Hybrid Ramjet Propulsion System Analysis

Overview

- The objective of this notebook is to perform a chemical equilibrium analysis for a hybrid ramjet propulsion system. The analysis will involve calculating various performance parameters, such as specific impulse (Isp), combustion temperature (Tc), and mass fraction (MF), for different oxidizer-to-fuel mixture ratios (MR) and chamber pressures (Pc). The analysis will be conducted using the "rocketcea" library, along with other Python libraries for data analysis and visualization.

Libraries Used

The following Python libraries will be utilized in this analysis:

- numpy: For numerical operations and array manipulation.
- pandas: For working with data in tabular form using DataFrames.
- matplotlib: For creating static plots and visualizations.
- seaborn: For setting plot styles and accessing color palettes.
- rocketcea: For chemical equilibrium analysis and propellant properties.
- bokeh: A Python library for creating interactive and web-based data visualizations.

Oxidizer and Fuel Cards:

- The chemical composition and thermodynamic properties of the oxidizer and fuel will be defined using the "rocketcea.cea_obj" module. The oxidizer and fuel cards will include properties such as composition, enthalpy, and temperature.

Performance Parameters

The main focus of the analysis will be on calculating key performance parameters for the hybrid ramjet system. These parameters include:

- Specific Impulse (Isp): A measure of propellant efficiency, representing the thrust produced per unit mass flow rate of propellant.
- Combustion Temperature (Tc): The temperature at which the combustion process occurs in the ramjet engine.
- Mass Fraction (MF): The average molecular weight of the combustion products.

The performance parameters will be calculated for various mixture ratios (MR), hybrid ratios(H_o_f) and chamber pressures (Pc).

Plotting

The calculated results will be visualized using line plots and graphs. The following visualizations will be generated:

- Temperature vs. Mixture Ratio: A line plot showing the variation of combustion temperature with different mixture ratios at a constant chamber pressure.

- Mass Flow Rate vs. Mixture Ratio: A line plot depicting the change in mass flow rate of air with different mixture ratios for various hybrid ratios.
- Several other parameters also plotted for a range of Mixture and Hybrid ratios.

Interactive Plots:

- The use of Bokeh for interactive visualization, enabling users to hide/show specific lines in the plots by clicking on the legend items.

CSV Saving

- The calculated results will be saved to CSV files, allowing for further analysis, data sharing, and presentation of the findings.

Some terminologies

- Hybrid ratios: ratio of oxygen to paraffin wax.
- Mixture ratios: ratio of air to fuel(combustion product of hybrid).

Install RocketCEA for Chemical Equilibrium Analysis

```
! pip install rocketcea
```

```
Requirement already satisfied: rocketcea in /usr/local/lib/python3.10/dist-packages (1.1.34)
Requirement already satisfied: future in /usr/local/lib/python3.10/dist-packages (from rocketcea) (0.18.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from rocketcea) (1.22.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from rocketcea) (1.10.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from rocketcea) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->rocketcea) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib->rocketcea) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->rocketcea) (4.41.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->rocketcea) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->rocketcea) (23.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->rocketcea) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->rocketcea) (3.1.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib->rocketcea) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib->rocketcea) (1.16.0)
```

Import the necessary libraries

```
import numpy as np
import math
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
from matplotlib.widgets import Cursor
import seaborn as sns
```

```
from pylab import *
import rocketcea
```

Starting with RocketCEA

Define fuel and oxidizer, get the cea output for a given mixer ratio

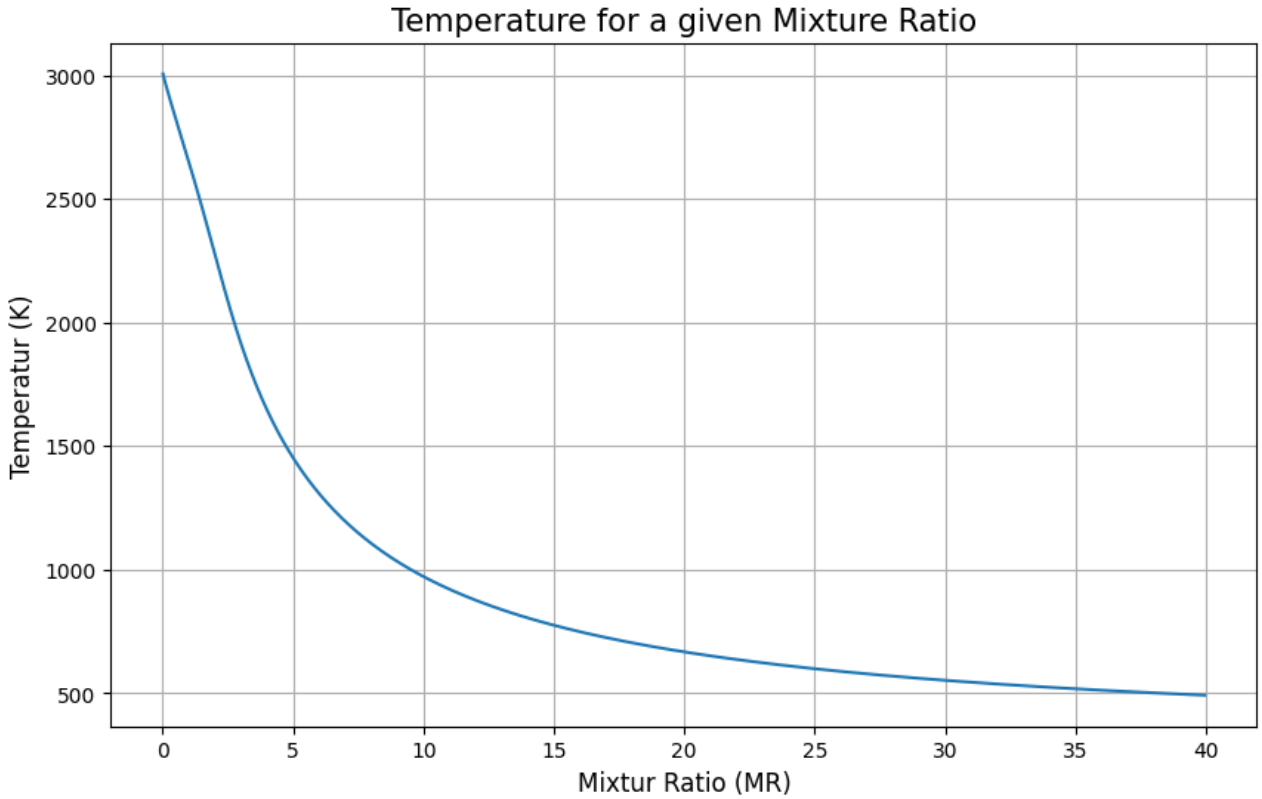
```
# Import fuel and oxidizer card
from rocketcea.cea_obj import CEA_Obj, add_new_fuel, add_new_oxidizer, add_new_propellant

# Add oxidizer
card_str = """
   oxid Air    wt%=100.00
   h,cal=-28.2     t(k)=300

   """
add_new_oxidizer( 'oxy', card_str )

# Add fuel
card_str = """
   fuel OXYGEN  O 2 wt%=81.82
   h,cal= 12.9918 t(k)=300
   fuel WAX C 26 H 54 wt%=18.18
   h,cal= -140439.7 t(k)=300

   """
add_new_fuel( 'fuel', card_str )

# Assign oxidizer and fuel name to cea_obj to process above two cards
C = CEA_Obj(oxName='oxy', fuelName="fuel")

# Get full cea output in SI units for a given mixture ratio and Chamber pressure
s = C.get_full_cea_output( Pc=10, MR=[MR for MR in range(20, 41)], frozen = 0, pc_units='bar',output='siunits')

print( s )
```

```
*N2             0.76301  0.76301
*O              0.00000  0.00131
*O2             0.20893  0.20893

 * THERMODYNAMIC PROPERTIES FITTED TO 20000.K

   PRODUCTS WHICH WERE CONSIDERED BUT WHOSE MOLE FRACTIONS
   WERE LESS THAN 5.000000E-06 FOR ALL ASSIGNED CONDITIONS

*C              *CH             CH2             CH3             CH2OH
CH3O            CH4             CH3OH           CH3OOH          *CN
CNN             *CO             COOH            *C2             C2H
C2H2,acetylene  C2H2,vinylidene CH2CO,ketene    O(CH)2O         HO(CO)2OH
C2H3,vinyl      CH3CN           CH3CO,acetyl    C2H4            C2H4O,ethylen-o
CH3CHO,ethanal  CH3COOH         OHCH2COOH       C2H5            C2H6
CH3N2CH3        C2H5OH          CH3OCH3         CH3O2CH3        CCN
CNC             OCCN            C2N2            C2O             *C3
C3H3,1-propynl  C3H3,2-propynl  C3H4,allene     C3H4,propyne    C3H4,cyclo-
C3H5,allyl      C3H6,propylene  C3H6,cyclo-     C3H6O,propylox  C3H6O,acetone
C3H6O,propanal  C3H7,n-propyl   C3H7,i-propyl   C3H8            C3H8O,1propanol
C3H8O,2propanol CNCOCN          C3O2            *C4             C4H2,butadiyne
C4H4,1,3-cyclo- C4H6,butadiene  C4H6,1butyne    C4H6,2butyne    C4H6,cyclo-
C4H8,1-butene   C4H8,cis2-buten C4H8,tr2-butene C4H8,isobutene  C4H8,cyclo-
(CH3COOH)2      C4H9,n-butyl    C4H9,i-butyl    C4H9,s-butyl    C4H9,t-butyl
C4H10,n-butane  C4H10,isobutane C4N2            *C5             C5H6,1,3cyclo-
C5H8,cyclo-     C5H10,1-pentene C5H10,cyclo-    C5H11,pentyl    C5H11,t-pentyl
C5H12,n-pentane C5H12,i-pentane CH3C(CH3)2CH3   C6H2            C6H5,phenyl
C6H5O,phenoxy   C6H6            C6H5OH,phenol   C6H10,cyclo-    C6H12,1-hexene
C6H12,cyclo-    C6H13,n-hexyl   C6H14,n-hexane  C7H7,benzyl     C7H8
C7H8O,cresol-mx C7H14,1-heptene C7H15,n-heptyl  C7H16,n-heptane C7H16,2-methylh
C8H8,styrene    C8H10,ethylbenz C8H16,1-octene  C8H17,n-octyl   C8H18,n-octane
C8H18,isooctane C9H19,n-nonyl   C10H8,naphthale C10H21,n-decyl  C12H9,o-bipheny
C12H10,biphenyl HCN             HCO             HCCN            HCCO
HNC             HNCO            HNO             HNO2            HNO3
HO2             *H2             HCHO,formaldehy HCOOH           H2O2
(HCOOH)2        NCO             *NH             NH2             NH3
NH2OH           *NO             NO2             NO3             NCN
N2H2            NH2NO2          N2H4            N2O             N2O3
N2O4            N2O5            N3              N3H             *OH
O3              C(gr)           H2O(cr)         H2O(L)

NOTE. WEIGHT FRACTION OF FUEL IN TOTAL FUELS AND OF OXIDANT IN TOTAL OXIDANTS
```

Plot Temperature vs Mixture Ratio for a Chamber pressure of 10 bar

```
T_c = []    # empty combustion temperature list
MR_lst = [] # empty mixture ratio list
MR = 0.01
while MR < 40:
  temperature = C.get_Tcomb(Pc=10, MR = MR)   # getting combustion temperature from rocketcea
```

```
  T_c.append(temperature * 5/9)
  MR_lst.append(MR)
  MR += 0.05
plt.figure(figsize = (10,6))  # figure size
sns.lineplot(x = MR_lst, y = T_c) # lineplot using seaborn
plt.title('Temperature for a given Mixture Ratio', fontsize = 15)
plt.xlabel('Mixtur Ratio (MR)', fontsize = 12)
plt.ylabel('Temperatur (K)', fontsize =12)
plt.grid()
```



Generating CEA output for various Hybrid and Mixture ratios

```
# Import fuel and oxidizer card
from rocketcea.cea_obj import CEA_Obj, add_new_fuel, add_new_oxidizer, add_new_propellant

# Add oxidizer
card_str = """
   oxid Air    wt%=100.00
```

```
   h,cal=-28.2     t(k)=300

   """
add_new_oxidizer( 'oxy', card_str )

# Add fuel
lst1 = [84, 85, 86, 89, 92]
lst2 = [16, 15, 14, 11, 8]
for i in range(len(lst1)):
  card_str = """
   fuel OXYGEN  O 2     wt%=""" + str(lst1[i]) + """
   h,cal= 12.9918 t(k)=300
   fuel WAX C 26 H 54    wt%=""" + str(lst2[i]) + """
   h,cal= -140439.7 t(k)=300

   """
  add_new_fuel( 'fuel', card_str )

  # Assign oxidizer and fuel name to cea_obj to process above two cards
  C = CEA_Obj(oxName='oxy', fuelName="fuel")

  # Get full cea output in SI units for a given mixture ratio and Chamber pressure
  s = C.get_full_cea_output( Pc=1, MR=[MR for MR in range(20, 41)], frozen = 0, pc_units='bar',output='siunits')

  print( s )
```

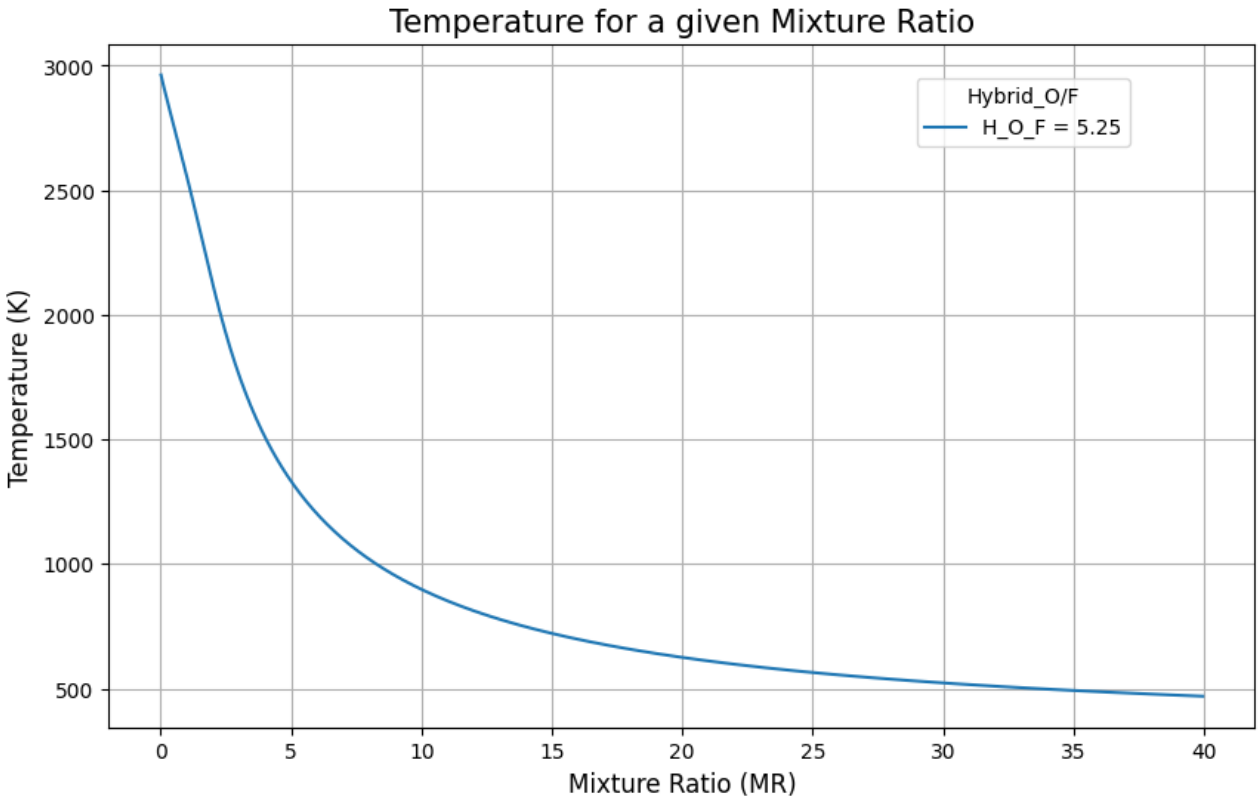
```
CNC             OCCN            C2N2            C2O             *C3
C3H3,1-propynl  C3H3,2-propynl  C3H4,allene     C3H4,propyne    C3H4,cyclo-
C3H5,allyl      C3H6,propylene  C3H6,cyclo-     C3H6O,propylox  C3H6O,acetone
C3H6O,propanal  C3H7,n-propyl   C3H7,i-propyl   C3H8            C3H8O,1propanol
C3H8O,2propanol CNCOCN          C3O2            *C4             C4H2,butadiyne
C4H4,1,3-cyclo- C4H6,butadiene  C4H6,1butyne    C4H6,2butyne    C4H6,cyclo-
C4H8,1-butene   C4H8,cis2-buten C4H8,tr2-butene C4H8,isobutene  C4H8,cyclo-
(CH3COOH)2      C4H9,n-butyl    C4H9,i-butyl    C4H9,s-butyl    C4H9,t-butyl
C4H10,n-butane  C4H10,isobutane C4N2            *C5             C5H6,1,3cyclo-
C5H8,cyclo-     C5H10,1-pentene C5H10,cyclo-    C5H11,pentyl    C5H11,t-pentyl
C5H12,n-pentane C5H12,i-pentane CH3C(CH3)2CH3   C6H2            C6H5,phenyl
C6H5O,phenoxy   C6H6            C6H5OH,phenol   C6H10,cyclo-    C6H12,1-hexene
C6H12,cyclo-    C6H13,n-hexyl   C6H14,n-hexane  C7H7,benzyl     C7H8
C7H8O,cresol-mx C7H14,1-heptene C7H15,n-heptyl  C7H16,n-heptane C7H16,2-methylh
C8H8,styrene    C8H10,ethylbenz C8H16,1-octene  C8H17,n-octyl   C8H18,n-octane
C8H18,isooctane C9H19,n-nonyl   C10H8,naphthale C10H21,n-decyl  C12H9,o-bipheny
C12H10,biphenyl *H              HCN             HCO             HCCN
HCCO            HNC             HNCO            HNO             HNO2
HNO3            HO2             *H2             HCHO,formaldehy HCOOH
H2O2            (HCOOH)2        *N              NCO             *NH
NH2             NH3             NH2OH           *NO             NO2
NO3             NCN             N2H2            NH2NO2          N2H4
N2O             N2O3            N2O4            N2O5            N3
N3H             *O              *OH             O3              C(gr)
H2O(cr)         H2O(L)

NOTE. WEIGHT FRACTION OF FUEL IN TOTAL FUELS AND OF OXIDANT IN TOTAL OXIDANTS
```

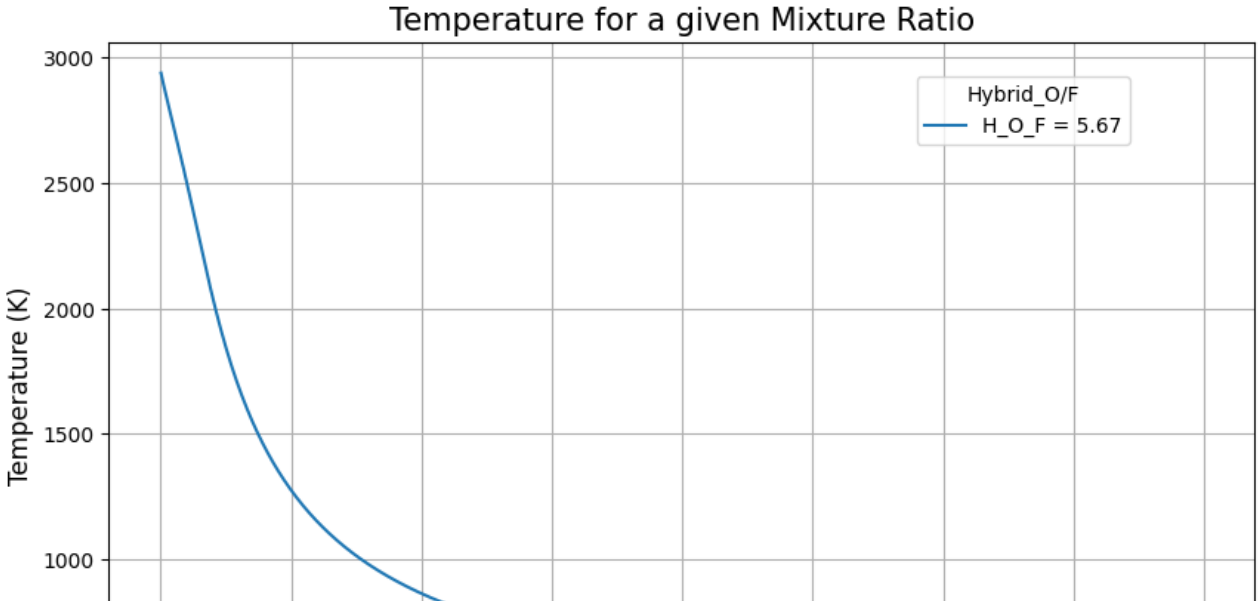
Single Plot for the teperature at each hybrid and various mixture ratios

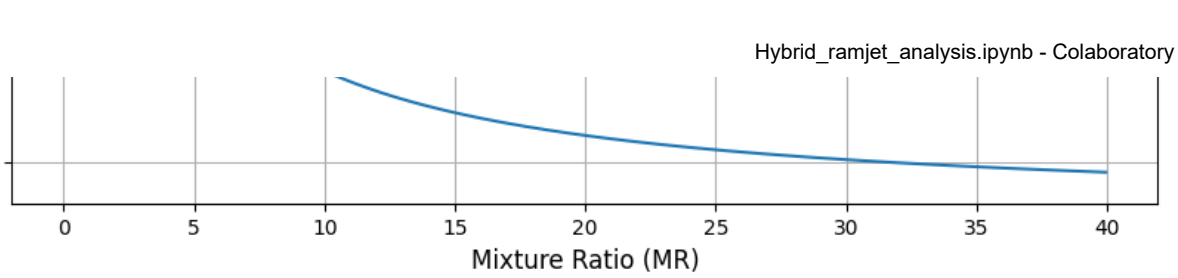
ChatGPT

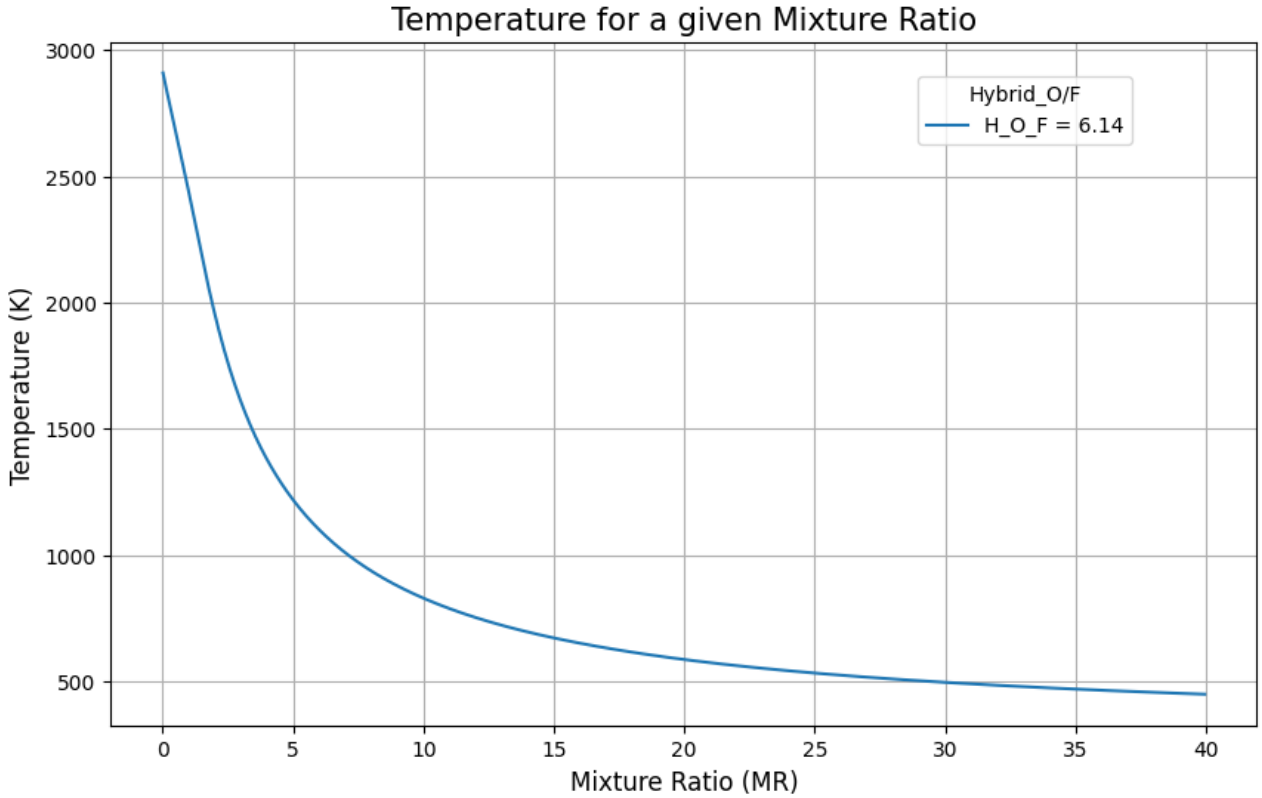
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from rocketcea.cea_obj import CEA_Obj, add_new_fuel, add_new_oxidizer, add_new_propellant

