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

# Aircraft parameters
Np = 240                # max number of passengers
range_max = 12000   # range at max payload (km)
wp = 40*1000            # max payload weight (kg)
we = 106*1000           # empty weight
wf = 74*1000    # fuel capacity at max payload
wto = 220*1000          # max take off weight
V = 256                 # cruise TAS
M = 0.85                # cruise mach number
h_initial = 9.5         # initial cruise altitude (km)
LD_optimum = 21          # cruise L/D
betastar = 1/LD_optimum   # 1/(L/D)
S = 315                 # wing area (m^2)
v = 1.0                 # speed ratio = Ve/Ve*

# Engine parameters
OPR_initial = 45        # overall pressure ratio = p03/p02
theta = 6       # turbine entry temp. ratio = T04/T02
nc = nt = 0.9   # turbine and compressor efficiencies
FPR = 1.45      # fan pressure ratio
nf = 0.92       # fan efficiency
ntr = 0.9       # transfer efficiency
cp = 1005
gamma = 1.4
fga = (gamma-1)/gamma
LCV = 42.7*10**6    # keroscene

# Modelling constants
k = 0.015               # Breguet range equation fuel burn offset
c1,c2 = 0.3, 1.0        # aircraft weight correlation
K1,K2 = 0.0125,0.0446   # parabolic drag low constants

#*******************************************************
#  ISA conditions
Tsl = 288.15
psl = 101325
rhosl = 1.225
asl = 340.3
dh = 0.1   # step size in height

hlow = np.arange(0,11+dh,dh)
Tlow = Tsl - 6.5*hlow
plow = psl * (Tlow/Tsl)**5.256
rholow = rhosl * (Tlow/Tsl)**4.256
```

```
alow = asl * (Tlow/Tsl)**0.5
hhigh = np.arange(11+dh, 20+dh, dh)
Thigh = np.array([216.65]*(len(hhigh)))
phigh = plow[-1]*np.exp(-0.1577*(hhigh-11))
rhohigh = rholow[-1]*np.exp(-0.1577*(hhigh-11))
ahigh = asl * (Thigh/Tsl)**0.5

hlist = np.concatenate((hlow,hhigh))
Tlist = np.concatenate((Tlow,Thigh))
plist = np.concatenate((plow,phigh))
rholist = np.concatenate((rholow,rhohigh))
alist = np.concatenate((alow,ahigh))

# print(T[np.where(h==11)])

def find_nearest(array, value):
    idx = (np.abs(array - value)).argmin()
    if idx==0 or idx==len(hlist)-1:
        print("Value lookup out of range!")
    return idx

# plt.plot(hlist,Tlist/Tsl)
# plt.plot(hlist,plist/psl)
# plt.plot(hlist,rholist/rhosl)
# plt.legend(["Temperature","Pressure","Density"])
# plt.xlabel("Altitude (km)")
# plt.ylabel("Ratio to Sea Level")
# plt.grid()
# plt.show()
#*******************************************************************

Nstage = 10
# h = [9.5]
h = np.arange(6,14,0.5)
# OPR = [45]
OPR = np.arange(10,60,5)
s = range_max*1000/Nstage   # distance of 1 stage, m
CO2_ovr = np.zeros((len(h),len(OPR)))
NOx_ovr = np.zeros((len(h),len(OPR)))
fuel_ppkm = np.zeros((len(h),len(OPR))) # fuel burnt per kg payload per km
M_max = np.zeros(len(h))

for i in range(len(h)): # CHANGE ALTITUDES
    index = find_nearest(hlist, h[i])  # altitude
    Ta = Tlist[index]
    pa = plist[index]
    rhoa = rholist[index]
    aa = alist[index]

```

```
    for j in range(len(OPR)):   # CHANGE OVERALL PRESSURE RATIO
        ncycle = (theta*(1-1/OPR[j]**fga)*nt-(OPR[j]**fga-1)/nc)/(theta-1-(OPR[j]**fga-1)/nc)
        w = we + wp + wf    # total mass
        Eppkm_CO2, Eppkm_NOx, M = [],[],[]
        for stage in range(Nstage):
            print("****************************************************" )
            print("h = ", h[i], "km", "OPR = ", OPR[j], "Stage ",stage+1, " out of ", Nstage)
            Ve_star = (w*9.81/(0.5*rhosl*S))**0.5 * (K2/K1)**0.25  # optimum EAS
            Ve = Ve_star * v
            V = Ve * (rhosl/rhoa)**0.5   # TAS
            print("TAS = ", "%.2f" % V, " m/s")
            rho = rhosl * (Ve/V)**2

