In [64]:
```
import numpy as np
import pandas as pd
from numba import njit
import matplotlib.pyplot as plt
import mplcyberpunk
plt.style.use("cyberpunk")
plt.rcParams["figure.figsize"] = (15, 10)
```

Lista 4 - PRP41

Leonardo Antonio Lugarini

Questão 1:

Parameter	FUEL (C_2H_5OH)	OXIDANT ($O_2(L)$)
REACTANT	WT FRACTION	TEMP (K)
FUEL	1.0000000	400.000
OXIDANT	1.0000000	90.170

- O/F= 1.60000

Pressure Ratio (Pinf/P)	CHAMBER	THROAT	EXIT	EXIT	EXIT
P, BAR	30.000	17.368	1.4809	0.49714	0.23281
T, K	3351.20	3188.32	2483.01	2134.41	1892.65
RHO, KG/CU M	2.4259	1.4946	1.7052-1	6.7005-2	3.5429-2

H, KJ/KG	-2148.83	-2804.10	-5295.27	-6174.14	-6704.48
U, KJ/KG	-3385.48	-3966.11	-6163.71	-6916.08	-7361.62
G, KJ/KG	-42110.3	-40823.3	-34904.0	-31626.0	-29273.5
S, KJ/(KG)(K)	11.9245	11.9245	11.9245	11.9245	11.9245
M, (1/n)	22.531	22.813	23.773	23.919	23.947
(dLV/dLP)t	-1.03227	-1.02681	-1.00431	-1.00079	-1.00018
(dLV/dLT)p	1.6108	1.5347	1.1102	1.0225	1.0056
Cp, KJ/(KG)(K)	6.5169	6.1254	3.1450	2.3353	2.1367
GAMMAs	1.1295	1.1278	1.1531	1.1832	1.1964
SON VEL,M/SEC	1181.9	1144.8	1000.7	937.0	886.7
MACH NUMBER	0.000	1.000	2.507	3.028	3.404

PERFORMANCE PARAMETERS

Ae/At	CSTAR, M/SEC	CF	Ivac, M/SEC	Isp, M/SEC
1.0000	1753.3	0.6529	2159.8	1144.8
4.0000	1753.3	1.4308	2854.8	2508.6
9.0000	1753.3	1.6183	3098.9	2837.4
16.000	1753.3	1.7216	3236.2	3018.5

In [65]:
```
#utilizando dados da CHAMBER
c_star = 1753.3 #m/s
P0 = 30*1e5 #Pa
```

```
At = np.pi*(3*1e-2)**2 #m^2
Isp = 1144.8 #m/s
Ivac = 2159.8 #m/s

dot_m = P0*At/c_star

g0 = 9.81 #m/s2
E = dot_m*Isp*g0 #N

data_dict = {
    "Vazão (kg/s)": [dot_m],
    "Impulso Esp. Vácuo (m/s)": [Ivac],
    "Empuxo (N)": [E]
}

df = pd.DataFrame(data_dict, index=["Values"])
df
```

Out[65]:

	Vazão (kg/s)	Impulso Esp. Vácuo (m/s)	Empuxo (N)
Values	4.837906	2159.8	54332.042555

Questão 2:

THEORETICAL ROCKET PERFORMANCE ASSUMING EQUILIBRIUM

COMPOSITION DURING EXPANSION FROM FINITE AREA COMBUSTOR

- Pin = 435.1 PSIA
- Ac/At = 4.0000
- Pinj/Pinf = 1.012641
- CASE = _______________

REACTANT COMPOSITION

Reactant	WT Fraction (See Note)	Energy (KJ/KG-MOL)	Temp (K)