# Add oxidizer
card_str = """
   oxid Air    wt%=100.00
   h,cal=-28.2     t(k)=300

   """
add_new_oxidizer('oxy', card_str)

H_O_F = []
# Add fuel
lst1 = [84, 85, 86, 89, 92]
lst2 = [16, 15, 14, 11, 8]
for i in range(len(lst1)):
```

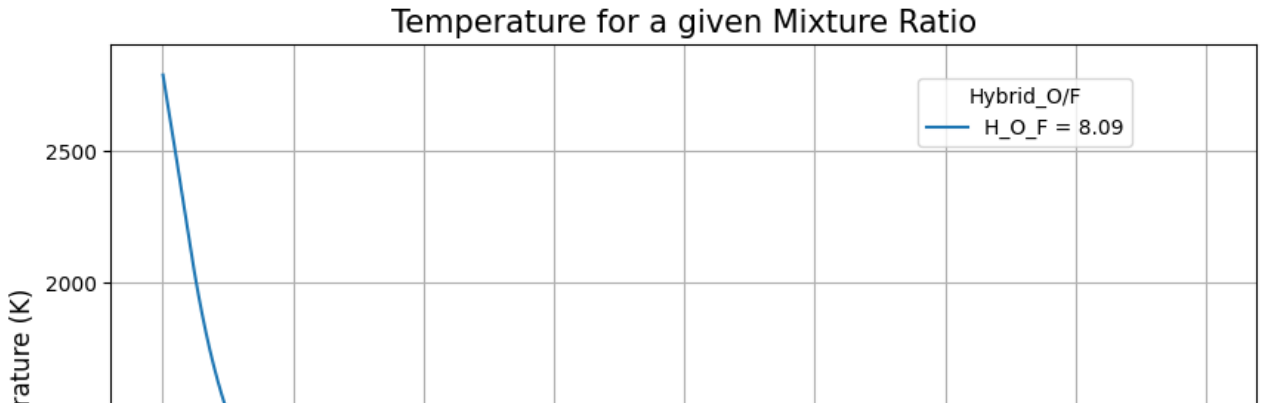
```
    card_str = """
       fuel OXYGEN  O 2     wt%=""" + str(lst1[i]) + """
       h,cal= 12.9918 t(k)=300
       fuel WAX C 26 H 54    wt%=""" + str(lst2[i]) + """
       h,cal= -140439.7 t(k)=300

       """
    add_new_fuel('fuel', card_str)

    # Assign oxidizer and fuel name to cea_obj to process above two cards
    C = CEA_Obj(oxName='oxy', fuelName="fuel")

    # Get full cea output in SI units for a given mixture ratio and Chamber pressure
    s = C.get_full_cea_output(Pc=1, MR=[MR for MR in range(20, 41)], frozen=0, pc_units='bar', output='siunits')
    H_O_F.append(round(lst1[i] / lst2[i], 2))
    T_c = []
    MR_lst = []
    MR = 0.01
    while MR < 40:
        temperature = C.get_Tcomb(Pc=10, MR=MR)
        T_c.append(temperature * 5 / 9)
        MR_lst.append(MR)
        MR += 0.05
    plt.figure(figsize=(10, 6))
    sns.lineplot(x=MR_lst, y=T_c, label=f'H_O_F = {H_O_F[i]}')
    plt.title('Temperature for a given Mixture Ratio', fontsize=15)
    plt.xlabel('Mixture Ratio (MR)', fontsize=12)
    plt.ylabel('Temperature (K)', fontsize=12)
    plt.legend(title="Hybrid_O/F", bbox_to_anchor=(0.9, 0.9), loc='right')
    plt.grid()

plt.show()
```

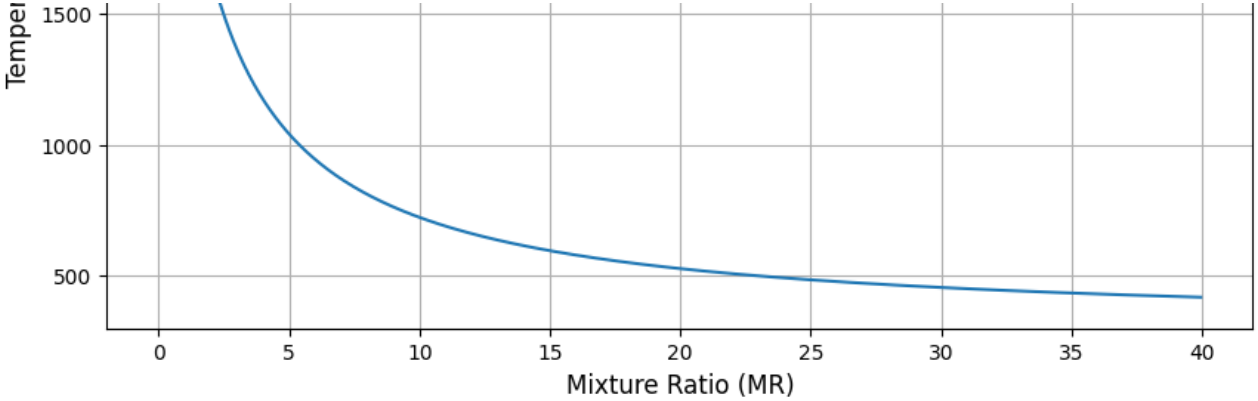

Temperature for a given Mixture Ratio
Hybrid_O/F
H_O_F = 5.25
Temperature (K)
Mixture Ratio (MR)
3000
2500
2000
1500
1000
500
0
5
10
15
20
25
30
35
40

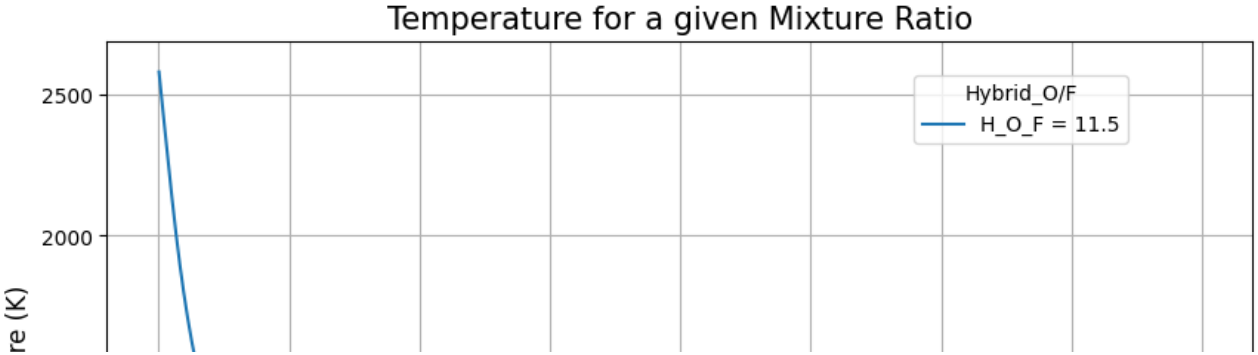

Temperature for a given Mixture Ratio
Hybrid_O/F
H_O_F = 5.67
Temperature (K)
3000
2500
2000
1500
1000


500
0
5
10
15
20
25
30
35
40
Mixture Ratio (MR)


Temperature for a given Mixture Ratio
Hybrid_O/F
H_O_F = 6.14
3000
2500
2000
1500
1000
500
Temperature (K)
0
5
10
15
20
25
30
35
40
Mixture Ratio (MR)


Temperature for a given Mixture Ratio
Hybrid_O/F
H_O_F = 8.09
2500
2000
rature (K)





Combustion temperature for Mixture of hybrid and air

Check temperature at various hybrid ratios

```
# Import fuel and oxidizer card
from rocketcea.cea_obj import CEA_Obj, add_new_fuel, add_new_oxidizer, add_new_propellant

# Add oxidizer
card_str = """
   oxid Air    wt%=100.00
   h,cal=-28.2     t(k)=300

   """
add_new_oxidizer( 'oxy', card_str )

# Add fuel
T_C = []
lst1 = []
lst2 = []
O_F = []
```

```
# Iterating for a range of oxidizer to fuel mixture (these are hybrid ratios)
for o_f in np.linspace(1, 5, 20):
  m_f = 100 / (1 + o_f)
  m_ox = 100 - m_f
  lst1.append(m_ox)
  lst2.append(m_f)
  O_F.append(round(o_f, 2))

# Iterating fuel card for several Hybrid ratios
for i in range(len(lst1)):
  card_str = """
   fuel OXYGEN  O 2     wt%=""" + str(lst1[i]) + """
   h,cal= 12.9918 t(k)=300
   fuel WAX C 26 H 54    wt%=""" + str(lst2[i]) + """
   h,cal= -140439.7 t(k)=300

   """
  add_new_fuel( 'fuel', card_str )

  # Assign oxidizer and fuel name to cea_obj to process above two cards
  C = CEA_Obj(oxName='oxy', fuelName="fuel")

  # Get full cea output in SI units for a given mixture ratio and Chamber pressure
  #s = C.get_full_cea_output( Pc=1, MR=[MR for MR in range(20, 41)], frozen = 0, pc_units='bar',output='siunits')


  MR = 0.01
  T_c = []
  MR_lst = []
  while MR < 40:
    temperature = C.get_Tcomb(Pc=10, MR = MR)
    T_c.append(temperature * 5/9)
    MR_lst.append(MR)
    MR += 0.05
  T_C.append(T_c)
print(T_C)
# print(O_F)
# print(MR_lst)
```

```
[[1506.5028225660526, 1512.5591193733542, 1518.4628380781278, 1524.1462061130874, 1529.6076779475566, 1534.8546656166936, 1554.4922911121437, 1595.5388393245198, 1634.654886541
```

Transpose the temperature cells and build the DataFrame

```
df_t = np.transpose(T_C)
df = pd.DataFrame(data = df_t, index = MR_lst, columns = O_F)
```

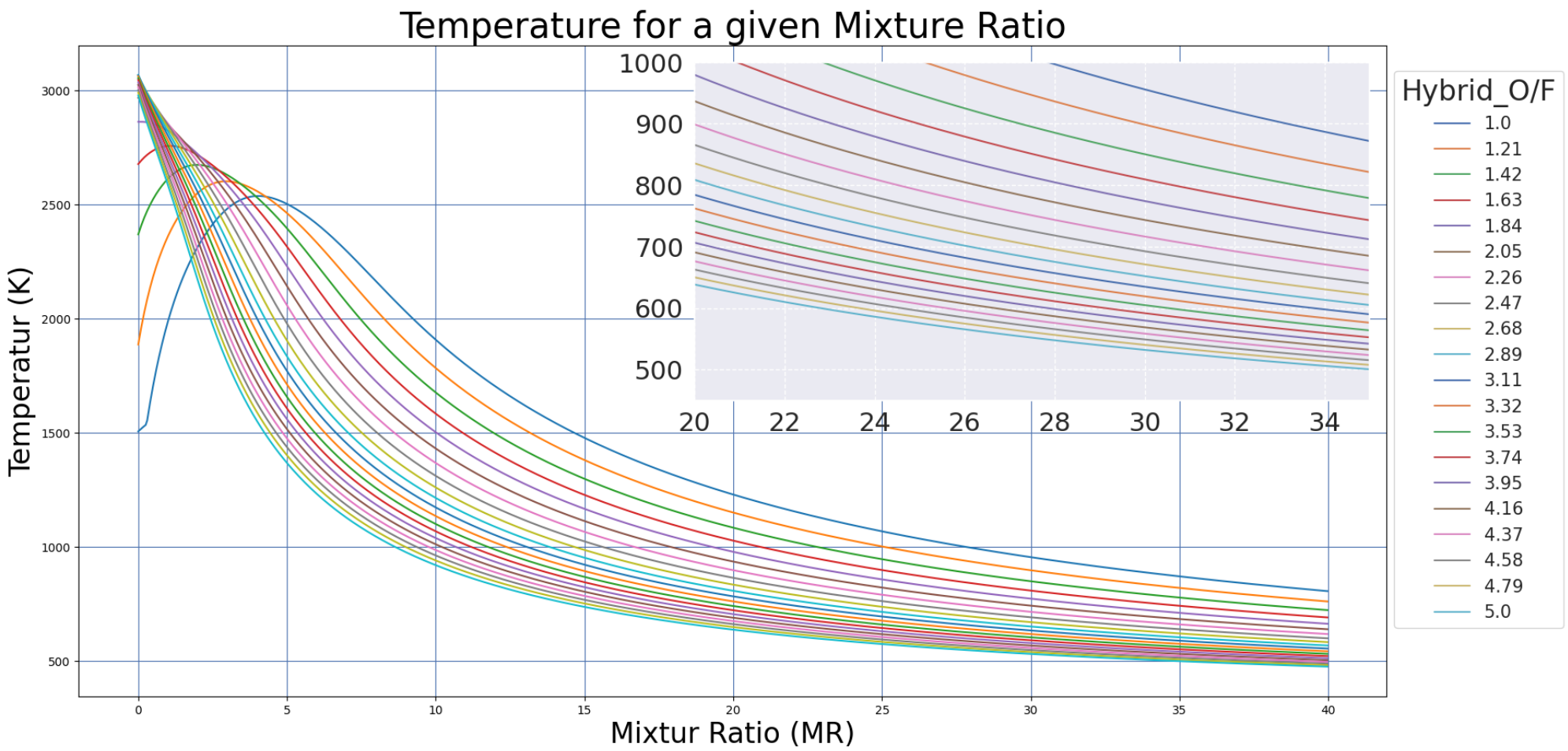
```
df = df.rename_axis(index='MR', columns="Hybrid_O/F")
df
```

Hybrid_O/F	1.00	1.21	1.42	1.63	1.84	2.05	2.26	2.47	2.68	2.89	3.11	3.32	3.53	
MR														
0.01	1506.502823	1887.123140	2369.204887	2677.174992	2862.201047	2967.160156	3022.841778	3050.926740	3063.683952	3067.558006	3066.015639	3061.019615	3053.740047	3
0.06	1512.559119	1925.266776	2389.142953	2685.052882	2862.190011	2961.381306	3013.255451	3038.884405	3049.952712	3052.563780	3050.003603	3044.137379	3036.079962	3
0.11	1518.462838	1961.129118	2407.846139	2692.503198	2862.205952	2956.021087	3004.351263	3027.668187	3037.127116	3038.525151	3034.984044	3028.278744	3019.472028	3
0.16	1524.146206	1994.891005	2425.421832	2699.481317	2862.025825	2950.699071	2995.662313	3016.775613	3024.690301	3024.916799	3020.423147	3012.899388	3003.358995	2
0.21	1529.607678	2026.715160	2441.967774	2705.996339	2861.610341	2945.340703	2987.090491	3006.096240	3012.525300	3011.617424	3006.195825	2997.870349	2987.607700	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
39.76	809.521089	764.089220	726.074103	693.791440	666.033647	641.911456	620.755007	602.049823	585.394063	570.469239	557.019530	544.837199	533.751708	
39.81	808.939827	763.555953	725.581435	693.333603	665.606042	641.510354	620.377336	601.693018	585.055963	570.148000	556.713569	544.545146	533.472367	
39.86	808.359886	763.023905	725.089900	692.876823	665.179429	641.110185	620.000546	601.337050	584.718657	569.827518	556.408331	544.253784	533.193689	
39.91	807.781263	762.493072	724.599492	692.421095	664.753803	640.710947	619.624636	600.981915	584.382144	569.507791	556.103813	543.963111	532.915672	
39.96	807.203953	761.963450	724.110209	691.966417	664.329162	640.312636	619.249601	600.627610	584.046419	569.188815	555.800014	543.673126	532.638313	

800 rows × 20 columns

Plot all Hybrid ratios in one plot

```
plt.figure(figsize = (20,10))
plt.plot(df)
plt.grid(color='b', linestyle='-')
plt.title('Temperature for a given Mixture Ratio', fontsize = 30)
plt.xlabel('Mixtur Ratio (MR)', fontsize = 24)
plt.ylabel('Temperatur (K)', fontsize =24)
sns.set(font_scale = 2)
ax2 = plt.axes([0.49, 0.46, .4, .4])
ax2.plot(df)
ax2.set_xlim([20,35])
ax2.set_ylim([450,1000])
plt.grid(color='w', linestyle='--')
plt.legend(O_F, bbox_to_anchor=(1.3, 1), loc='upper right', fontsize = 15, title = "Hybrid_O/F", facecolor = "w")
#plt.tight_layout()
plt.show()
```



Mass flow rate required

Find the mass flow rate required

```
# Generate mass flow rate for fuel, oxidizer and together
m_ox =[i for i in np.linspace(30, 210, 20)]
m_f = [m_ox / O_F for m_ox, O_F in zip(m_ox, O_F)]
m_h = [m_ox + m_f for m_ox, m_f in zip(m_ox, m_f)]
# print(m_ox)
# print(m_f)
# print(m_h)
```

```
# convert mixture ratio list to numpy array
MR_Lst = np.array(MR_lst)
m_hy = np.array(m_h)

# Reshape mixture ratio and mass flow rate of hybrid arrays
MR_Lst = np.reshape(MR_Lst, (800, 1))
MR_Lst.shape
# print(MR_Lst)

m_hy = np.reshape(m_hy, (1,20))
m_hy.shape
# print(m_hy)

# Air flow rate for a given mixture ratio and hybrid flow rate
m_a = MR_Lst*m_hy
m_a
```

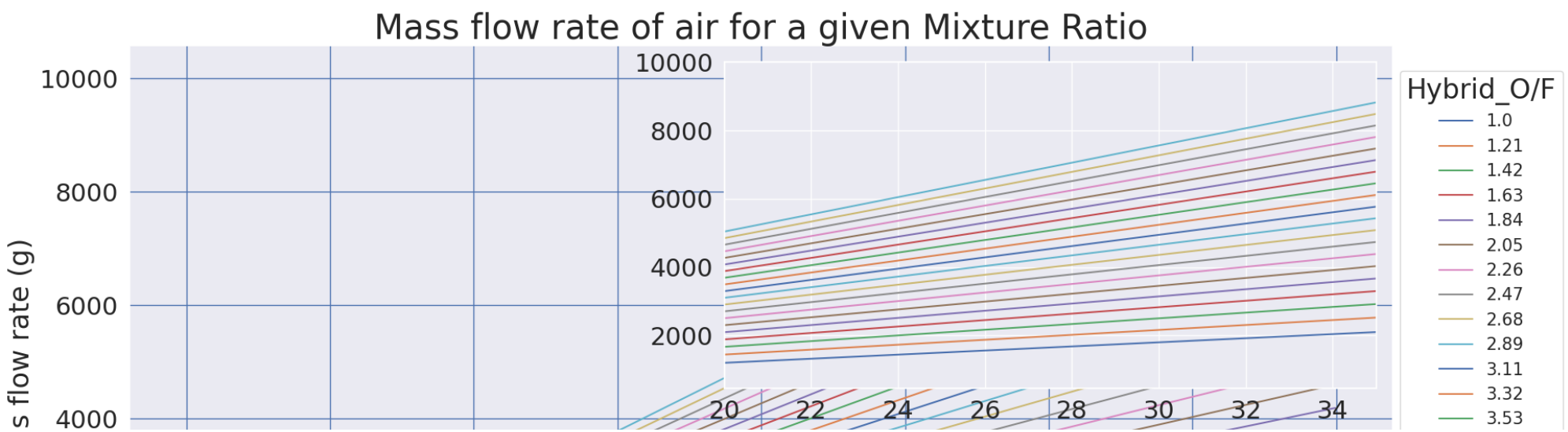
```
array([[6.00000000e-01, 7.20965637e-01, 8.34173462e-01, ...,
        2.32767180e+00, 2.42389847e+00, 2.52000000e+00],
       [3.60000000e+00, 4.32579382e+00, 5.00504077e+00, ...,
        1.39660308e+01, 1.45433908e+01, 1.51200000e+01],
       [6.60000000e+00, 7.93062201e+00, 9.17590808e+00, ...,
        2.56043898e+01, 2.66628832e+01, 2.77200000e+01],
       ...,
       [2.39160000e+03, 2.87376903e+03, 3.32501542e+03, ...,
        9.27809979e+03, 9.66165931e+03, 1.00447200e+04],
       [2.39460000e+03, 2.87737386e+03, 3.32918629e+03, ...,
        9.28973815e+03, 9.67377880e+03, 1.00573200e+04],
       [2.39760000e+03, 2.88097869e+03, 3.33335715e+03, ...,
        9.30137651e+03, 9.68589830e+03, 1.00699200e+04]])
```

```
df_ma = pd.DataFrame(data = m_a, index = MR_lst, columns = O_F)     # Create data frame for mass flow rate
#df_ma = df_ma.MultiIndex(m_h)
df_ma = df_ma.rename_axis(index='MR', columns="Hybrid_O/F")
df_ma
```

Hybrid_O/F MR	1.00	1.21	1.42	1.63	1.84	2.05	2.26	2.47	2.68	2.89	3.11	3.32	3.53	
0.01	0.6	0.720966	0.834173	0.942622	1.047941	1.151091	1.252678	1.353100	1.452632	1.551466	1.648452	1.746354	1.843880	1.94
0.06	3.6	4.325794	5.005041	5.655731	6.287643	6.906547	7.516069	8.118602	8.715789	9.308796	9.890709	10.478123	11.063277	11.64
0.11	6.6	7.930622	9.175908	10.368841	11.527346	12.662003	13.779460	14.884104	15.978947	17.066126	18.132967	19.209892	20.282675	21.35
0.16	9.6	11.535450	13.346775	15.081950	16.767048	18.417458	20.042850	21.649606	23.242105	24.823457	26.375224	27.941661	29.502072	31.05
0.21	12.6	15.140278	17.517643	19.795060	22.006751	24.172914	26.306241	28.415108	30.505263	32.580787	34.617482	36.673431	38.721470	40.76

Mass flow rate of air plot

```
ax = plt.figure(figsize = (20,10))
ax1 = ax.subplots()
ax1.plot(df_ma)
plt.grid(color='b', linestyle='-')
plt.title('Mass flow rate of air for a given Mixture Ratio', fontsize = 30)
plt.xlabel('Mixtur Ratio (MR)', fontsize = 24)
plt.ylabel('Mass flow rate (g)', fontsize =24)
sns.set(font_scale = 2)
ax2 = plt.axes([0.49, 0.46, .4, .4])
ax2.plot(df_ma)
ax2.set_xlim([20,35])
ax2.set_ylim([450,10000])
plt.grid(color='w', linestyle='-')
plt.legend(O_F, bbox_to_anchor=(1.3, 1), loc='upper right', fontsize = 15, title = "Hybrid_O/F", facecolor = "w")
#plt.tight_layout()
cursor = Cursor(ax1, horizOn = True, vertOn = True, useblit = False)
plt.show()
```



Analyzing various parameters of system

considering all important parameters in the evaluation of Hybrid

ChatGPT

