co_p

        n_products = a + b + c + d + e
        nRT_products = n_products * R * T

        # Reactants At Standard Temperature

        T_std = 298.15
        h_ch4_r = h(T_std, ch4_coeffs_l)
        h_o2_r = h(T_std, o2_coeffs_l)
        h_n2_r = h(T_std, n2_coeffs_l)
        H_reactants = h_ch4_r + g*h_o2_r + 3.76*g*h_n2_r

        n_reactants = 1 + g + 3.76*g
        nRT_reactants = n_reactants * R * T_std

        return (H_products - H_reactants) - (nRT_products - nRT_reactants)


# Function that represents the derivative of the root finding problem

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

# Calculating the temperature and storing the values

T_py = []
phi_min = 0.1
phi_max = 2
phi_diff = 0.1
phi = np.arange(phi_min, phi_max + phi_diff, phi_diff)

for i in range (0, len(phi)):

        if (phi[i] == 1):
                ar = x + y/4
                g = ar
                a = x
                b = y/2
                c = 3.76 * ar
                d = 0
                e = 0

        if (phi[i] < 1):
                ar = x + y/4
                g = ar * (1/phi[i])
                a = x
                b = y/2
                c = 3.76 * ar * (1/phi[i])
                d = ar * ((1/phi[i])-1)
                e = 0

        if (phi[i] > 1):
                ar = x + y/4
                g = ar * (1/phi[i])
                a = (2*ar/phi[i]) - x - y/2
                b = y/2
                c = 3.76 * ar * (1/phi[i])
                d = 0
                e = 2*x + y/2 - (2*ar)/phi[i]

        T_guess = 1500
        tol = 1e-6
        alpha = 0.2
        ct = 0

        while (abs(f(T_guess)) > tol):
                T_guess = T_guess - alpha * (f(T_guess) / fprime(T_guess))
                ct = ct + 1

        T_py.append(T_guess)


# Cantera Code - Exact Solution

import cantera as ct

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

T_AFT_ct = []

for i in range(0, len(phi)):

        n_total = 1 + (ar/phi[i]) + 3.76*(ar/phi[i])
        n_ch4 = 1/n_total
        n_o2 = (ar/phi[i])/n_total
        n_n2 = (3.76*(ar/phi[i]))/n_total

        gas.TPX = 298.15, 101325, {'CH4':n_ch4, 'O2':n_o2, 'N2':n_n2}
        gas.equilibrate('UV', 'auto')
        T_AFT_ct.append(gas.T)

print('Equivalence Ratio = ', phi)
print('Temperature (Python) = ', T_py)
print('Temperature (Cantera) = ', T_AFT_ct)