            M.append(V/aa)   # Mach number
            if M[-1]>M_max[i]:
                M_max[i]=M[-1]
            print("M = ", "%.2f" % M[-1])
            if M[-1]>0.85:
                print("Mach number exceed transonic drag rise, M = ", M[-1])

            p02 = pa * (1+(gamma-1)/2*M[-1]**2)**(1/fga)
            Mj = ((2/(gamma-1))*((FPR*p02/pa)**fga-1))**0.5
            print("Mj = ", Mj)
            Tj = Ta * (1+0.5*(gamma-1)*M[-1]**2)/(1+0.5*(gamma-1)*Mj**2)*FPR**(fga/nf)
            nprop = 2*(1+Mj/M[-1]*(Tj/Ta)**0.5)**-1
            beta = 0.5*betastar*(v**2+1/v**2)
            H = nprop*ncycle*ntr * 1/beta * LCV / 9.81
            print("H = ", "%.2f" % (H/1000), " km")
            wnew = w / math.exp(s/H)
            wf_burnt = w - wnew     # this stage, in kg
            print("mf = ", "%.2f" % wf_burnt, " kg")
            w = wnew
            # T02 = Ta + V**2/(2*cp)
            T02 = Ta * (1+(gamma-1)/2*M[-1]**2)
            T03 = T02 * (1+(OPR[j]**fga-1)/nc)
            EI_NOx = 0.011445*math.exp(0.00676593*T03)  # gNOx / kg air
            EI_CO2 = 3088   # gCO2/kg fuel, depends only on fuel

            Eppkm_CO2.append(wf_burnt / (s/1000*Np) * EI_CO2)  # gCO2 per passenger km
            Eppkm_NOx.append(EI_NOx * wf_burnt / (s/1000*Np) * 15.1 * 2)   # 15.1 is Stoichiometric, assume 2x stoichiometric
            print("CO2 Emissions = ", "%.2f" % Eppkm_CO2[-1], " gCO2/pas/km")
            print("NOx Emissions = ", "%.2f" % Eppkm_NOx[-1], " gNO2/pas/km")

```

```python
            CO2_ovr[i][j] = sum(Eppkm_CO2)/len(Eppkm_CO2)
            NOx_ovr[i][j] = sum(Eppkm_NOx)/len(Eppkm_NOx)
            fuel_ppkm[i][j] = (wto-w)/range_max/wp + 0.015*wto/range_max/wp     # fuel burnt per payload per km (kg/kgkm)


for i in range(len(h)):
    for j in range(len(OPR)):
        print("**********************************************************")
        print("h = ", h[i], "km", "OPR = ", OPR[j])
        print("Overall CO2 Emissions = ", "%.2f" % CO2_ovr[i][j], " gCO2/pas/km")
        print("Overall NOx Emissions = ", "%.2f" % NOx_ovr[i][j], " gNOx/pas/km")
        print("Total fuel burnt = ", fuel_ppkm[i][j]*range_max*wp,  " kg")
        print("Fuel per payload km = ", "%.2e" % fuel_ppkm[i][j], " kg fuel/kg payload/km")
        print("**********************************************************")

# SET UP GWP FOR GREEN AND SVENSSON
hlist_s = np.arange(0,15.5,0.5)
GWP_NOx_old = [-7.1,-7.1,-7.1,-4.3,-1.5,6.5,14.5,37.5,60.5,64.7,68.9,57.7,46.5,25.6,4.6,0.6]    # Svensson
GWP_NOx_s = np.zeros(len(hlist_s))
for i in range(len(GWP_NOx_old)):
    if i==0:
        GWP_NOx_s[0] = GWP_NOx_old[0]
    else:
        GWP_NOx_s[i*2-1] = (GWP_NOx_old[i]+GWP_NOx_old[i-1])/2
        GWP_NOx_s[i*2] = GWP_NOx_old[i]

hlist_g = np.arange(5,12.5,0.5)
GWP_NOx_old = np.concatenate((np.linspace(10, 47,5),np.array([63,105,126])))    # Green
GWP_CO2_old = np.concatenate((np.linspace(147, 126,5),np.array([110,100,100])))
GWP_NOx_g = np.zeros(len(hlist_g))
GWP_CO2_g = np.zeros(len(hlist_g))
for i in range(len(GWP_NOx_old)):
    if i==0:
        GWP_CO2_g[0] = GWP_CO2_old[0]
        GWP_NOx_g[0] = GWP_NOx_old[0]
    else:
        GWP_CO2_g[i*2-1] = (GWP_CO2_old[i]+GWP_CO2_old[i-1])/2
        GWP_CO2_g[i*2] = GWP_CO2_old[i]
        GWP_NOx_g[i*2-1] = (GWP_NOx_old[i]+GWP_NOx_old[i-1])/2
        GWP_NOx_g[i*2] = GWP_NOx_old[i]
for i in range(len(hlist_g)):
    GWP_NOx_g[i] = GWP_NOx_g[i]/GWP_CO2_g[i]
```