```
import numpy as np
import math
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
from pylab import *
from rocketcea.cea_obj import CEA_Obj, add_new_fuel, add_new_oxidizer, add_new_propellant


card_str = """
   oxid Air    wt%=100.00
   h,cal=-28.2     t(k)=300

   """
add_new_oxidizer( 'oxy', card_str )
# Add fuel
T_C = []
lst1 = []
lst2 = []
O_F = []
Result = []

# Iterating for a range of oxidizer to fuel mixture
```

```
for o_f in np.linspace(1, 5, 20):
  m_f = 100 / (1 + o_f)
  m_ox = 100 - m_f
  lst1.append(m_ox)
  lst2.append(m_f)
  O_F.append(round(o_f, 2))

# Iterating fuel card for several Hybrid ratios
for i in range(len(lst1)):

          card_str = """
            fuel OXYGEN  O 2     wt%=""" + str(lst1[i]) + """
            h,cal= 12.9918 t(k)=300
            fuel WAX C 26 H 54    wt%=""" + str(lst2[i]) + """
            h,cal= -140439.7 t(k)=300

            """
          add_new_fuel( 'fuel', card_str )

          from rocketcea.cea_obj_w_units import CEA_Obj
          C = CEA_Obj(oxName='oxy', fuelName="fuel",isp_units='sec', cstar_units='m/s', pressure_units='Bar', temperature_units='K', sonic_velocity_units='m/s', enthalpy_units='BTU/
          n = []
          e = []
          t = []
          result = []
          H_o_f = lst1[i]/lst2[i]
          def show_perf( Pc=10, eps=1, MR=1.0 ):

              IspVac, Cstar, Tc, MW, gamma = C.get_IvacCstrTc_ChmMwGam(Pc=Pc, MR=MR, eps=eps)          # Performance parameters
              molWtD, massFracD = C.get_SpeciesMassFractions(Pc=Pc, eps=eps, MR=MR)                    # Gives the set containing species mol weight and mass fraction
              s = list(massFracD.items())                                                             # Gives the list of massFrac of species
              for specie in s:                # iterating over each species

                oxy_spe = "*O2"

                if oxy_spe in specie:         # Considering only oxygen species
                  oxy_spe = specie

                else:
                  oxy_spe = [0,[0,0,0,0]]

              l = oxy_spe[0]                  # Oxygen string
              m_F_cc = oxy_spe[1][0]          # mass fraction of oxygen
                                              # Store the values in result for a given hybrid ratio and mixture ratio
              result.append([Pc, eps, round(MR, 2), round(H_o_f,2), round(IspVac, 2), round(Cstar, 2), round(Tc, 2), round(MW, 2), gamma, m_F_cc])

          Pc = 10
          eps = 1
          MR_lst = []
          for MR in [1.0 + i*0.1 for i in range(400)]:            # For every MR show_performance is called
            show_perf( Pc=Pc, eps=eps, MR=MR )
```

```
            MR_lst.append(MR)
          Result.append(result)                                  # For result to Result(contains all mixture ratios for each Hybrid ratio i.e for one loop of hybrid ratio)
                                                                 # Result [ result(hybrid_ratio1([[MR_1], [MR_2]]), hybrid_ratio2([[MR_1], [MR_2]])) ]
print('result', result)

MR_Lst = MR_lst
print('Result', Result)
```

```
result [[10, 1, 1.0, 5.0, 172.73, 1370.22, 2730.44, 29.49, 1.1531904056569735, 0.23102314502295032], [10, 1, 1.1, 5.0, 170.5, 1351.33, 2673.56, 29.55, 1.1597813498916645, 0.236
Result [[[10, 1, 1.0, 1.0, 174.51, 1365.47, 1989.0, 20.09, 1.2808104231427178, 0], [10, 1, 1.1, 1.0, 174.7, 1367.57, 2034.55, 20.53, 1.2768426751473334, 0], [10, 1, 1.2, 1.0, 1
```

```
import pandas as pd

# Result is a list of lists where each inner list represents a row of data
# Convert the Result list to a DataFrame
df = pd.DataFrame([item for sublist in Result for item in sublist], columns=['Pc(psia)', 'eps', 'MixtureRatio', 'H_o_f', 'IspVac(sec)', 'Cstar(m/sec)', 'Tc(k)', 'MolWt', 'gamma', 'O

# Save the DataFrame to a CSV file
df.to_csv('result.csv', index=False)
```

```
df
```

	Pc(psia)	eps	MixtureRatio	H_o_f	IspVac(sec)	Cstar(m/sec)	Tc(k)	MolWt	gamma	O2_MF
0	10	1	1.0	1.0	174.51	1365.47	1989.00	20.09	1.280810	0.000000
1	10	1	1.1	1.0	174.70	1367.57	2034.55	20.53	1.276843	0.000000
2	10	1	1.2	1.0	174.86	1369.29	2077.05	20.94	1.273128	0.000000
3	10	1	1.3	1.0	174.98	1370.71	2116.80	21.34	1.269632	0.000000
4	10	1	1.4	1.0	175.07	1371.89	2154.07	21.72	1.266326	0.000000
...	...	...	...	...	...	...	...	...	...	...
7995	10	1	40.5	5.0	69.72	539.77	474.31	29.01	1.386055	0.232077
7996	10	1	40.6	5.0	69.69	539.54	473.90	29.01	1.386102	0.232076
7997	10	1	40.7	5.0	69.66	539.30	473.49	29.01	1.386149	0.232074
7998	10	1	40.8	5.0	69.63	539.06	473.08	29.01	1.386196	0.232072
7999	10	1	40.9	5.0	69.60	538.83	472.67	29.01	1.386243	0.232071

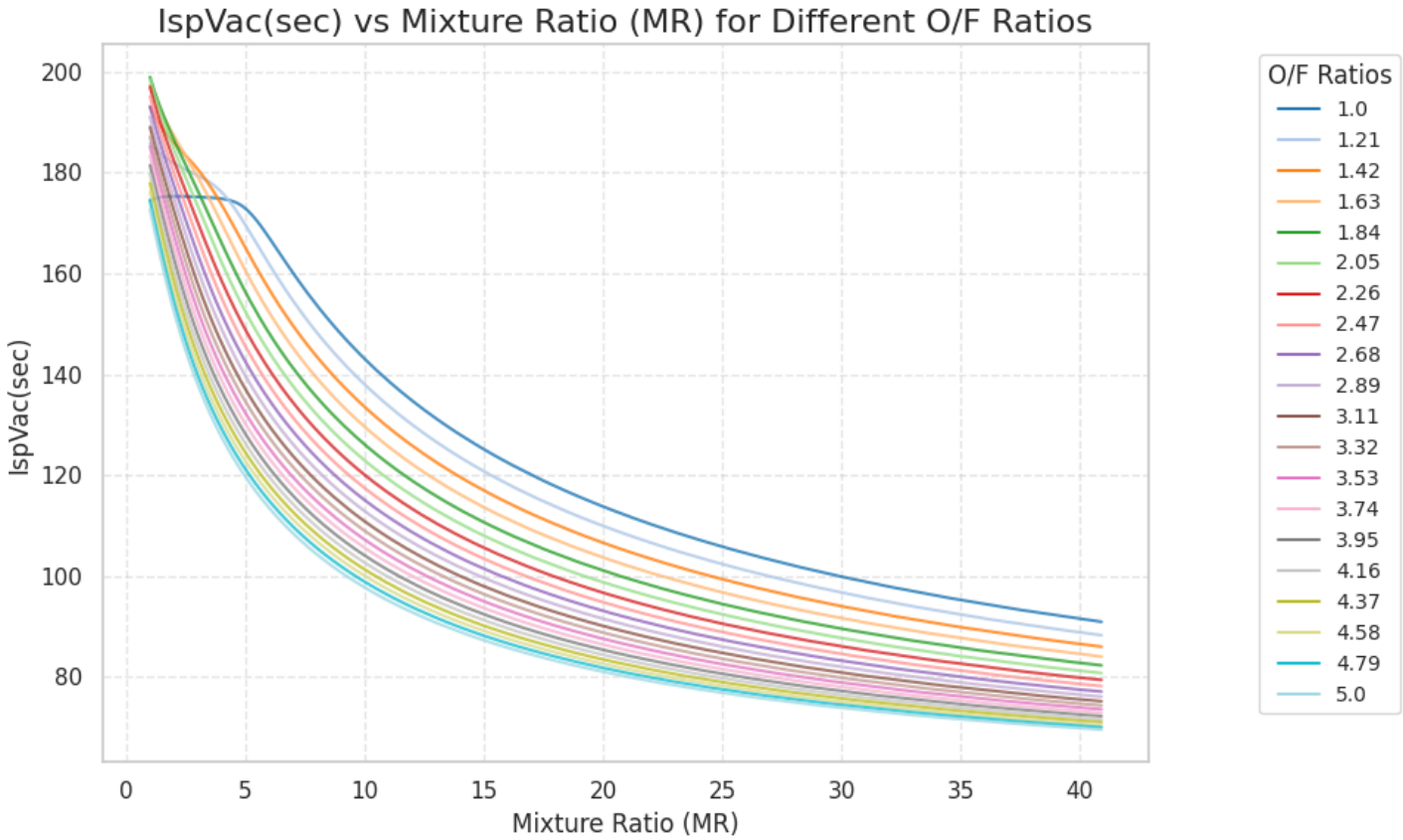
8000 rows × 10 columns

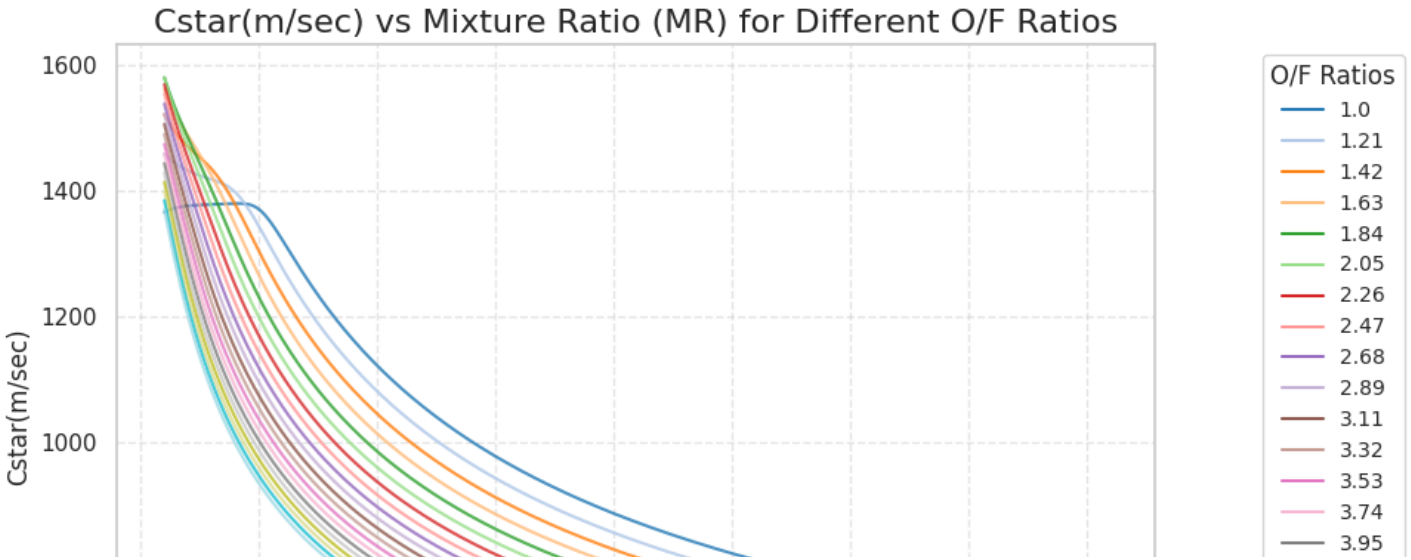
```
# # For Short form data
# import pandas as pd

# # we have already calculated and stored the 'Result' list and 'O_F' list

# # Convert the 'Result' list to a pandas DataFrame
# columns = ['Pc(psia)', 'eps', 'MixtureRatio', 'H_o_f', 'IspVac(sec)', 'Cstar(m/sec)', 'Tc(k)', 'MolWt', 'gamma', 'comp']
# df = pd.DataFrame([item for sublist in Result for item in sublist], columns=columns)

# # Convert 'H_o_f' column to float data type
# df['H_o_f'] = df['H_o_f'].astype(float)

# # Define the parameters for melting (columns to be unpivoted)
# parameters = ['IspVac(sec)', 'Cstar(m/sec)', 'Tc(k)', 'MolWt', 'gamma', 'comp']

# # Melt the DataFrame to convert it to short-form data
# df_short = pd.melt(df, id_vars=['Pc(psia)', 'eps', 'MixtureRatio', 'H_o_f'], value_vars=parameters,
#                    var_name='Parameter', value_name='Value')

# # Print the short-form DataFrame
# print(df_short)
# # Save the DataFrame to a CSV file
# df_short.to_csv('result2.csv', index=False)
```

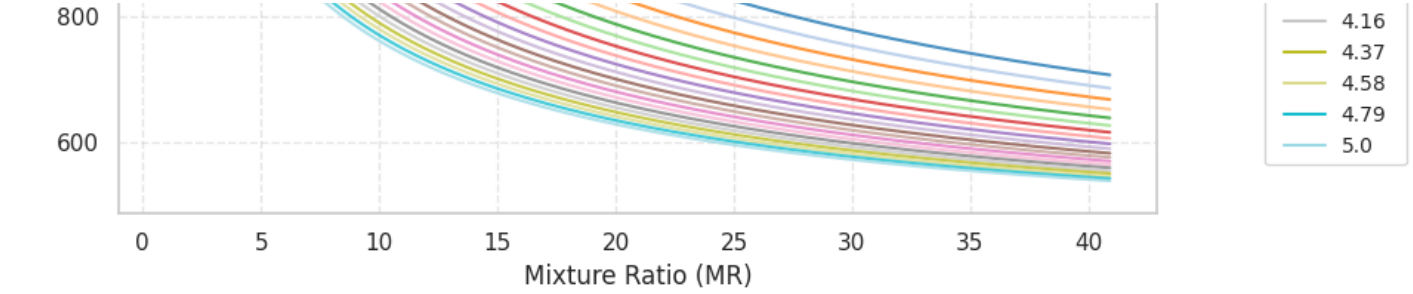
Plot the parameters for Hybrid and mixture ratio.

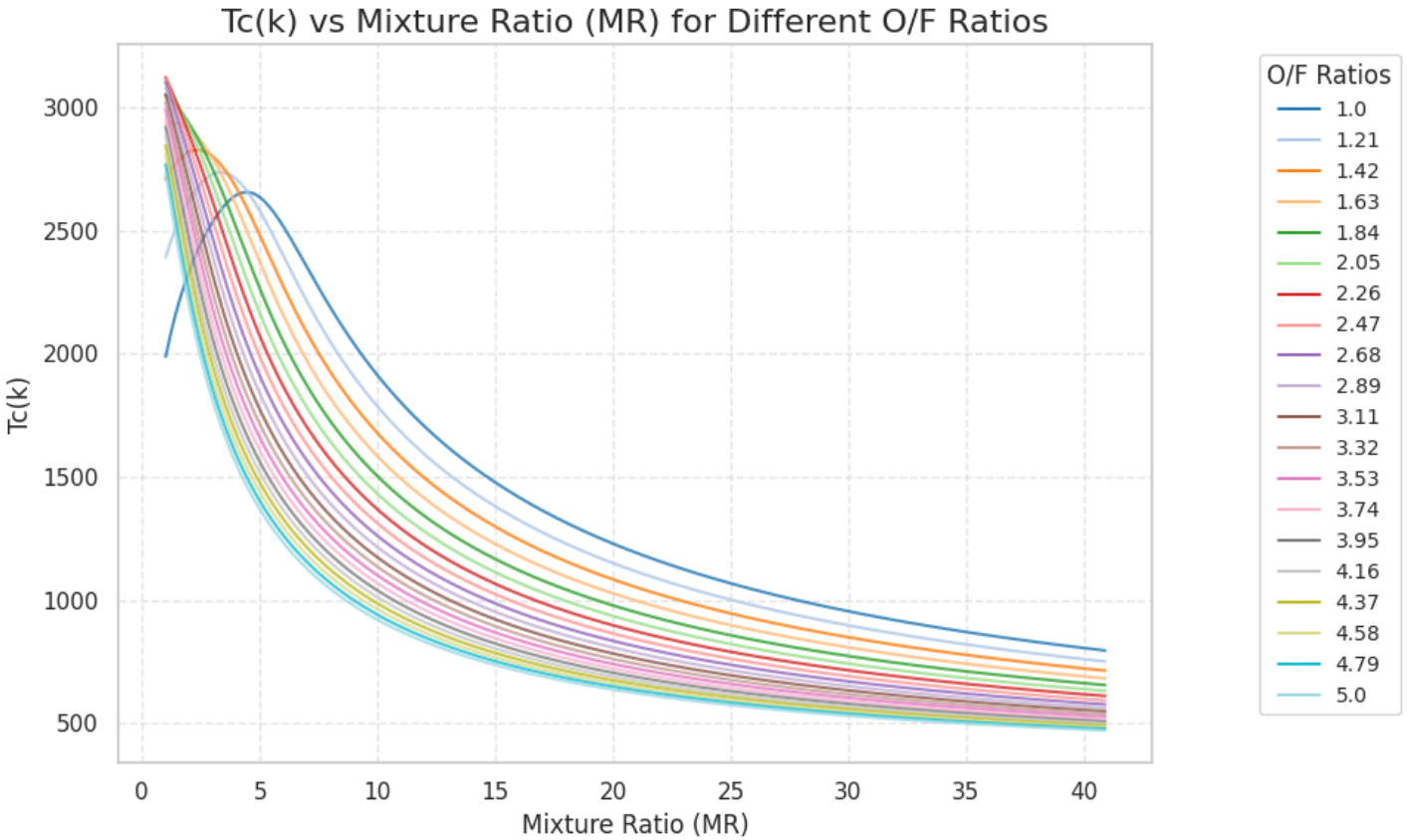
```
# Convert the 'Result' list to a pandas DataFrame
columns = ['Pc(psia)', 'eps', 'MixtureRatio', 'H_o_f', 'IspVac(sec)', 'Cstar(m/sec)', 'Tc(k)', 'MolWt', 'gamma', 'O2_MF']
df = pd.DataFrame([item for sublist in Result for item in sublist], columns=columns)


# Convert 'H_o_f' column to string data type
df['H_o_f'] = df['H_o_f'].astype(str)

# Create a list of parameter names
parameters = ['IspVac(sec)', 'Cstar(m/sec)', 'Tc(k)', 'MolWt', 'gamma', 'O2_MF']

# Set the style for the plots
sns.set(style='whitegrid')
sns.set_palette('tab20')       # 20 distinguishable colors

# Loop through each parameter and create the plots
for param in parameters:
    plt.figure(figsize=(10, 6))
    sns.lineplot(x='MixtureRatio', y=param, hue='H_o_f', data=df, markers=True, alpha=0.8)