# Plotting the results obtained from Python and Cantera
```

```
plt.plot(phi, T_py, '-o', color = 'red', label = 'Python')
plt.plot(phi, T_AFT_ct, '-o', color = 'blue', label = 'Cantera')
plt.xlabel('Equivalence Ratio, $\phi$')
plt.ylabel('Adiabatic Flame Temperature ($K$)')
plt.title('Effect of Equivalence Ratio on Adiabatic Flame Temperature\n'
                        + 'Constant Volume Chamber')
plt.legend()
plt.xticks(np.arange(0, 2.1, 0.2))
plt.yticks(np.arange(400, 3000, 200))
plt.grid(which='major')
plt.minorticks_on()
plt.grid(which='minor', alpha=0.2)
plt.show()
```

4. OUTPUT:



5. RESULTS TABLE:



Equivalence Ratio	Temperature (Python)	Temperature (Cantera)
0.1	681.253	679.619
0.2	1009.553	1009.44
0.3	1301.078	1300.749
0.4	1562.091	1563.556
0.5	1794.974	1802.089
0.6	2004.338	2018.373
0.7	2197.852	2211.714
0.8	2385.686	2377.514
0.9	2582.36	2506.673
1	2817.831	2585.878
1.1	2704.167	2600.539
1.2	2592.954	2556.491
1.3	2484.005	2484.065
1.4	2377.133	2403.256
1.5	2272.147	2321.029
1.6	2168.858	2239.55
1.7	2067.071	2159.515
1.8	1966.592	2081.142
1.9	1867.22	2004.484
2	1768.753	1929.533

6. RESULTS:

The AFT obtained from Python and Cantera are almost equal at an equivalence ratio lower than 0.8. This is because enough oxygen is present in the system to burn all the fuel into its corresponding products. These products have negligible additional species and hence, the results are almost equal for both Python and Cantera.

At an equivalence ratio greater than 0.8, the AFT curve obtained from Python deviates from the Cantera's curve, because the combustion process at these ratios results in the formation of some additional minor species which are not considered by us in the Python program.

CASE 2 - EFFECT OF HEAT LOSS ON ADIABATIC FLAME TEMPERATURE

1. PROCESS:

- Combustion of hydrocarbon in a Constant Pressure Chamber

- Equivalence Ratio, $\phi = 1$

2. HYDROCARBON:

- **Methane:** x = 1; y = 4

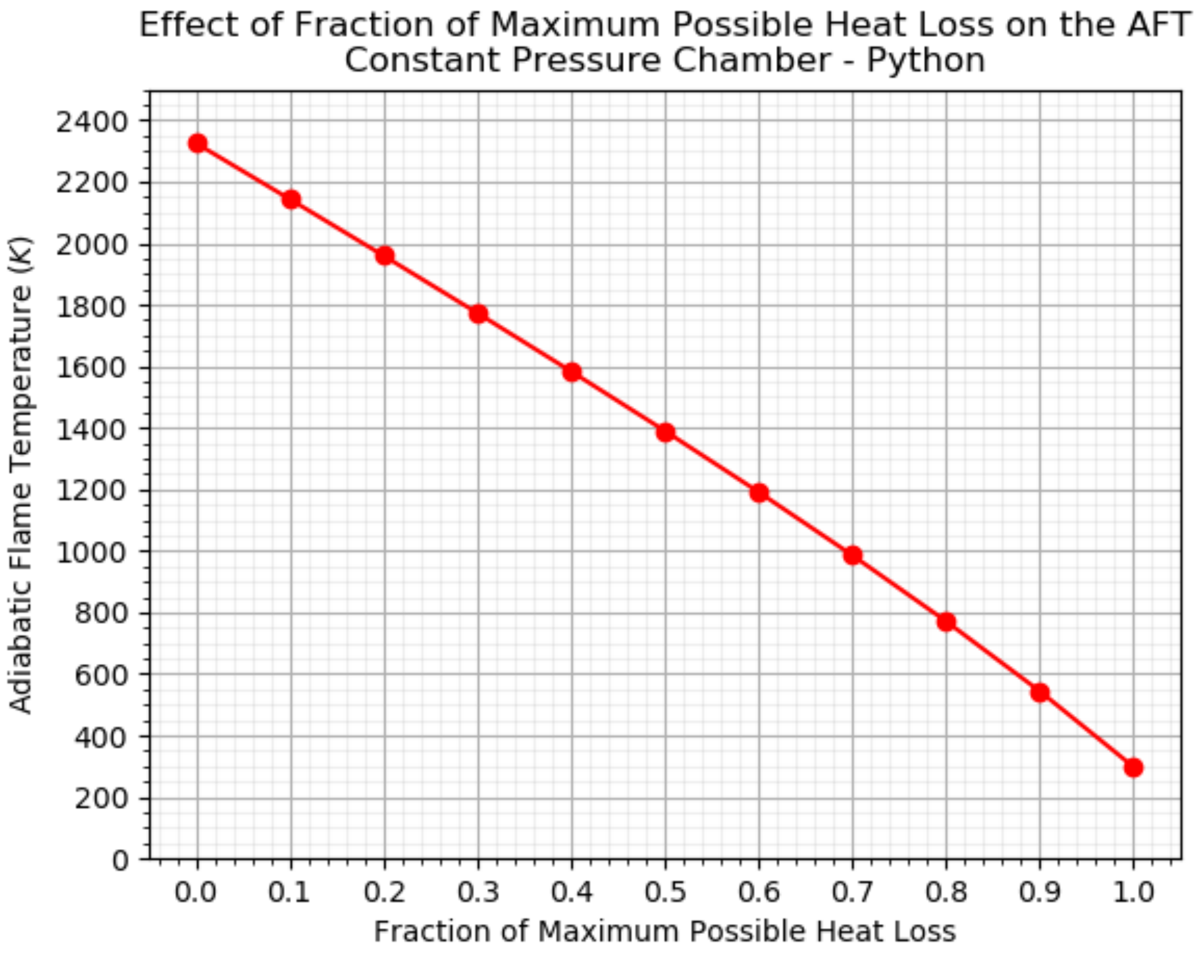
3. CODE:

```
"""
        In this program we are going to calculate the AFT of Methane
        at different Fraction of Heat Loss
        General Equation:
                CxHy + g(O2 + 3.76N2) = aCO2 + bH2O + cN2 + dO2 + eCO
                x = 1; y = 4; ar = x + y/4; g = ar * (1/phi)

        Stoichiometric Equation (phi = 1):
                CxHy + ar(O2 + 3.76N2) = xCO2 + (y/2)H2O + 3.76(x + y/4)N2
        Lean Mixture Equation (phi < 1):
                CxHy + ar(1/phi)(O2 + 3.76N2) = xCO2 + (y/2)H2O + 3.76(ar/phi)N
        Rich Mixture Equation (phi > 1):
                CxHy + ar(1/phi)(O2 + 3.76N2) = ((ar/2phi) - x - y/2)CO2 + (y/2
"""

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

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

# NASA Polynomial Constants

ch4_coeffs_l = [5.14987613E+00, -1.36709788E-02, 4.91800599E-05, -4.84743026E-0
o2_coeffs_l = [3.78245636E+00, -2.99673416E-03, 9.84730201E-06, -9.68129509E-09
n2_coeffs_l = [0.03298677E+02, 0.14082404E-02, -0.03963222E-04, 0.05641515E-07,

n2_coeffs_h = [0.02926640E+02, 0.14879768E-02, -0.05684760E-05, 0.10097038E-09,
co2_coeffs_h = [3.85746029E+00, 4.41437026E-03, -2.21481404E-06, 5.23490188E-10
h2o_coeffs_h = [3.03399249E+00, 2.17691804E-03, -1.64072518E-07, -9.70419870E-1
o2_coeffs_h =  [3.78245636E+00, -2.99673416E-03, 9.84730201E-06, -9.68129509E-0
co_coeffs_h = [2.71518561E+00, 2.06252743E-03, -9.98825771E-07, 2.30053008E-10,