FUEL C2H5OH	1.0000000	-227485.078	400.000
OXIDANT O2(L)	1.0000000	-12979.000	90.170

- O/F = 1.60000
- %FUEL = 38.461538
- R,EQ.RATIO = 1.248436
- PHI,EQ.RATIO = 1.302361

PERFORMANCE PARAMETERS

Parameter	INJECTOR	COMB END	THROAT	EXIT 1	EXIT 2	EXIT 3
Pinj/P	1.0000	1.0258	1.7491	20.509	61.086	130.44
P, BAR	30.000	29.247	17.152	1.4628	0.49111	0.23000
T, K	3351.20	3345.81	3187.08	2482.90	2134.69	1892.99
RHO, KG/CU M	2.4259 0	2.3692 0	1.4764 0	1.6844-1	6.6183-2	3.4994-2
H, KJ/KG	-2148.83	-2164.73	-2803.88	-5294.45	-6173.29	-6703.71
U, KJ/KG	-3385.48	-3399.20	-3965.59	-6162.88	-6915.34	-7360.97
G, KJ/KG	-42110.3	-42077.5	-40823.2	-34913.4	-31638.4	-29285.6
S, KJ/(KG)(K)	11.9245	11.9292	11.9292	11.9292	11.9292	11.9292
M, (1/n)	22.531	22.535	22.810	23.772	23.919	23.947
(dLV/dLP)t	-1.03227	-1.03221	-1.02687	-1.00434	-1.00079	-1.00018
(dLV/dLT)p	1.6108	1.6106	1.5362	1.1109	1.0227	1.0057
Cp, KJ/(KG)(K)	6.5169	6.5215	6.1377	3.1518	2.3373	2.1373
GAMMAs	1.1295	1.1294	1.1277	1.1529	1.1831	1.1963

SON VEL,M/SEC	1181.9	1180.8	1144.6	1000.6	937.0	886.7
MACH NUMBER	0.000	0.151	1.000	2.507	3.028	3.404

TRANSPORT PROPERTIES (GASES ONLY)

CONDUCTIVITY IN UNITS OF MILLIWATTS/(CM)(K)

Parameter	INJECTOR	COMB END	THROAT	EXIT 1	EXIT 2	EXIT 3
VISC,MILLIPOISE	1.0766	1.0755	1.0419	0.88075	0.79081	0.72497

In [66]:

```
#### Dados do CEA Online

T = np.array([3351.20, 3345.81, 3187.08, 2482.9, 2134.69]) #K

Gammas = np.array([1.1295, 1.1294, 1.1277, 1.1529, 1.1831])

Mach = np.array([0.0, 0.151, 1.0, 2.507, 3.028])

A_At = np.array([4.0, 4.0, 1.0, 4.0, 9.0])

MM = np.array([22.531, 22.535, 22.810, 23.772, 23.919])/1000 #kg/mol

##Constantes

R = 8.3144621 #J/(K mol)
R_t = (R/MM[2]) #J/ (Kg K)
T_k = T[0]
T_w = 1000
D_t = 6*1e-2 #m
r_m = 2*1e-2 #m
u_t = 1.0419*1e-4 #Pa.s

Cp = Gammas[2]*R_t/(Gammas[2] - 1)
Pr_t = (4*Gammas[2])/(9*Gammas[2] - 5)
C = (0.026/(D_t * r_m)**0.1)*((u_t**0.2*Cp)/Pr_t**0.6)*((dot_m/At)**0.8)

##Calculo dos coeficientes de transferencia de calor
```

```python
tau = 1 + ((Gammas - 1)/2)*Mach**2
h_g = C * (A_At**-0.9) * (((T_w/T_k)*tau/2 + 1/2)**-0.68) * tau**-0.12

df = pd.DataFrame({'Temperature (Tg)': T, 'Heat Transfer Coefficient (h_g)': h_g}, index=["x1", "x2", "x3", "x4", "x
df.round(2)
```

Out[66]:

	Temperature (Tg)	Heat Transfer Coefficient (h_g)
x1	3351.20	4218.61
x2	3345.81	4216.89
x3	3187.08	14437.62
x4	2482.90	3747.89
x5	2134.69	1676.29

Questão 3:

In [67]:
```python
#a)
l_canal = 1.5*1e-3
l_rib = 1.5*1e-3
h_canal = 1e-3

l_t = np.pi*D_t

n_ch = int(l_t/(l_canal+l_rib))

print(f"n_ch = {n_ch}")
```

```
n_ch = 62
```

In [68]:
```python
#b)
Cp_etanol = 3414
lambda_etanol = 0.1523
mu_etanol = 2.54*1e-4 #Pa.s
f_a = 2
```

```
A_ch = (l_canal)*h_canal
d_h = 2*(A_ch/(l_canal + h_canal))

h_l = 0.023*((dot_m/(n_ch*A_ch))**0.8)*((Cp_etanol**0.4)*(lambda_etanol**0.6)/(mu_etanol**0.4))*(f_a/(d_h**0.2))
print(f"h_l = {h_l:.2f}")
```

```
h_l = 240354.92
```

Questão 4:

In [69]:

```
from scipy.optimize import root

dados_tabela = {
    "Estação": ["x1", "x1", "x2", "x2", "x3", "x3", "x4", "x4", "x5", "x5"],
    "Segmento": ["s1,1", "s1,2", "s2,1", "s2,2", "s3,1", "s3,2", "s4,1", "s4,2", "s5,1", "s5,2"],
    "Comprimento de canal (mm)": [2.5, 2.5, 2.5, 3.0, 3.0, 2.7, 2.97, 2.97, 2.97, 2.97],
}

df_camara = pd.DataFrame(dados_tabela)

k_m = 1.6
dot_m_O = dot_m/(1 + (1/k_m))
dot_m_F = dot_m_O/k_m

eps_ch = 1.5*1e-3
rho_etanol = 790 #kg/m3
v = dot_m_F/(rho_etanol*(A_ch*n_ch))
Re = rho_etanol*v*d_h/mu_etanol

def function_f_d(f_d):
    return f_d**-0.5 + 2*np.log10(eps_ch/(3.7*d_h) + (2.51*(f_d**-0.5))/Re)

sol = root(function_f_d, 1)
f_D = sol.x[0]

def delta_p(L):
    return L*((f_D*rho_etanol*v**2)/(2*d_h))

#Como a entrada da jaqueta é no segmento s4,2, é feito a soma das perdas de pressão a partir de s4,2 ate s1,1
```

```python
df_camara = df_camara.iloc[0:8]
df_camara["Δp (bar)"] = delta_p(df_camara["Comprimento de canal (mm)"]*1e-3)/1e5

P_entrada = df_camara["Δp (bar)"].sum() + 33

df_camara["P (bar)"] = P_entrada - df_camara.iloc[::-1]["Δp (bar)"].cumsum().shift(1)
df_camara = df_camara.fillna(P_entrada)

dados_entrada = {
    "Estação": ["Entrada Jaqueta"],
    "Segmento": [""],
    "Comprimento de canal (mm)": np.nan,
    "Δp (bar)": np.nan,
    "P (bar)": P_entrada
}
entrada = pd.DataFrame(dados_entrada,index=[8])

dados_saida = {
    "Estação": ["Saida Jaqueta"],
    "Segmento": [""],
    "Comprimento de canal (mm)": np.nan,
    "Δp (bar)": np.nan,
    "P (bar)": 33
}
saida = pd.DataFrame(dados_saida,index=[0])

df_camara = pd.concat([saida, df_camara, entrada]).reset_index(drop=True).fillna("").iloc[::-1]
df_camara.round(2)
```

Out[69]:

	Estação	Segmento	Comprimento de canal (mm)	Δp (bar)	P (bar)
9	Entrada Jaqueta				85.62
8	x4	s4,2	2.97	7.058949	85.62
7	x4	s4,1	2.97	7.058949	78.56
6	x3	s3,2	2.7	6.417226	71.50
5	x3	s3,1	3.0	7.130251	65.09

4	x2	s2,2	3.0	7.130251	57.96
3	x2	s2,1	2.5	5.941876	50.83
2	x1	s1,2	2.5	5.941876	44.88
1	x1	s1,1	2.5	5.941876	38.94
0	Saida Jaqueta				33.00