```python
# plt.plot(hlist_s,GWP_NOx_s)
# plt.xlabel('Altitude (km)')
# plt.ylabel('GWP for NOx')
# # plt.plot(hlist_g,GWP_NOx_g/10*147)
# plt.show()



# # PLOT OVERALL EMISSIONS FOR EACH OPR
# CO2_ovr = np.transpose(CO2_ovr)
# NOx_ovr = np.transpose(NOx_ovr)
# fuel_ppkm = np.transpose(fuel_ppkm)
# fig, ax1 = plt.subplots(3)
# ax1[0].set_ylabel('CO2',color='b')
# ax1[0].set_title('Emissions (g/passenger/km)')
# # ax1[0].set_xlabel('Cruise Altitude (km)')
# ax1[0].plot(h, CO2_ovr[0], '-ob')
# ax1[0].tick_params(axis='y',labelcolor='b')
# ax2 = ax1[0].twinx()  # instantiate a second axes that shares the same x-axis
# ax2.set_ylabel('NOx',color='r')  # we already handled the x-label with ax1
# ax2.plot(h, NOx_ovr[0], '-or')
# ax2.tick_params(axis='y',labelcolor='r')
# fig.tight_layout()  # otherwise the right y-label is slightly clipped
# # plt.show()

# # plt.plot(h,M_max,'-ob')
# # plt.xlabel('Cruise Altitude (km)')
# # plt.ylabel('Maximum Mach Number')
# # plt.plot([10.18]*2,[0.7,1.0],'-r')
# # plt.legend(['Max M','Transonic Drag Onset'])
# # plt.show()




# PLOT OVERALL GWP
ovr_s = np.zeros((len(h),len(OPR)))
ovr_g = np.zeros(len(h))
# NOx_ovr_new = np.zeros(len(h))
for i in range(len(h)):
    for j in range(len(OPR)):
        si = find_nearest(hlist_s, h[i])
        NOx_ovr_new = NOx_ovr[i][j]*GWP_NOx_s[si]
        ovr_s[i][j] = CO2_ovr[i][j]+NOx_ovr_new

    # gi = find_nearest(hlist_g, h[i])
    # NOx_ovr_new = NOx_ovr[0][i]*GWP_NOx_g[gi]
    # ovr_g[i] = CO2_ovr[0][i]+NOx_ovr_new

```

```
# # ax1[1].set_xlabel('Cruise Altitude (km)')
# # ax1[1].set_ylabel('CO2 Emissions (gCO2/passenger/km)', color='b')
# ax1[1].set_title('Relative Greenhouse Effects, normalised with CO2')
# ax1[1].plot(h, CO2_ovr[0], '-ob')
# ax1[1].plot(h, NOx_ovr_new, '-or')
# ax1[1].legend(['CO2','NOx'])

# ax1[2].set_xlabel('Cruise Altitude (km)')
# # ax1[2].ylabel('Overall GWP, normalised with h=9.5km')
# ax1[2].set_title('Overall GWP, normalised with h=9.5km')
# ax1[2].plot(h, ovr_s/ovr_s[find_nearest(h, 9.5)], '-om')
# # plt.plot(h, ovr_g/ovr_g[find_nearest(h, 9.5)], '-or')
# plt.show()

# ## PLOT FUEL BURNT
# # plt.plot(h, fuel_ppkm[0],'-ob')
# # plt.xlabel('Cruise Altitude (km)')
# # plt.ylabel('Fuel Burnt (kg fuel/kg payload/km')
# # plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
# # plt.show()