    plt.title(f'{param} vs Mixture Ratio (MR) for Different O/F Ratios', fontsize=16)
    plt.xlabel('Mixture Ratio (MR)', fontsize=12)
```

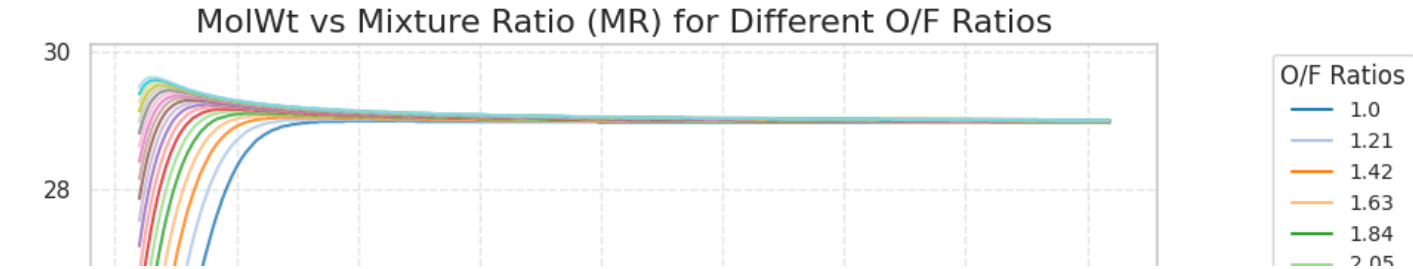
```
    plt.ylabel(param, fontsize=12)
    plt.legend(title='O/F Ratios', loc='upper right', bbox_to_anchor=(1.25, 1), fontsize=10, title_fontsize=12)
    plt.grid(True, linestyle='dashed', alpha=0.5)
    plt.tight_layout()
    plt.show()
```

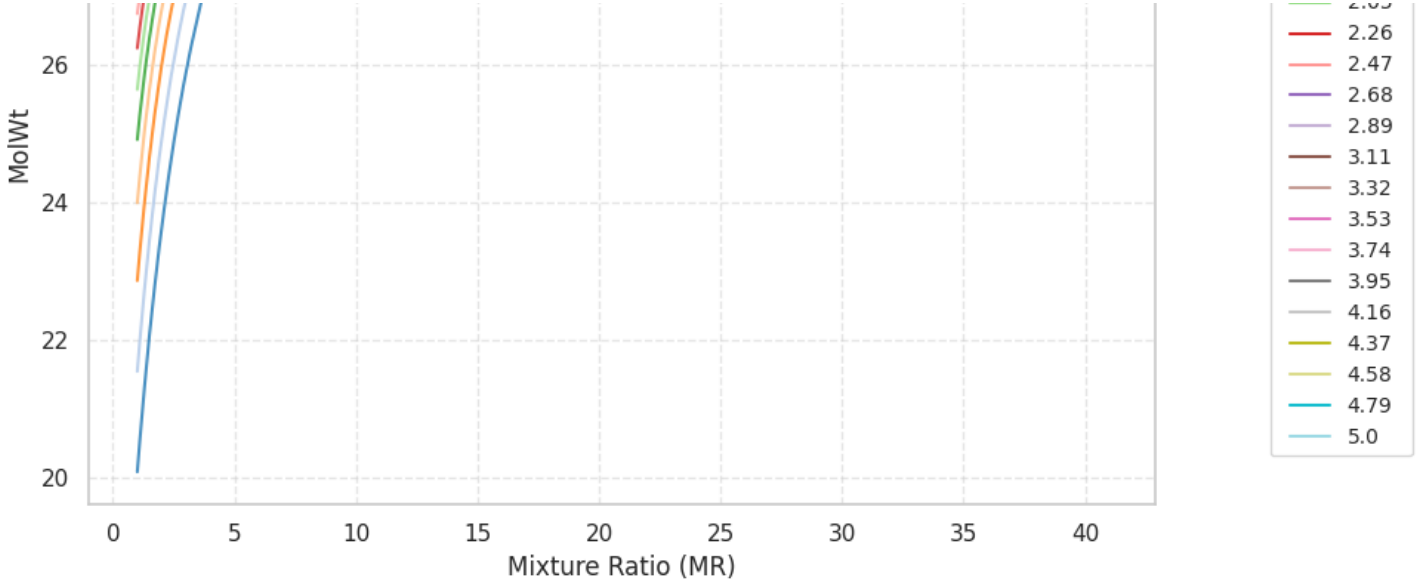

IspVac(sec) vs Mixture Ratio (MR) for Different O/F Ratios
IspVac(sec)
Mixture Ratio (MR)
O/F Ratios
1.0
1.21
1.42
1.63
1.84
2.05
2.26
2.47
2.68
2.89
3.11
3.32
3.53
3.74
3.95
4.16
4.37
4.58
4.79
5.0

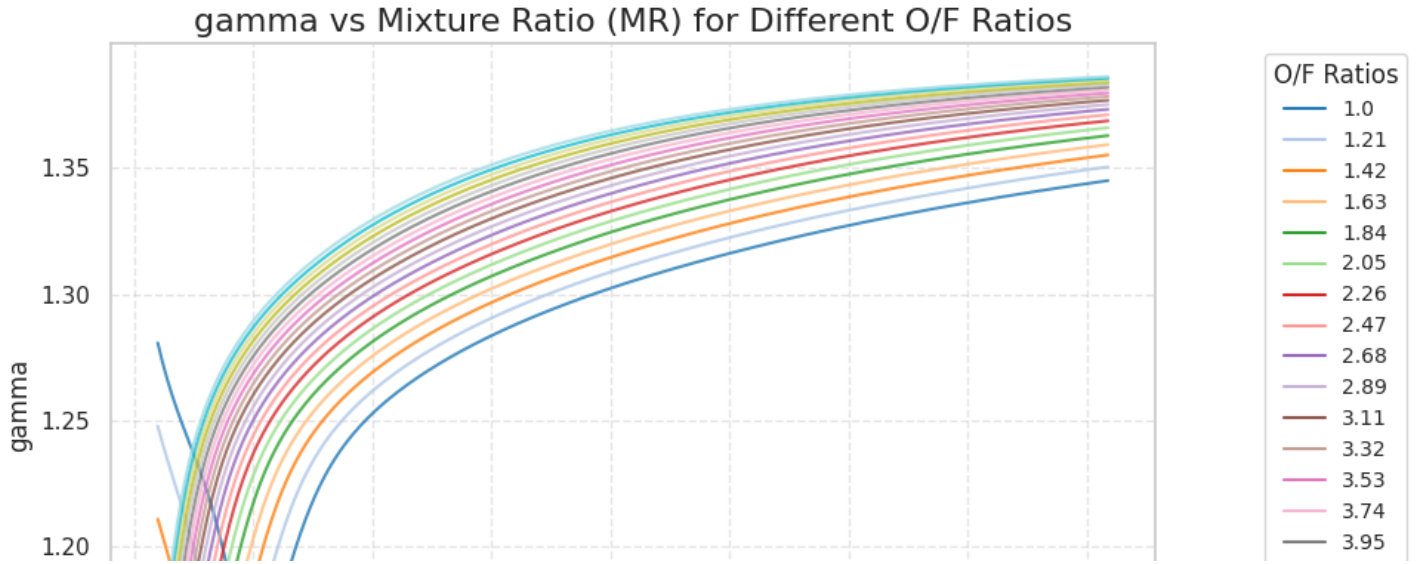

Cstar(m/sec) vs Mixture Ratio (MR) for Different O/F Ratios
Cstar(m/sec)
O/F Ratios
1.0
1.21
1.42
1.63
1.84
2.05
2.26
2.47
2.68
2.89
3.11
3.32
3.53
3.74
3.95


800
600
0
5
10
15
20
25
30
35
40
Mixture Ratio (MR)
4.16
4.37
4.58
4.79
5.0


Tc(k) vs Mixture Ratio (MR) for Different O/F Ratios
Tc(k)
3000
2500
2000
1500
1000
500
0
5
10
15
20
25
30
35
40
Mixture Ratio (MR)
O/F Ratios
1.0
1.21
1.42
1.63
1.84
2.05
2.26
2.47
2.68
2.89
3.11
3.32
3.53
3.74
3.95
4.16
4.37
4.58
4.79
5.0


MolWt vs Mixture Ratio (MR) for Different O/F Ratios
30
28
O/F Ratios
1.0
1.21
1.42
1.63
1.84

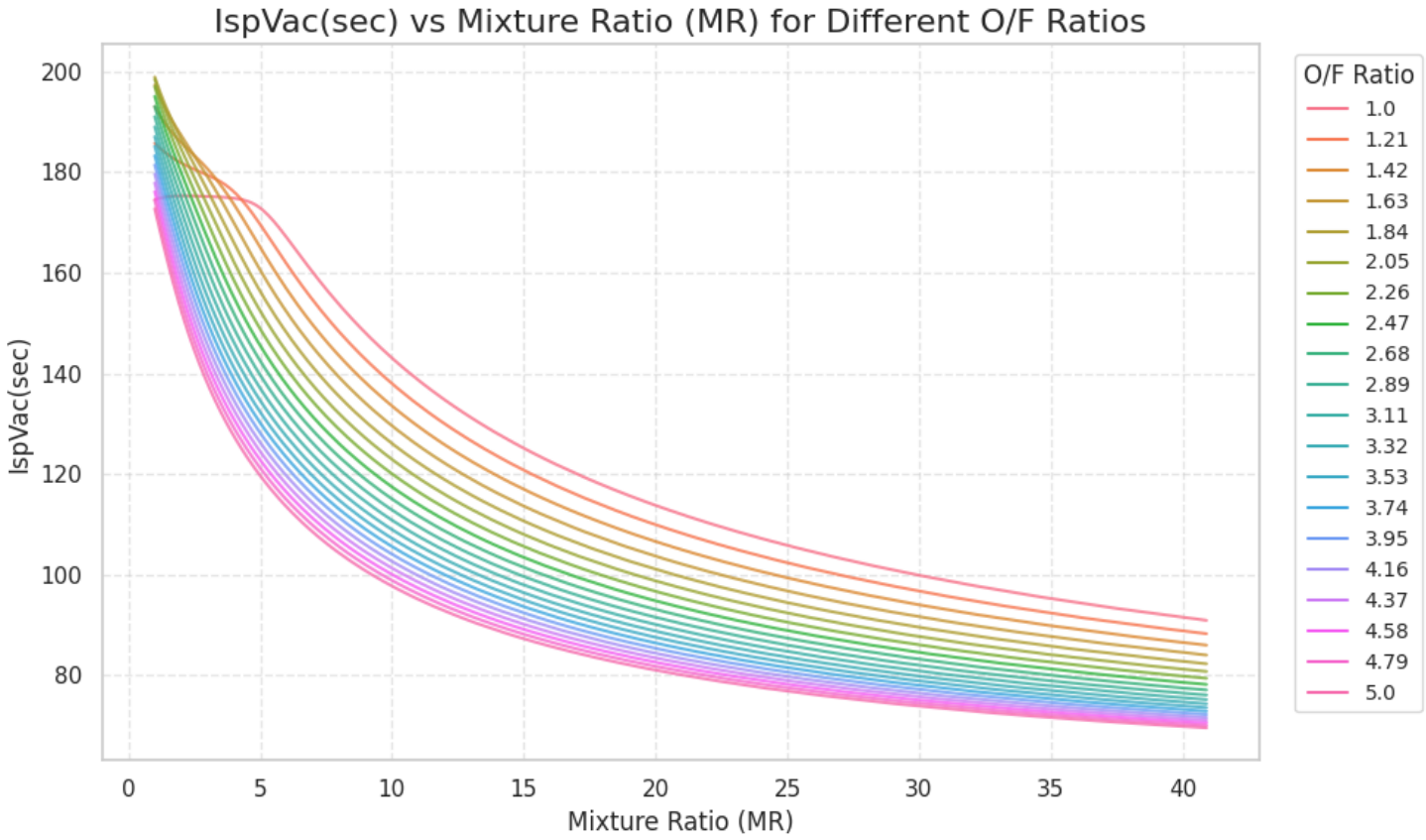


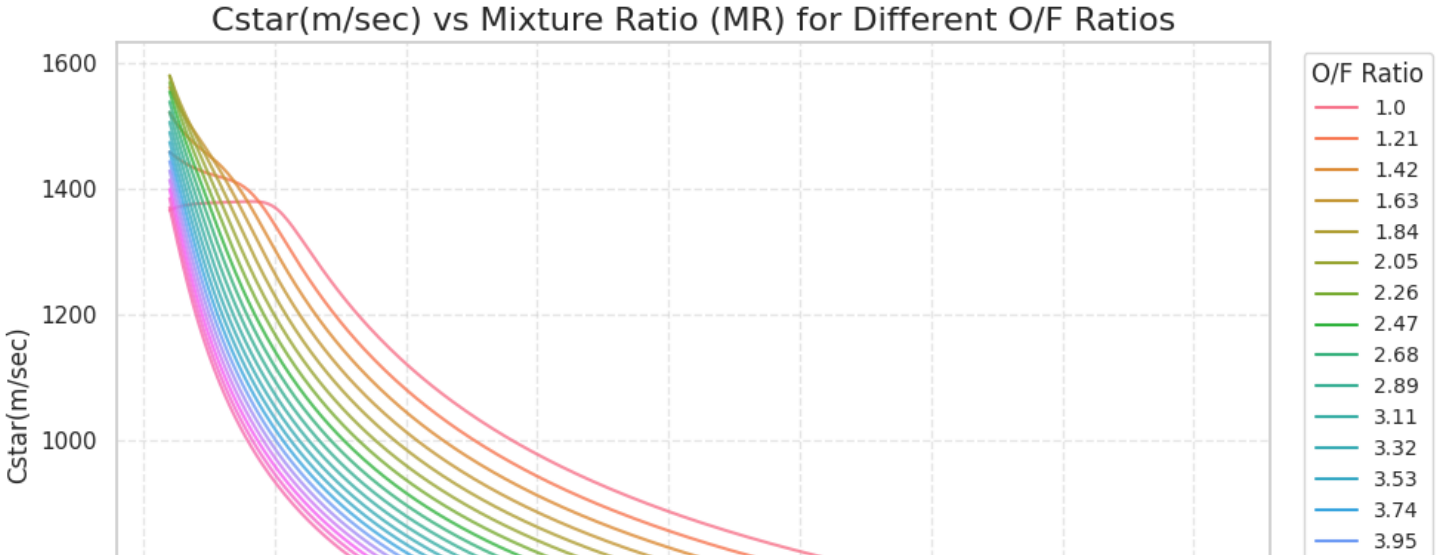


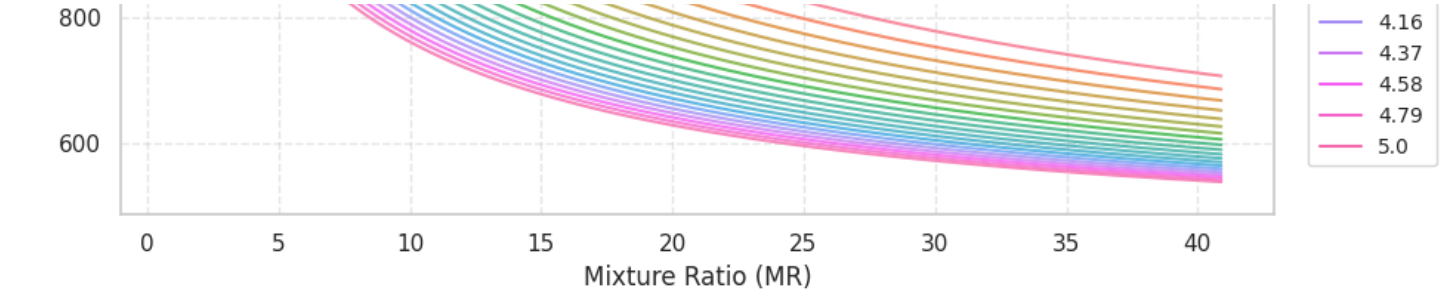
Plotting the above for a realative color style

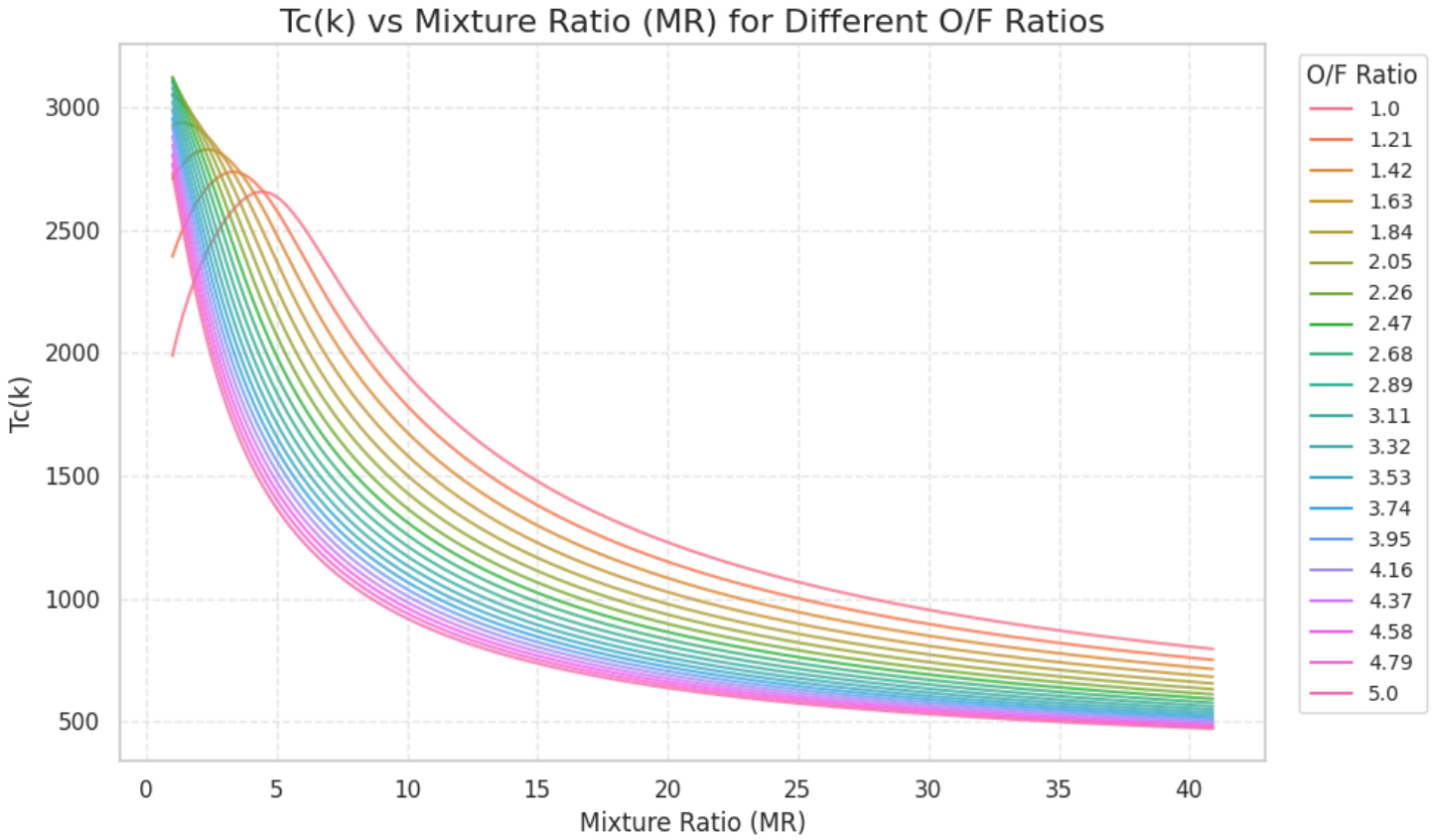
```
# Set the style for the plots
sns.set(style='whitegrid')

# Loop through each parameter and create the plots
for param in parameters:
    plt.figure(figsize=(10, 6))
```

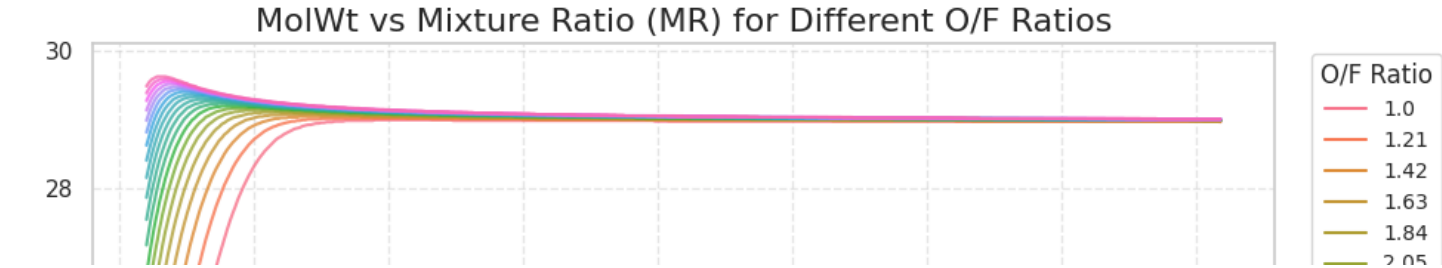
```
    sns.lineplot(x='MixtureRatio', y=param, hue='H_o_f', data=df, markers=True, err_style=None, alpha=0.8)
    plt.title(f'{param} vs Mixture Ratio (MR) for Different O/F Ratios', fontsize=16)
    plt.xlabel('Mixture Ratio (MR)', fontsize=12)
    plt.ylabel(param, fontsize=12)
    plt.legend(title='O/F Ratio', loc='upper right', bbox_to_anchor=(1.15, 1), fontsize=10, title_fontsize=12)
    plt.grid(True, linestyle='dashed', alpha=0.5)
    plt.tight_layout()
    plt.show()
```

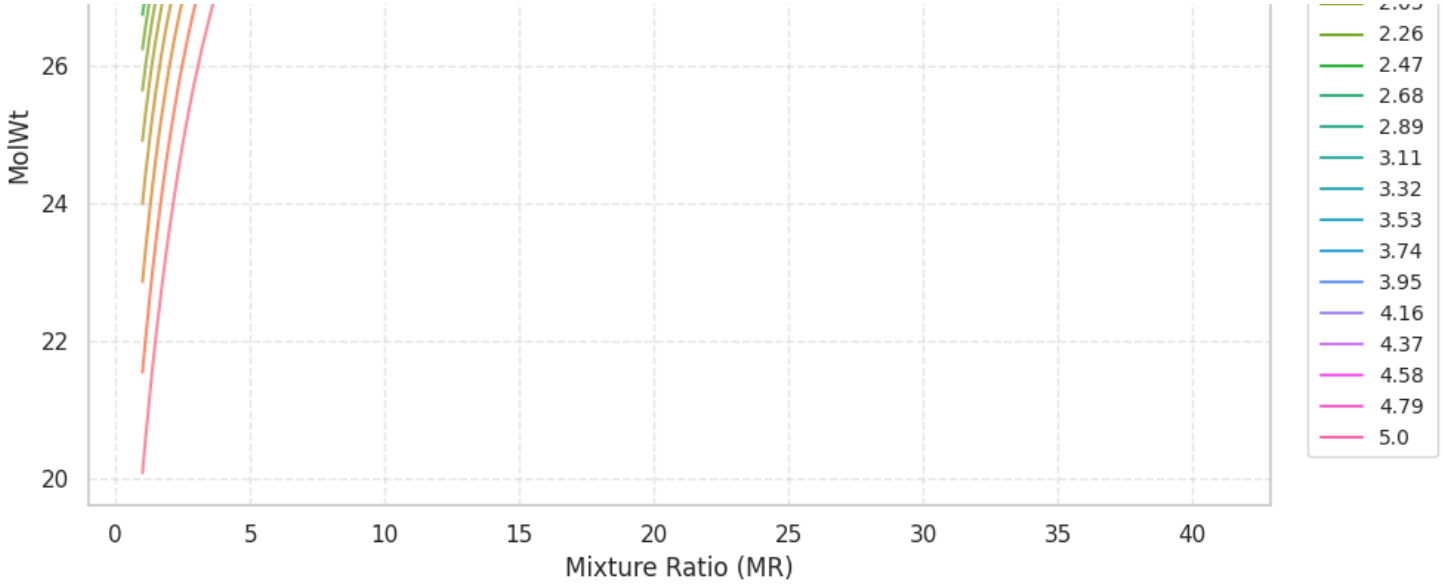

IspVac(sec) vs Mixture Ratio (MR) for Different O/F Ratios
IspVac(sec)
Mixture Ratio (MR)
O/F Ratio
1.0
1.21
1.42
1.63
1.84
2.05
2.26
2.47
2.68
2.89
3.11
3.32
3.53
3.74
3.95
4.16
4.37
4.58
4.79
5.0

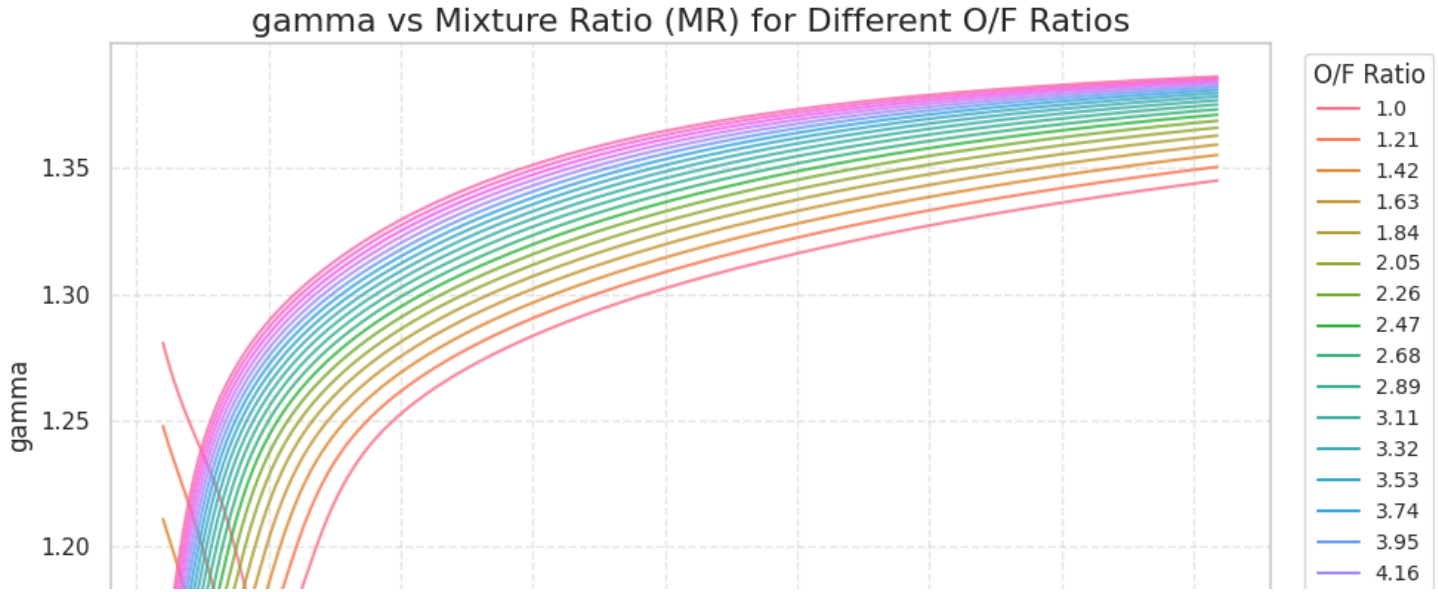

Cstar(m/sec) vs Mixture Ratio (MR) for Different O/F Ratios
Cstar(m/sec)
O/F Ratio
1.0
1.21
1.42
1.63
1.84
2.05
2.26
2.47
2.68
2.89
3.11
3.32
3.53
3.74
3.95


800
600
0
5
10
15
20
25
30
35
40
Mixture Ratio (MR)
4.16
4.37
4.58
4.79
5.0


Tc(k) vs Mixture Ratio (MR) for Different O/F Ratios
Tc(k)
3000
2500
2000
1500
1000
500
0
5
10
15
20
25
30
35
40
Mixture Ratio (MR)
O/F Ratio
1.0
1.21
1.42
1.63
1.84
2.05
2.26
2.47
2.68
2.89
3.11
3.32
3.53
3.74
3.95
4.16
4.37
4.58
4.79
5.0


MolWt vs Mixture Ratio (MR) for Different O/F Ratios
30
28
O/F Ratio
1.0
1.21
1.42
1.63
1.84
2.05

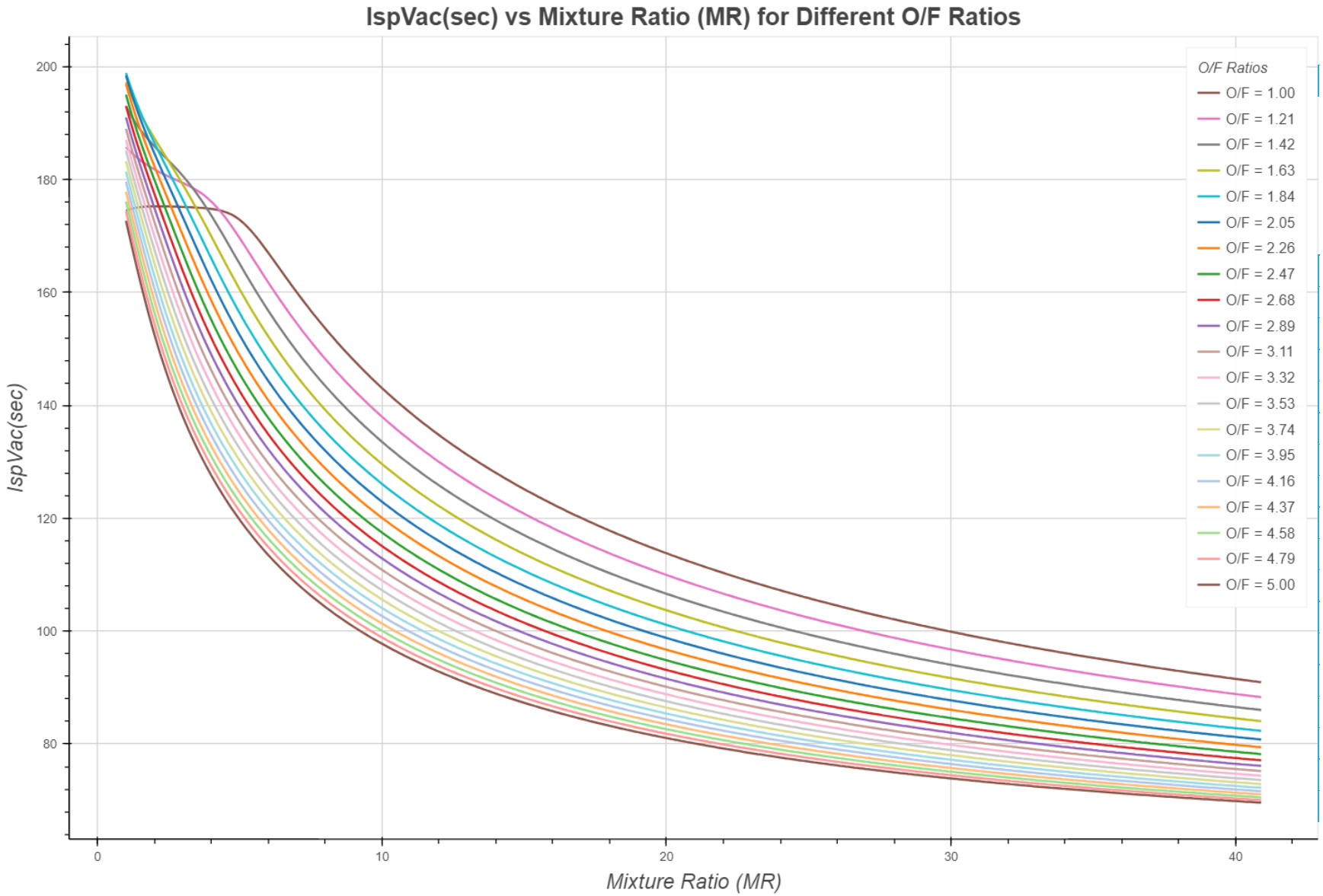


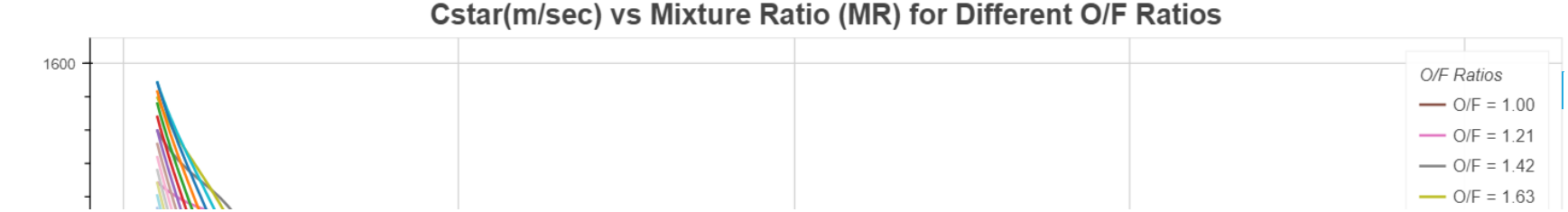


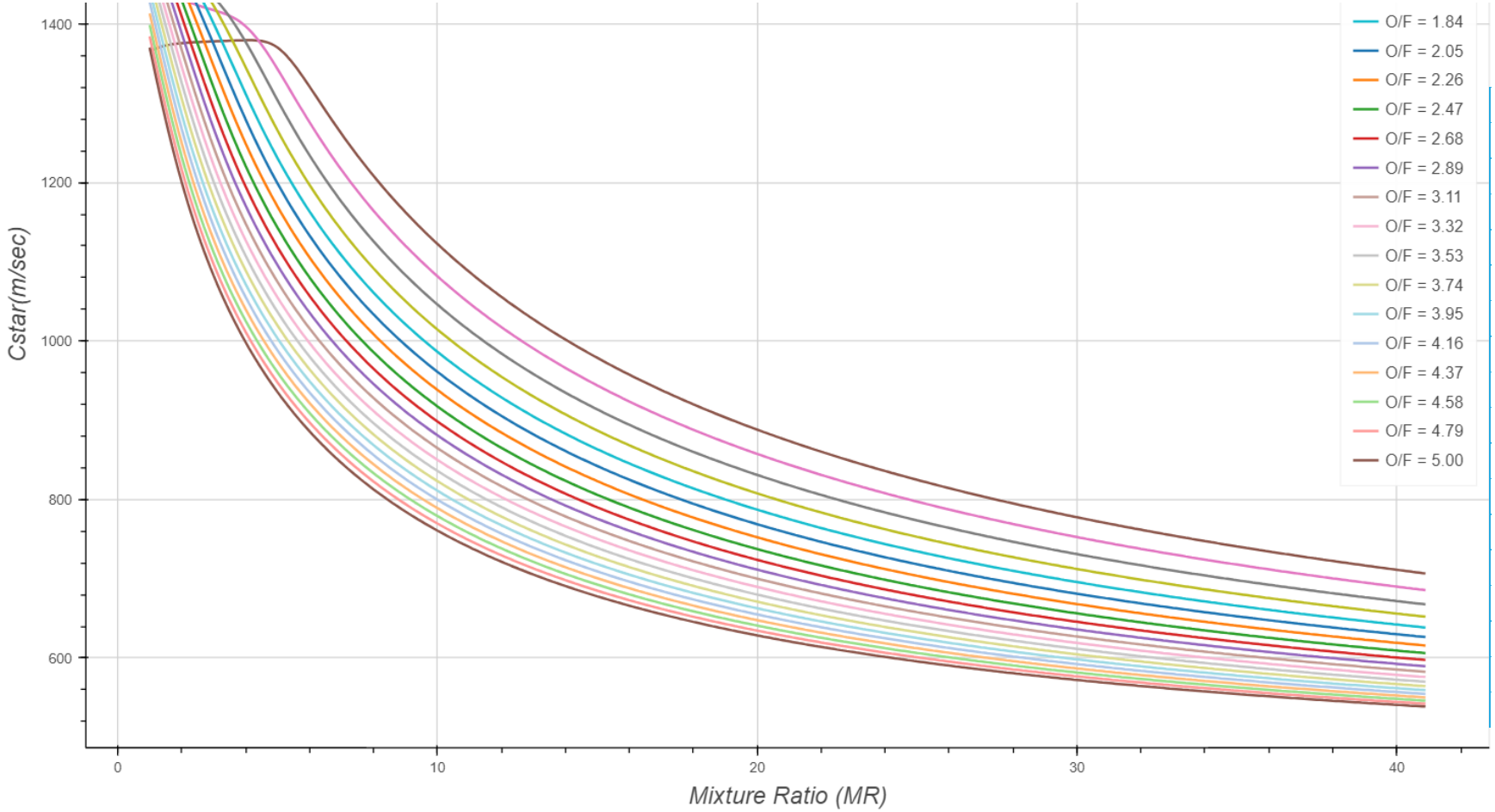
Creating more interactive plot using bokeh