# Function to evaluate Enthalpy

def h(T, co_effs):

        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


# Function that represents the root finding problem
```

```
def f(T, H_loss):

        # Products At Temperature T

        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_products = a*h_co2_p + b*h_h2o_p + c*h_n2_p + d*h_o2_p + e*h_co_p

        # Products At Standard Temperature

        T_std = 298.15
        h_co2_p_std = h(T_std, co2_coeffs_h)
        h_h2o_p_std = h(T_std, h2o_coeffs_h)
        h_n2_p_std = h(T_std, n2_coeffs_h)
        h_o2_p_std = h(T_std, o2_coeffs_h)
        h_co_p_std = h(T_std, co_coeffs_h)
        H_products_std = a*h_co2_p_std + b*h_h2o_p_std + c*h_n2_p_std + d*h_o2_

        # Reactants At Standard Temperature

        h_ch4_r = h(T_std, ch4_coeffs_l)
        h_o2_r = h(T_std, o2_coeffs_l)
        h_n2_r = h(T_std, n2_coeffs_l)
        H_reactants = h_ch4_r + g*h_o2_r + 3.76*g*h_n2_r

        # Lower Heating Value

        LHV = H_reactants - H_products_std

        return H_products - H_reactants + (H_loss * LHV)


# Function that represents the derivative of the root finding problem

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


# Calculating the temperature and storing the values

H_loss_py = []
T_py = []
phi_py = 1
H_loss = 0
H_loss_max = 1
H_loss_diff = 0.1

if (phi_py == 1):
        ar = x + y/4
        g = ar
        a = x
        b = y/2
        c = 3.76 * ar
        d = 0
        e = 0
```

```
if (phi_py < 1):
        ar = x + y/4
        g = ar * (1/phi_py)
        a = x
        b = y/2
        c = 3.76 * ar * (1/phi_py)
        d = ar * ((1/phi_py)-1)
        e = 0

if (phi_py > 1):
        ar = x + y/4
        g = ar * (1/phi_py)
        a = (2*ar/phi_py) - x - y/2
        b = y/2
        c = 3.76 * ar * (1/phi_py)
        d = 0
        e = 2*x + y/2 - (2*ar)/phi_py

T_guess = 1500
tol = 1e-6
alpha = 0.2
ct = 0

while (H_loss <= H_loss_max):

        while (abs(f(T_guess, H_loss)) > tol):
                T_guess = T_guess - alpha * (f(T_guess, H_loss) / fprime(T_gues
                ct = ct + 1

        H_loss_py.append(H_loss)
        T_py.append(T_guess)
        H_loss = H_loss + H_loss_diff
        H_loss = round(H_loss, 1)

print('Heat Loss Fraction = ', H_loss_py)
print('Temperature = ', T_py)

# Plotting the results obtained from Python

plt.plot(H_loss_py, T_py, '-o', color = 'red')
plt.ylim(0,2500)
plt.xticks(np.arange(0, 1.1, 0.1))
plt.yticks(np.arange(0, 2500, 200))
plt.xlabel('Fraction of Maximum Possible Heat Loss')
plt.ylabel('Adiabatic Flame Temperature ($K$)')
plt.title('Effect of Fraction of Maximum Possible Heat Loss on the AFT\n'
                        + 'Constant Pressure Chamber - Python')
plt.grid(which='major')
plt.minorticks_on()
plt.grid(which='minor', alpha=0.2)
plt.show()
```

4. OUTPUT:



5. RESULTS TABLE:

Heat Loss Fraction	Temperature (Python)
0	2325.598
0.1	2143.828
0.2	1960.024
0.3	1773.686
0.4	1584.162
0.5	1390.592
0.6	1191.819
0.7	986.239
0.8	771.534
0.9	544.151
1	298.15

6. RESULTS:

The increase in the fraction of the maximum possible heat loss causes a corresponding decrease in the AFT.

For a given reaction, the maximum temperature can be obtained when the heat loss is zero, and the minimum temperature i.e. standard temperature is obtained when the heat is completely lost.

CASE 3 - CALCULATION OF AFT FOR A GIVEN EQUIVALENCE RATIO AND HEAT LOSS FRACTION

1. PROCESS:

- Combustion of hydrocarbon in a Constant Pressure Chamber
- Equivalence Ratio, $\phi = 1$
- Heat Loss Fraction = 0.35

2. HYDROCARBON:

- **Methane:** $x = 1$; $y = 4$

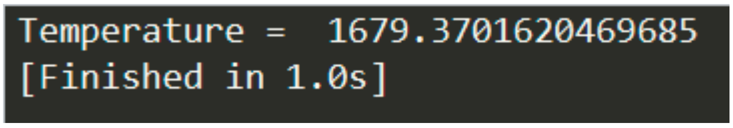
3. CODE:

```
"""
	In this program, we will calculate the AFT for a given equivalence rati
	General Equation:
		CxHy + g(O2 + 3.76N2) = aCO2 + bH2O + cN2 + dO2 + eCO
```

```
                x = 1; y = 4; ar = x + y/4; g = ar * (1/phi)

        Stoichiometric Equation (phi = 1):
                CxHy + ar(O2 + 3.76N2) = xCO2 + (y/2)H2O + 3.76(x + y/4)N2
        Lean Mixture Equation (phi < 1):
                CxHy + ar(1/phi)(O2 + 3.76N2) = xCO2 + (y/2)H2O + 3.76(ar/phi)N
        Rich Mixture Equation (phi > 1):
                CxHy + ar(1/phi)(O2 + 3.76N2) = ((ar/2phi) - x - y/2)CO2 + (y/2
"""

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

R = 8.314 # J/mol-K
x = 1
y = 4
phi_py = 1
H_loss = 0.35

# NASA Polynomial Constants

ch4_coeffs_l = [5.14987613E+00, -1.36709788E-02, 4.91800599E-05, -4.84743026E-0
o2_coeffs_l = [3.78245636E+00, -2.99673416E-03, 9.84730201E-06, -9.68129509E-09
n2_coeffs_l = [0.03298677E+02, 0.14082404E-02, -0.03963222E-04, 0.05641515E-07,

n2_coeffs_h = [0.02926640E+02, 0.14879768E-02, -0.05684760E-05, 0.10097038E-09,
co2_coeffs_h = [3.85746029E+00, 4.41437026E-03, -2.21481404E-06, 5.23490188E-10
h2o_coeffs_h = [3.03399249E+00, 2.17691804E-03, -1.64072518E-07, -9.70419870E-1
o2_coeffs_h =  [3.78245636E+00, -2.99673416E-03, 9.84730201E-06, -9.68129509E-0
co_coeffs_h = [2.71518561E+00, 2.06252743E-03, -9.98825771E-07, 2.30053008E-10,

# Function to evaluate Enthalpy

def h(T, co_effs):

        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


# Function that represents the root finding problem

def f(T, H_loss):

        # Products At Temperature T

        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_products = a*h_co2_p + b*h_h2o_p + c*h_n2_p + d*h_o2_p + e*h_co_p

        # Products At Standard Temperature
        T_std = 298.15
        h_co2_p_std = h(T_std, co2_coeffs_h)
        h_h2o_p_std = h(T_std, h2o_coeffs_h)
        h_n2_p_std = h(T_std, n2_coeffs_h)
        h_o2_p_std = h(T_std, o2_coeffs_h)
        h_co_p_std = h(T_std, co_coeffs_h)
        H_products_std = a*h_co2_p_std + b*h_h2o_p_std + c*h_n2_p_std + d*h_o2_

        # Reactants At Standard Temperature

        h_ch4_r = h(T_std, ch4_coeffs_l)
        h_o2_r = h(T_std, o2_coeffs_l)
        h_n2_r = h(T_std, n2_coeffs_l)
        H_reactants = h_ch4_r + g*h_o2_r + 3.76*g*h_n2_r

        # Lower Heating Value

        LHV = H_reactants - H_products_std

        return H_products - H_reactants + (H_loss * LHV)


# Function that represents the derivative of the root finding problem

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


# Calculating the temperature and storing the values

if (phi_py == 1):
        ar = x + y/4
        g = ar
        a = x
        b = y/2
        c = 3.76 * ar
        d = 0
        e = 0

if (phi_py < 1):
        ar = x + y/4
        g = ar * (1/phi_py)
        a = x
        b = y/2
        c = 3.76 * ar * (1/phi_py)
        d = ar * ((1/phi_py)-1)
        e = 0

if (phi_py > 1):
        ar = x + y/4
        g = ar * (1/phi_py)
        a = (2*ar/phi_py) - x - y/2
        b = y/2
        c = 3.76 * ar * (1/phi_py)
        d = 0
        e = 2*x + y/2 - (2*ar)/phi_py

T_guess = 1500
tol = 1e-6
```

```
alpha = 0.2
ct = 0

while (abs(f(T_guess, H_loss)) > tol):
        T_guess = T_guess - alpha * (f(T_guess, H_loss) / fprime(T_guess))
        ct = ct + 1

print('Temperature = ', T_guess)
```

4. OUTPUT:



```
Temperature =  1679.3701620469685
[Finished in 1.0s]
```

CASE 4 - VARIATION OF ADIABATIC FLAME TEMPERATURE FOR DIFFERENT HYDROCARBONS

1. PROCESS:

- Combustion of hydrocarbon in a Constant Pressure Chamber.
- Equivalence Ratio, $\phi = 1$
- Heat Loss Fraction = 0.35

2. HYDROCARBON:

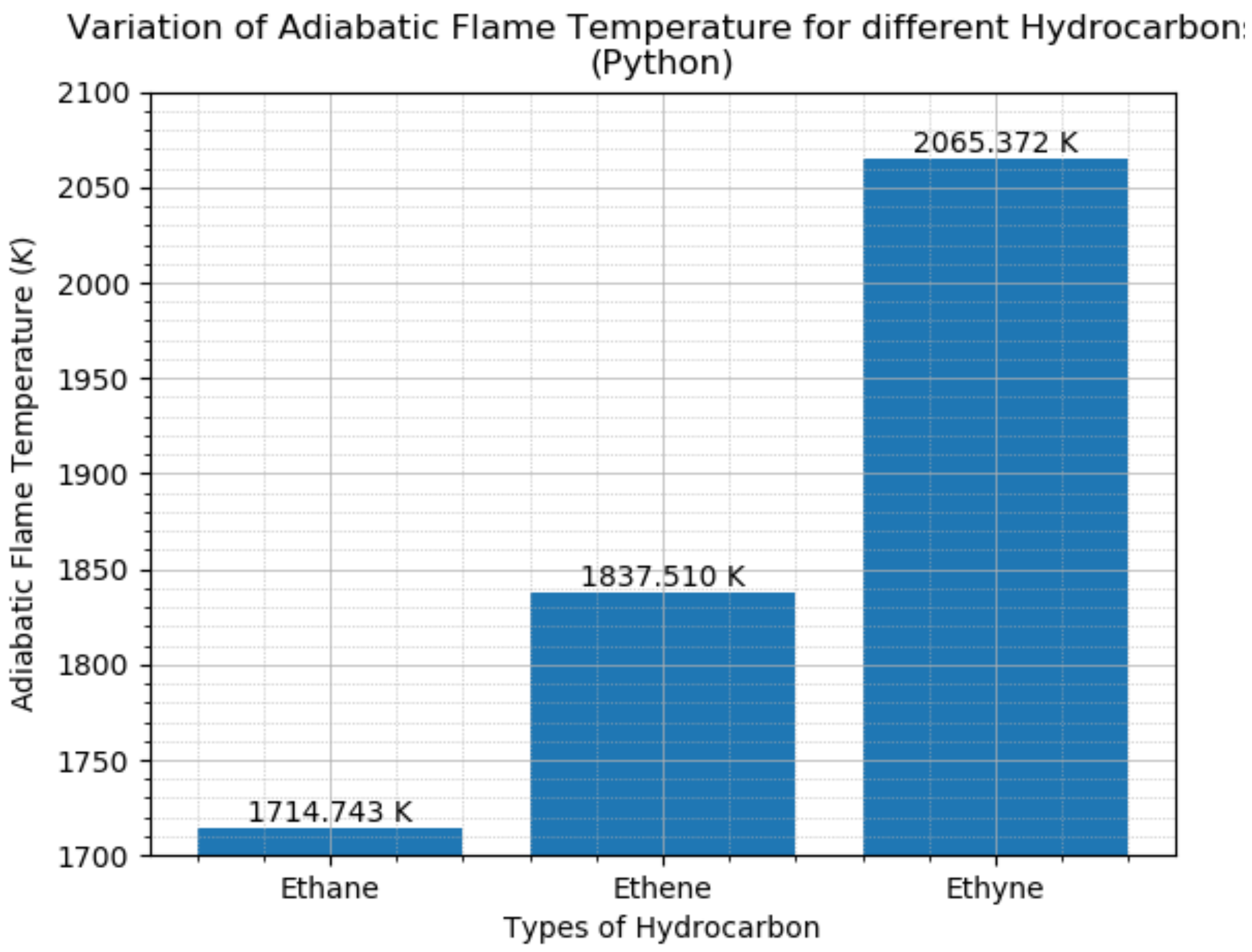
- **Ethane:** x = 2; y = 6
- **Ethene:** x = 2; y = 4
- **Ethyne:** x = 2; y = 2

3. CODE

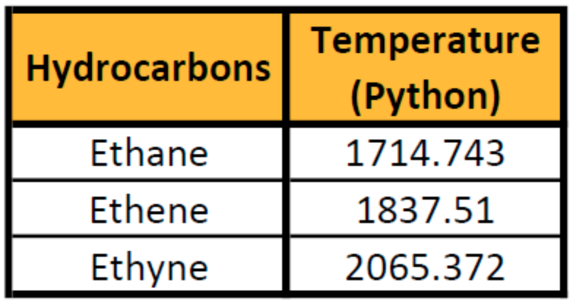
```
"""
        In this program we are going to analyse the variation of AFT for differ

        General Equation:
                CxHy + g(O2 + 3.76N2) = aCO2 + bH2O + cN2 + dO2 + eCO
                x = 1; y = 4; ar = x + y/4; g = ar * (1/phi)

        Stoichiometric Equation (phi = 1):
                CxHy + ar(O2 + 3.76N2) = xCO2 + (y/2)H2O + 3.76(x + y/4)N2
        Lean Mixture Equation (phi < 1):
                CxHy + ar(1/phi)(O2 + 3.76N2) = xCO2 + (y/2)H2O + 3.76(ar/phi)N
        Rich Mixture Equation (phi > 1):
                CxHy + ar(1/phi)(O2 + 3.76N2) = ((ar/2phi) - x - y/2)CO2 + (y/2
"""