# # PLOT OVERALL EMISSIONS FOR EACH OPR
# fig, ax1 = plt.subplots(3)
# # ax1[0].set_xlabel('Overall Pressure Ratio')
# # ax1[0].set_ylabel('CO2 Emissions (gCO2/passenger/km)', color='b')
# ax1[0].set_ylabel('CO2',color='b')
# ax1[0].set_title('Emissions (g/passenger/km)')
# ax1[0].plot(OPR, CO2_ovr[0], '-ob')
# ax1[0].tick_params(axis='y',labelcolor='b')
# # ax1[0].legend(['CO2','NOx'])
# ax2 = ax1[0].twinx()  # instantiate a second axes that shares the same x-axis
# # ax2.set_ylabel('NOx Emissions (gNOx/passenger/km)',color='r')  # we already 
   handled the x-label with ax1
# ax2.set_ylabel('NOx',color='r')
# ax2.plot(OPR, NOx_ovr[0], '-or')
# # ax2.legend(['NOx'])
# ax2.tick_params(axis='y',labelcolor='r')
# fig.tight_layout()  # otherwise the right y-label is slightly clipped
# # plt.show()

# ax1[1].plot(OPR, CO2_ovr[0], '-ob')
# ax1[1].plot(OPR, NOx_ovr[0]*66.8, '-or')
# ax1[1].legend(['CO2','NOx'])
# ax1[1].set_title('Relative Greenhouse Effects, normalised with CO2')
# # plt.show()
```

```python

# # PLOT OVERALL GWP
# # plt.xlabel('Overall Pressure Ratio')
# ax1[2].set_title('Overall GWP, normalised with OPR=45')
# ovr = CO2_ovr[0]+NOx_ovr[0]*66.8
# ax1[2].plot(OPR, ovr/ovr[5], '-om')
# ax1[2].set_xlabel('Overall Pressure Ratio')
# plt.show()

# # plt.plot(OPR, fuel_ppkm[0],'-ob')
# # plt.xlabel('Cruise Altitude (km)')
# # plt.ylabel('Fuel Burnt (kg fuel/kg payload/km')
# # plt.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
# # plt.show()


# PLOT CONTOUR
colour = plt.get_cmap('plasma_r')
# colour = matplotlib.colors.LinearSegmentedColormap.from_list("", ["green","yellow","red"])
X, Y = np.meshgrid(h, OPR)
cp = plt.contourf(X,Y, np.transpose(ovr_s)/194.074,100,cmap=colour,norm=matplotlib.colors.CenteredNorm(vcenter=0.7, halfrange=0.3))
plt.colorbar(cp,label='Normalised GWP')
plt.xlabel('Cruise Altitude (km)')
plt.ylabel('Overall Pressure Ratio')
plt.show()

cp = plt.contourf(h,OPR, np.transpose(fuel_ppkm),100,cmap=colour,norm=matplotlib.colors.CenteredNorm(vcenter=1.5e-4, halfrange=0.125e-4))
plt.colorbar(cp, label='kg fuel / kg payload / km',format='%.2e')
plt.xlabel('Cruise Altitude (km)')
plt.ylabel('Overall Pressure Ratio')
plt.show()












s_total = [s/1000]
```

```
for i in range(Nstage-1):
    s_total.append(s_total[-1]+s/1000)

## PLOT VARIATION OF MACH AND X-T DIAGRAM
# plt.plot(s_total,M)
# plt.xlabel('Distance Travelled (km)')
# plt.ylabel('Mach Number')
# plt.show()
# speed=[]
# time=[]
# time2 = 12000000/M[0]/aa/3600
# for i in range(Nstage):
#     speed.append(M[i]*aa)
#     time.append(s/speed[i]/3600)
# for i in range(Nstage-1):
#     time[i+1]=time[i]+time[i+1]
# plt.plot([0]+time,[0]+s_total)
# plt.plot([0,time2],[0,12000])
# plt.xlabel('Time (hours)')
# plt.ylabel('Distance Travelled (km)')
# plt.xlim([0, time[-1]])
# plt.ylim([0, s_total[-1]])
# plt.legend(['Optimum Mach across 10 stages','Constant M = $M_{initial}$'])
# plt.show()

## PLOT VARIATION OF EMISSIONS ACROSS STAGES
# plt.plot(s_total,Eppkm_CO2)
# plt.plot([s_total[0],s_total[-1]],[CO2_ovr[0]]*2)
# plt.xlabel('Distance Travelled (km)')
# plt.ylabel('CO2 Emissions (gCO2/passenger/km)')
# plt.legend(['Emissions each stage','Overall Emissions'])
# plt.show()
# plt.plot(s_total,Eppkm_NOx)
# plt.plot([s_total[0],s_total[-1]],[NOx_ovr[0]]*2)
# plt.xlabel('Distance Travelled (km)')
# plt.ylabel('NOx Emissions (gNOx/passenger/km)')
# plt.legend(['Emissions each stage','Overall Emissions'])
# plt.show()
```