```
import numpy as np
import pandas as pd
import seaborn as sns
from bokeh.plotting import figure, show
from bokeh.models import ColumnDataSource, HoverTool
```

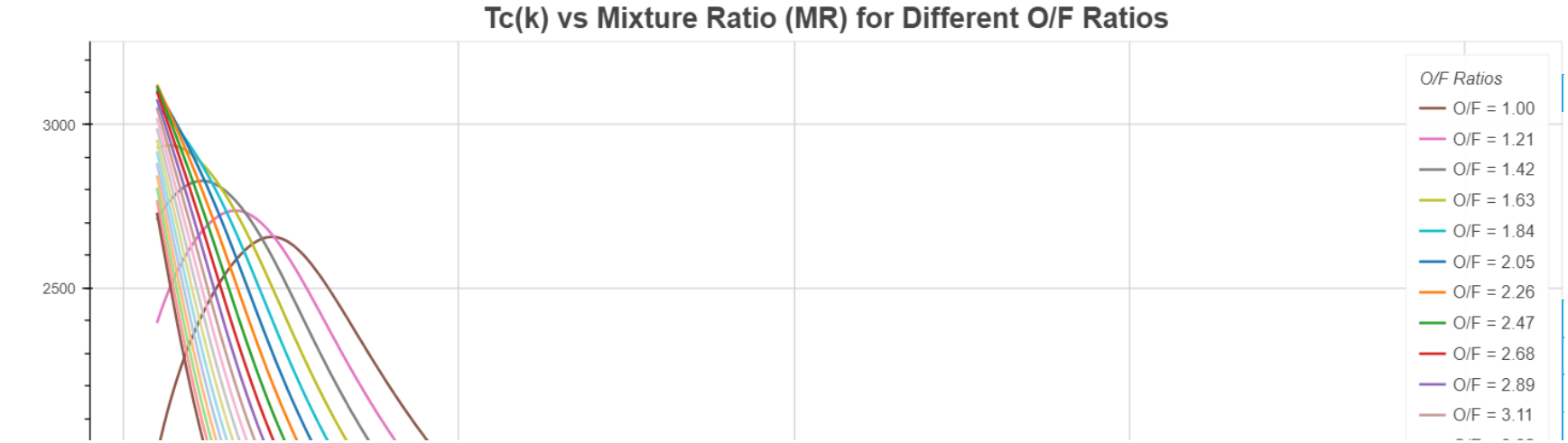
```
from bokeh.io import output_notebook
from matplotlib.colors import rgb2hex
output_notebook()

# Convert the 'Result' list to a pandas DataFrame
columns = ['Pc(psia)', 'eps', 'MixtureRatio', 'H_o_f', 'IspVac(sec)', 'Cstar(m/sec)', 'Tc(k)', 'MolWt', 'gamma', 'O2_MF']
df = pd.DataFrame([item for sublist in Result for item in sublist], columns=columns)

# Convert 'H_o_f' column to float data type
df['H_o_f'] = df['H_o_f'].astype(float)

# Create a list of parameter names
parameters = ['IspVac(sec)', 'Cstar(m/sec)', 'Tc(k)', 'MolWt', 'gamma', 'O2_MF']

# Set the style for the plots
sns.set(style='whitegrid')
sns.set_palette('tab20')

# Convert Seaborn's RGB color palette to Bokeh color representations
bokeh_palette = [rgb2hex(c) for c in sns.color_palette('tab20')]

# Loop through each parameter and create the interactive plots
for param in parameters:
    p = figure(width=1200, height=800, title=f'{param} vs Mixture Ratio (MR) for Different O/F Ratios',
               x_axis_label='Mixture Ratio (MR)', y_axis_label=param)
    # Center the title
    p.title.align = 'center'
    p.title.text_font_size = '16pt'

    unique_h_o_f = df['H_o_f'].unique()
    for o_f in unique_h_o_f:
        data = df[df['H_o_f'] == o_f]
        source = ColumnDataSource(data=dict(x=data['MixtureRatio'], y=data[param], H_o_f=data['H_o_f']))
        line = p.line('x', 'y', source=source, line_width=2, line_color=bokeh_palette[int(o_f * 10) % len(bokeh_palette)],
                      legend_label=f'O/F = {o_f:.2f}')
        p.add_tools(HoverTool(renderers=[line], tooltips=[('O/F', '@H_o_f'), ('Mixture Ratio', '@x'), (param, '@y')]))

    p.legend.title = 'O/F Ratios'
    p.legend.location = 'top_right'  # Change the legend location here
    p.legend.click_policy = 'hide'

    # Add grid lines, axes labels, and plot title
    p.grid.grid_line_color = 'lightgray'
    p.xaxis.axis_label_text_font_size = '14pt'
    p.yaxis.axis_label_text_font_size = '14pt'
    p.title.text_font_size = '16pt'

    show(p)
```

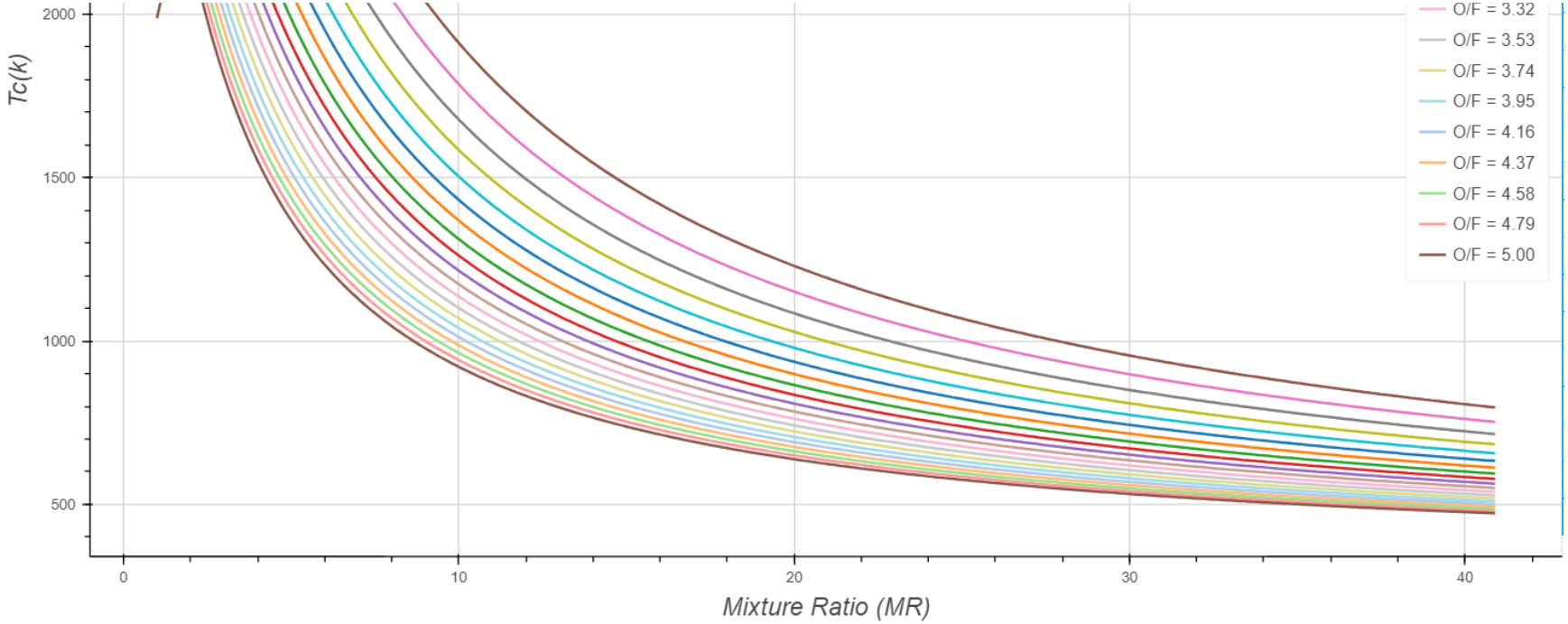

IspVac(sec) vs Mixture Ratio (MR) for Different O/F Ratios
IspVac(sec)
Mixture Ratio (MR)
O/F Ratios
O/F = 1.00
O/F = 1.21
O/F = 1.42
O/F = 1.63
O/F = 1.84
O/F = 2.05
O/F = 2.26
O/F = 2.47
O/F = 2.68
O/F = 2.89
O/F = 3.11
O/F = 3.32
O/F = 3.53
O/F = 3.74
O/F = 3.95
O/F = 4.16
O/F = 4.37
O/F = 4.58
O/F = 4.79
O/F = 5.00
200
180
160
140
120
100
80
0
10
20
30
40


Cstar(m/sec) vs Mixture Ratio (MR) for Different O/F Ratios
O/F Ratios
O/F = 1.00
O/F = 1.21
O/F = 1.42
O/F = 1.63
1600

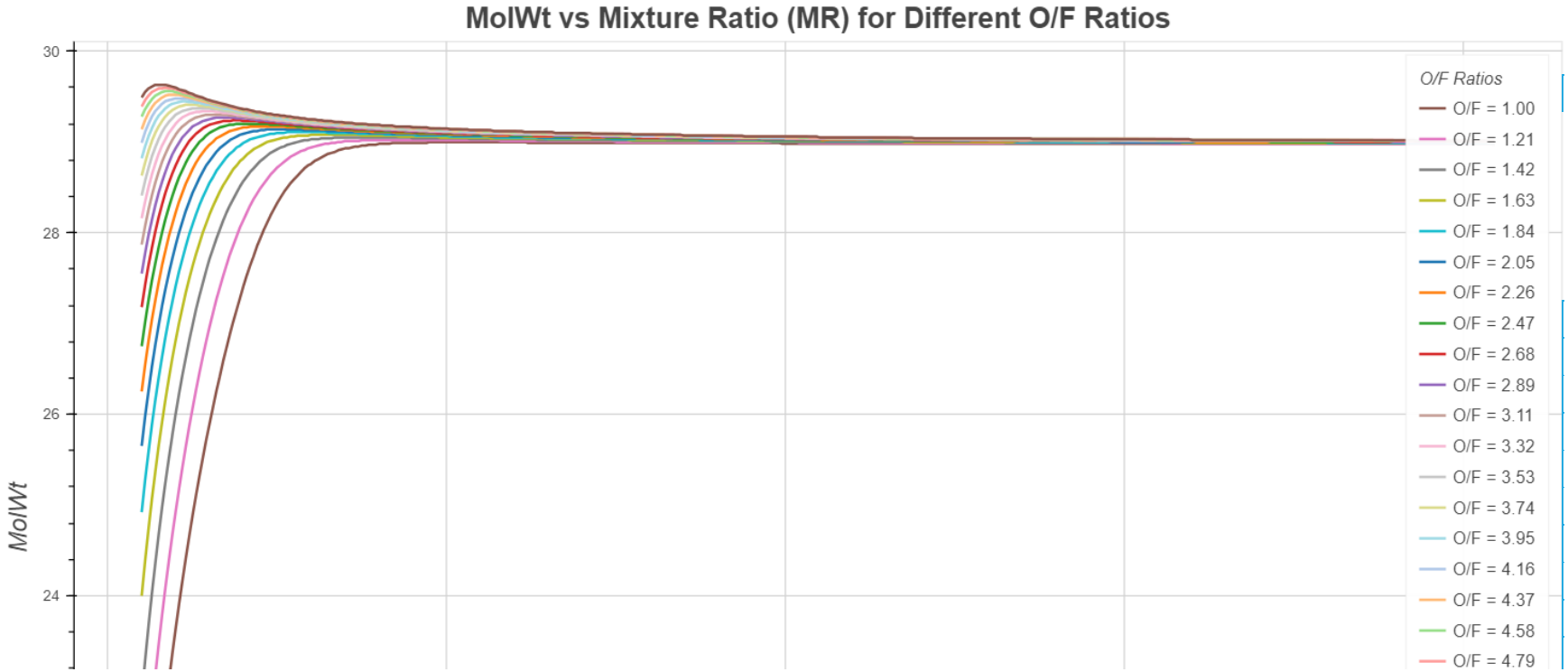

Cstar(m/sec)
1400
1200
1000
800
600
0
10
20
30
40
Mixture Ratio (MR)
O/F = 1.84
O/F = 2.05
O/F = 2.26
O/F = 2.47
O/F = 2.68
O/F = 2.89
O/F = 3.11
O/F = 3.32
O/F = 3.53
O/F = 3.74
O/F = 3.95
O/F = 4.16
O/F = 4.37
O/F = 4.58
O/F = 4.79
O/F = 5.00

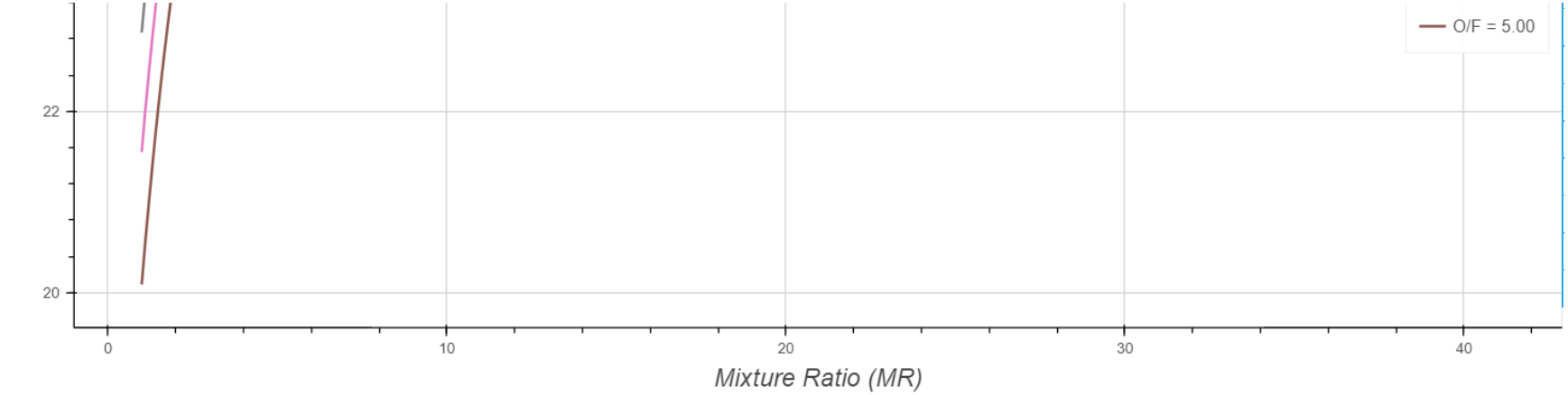
...

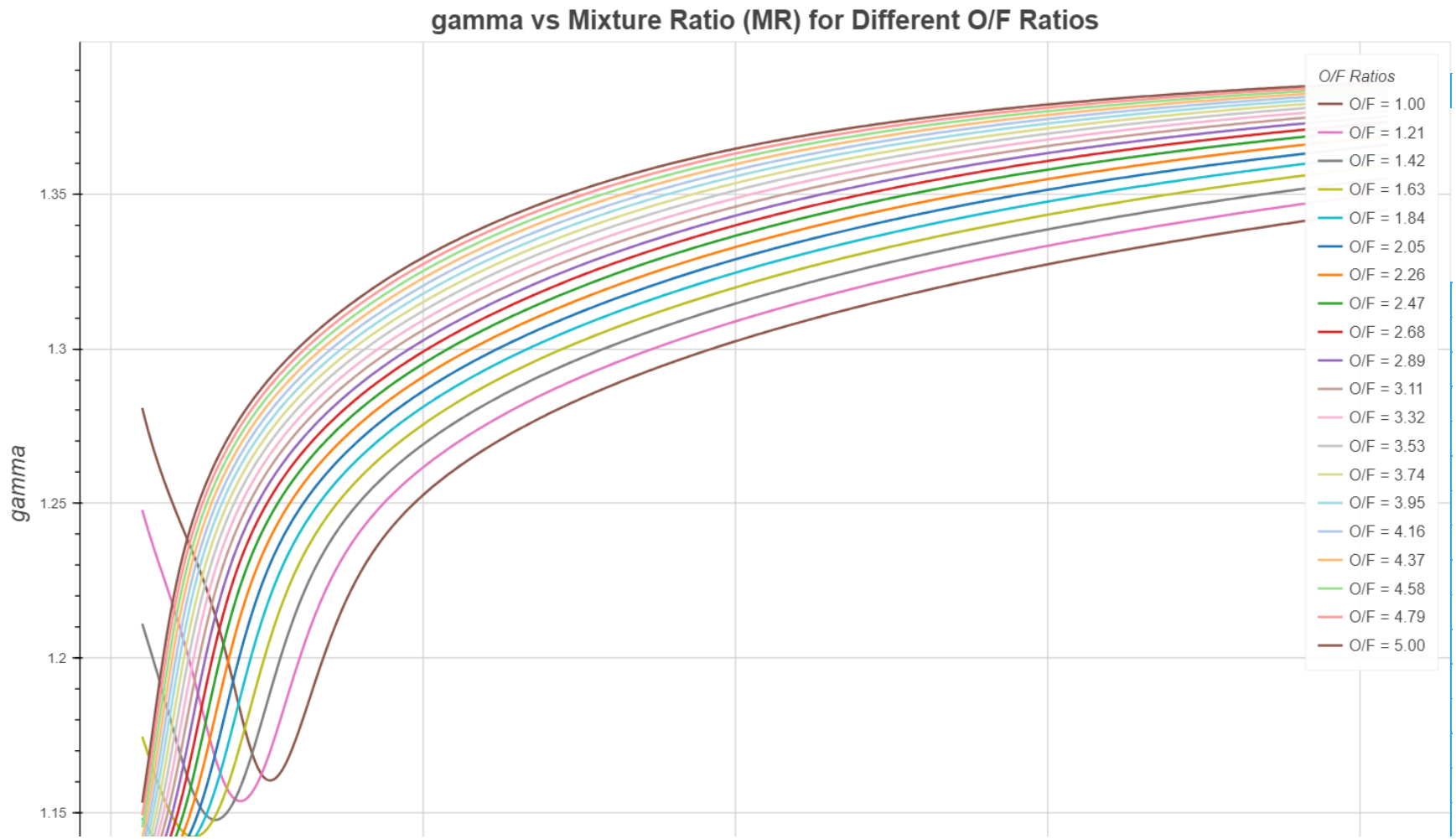

Tc(k) vs Mixture Ratio (MR) for Different O/F Ratios
3000
2500
O/F Ratios
O/F = 1.00
O/F = 1.21
O/F = 1.42
O/F = 1.63
O/F = 1.84
O/F = 2.05
O/F = 2.26
O/F = 2.47
O/F = 2.68
O/F = 2.89
O/F = 3.11

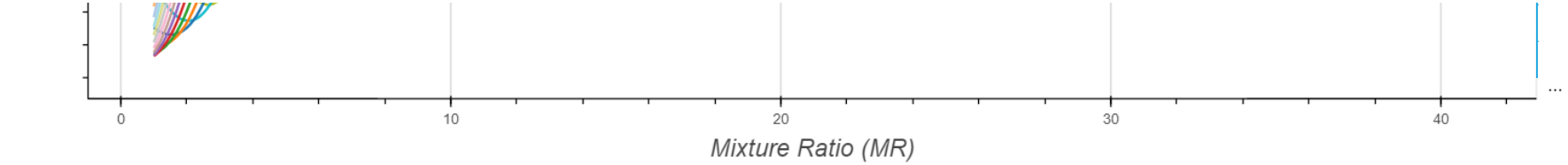

Tc(k)
2000
1500
1000
500
0
10
20
30
40
Mixture Ratio (MR)
O/F = 3.32
O/F = 3.53
O/F = 3.74
O/F = 3.95
O/F = 4.16
O/F = 4.37
O/F = 4.58
O/F = 4.79
O/F = 5.00

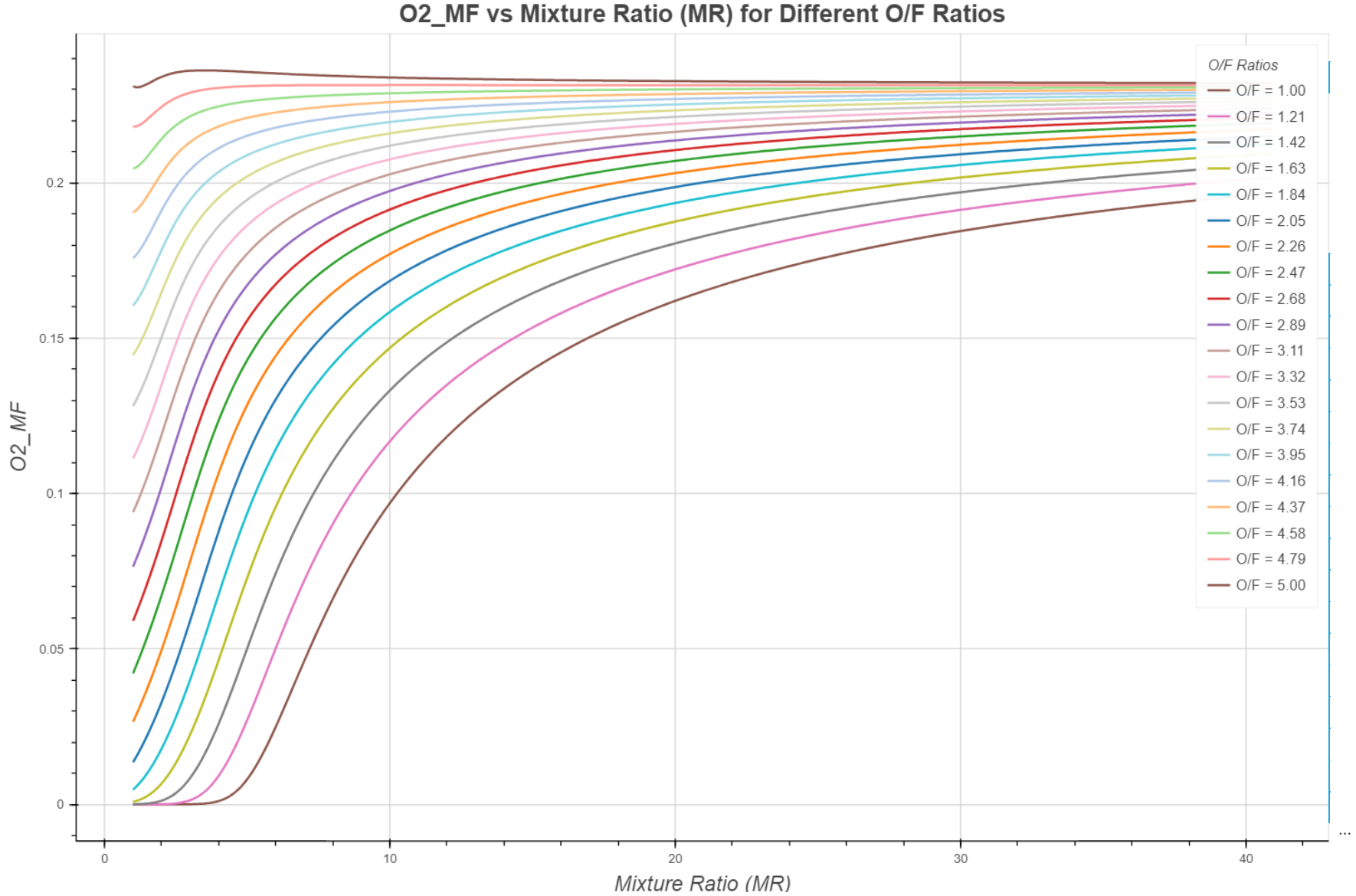
...


MolWt vs Mixture Ratio (MR) for Different O/F Ratios
MolWt
30
28
26
24
O/F Ratios
O/F = 1.00
O/F = 1.21
O/F = 1.42
O/F = 1.63
O/F = 1.84
O/F = 2.05
O/F = 2.26
O/F = 2.47
O/F = 2.68
O/F = 2.89
O/F = 3.11
O/F = 3.32
O/F = 3.53
O/F = 3.74
O/F = 3.95
O/F = 4.16
O/F = 4.37
O/F = 4.58
O/F = 4.79


O/F = 5.00
22
20
0
10
20
30
40
Mixture Ratio (MR)


gamma vs Mixture Ratio (MR) for Different O/F Ratios
gamma
1.35
1.3
1.25
1.2
1.15
O/F Ratios
O/F = 1.00
O/F = 1.21
O/F = 1.42
O/F = 1.63
O/F = 1.84
O/F = 2.05
O/F = 2.26
O/F = 2.47
O/F = 2.68
O/F = 2.89
O/F = 3.11
O/F = 3.32
O/F = 3.53
O/F = 3.74
O/F = 3.95
O/F = 4.16
O/F = 4.37
O/F = 4.58
O/F = 4.79
O/F = 5.00





▸ More information on creating above plot

1 cell hidden

Analysis on Mass Fraction of species

ChatGPT