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

R = 8.314
x = 2
y = 6
```

```
carbon_type = 1

# NASA Polynomial Constants

c2h6_coeffs_l = [4.29142492E+00, -5.50154270E-03, 5.99438288E-05, -7.08466285E-
c2h4_coeffs_l = [3.95920148E+00, -7.57052247E-03, 5.70990292E-05, -6.91588753E-
c2h2_coeffs_l = [8.08681094E-01, 2.33615629E-02, -3.55171815E-05, 2.80152437E-0

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

n2_coeffs_h = [0.02926640E+02, 0.14879768E-02, -0.05684760E-05, 0.10097038E-09,
co2_coeffs_h = [3.85746029E+00, 4.41437026E-03, -2.21481404E-06, 5.23490188E-10
h2o_coeffs_h = [3.03399249E+00, 2.17691804E-03, -1.64072518E-07, -9.70419870E-1
o2_coeffs_h =  [3.78245636E+00, -2.99673416E-03, 9.84730201E-06, -9.68129509E-0
co_coeffs_h = [2.71518561E+00, 2.06252743E-03, -9.98825771E-07, 2.30053008E-10,


# Function to evaluate Enthalpy

def h(T, co_effs):

        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


# Function that represents the root finding problem

def f(T, H_loss):

        # Products At Temperature T

        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_products = a*h_co2_p + b*h_h2o_p + c*h_n2_p + d*h_o2_p + e*h_co_p

        # Products At Standard Temperature

        T_std = 298.15
        h_co2_p_std = h(T_std, co2_coeffs_h)
        h_h2o_p_std = h(T_std, h2o_coeffs_h)
        h_n2_p_std = h(T_std, n2_coeffs_h)
        h_o2_p_std = h(T_std, o2_coeffs_h)
        h_co_p_std = h(T_std, co_coeffs_h)
        H_products_std = a*h_co2_p_std + b*h_h2o_p_std + c*h_n2_p_std + d*h_o2_

        # Reactants At Standard Temperature

        if carbon_type == 1:
                h_hydrocarbon_r = h(T_std, c2h6_coeffs_l)
        elif carbon_type == 2:
                h_hydrocarbon_r = h(T_std, c2h4_coeffs_l)

        elif carbon_type == 3:
                h_hydrocarbon_r = h(T_std, c2h2_coeffs_l)

        h_o2_r = h(T_std, o2_coeffs_l)
        h_n2_r = h(T_std, n2_coeffs_l)
        H_reactants = h_hydrocarbon_r + g*h_o2_r + 3.76*g*h_n2_r

        # Lower Heating Value
        LHV = H_reactants - H_products_std

        return H_products - H_reactants + (H_loss * LHV)


# Function that represents the derivative of the root finding problem

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


# Calculating the temperature and storing the values

T_py = []

for carbon_type in range (1, 4):

        phi_py = 1
        H_loss = 0.35

        if carbon_type == 1:
                x = 2
                y = 6
        elif carbon_type == 2:
                x = 2
                y = 4
        elif carbon_type == 3:
                x = 2
                y = 2


        if (phi_py == 1):
                ar = x + y/4
                g = ar
                a = x
                b = y/2
                c = 3.76 * ar
                d = 0
                e = 0
        elif (phi_py < 1):
                ar = x + y/4
                g = ar * (1/phi_py)
                a = x
                b = y/2
                c = 3.76 * ar * (1/phi_py)
                d = ar * ((1/phi_py)-1)
                e = 0
        elif (phi_py > 1):
                ar = x + y/4
                g = ar * (1/phi_py)
                a = (2*ar/phi_py) - x - y/2
                b = y/2
                c = 3.76 * ar * (1/phi_py)
                d = 0
                e = 2*x + y/2 - (2*ar)/phi_py

        T_guess = 1500
        tol = 1e-6
        alpha = 0.2
        ct = 0

        while (abs(f(T_guess, H_loss)) > tol):
                T_guess = T_guess - alpha * (f(T_guess, H_loss) / fprime(T_gues
                ct = ct + 1

        T_py.append(T_guess)

print('Temperature (Python) = ', T_py)

# Plotting the results obtained from Python

x_bar = ['Ethane', 'Ethene', 'Ethyne']
x_pos = [i for i, _ in enumerate(x_bar)]

fig, ax = plt.subplots()
rects1 = ax.bar(x_pos, T_py)
plt.ylim(1700, 2100)
plt.xlabel("Types of Hydrocarbon")
plt.ylabel("Adiabatic Flame Temperature ($K$)")
plt.title("Variation of Adiabatic Flame Temperature for different Hydrocarbons\
                        + "(Python)")
plt.xticks(x_pos, x_bar)

plt.minorticks_on()
plt.grid(which='major', linestyle='-', linewidth='0.5')
plt.grid(which='minor', linestyle=':', linewidth='0.5')

def autolabel(rects):

