CUED - Engineering Tripos Part IIB 2021-2022 **Module Coursework**

Module	4A7	Title of report	Aviation and the Environment Coursework

Date submitted: 07/12/2021	Assessment for this module is 100% coursework of which this assignment forms 50%

UNDERGRADUATE STUDENTS ONLY		**POST GRADUATE STUDENTS ONLY**			
Candidate number:	5726A	Name:		College:	

Feedback to the student

☐ **See also comments in the text**

		Very good	**Good**	Needs improvmt
CONTENT	**Completeness, quantity of content:** Has the report covered all aspects of the lab? Has the analysis been carried out thoroughly?			
	Correctness, quality of content Is the data correct? Is the analysis of the data correct? Are the conclusions correct?			
	Depth of understanding, quality of discussion Does the report show a good technical understanding? Have all the relevant conclusions been drawn?			
	Comments:			
PRESENTATION	**Attention to detail, typesetting and typographical errors** Is the report free of typographical errors? Are the figures/tables/references presented professionally?			
	Comments:			

Marker: Date:

Aviation and the Environment Coursework

CCN: 5726A

08/12/2021

1 Summary

Aircraft engines and their cruise operating conditions have been optimised to maximise commercial profits which have led to some focus on fuel efficiency but not on their environmental impacts. This coursework asks the question: How could the environmental impact be improved by operating aircraft differently? A code is developed to investigate effects of varying cruise altitude and overall pressure ratio (OPR) to help answer this question. Other limits on operating conditions are considered and the need to compromise between cost (fuel burn) and environmental impacts is discussed.

2 Model

This coursework was carried out using Python 3.7. The full script can be found in the Appendix. The atmosphere is modelled with the International Standard Atmosphere (ISA) conditions:

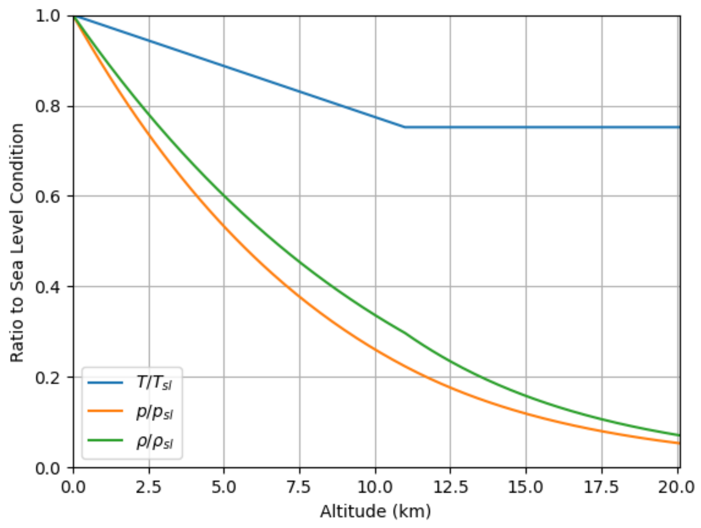


Figure 1: ISA conditions at different altitudes, plotted with matplotlib

A look-up table was created to allow the code to get atmospheric conditions at any altitude. Next, basic reference data of the aircraft and engine were set:

Table 1: Model Reference Data

Aircraft		**Engine**	
Number of Passengers:	240	Temperature Ratio θ:	6
Range:	12000 km	Turbine, Compressor Efficiencies η_t, η_c	0.9
Payload Mass w_p:	40 tonnes	Fan Pressure Ratio FPR:	1.45
Empty Mass w_e:	106 tonnes	Fan Efficiency η_f:	0.92
Initial Fuel Mass w_f:	74 tonnes	Transfer Efficiency η_{tr}:	0.9
Take Off Mass w_{TO}:	220 tonnes	Parabolic drag law constants $K1$:	0.0125
Wing Area S:	315 m^2	Parabolic drag law constants $K2$:	0.0446

2.1 Multiple Staged Flight

The flight is split into 10 stages, with all calculated quantities being constant within each stage. Sequence of calculated values are summarised (detailed calculations in Appendix):