```
import numpy as np
import pandas as pd
from rocketcea.cea_obj import CEA_Obj, add_new_fuel, add_new_oxidizer

card_str = """
   oxid Air    wt%=100.00
   h,cal=-28.2     t(k)=300
   """
add_new_oxidizer('oxy', card_str)

# Add fuel
lst1 = []
lst2 = []
O_F = []
Result_mf = []
species_to_include = ['*CO2', '*N2', '*H2', '*CO', '*H', '*Ar', '*NO', 'NO2', 'HO2', '*O2', '*OH', 'H2O', '*O']

# Iterating for a range of oxidizer to fuel mixture
for o_f in np.linspace(1, 5, 20):
    m_f = 100 / (1 + o_f)
    m_ox = 100 - m_f
    lst1.append(m_ox)
    lst2.append(m_f)
    O_F.append(round(o_f, 2))

# Iterating fuel card for several Hybrid ratios
for i in range(len(lst1)):
    card_str = """
    fuel OXYGEN  O 2     wt%=""" + str(lst1[i]) + """
    h,cal= 12.9918 t(k)=300
    fuel WAX C 26 H 54    wt%=""" + str(lst2[i]) + """
    h,cal= -140439.7 t(k)=300
    """
    add_new_fuel('fuel', card_str)

    from rocketcea.cea_obj_w_units import CEA_Obj
    C = CEA_Obj(oxName='oxy', fuelName="fuel", isp_units='sec', cstar_units='m/s', pressure_units='Bar', temperature_units='K', sonic_velocity_units='m/s', enthalpy_units='BTU/lbm',

    result_mf = []
    H_o_f = lst1[i] / lst2[i]

    def show_perf(Pc=10, eps=1, MR=1.0):
        molWtD, massFracD = C.get_SpeciesMassFractions(Pc=Pc, eps=eps, MR=MR)  # Gives the set containing species mol weight and mass fraction
```

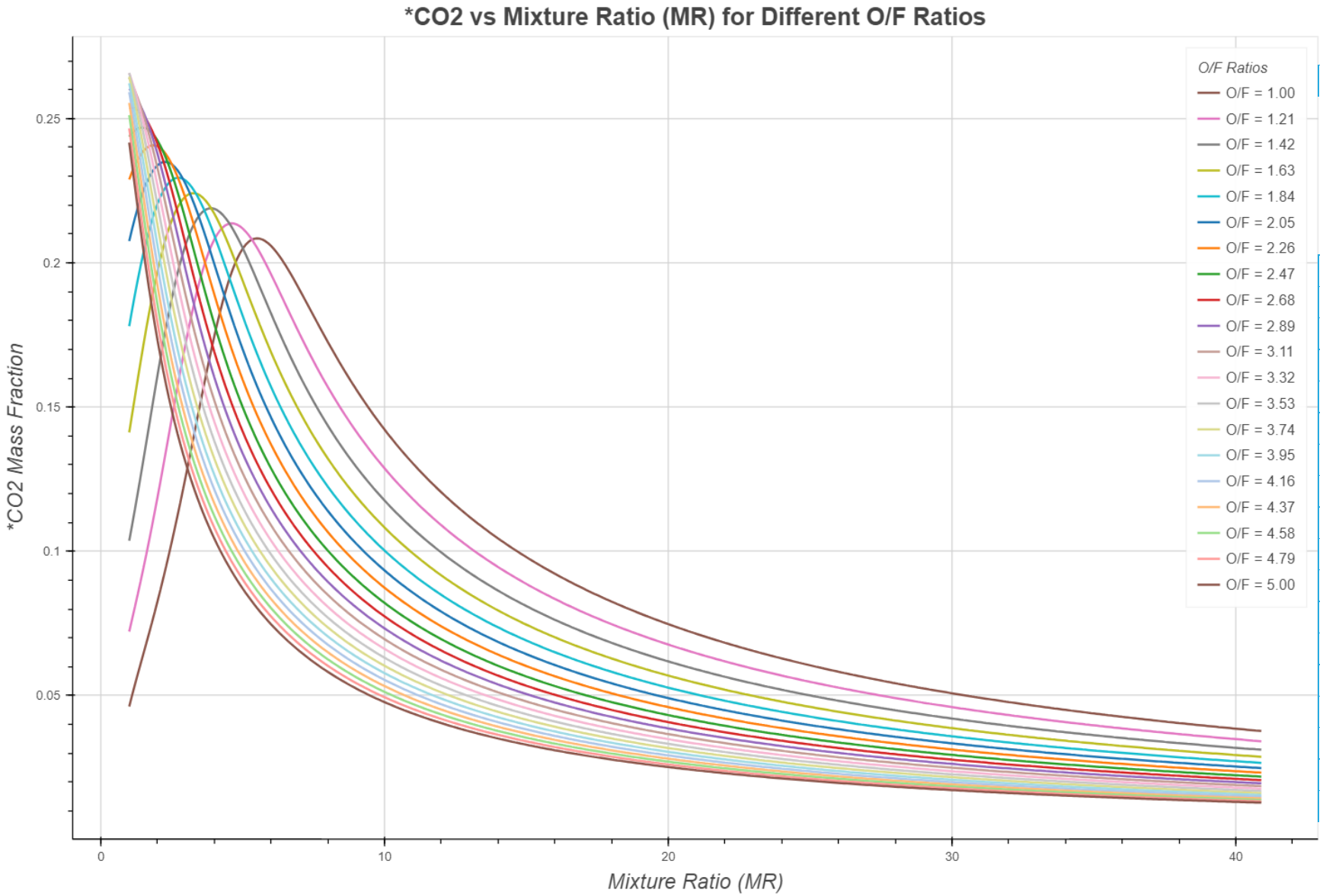
```
        mass_fractions = [massFracD[specie][1] if specie in massFracD else 0.0 for specie in species_to_include]

        result_mf.append([Pc, eps, round(MR, 2), round(H_o_f, 2)] + mass_fractions)

    Pc = 10
    eps = 1
    MR_lst = []
    for MR in [1.0 + i * 0.1 for i in range(400)]:  # For every MR show_performance is called
        show_perf(Pc=Pc, eps=eps, MR=MR)
        MR_lst.append(MR)
    Result_mf.append(result_mf)

# After the loop, convert the 'Result' list to a DataFrame
df_columns = ['Pc(psia)', 'eps', 'MixtureRatio', 'H_o_f'] + species_to_include
res = [i for lst in Result_mf for i in lst]
df_MF = pd.DataFrame(res, columns=df_columns)

df_MF
```

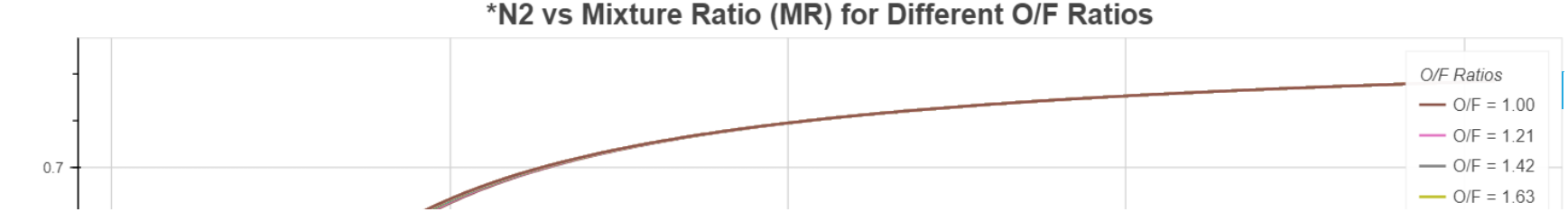
	Pc(psia)	eps	MixtureRatio	H_o_f	*CO2	*N2	*H2	*CO	*H	*Ar	*NO	NO2	HO2	*O2	*OH	H2O	*O
0	10	1	1.0	1.0	0.046188	0.377575	0.028849	0.467234	0.000013	0.006458	0.0	0.0	0.0	0.000000	0.000007	0.073650	0.0
1	10	1	1.1	1.0	0.050059	0.395558	0.026360	0.441137	0.000017	0.006766	0.0	0.0	0.0	0.000000	0.000011	0.080069	0.0
2	10	1	1.2	1.0	0.053772	0.411906	0.024106	0.417289	0.000021	0.007045	0.0	0.0	0.0	0.000000	0.000018	0.085823	0.0
3	10	1	1.3	1.0	0.057386	0.426832	0.022057	0.395373	0.000025	0.007300	0.0	0.0	0.0	0.000000	0.000027	0.090982	0.0
4	10	1	1.4	1.0	0.060947	0.440514	0.020191	0.375124	0.000030	0.007534	0.0	0.0	0.0	0.000000	0.000039	0.095605	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7995	10	1	40.5	5.0	0.013004	0.736986	0.000000	0.000000	0.000000	0.012605	0.0	0.0	0.0	0.232077	0.000000	0.005327	0.0
7996	10	1	40.6	5.0	0.012974	0.737030	0.000000	0.000000	0.000000	0.012606	0.0	0.0	0.0	0.232076	0.000000	0.005314	0.0
7997	10	1	40.7	5.0	0.012944	0.737074	0.000000	0.000000	0.000000	0.012606	0.0	0.0	0.0	0.232074	0.000000	0.005301	0.0
7998	10	1	40.8	5.0	0.012915	0.737117	0.000000	0.000000	0.000000	0.012607	0.0	0.0	0.0	0.232072	0.000000	0.005289	0.0
7999	10	1	40.9	5.0	0.012885	0.737160	0.000000	0.000000	0.000000	0.012608	0.0	0.0	0.0	0.232071	0.000000	0.005276	0.0

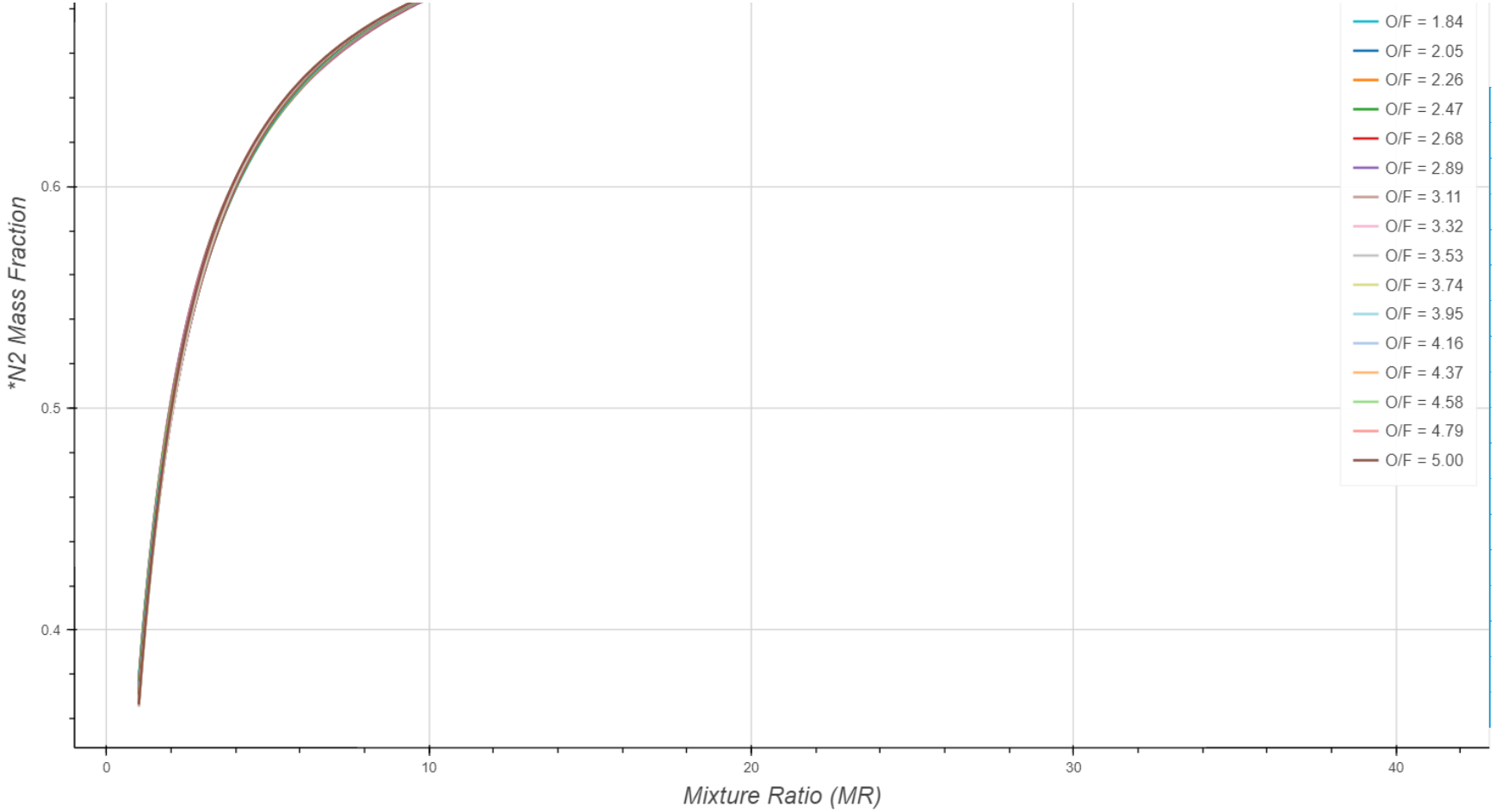
8000 rows × 17 columns

```
from bokeh.plotting import figure, show, output_notebook
from bokeh.models import ColumnDataSource, HoverTool
import seaborn as sns

species_to_include = ['*CO2', '*N2', '*H2', '*CO', '*H', '*Ar', '*NO', 'NO2', 'HO2', '*O2', '*OH', 'H2O', '*O']

# Set the style for the plots
sns.set(style='whitegrid')
```

```
sns.set_palette('tab20')  # 20 distinguishable colors

# Convert Seaborn's RGB color palette to Bokeh color representations
bokeh_palette = [rgb2hex(c) for c in sns.color_palette('tab20')]

# Loop through each species and create individual plots
for species in species_to_include:
    # Create the Bokeh figure
    p = figure(width=1200, height=800, title=f"{species} vs Mixture Ratio (MR) for Different O/F Ratios",
               x_axis_label="Mixture Ratio (MR)", y_axis_label=f"{species} Mass Fraction")

    # Center the title
    p.title.align = 'center'
    p.title.text_font_size = '16pt'

    # Loop through each unique H_o_f value and add lines to the plot
    unique_h_o_f = df_MF['H_o_f'].unique()
    for o_f in unique_h_o_f:
        data = df_MF[df_MF['H_o_f'] == o_f]
        source = ColumnDataSource(data=dict(x=data['MixtureRatio'], y=data[species], H_o_f=data['H_o_f']))
        line = p.line('x', 'y', source=source, line_width=2, line_color=bokeh_palette[int(o_f * 10) % len(bokeh_palette)],
                      legend_label=f'O/F = {o_f:.2f}')
        p.add_tools(HoverTool(renderers=[line], tooltips=[('O/F', '@H_o_f'), ('Mixture Ratio', '@x'), (species, '@y')]))



    p.legend.title = 'O/F Ratios'
    p.legend.location = 'top_right'
    p.legend.click_policy = 'hide'

    # Add grid lines, axes labels, and plot title
    p.grid.grid_line_color = 'lightgray'
    p.xaxis.axis_label_text_font_size = '14pt'
    p.yaxis.axis_label_text_font_size = '14pt'
    p.title.text_font_size = '16pt'