Questão 5:

In [70]:

```
#a)
area_df = pd.DataFrame({"Segmento": ["s1,1", "s1,2", "s2,1", "s2,2", "s3,1", "s3,2", "s4,1", "s4,2"],
"ΔS": [94.25*1e-4, 94.25*1e-4, 94.25*1e-4, 110.64*1e-4, 79.03*1e-4, 72.77*1e-4, 109.76*1e-4, 141.12*1e-4]
})

area_df
```

Out[70]:

	Segmento	ΔS
0	s1,1	0.009425
1	s1,2	0.009425
2	s2,1	0.009425
3	s2,2	0.011064
4	s3,1	0.007903
5	s3,2	0.007277
6	s4,1	0.010976
7	s4,2	0.014112

In [71]:
```python
e = 1e-3 #m
k = 400 #W/m/K

AUX = (1/h_g + e/k + 1/h_l)*dot_m_F*Cp_etanol

T_l = [300]
q = []
T_q = []
T_f = []

i = 0
while i<=3:
    ponteiro = 3-i

    num1 = AUX[ponteiro]*T_l[i]
    if i == 0:
        num2 = 0
    else:
        num2 = q[i-1]*(AUX[ponteiro]/(dot_m_F*Cp_etanol))*area_df[area_df["Segmento"] == f"s{ponteiro+2},1"]["ΔS"].v
    num3 = T[ponteiro]*area_df[area_df["Segmento"] == f"s{ponteiro+1},2"]["ΔS"].values[0]

    den1 = AUX[ponteiro]
    den2 = area_df[area_df["Segmento"] == f"s{ponteiro+1},2"]["ΔS"].values[0]

    T_aux = (num1 + num2 + num3)/(den1+ den2)
    T_l.append(T_aux)
    q.append(dot_m_F*Cp_etanol*(T[ponteiro] - T_l[i+1])/AUX[ponteiro])
    T_q.append(T[ponteiro] - q[i]/h_g[ponteiro])
    T_f.append(T_q[i] - (e/k)*q[i])

    i +=1

final_df = pd.DataFrame({"Tq": T_q, "Tf":T_f, "Tl":T_l[1:], "q":q}, index=["4", "3", "2", "1"])
final_df
```

Out[71]:

	Tq	Tf	Tl	q
4	370.324957	350.530693	317.588968	7.917706e+06

3	620.521949	527.884489	373.716709	3.705498e+07
2	519.939281	490.148311	440.570013	1.191639e+07
1	554.341730	524.844602	475.755316	1.179885e+07

In [72]:

```
#b)
k_718 = 26.3 #W/m/K

sigma = 5.67*1e-8 #W/m^2/K^4

epsilon = 0.95 #emissividade

def T_function(T_f5):
    AUX = sigma*epsilon*(1/h_g[4] + e/k_718)
    return AUX*(T_f5**4) + T_f5 - T[4]

sol = root(T_function, 400)
T_f5 = sol.x[0]

q_5 = sigma*epsilon*(T_f5**4)

T_q5 = q_5*(e/k_718) + T_f5
T_q5

dados_5 = {
    "Tq": T_q5,
    "Tf": T_f5,
    "Tl": np.nan,
    "q": q_5
}
secao5 = pd.DataFrame(dados_5,index=[5])

final_df = pd.concat([secao5, final_df]).iloc[::-1].fillna("")
final_df["Tg"] = T
final_df
```

Out[72]:

	Tq	Tf	Tl	q	Tg

1	554.341730	524.844602	475.755316	1.179885e+07	3351.20
2	519.939281	490.148311	440.570013	1.191639e+07	3345.81
3	620.521949	527.884489	373.716709	3.705498e+07	3187.08
4	370.324957	350.530693	317.588968	7.917706e+06	2482.90
5	1807.368476	1786.505863		5.486867e+05	2134.69

c)