    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2., height, '%0.3f K' % float(h
autolabel(rects1)

plt.show()
```

4. OUTPUT:



5. RESULTS TABLE:



Hydrocarbons	Temperature (Python)
Ethane	1714.743
Ethene	1837.51
Ethyne	2065.372

6. RESULTS:

The graph obtained by Python shows that the AFT increases as the number of bond increases from single to double and then to triple. This is because as the number of bonds increases, the energy released during the breakage of the bond also increases and there is a corresponding increase in the AFT.

CASE 5 - VARIATION OF ADIABATIC FLAME TEMPERATURE FOR

DIFFERENT ALKANES

1. PROCESS:

- Combustion of hydrocarbon in a Constant Pressure Chamber.
- Equivalence Ratio, ϕ = 1
- Heat Loss Fraction = 0

2. HYDROCARBON:

- **Methane:** x = 1; y = 4
- **Ethane:** x = 2; y = 6
- **Propane:** x = 3; y = 8

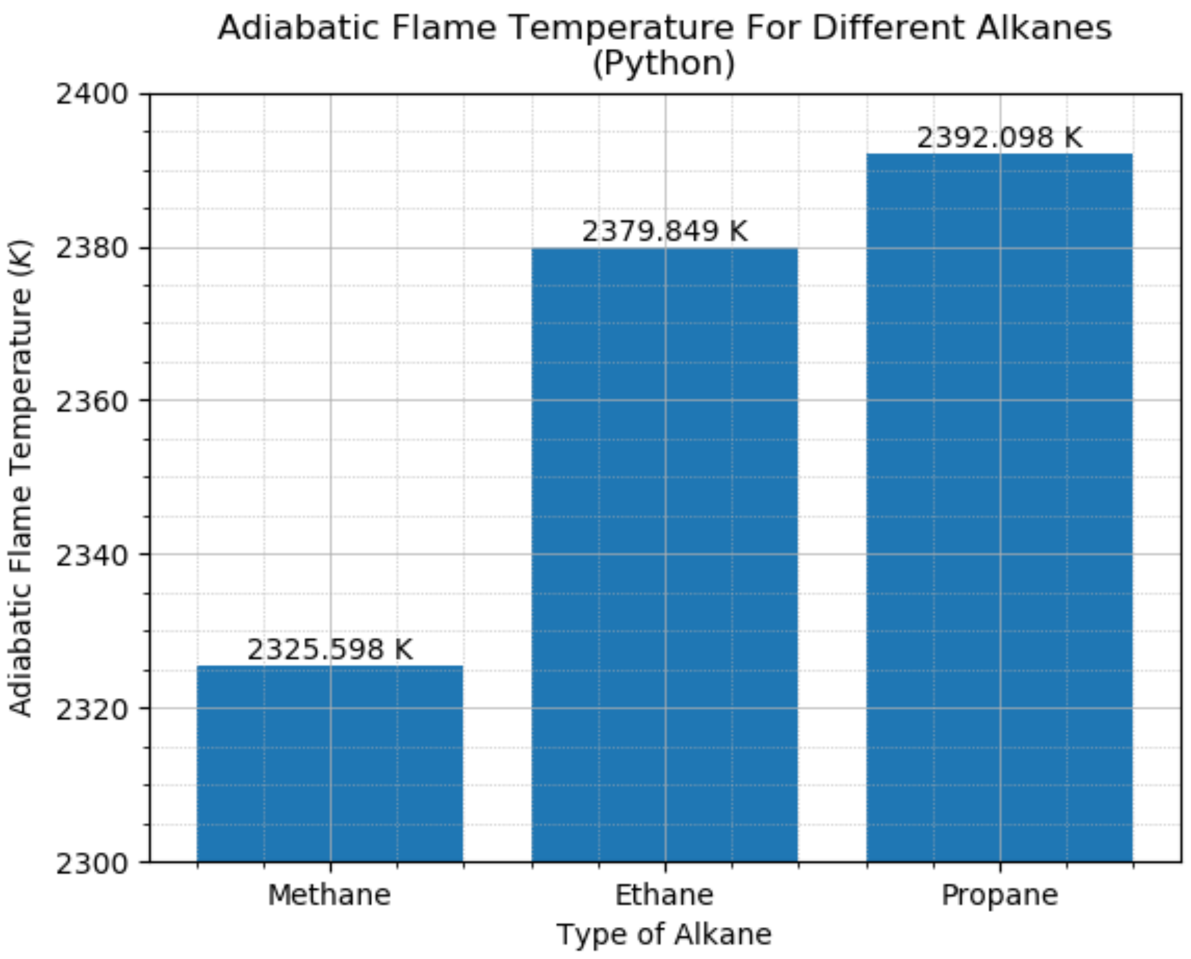
3. CODE

```
"""
        In this program we are going to analyse the variation of AFT for differ

        General Equation:
                CxHy + g(O2 + 3.76N2) = aCO2 + bH2O + cN2 + dO2 + eCO
                x = 1; y = 4; ar = x + y/4; g = ar * (1/phi)

        Stoichiometric Equation (phi = 1):
                CxHy + ar(O2 + 3.76N2) = xCO2 + (y/2)H2O + 3.76(x + y/4)N2
        Lean Mixture Equation (phi < 1):
                CxHy + ar(1/phi)(O2 + 3.76N2) = xCO2 + (y/2)H2O + 3.76(ar/phi)N
        Rich Mixture Equation (phi > 1):
                CxHy + ar(1/phi)(O2 + 3.76N2) = ((ar/2phi) - x - y/2)CO2 + (y/2
"""