    # Show the plot
    output_notebook()
    show(p)
```

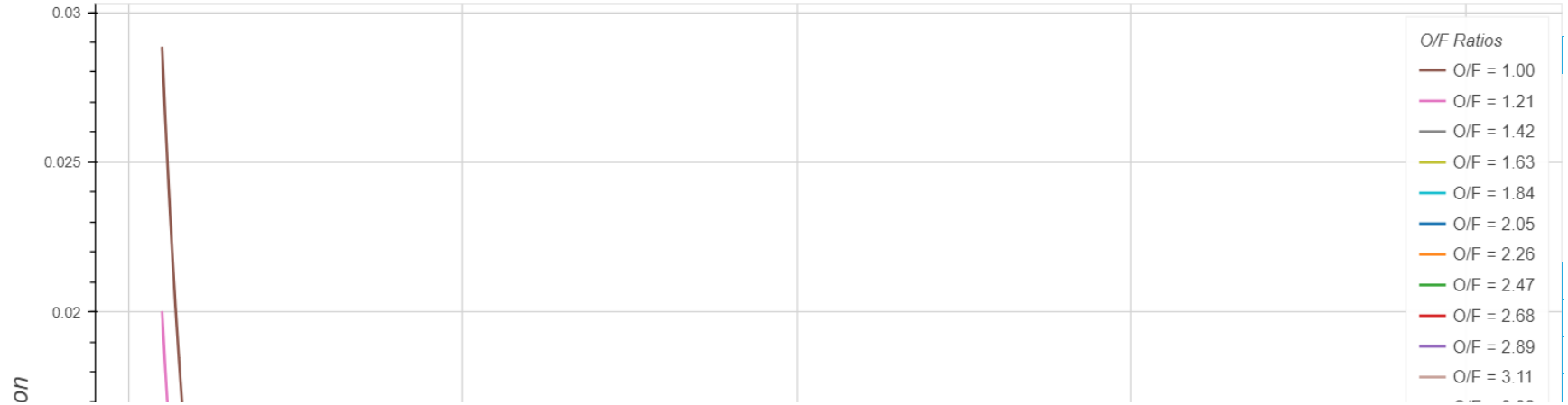

*CO2 vs Mixture Ratio (MR) for Different O/F Ratios
*CO2 Mass Fraction
Mixture Ratio (MR)
O/F Ratios
O/F = 1.00
O/F = 1.21
O/F = 1.42
O/F = 1.63
O/F = 1.84
O/F = 2.05
O/F = 2.26
O/F = 2.47
O/F = 2.68
O/F = 2.89
O/F = 3.11
O/F = 3.32
O/F = 3.53
O/F = 3.74
O/F = 3.95
O/F = 4.16
O/F = 4.37
O/F = 4.58
O/F = 4.79
O/F = 5.00
0.25
0.2
0.15
0.1
0.05
0
10
20
30
40

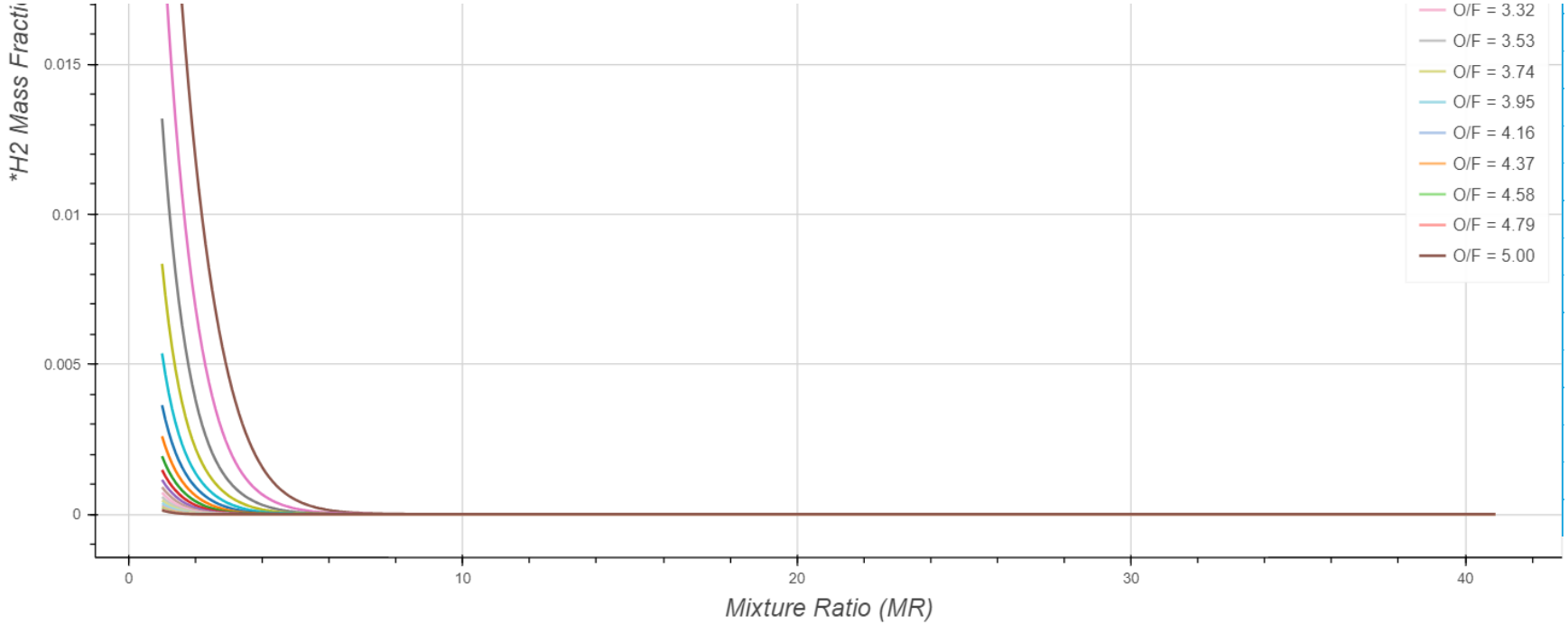

*N2 vs Mixture Ratio (MR) for Different O/F Ratios
O/F Ratios
O/F = 1.00
O/F = 1.21
O/F = 1.42
O/F = 1.63
0.7



...

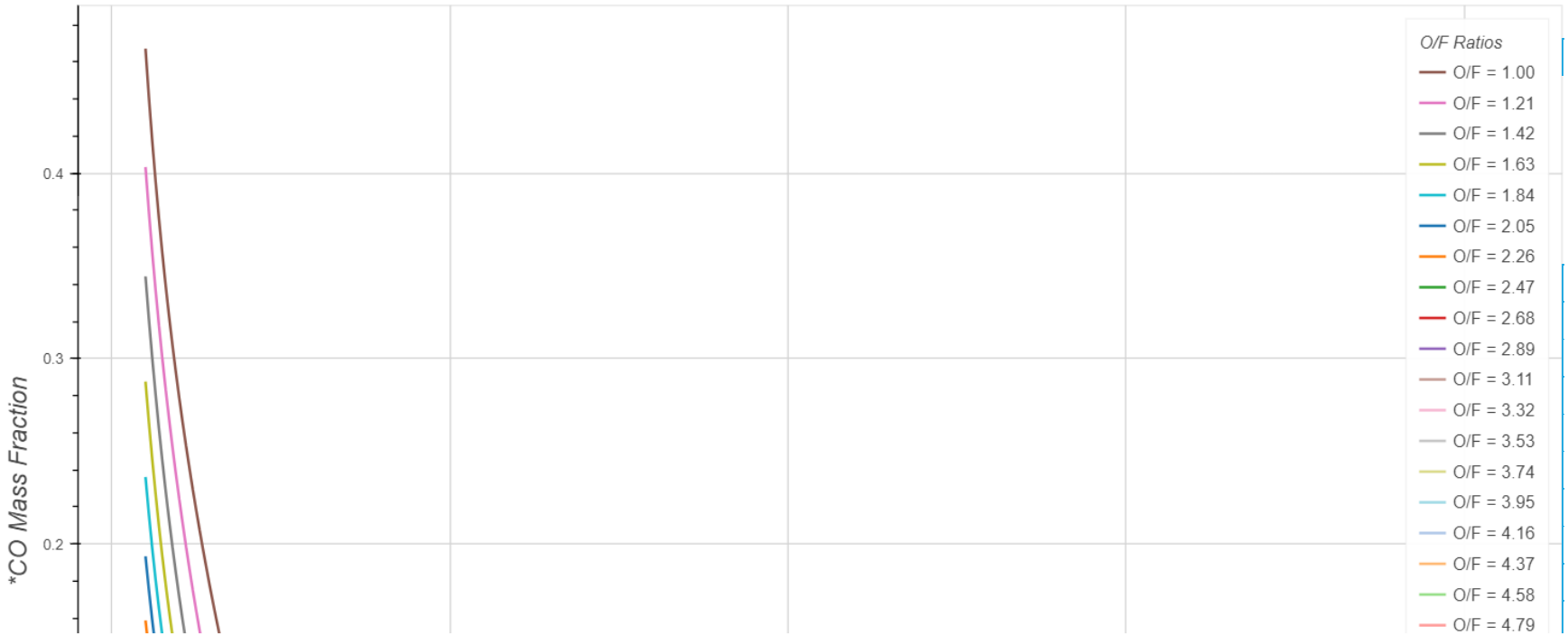
***H2 vs Mixture Ratio (MR) for Different O/F Ratios**

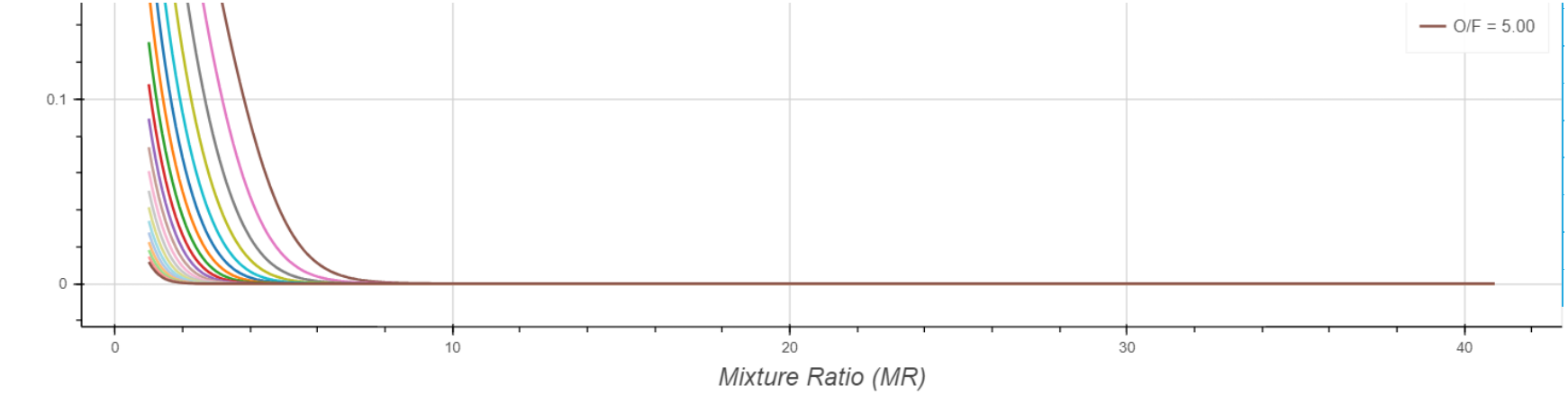




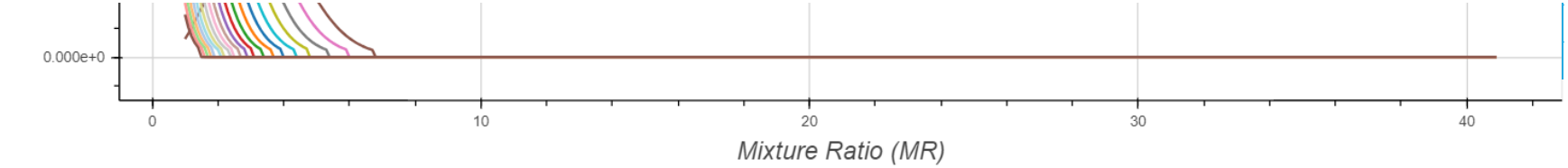
...

***CO vs Mixture Ratio (MR) for Different O/F Ratios**

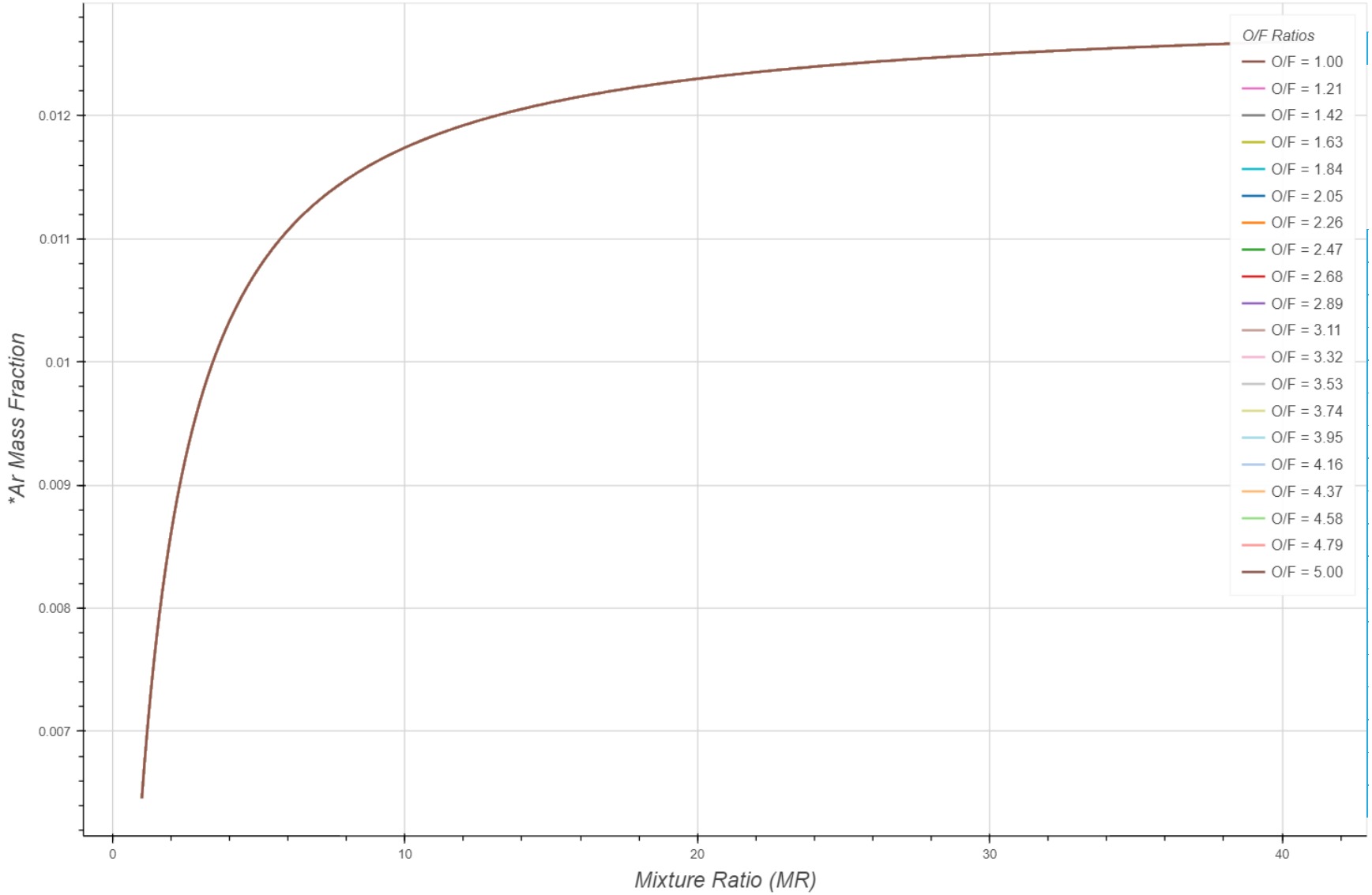



O/F = 5.00
0.1
0
0
10
20
30
40
Mixture Ratio (MR)

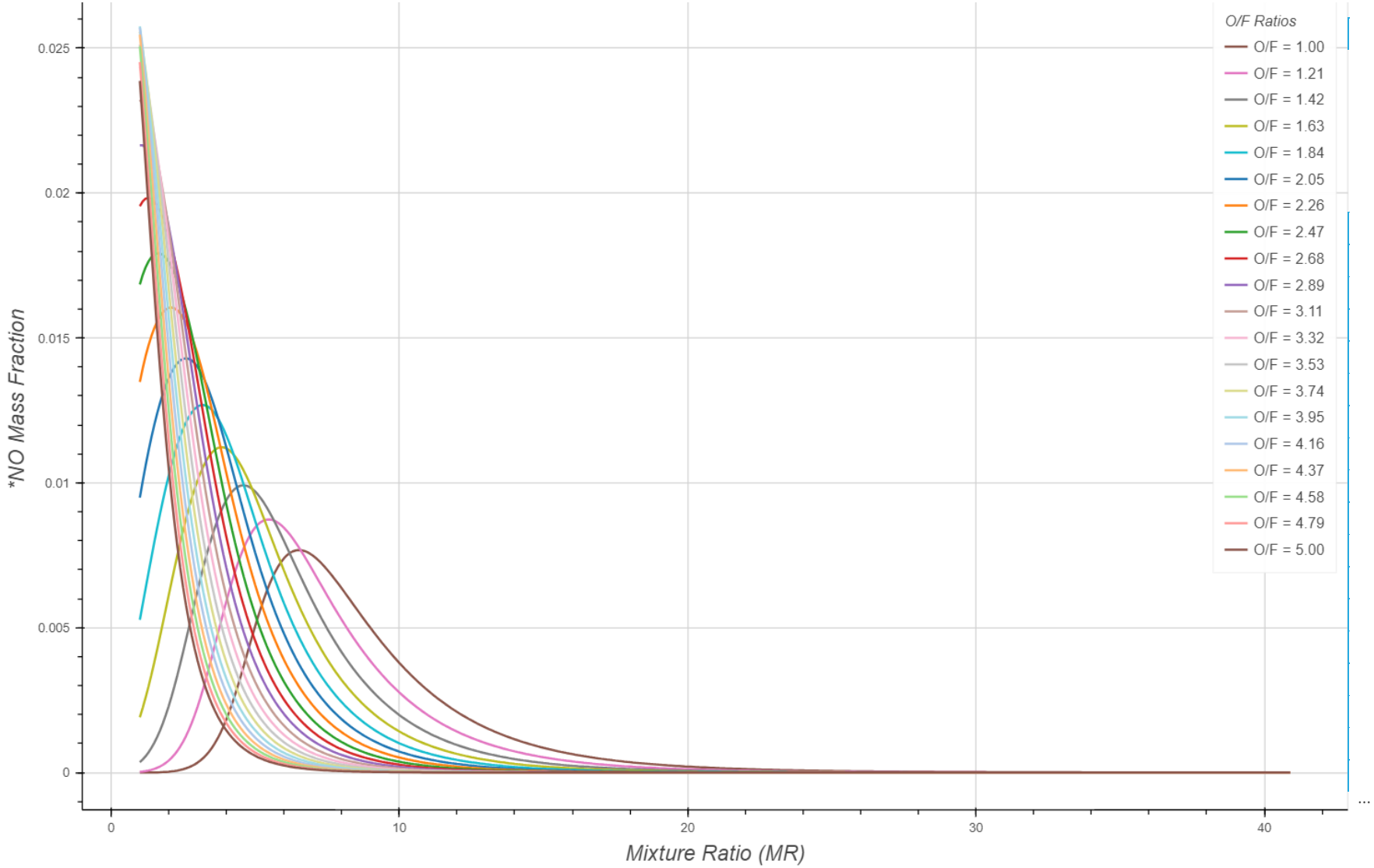

*H vs Mixture Ratio (MR) for Different O/F Ratios
*H Mass Fraction
6.000e−4
5.000e−4
4.000e−4
3.000e−4
2.000e−4
1.000e−4
O/F Ratios
O/F = 1.00
O/F = 1.21
O/F = 1.42
O/F = 1.63
O/F = 1.84
O/F = 2.05
O/F = 2.26
O/F = 2.47
O/F = 2.68
O/F = 2.89
O/F = 3.11
O/F = 3.32
O/F = 3.53
O/F = 3.74
O/F = 3.95
O/F = 4.16
O/F = 4.37
O/F = 4.58
O/F = 4.79
O/F = 5.00



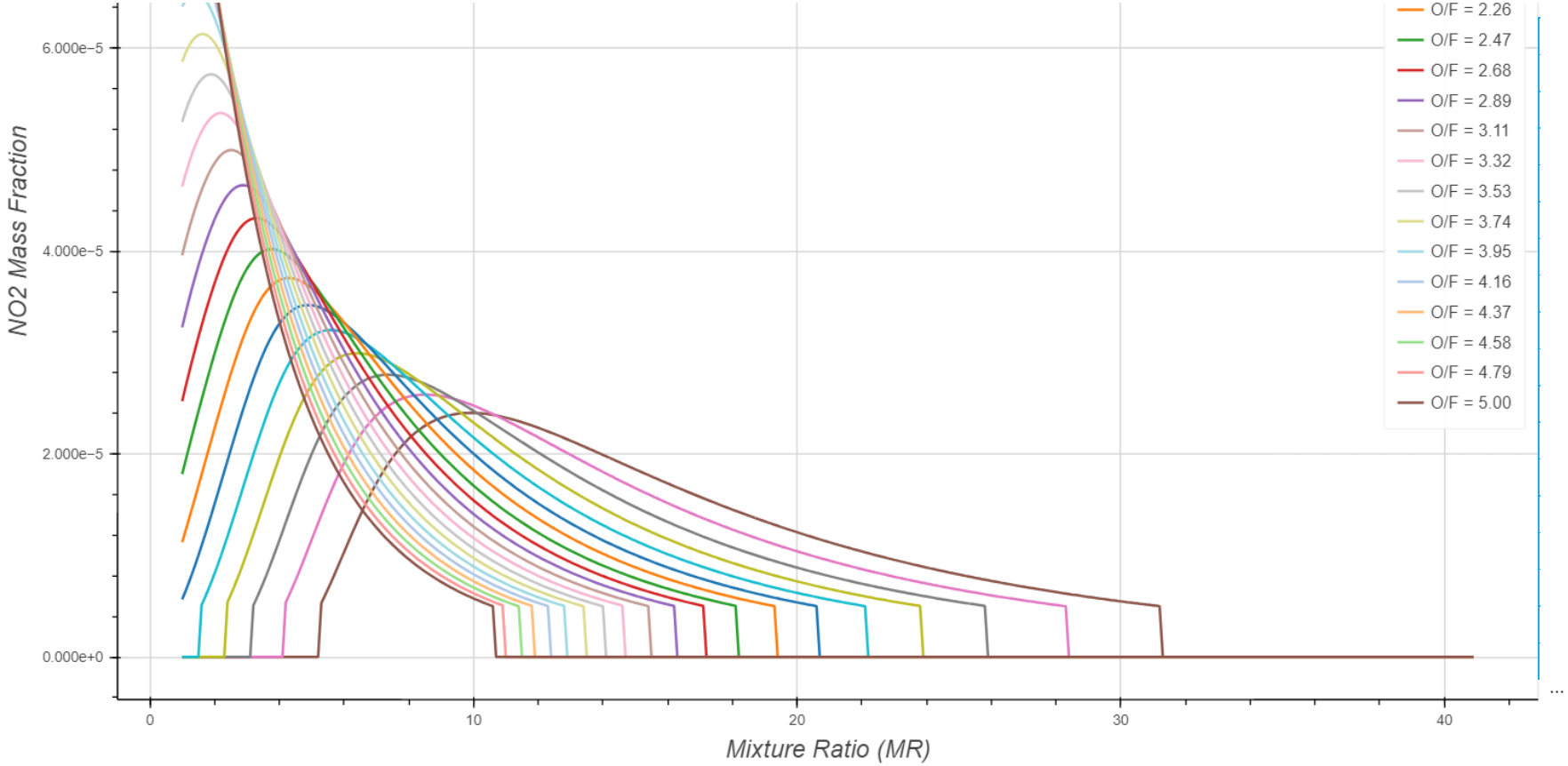
*Ar vs Mixture Ratio (MR) for Different O/F Ratios



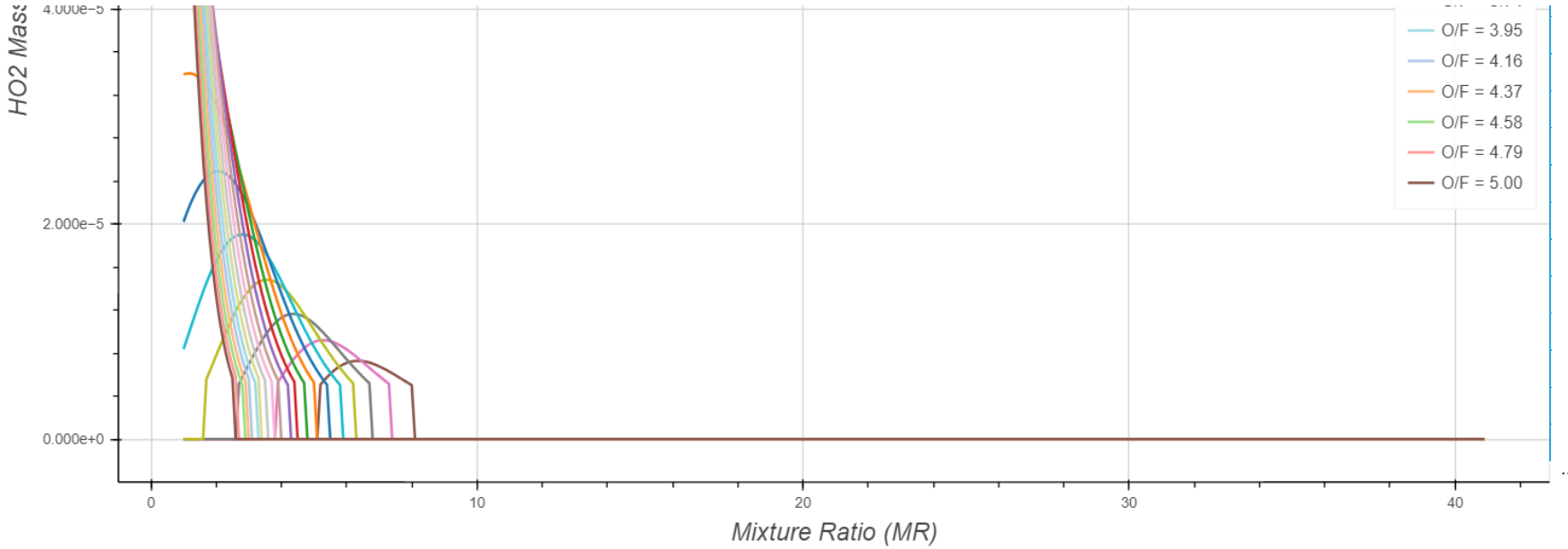
*NO vs Mixture Ratio (MR) for Different O/F Ratios

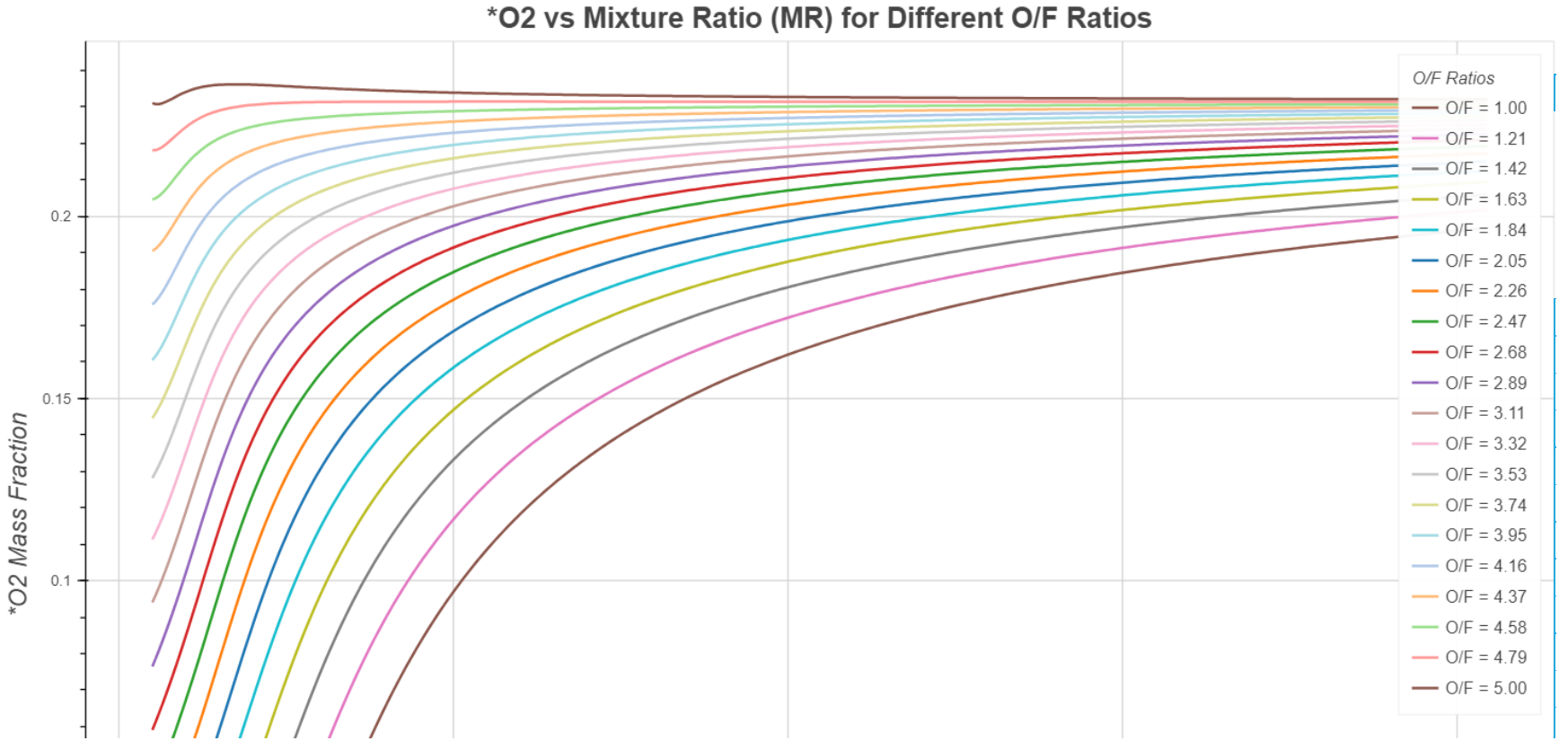


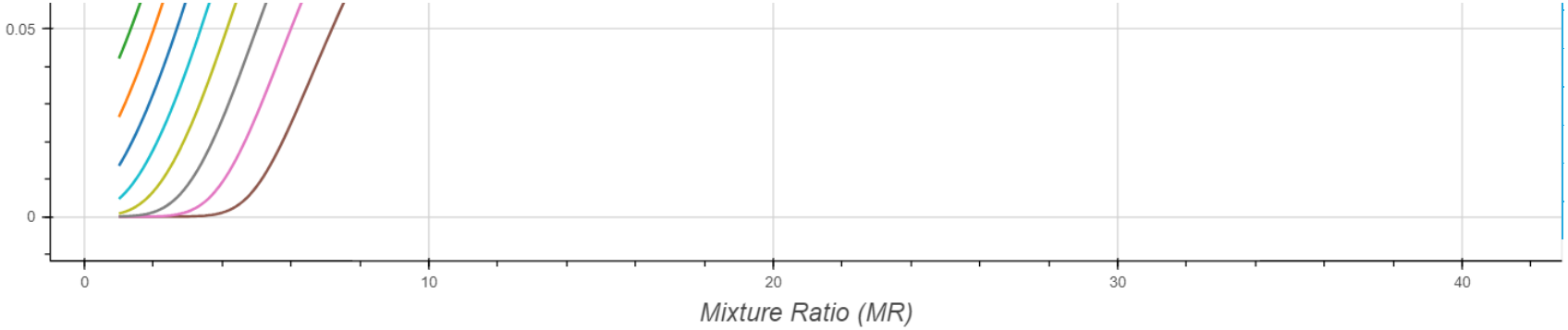
NO2 vs Mixture Ratio (MR) for Different O/F Ratios

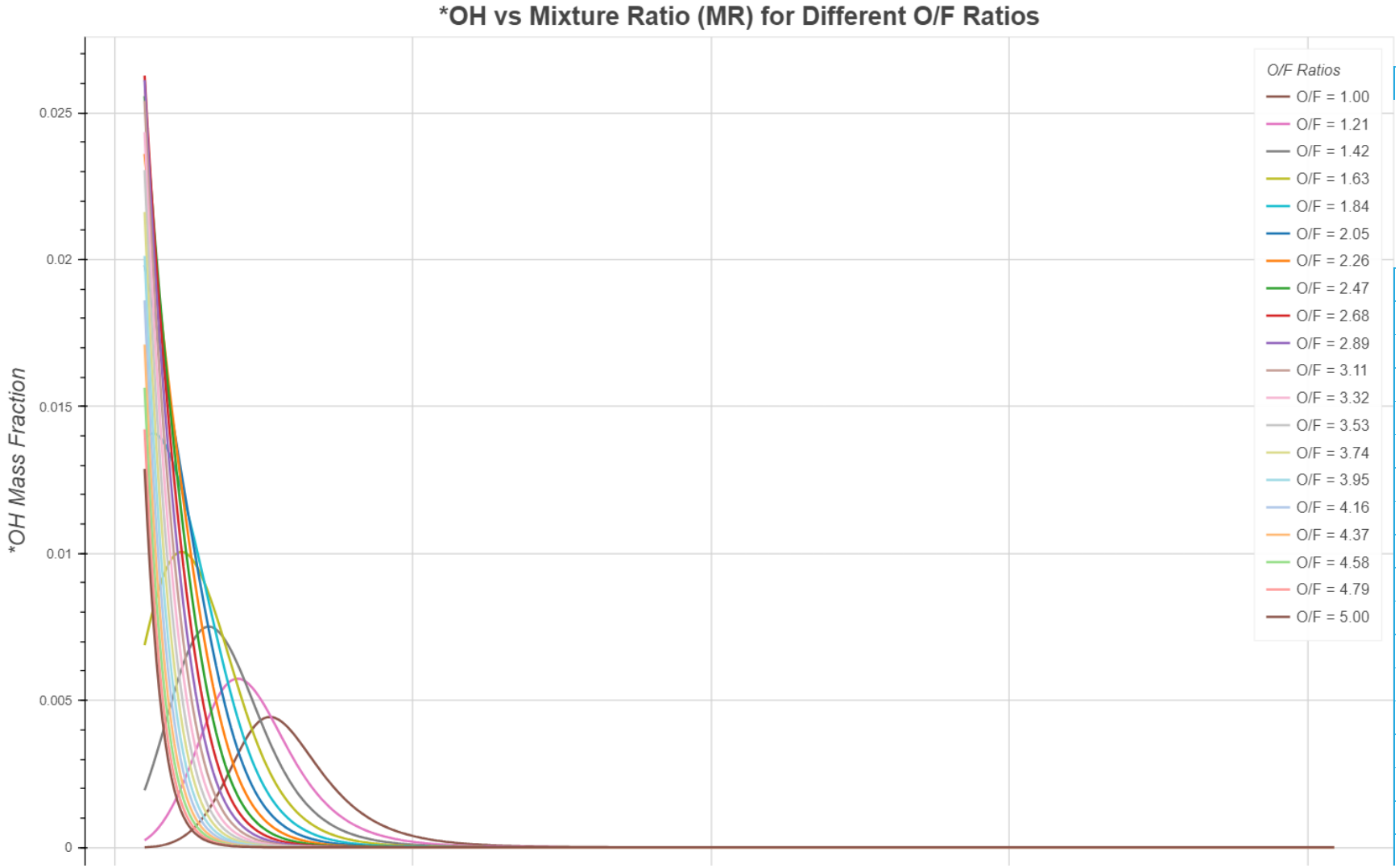


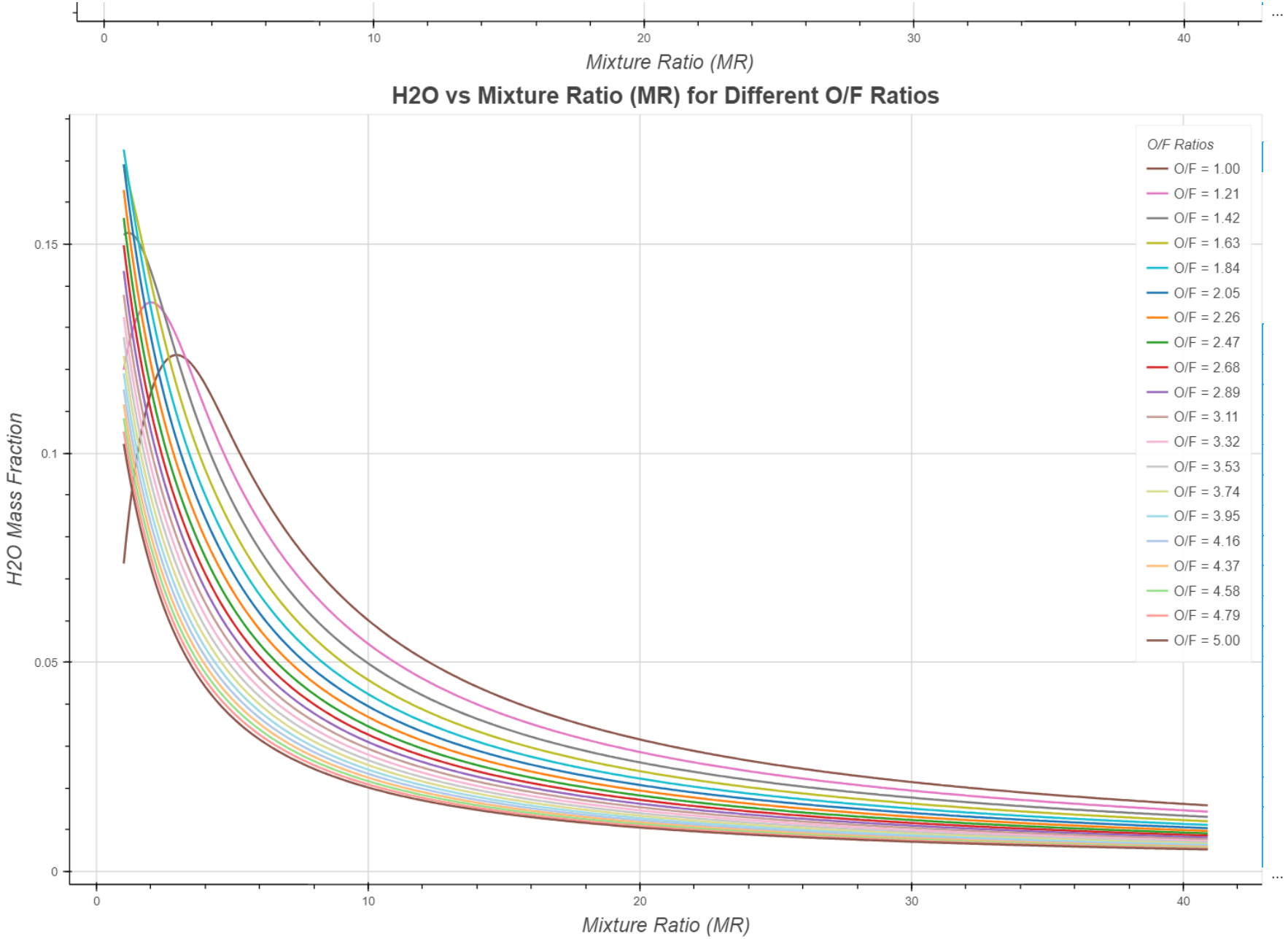
HO2 vs Mixture Ratio (MR) for Different O/F Ratios


HO2 Mass
4.000e−5
2.000e−5
0.000e+0
0
10
20
30
40
Mixture Ratio (MR)
O/F = 3.95
O/F = 4.16
O/F = 4.37
O/F = 4.58
O/F = 4.79
O/F = 5.00


*O2 vs Mixture Ratio (MR) for Different O/F Ratios
*O2 Mass Fraction
0.2
0.15
0.1
O/F Ratios
O/F = 1.00
O/F = 1.21
O/F = 1.42
O/F = 1.63
O/F = 1.84
O/F = 2.05
O/F = 2.26
O/F = 2.47
O/F = 2.68
O/F = 2.89
O/F = 3.11
O/F = 3.32
O/F = 3.53
O/F = 3.74
O/F = 3.95
O/F = 4.16
O/F = 4.37
O/F = 4.58
O/F = 4.79
O/F = 5.00


0.05
0
0
10
20
30
40
Mixture Ratio (MR)


*OH vs Mixture Ratio (MR) for Different O/F Ratios
*OH Mass Fraction
0.025
0.02
0.015
0.01
0.005
0
O/F Ratios
O/F = 1.00
O/F = 1.21
O/F = 1.42
O/F = 1.63
O/F = 1.84
O/F = 2.05
O/F = 2.26
O/F = 2.47
O/F = 2.68
O/F = 2.89
O/F = 3.11
O/F = 3.32
O/F = 3.53
O/F = 3.74
O/F = 3.95
O/F = 4.16
O/F = 4.37
O/F = 4.58
O/F = 4.79
O/F = 5.00


0
10
20
30
40
Mixture Ratio (MR)
H2O vs Mixture Ratio (MR) for Different O/F Ratios
H2O Mass Fraction
0.15
0.1
0.05
0
O/F Ratios
O/F = 1.00
O/F = 1.21
O/F = 1.42
O/F = 1.63
O/F = 1.84
O/F = 2.05
O/F = 2.26
O/F = 2.47
O/F = 2.68
O/F = 2.89
O/F = 3.11
O/F = 3.32
O/F = 3.53
O/F = 3.74
O/F = 3.95
O/F = 4.16
O/F = 4.37
O/F = 4.58
O/F = 4.79
O/F = 5.00
0
10
20
30
40
Mixture Ratio (MR)


*O vs Mixture Ratio (MR) for Different O/F Ratios
7.000e−3
O/F Ratios
O/F = 1.00

3D plot for Mixture Ratio, O2 Mass Fraction, and Temperature for Different Hybrid O/F Ratios

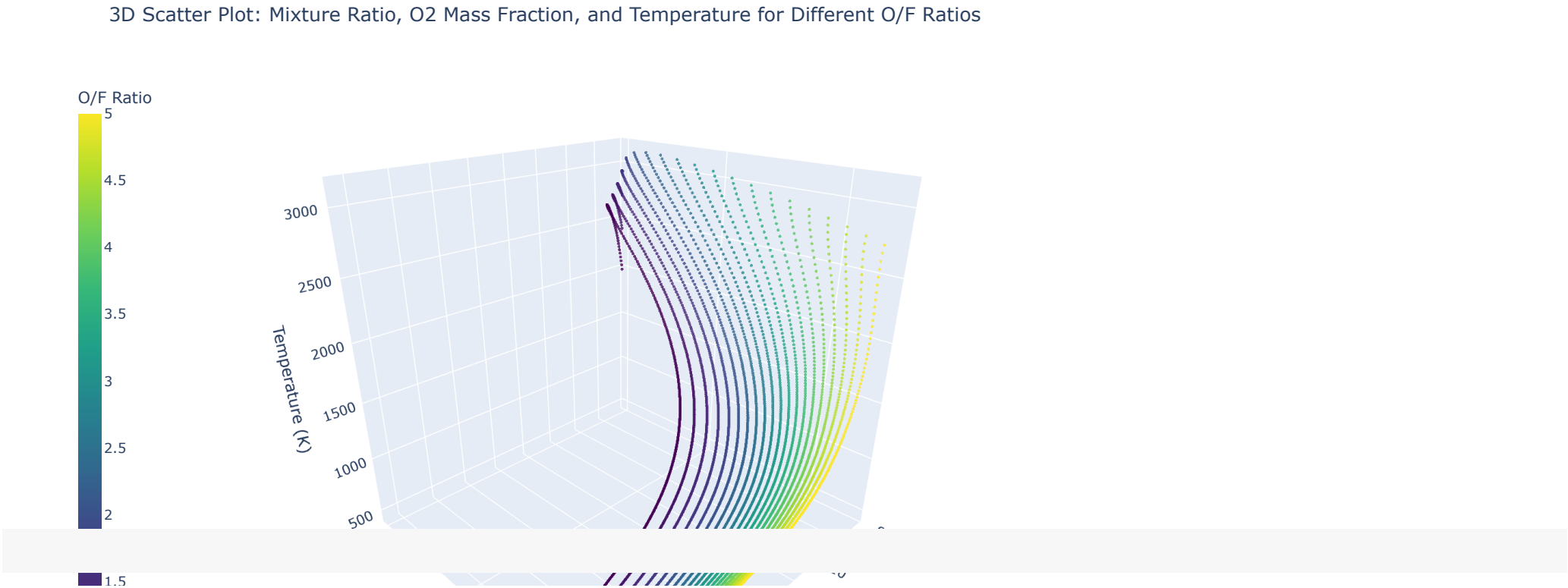
```
!pip install plotly
```