Optimum Equivalent Air Speed (EAS) $\rightarrow$ Actual EAS $\rightarrow$ True Air Speed (TAS) $\rightarrow$ Aircraft Mach $\rightarrow$ Jet Mach $\rightarrow$ Jet Temperature $\rightarrow$ Propulsive Efficiency $\rightarrow$ L/D $\rightarrow$ Range Parameter H $\rightarrow$ New Weight & Fuel Burnt $\rightarrow$ Emission Indices (EI_{CO_2} & EI_{NO_X}) $\rightarrow$ Total Emissions per stage

After each stage has been iterated, the total fuel burnt and emissions for both Carbon Dioxide (CO_2) and Nitrogen Oxides (NO_X) are summed up to calculate the overall fuel burnt per passenger-km and emissions per passenger-km. These are the most important quantities that are compared when varying parameters.

At a flight of $altitude = 9.5km$ and $OPR = 45$, the variation of Mach numbers and emissions during the whole flight can be visualised:

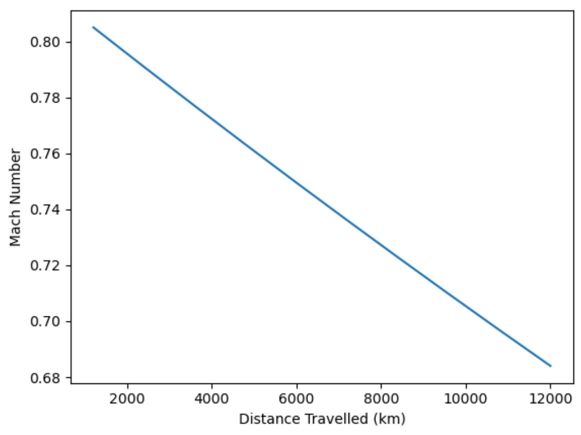


Figure (2a) Mach number during flight

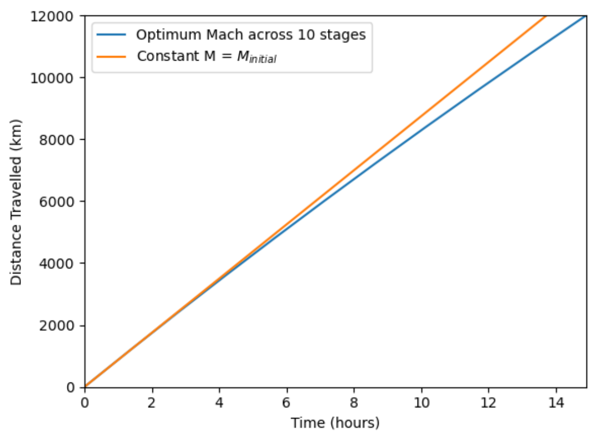
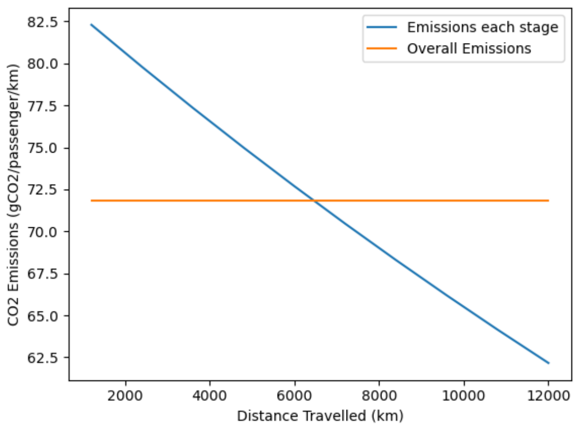