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

R = 8.314
x = 2
y = 6
carbon_type = 1

# NASA Polynomial Constants

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

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

n2_coeffs_h = [0.02926640E+02, 0.14879768E-02, -0.05684760E-05, 0.10097038E-09,
co2_coeffs_h = [3.85746029E+00, 4.41437026E-03, -2.21481404E-06, 5.23490188E-10
h2o_coeffs_h = [3.03399249E+00, 2.17691804E-03, -1.64072518E-07, -9.70419870E-1
o2_coeffs_h =  [3.78245636E+00, -2.99673416E-03, 9.84730201E-06, -9.68129509E-0
co_coeffs_h = [2.71518561E+00, 2.06252743E-03, -9.98825771E-07, 2.30053008E-10,
# Function to evaluate Enthalpy

def h(T, co_effs):

        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


# Function that represents the root finding problem

def f(T, H_loss):

        # Products At Temperature T

        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_products = a*h_co2_p + b*h_h2o_p + c*h_n2_p + d*h_o2_p + e*h_co_p

        # Products At Standard Temperature

        T_std = 298.15
        h_co2_p_std = h(T_std, co2_coeffs_h)
        h_h2o_p_std = h(T_std, h2o_coeffs_h)
        h_n2_p_std = h(T_std, n2_coeffs_h)
        h_o2_p_std = h(T_std, o2_coeffs_h)
        h_co_p_std = h(T_std, co_coeffs_h)
        H_products_std = a*h_co2_p_std + b*h_h2o_p_std + c*h_n2_p_std + d*h_o2_

        # Reactants At Standard Temperature

        if carbon_type == 1:
                h_hydrocarbon_r = h(T_std, ch4_coeffs_l)

        if carbon_type == 2:
                h_hydrocarbon_r = h(T_std, c2h6_coeffs_l)

        if carbon_type == 3:
                h_hydrocarbon_r = h(T_std, c3h8_coeffs_l)

        h_o2_r = h(T_std, o2_coeffs_l)
        h_n2_r = h(T_std, n2_coeffs_l)
        H_reactants = h_hydrocarbon_r + g*h_o2_r + 3.76*g*h_n2_r

        # Lower Heating Value
        LHV = H_reactants - H_products_std

        return H_products - H_reactants + (H_loss * LHV)


# Function that represents the derivative of the root finding problem
```

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


# Calculating the temperature and storing the values

T_py = []

for carbon_type in range (1, 4):

        phi_py = 1
        H_loss = 0

        if carbon_type == 1:
                x = 1
                y = 4
        elif carbon_type == 2:
                x = 2
                y = 6
        elif carbon_type == 3:
                x = 3
                y = 8


        if (phi_py == 1):
                ar = x + y/4
                g = ar
                a = x
                b = y/2
                c = 3.76 * ar
                d = 0
                e = 0
        elif (phi_py < 1):
                ar = x + y/4
                g = ar * (1/phi_py)
                a = x
                b = y/2
                c = 3.76 * ar * (1/phi_py)
                d = ar * ((1/phi_py)-1)
                e = 0
        elif (phi_py > 1):
                ar = x + y/4
                g = ar * (1/phi_py)
                a = (2*ar/phi_py) - x - y/2
                b = y/2
                c = 3.76 * ar * (1/phi_py)
                d = 0
                e = 2*x + y/2 - (2*ar)/phi_py

        T_guess = 1500
        tol = 1e-6
        alpha = 0.2
        ct = 0

        while (abs(f(T_guess, H_loss)) > tol):
                T_guess = T_guess - alpha * (f(T_guess, H_loss) / fprime(T_gues
                ct = ct + 1

        T_py.append(T_guess)
```

```
print('Temperature (Python) = ', T_py)

# Plotting the results obtained from Python

x_bar = ['Methane', 'Ethane', 'Propane']
x_pos = [i for i, _ in enumerate(x_bar)]

fig, ax = plt.subplots()
rects1 = ax.bar(x_pos, T_py)
plt.ylim(2300, 2400)
plt.xlabel("Type of Alkane")
plt.ylabel("Adiabatic Flame Temperature ($K$)")
plt.title("Adiabatic Flame Temperature For Different Alkanes\n"
                        + "(Python)")
plt.xticks(x_pos, x_bar)

plt.minorticks_on()
plt.grid(which='major', linestyle='-', linewidth='0.5')
plt.grid(which='minor', linestyle=':', linewidth='0.5')

def autolabel(rects):

    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2., height, '%0.3f K' % float(h
autolabel(rects1)

plt.show()
```

4. OUTPUT:



5. RESULTS TABLE:

Hydrocarbons	Temperature (Python)
Methane	2325.59813
Ethane	2379.849459
Propane	2392.097596

6. RESULTS:

The graph obtained by Python shows that the AFT slightly increases as the number of carbon atoms increases. However, due to the absence of double and triple bonds in the hydrocarbon, the increase in the adiabatic flame temperature is very small.

IV. CONCLUSIONS

The effect of equivalence ratio and fractional heat loss on a combustion reaction under different conditions is observed. Also, the variation of AFT for different types of hydrocarbons is determined.