```
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (5.13.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly) (8.2.2)
```

ChatGPT

```
import pandas as pd
import plotly.graph_objects as go


# Create the 3D scatter plot with thicker lines
fig = go.Figure(data=go.Scatter3d(
    x=df['MixtureRatio'],
    y=df['O2_MF'],
    z=df['Tc(k)'],
    mode='markers',
    marker=dict(
        size=1.5,  # Increase the marker size for thicker lines
        color=df['H_o_f'],  # Use the O/F ratio as the color scale
```

```
        colorscale='Viridis',  # Choose a color scale
        colorbar=dict(title='O/F Ratio', thickness=20, xanchor='left', x=-0.1),
        opacity=0.8,  # Adjust the marker opacity for better visibility
    )
))

# Set axis labels and title
fig.update_layout(scene=dict(
    xaxis_title='Mixture Ratio',
    yaxis_title='O2 Mass Fraction',
    zaxis_title='Temperature (K)',
), title='3D Scatter Plot: Mixture Ratio, O2 Mass Fraction, and Temperature for Different O/F Ratios')

# Add hover text to display additional information on each data point
hover_text = (
    "Mixture Ratio: %{x:.2f}<br>"
    "O2 Mass Fraction: %{y:.4f}<br>"
    "Temperature (K): %{z:.2f}<br>"
    "O/F Ratio: %{marker.color:.2f}"
)
fig.update_traces(text=hover_text, hoverinfo='text')

# Customize the layout and interaction
fig.update_layout(
    scene=dict(
        aspectmode='cube',  # Use a cube aspect ratio for a more balanced view
    ),
    scene_camera=dict(
        eye=dict(x=1.2, y=1.2, z=0.8),  # Adjust the initial camera view
    ),
    scene_dragmode='orbit',  # Allow users to orbit the plot
    hoverlabel=dict(bgcolor='white', font_size=12),  # Set the hover label style
    height=700,  # Set the height of the plot (increase from the default)
    width=1000,   # Set the width of the plot (decrease from the default)
)

# Show the 3D scatter plot
fig.show()
```



Conclusion

- In this notebook, we performed a comprehensive chemical equilibrium analysis for a hybrid ramjet propulsion system. By using the "rocketcea" library and various Python tools, we calculated important performance parameters such as specific impulse, combustion temperature, and mass fraction of species. The visualizations helped us understand the relationships between different parameters and their variations with mixture ratios and chamber pressures for a given hybrid ratio.

Key Findings:

- The specific impulse increased with higher mixture ratios, indicating improved propellant efficiency.
- Combustion temperature exhibited a non-linear relationship with mixture ratios, with an optimal value for efficient combustion.
- Mass fraction of species are moving twoards saturation as the oxygen percent increases in the hybrid also with the air.
- The CSV files generated during the analysis allow for further in-depth investigations and comparisons with other propulsion systems.

Overall, this analysis provides valuable insights into the performance characteristics of the hybrid ramjet system, helping us make informed decisions regarding the system's design and operational parameters.

Additional Resources

- For further exploration and understanding of the topics covered in this notebook, you may find the following resources helpful:
- rocketcea Documentation: Link to the rocketcea library documentation, providing detailed information about the library's functions and capabilities.
- Bokeh Documentation: Link to the Bokeh library documentation, offering guidance on creating interactive visualizations using Bokeh.
- Chemical Equilibrium Analysis: Additional reading on chemical equilibrium and its applications in propulsion systems.
- Ramjet and Hybrid Propulsion: Deepen your knowledge of ramjet engines and hybrid propulsion technologies to understand their significance in the aerospace industry.

Feel free to explore these resources to expand your understanding of the subject matter.

✓ 0s completed at 2:40 PM