Figure (2b) x-t diagram of the whole flight



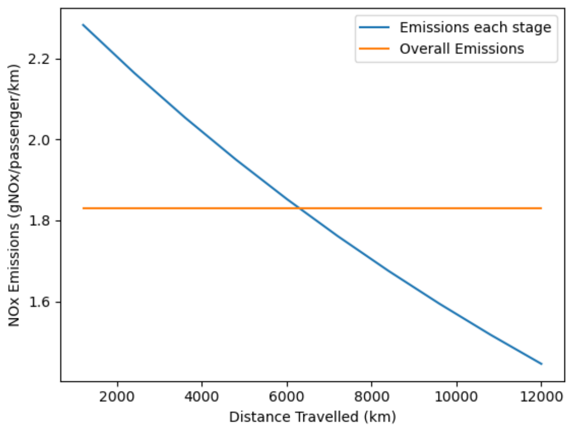


Figure 3: Emissions / pass-km for CO_2 and NO_X

Because the altitude and optimum L/D are fixed, the velocity and Mach number decrease as fuel is burnt. In Figure 2a, the Mach number almost decreases linearly. This decrease lengthens the flight duration compared to if the Mach number was kept constant at the initial speed. This can be seen in Figure 2b, where the flight duration increases from 13.6 to 14.8 hours.

This optimising of flight speed leads to a decrease in both CO_2 and NO_X emissions per passenger-km. In Figure 3, both emissions decrease almost linearly relative to the overall emission per passenger-km. This is evidence for how shorter journeys have a lower emission per passenger-km, because less fuel needs to be carried, leading to lower optimum Mach numbers and hence emissions.

2.2 Model Limitations

As weight decreases due to fuel burn, to keep the lift coefficient (C_L) constant the velocity or density must decrease. The latter can be done by flying at higher altitudes. In this model, each flight has a fixed cruise altitude and an decreasing flight speed. Whereas in practice flight speed is fixed with an increasing cruise altitude, known as 'cruise climb'.

Secondly, the calculations for emissions do not take into account 'k' which adjusts the fuel burn equation for the initial take off and climb. However, this is negligible as k is typically 1.5% for long journeys.

Lastly, this model ignores the effects of transonic drag rise, such as buffeting and shock-induced separations, in the optimum velocity. A check has been implemented to print out a warning if Mach exceeds 0.85, which it can be seen later this occurs for some higher cruise altitudes. This must be considered when analysing the results.

3 Effects of Cruise Altitude

Although in the model cruise altitude is fixed for each flight, it can be varied for different flights. Altitudes from 8km to 12.5km are investigated, with OPR fixed at 45.

3.1 Fuel Burnt

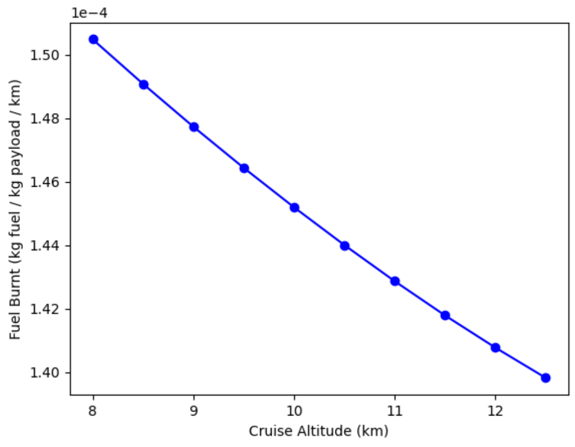


Figure (4a) Overall fuel burnt

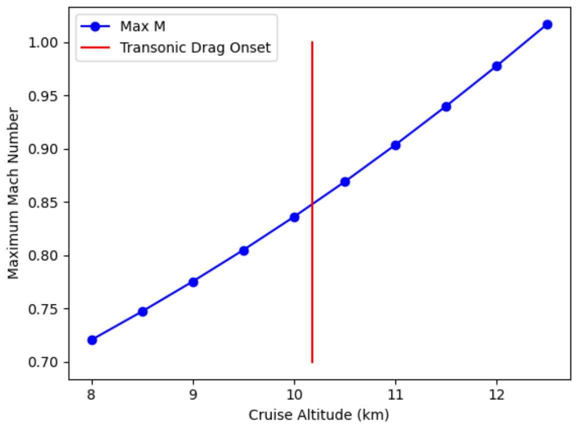


Figure (4b) Maximum Mach number

Figure 4a suggests that the higher the cruise altitude the better the fuel economy. However, the Mach number also increases due to a lower density and a fixed C_L. Figure 4b shows that the maximum Mach reaches 0.85 at an altitude of 10.2km, where the transonic drag becomes significant. This implies the optimum altitude for fuel economy is just beyond 10.2km, where the drawbacks of transonic drag outweigh the decrease in fuel burnt. This agrees with practice, which places this optimum height at 35000ft, or 10.8km.

One might make the assumption that lower fuel burnt means lower CO_2 and NO_X emissions. However, not only does altitude affect the atmospheric temperature, hence T_{03} and the NO_X emission index (EI_{NO_X}), but more significantly the Global Warming Potential (GWP) also scales with height.

3.2 Global Warming Potential

GWP describes the relative global warming impacts of 1 tonne of a greenhouse gas relative to 1 tonne of CO_2. Svennson et al. (2004) quantified the GWP of NO_X for varying altitudes. Figure 5 shows that the impacts of NO_X are significant around $h = 8km$ to $12km$, 50-60 times larger than CO_2. However, this is the impacts of the same mass of gas, in our analysis it can be seen that the amount of NO_X produced is also a magnitude smaller than CO_2.

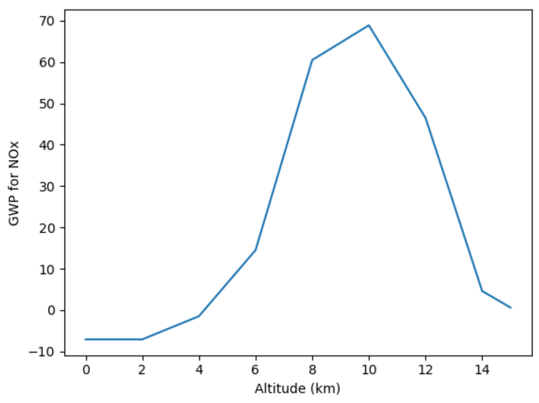


Figure 5: Svennson et al.'s GWP of NO_X relative to CO_2

3.3 Emissions and GWP

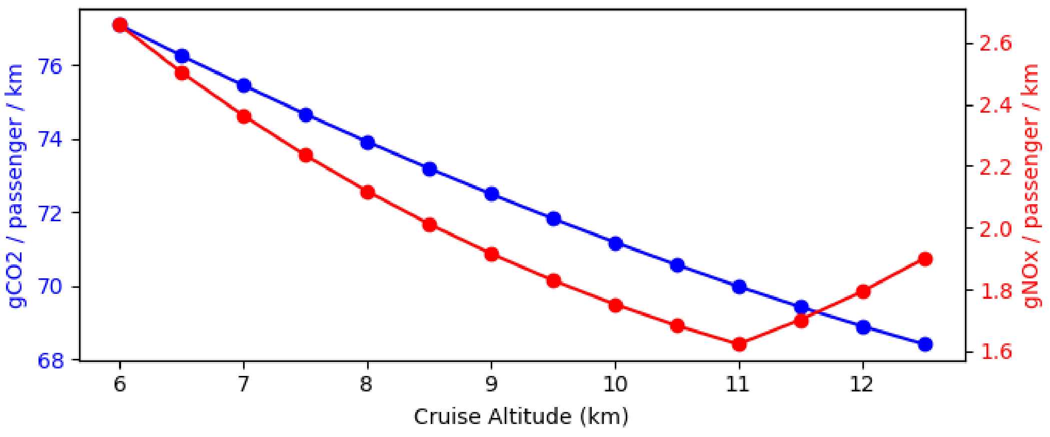


Figure 6: Emissions per passenger-km against altitude

Figure 6 shows that increasing altitudes decreases both gases' emissions but up to $11km$ for NO_X. CO_2 follows the same trend as fuel burnt because EI_{CO_2} is constant at $3088gCO_2/kgfuel$. Above $11km$ the atmospheric temperature is constant, which with an increasing velocity means T_{02}, T_{03} and EI_{NO_X} all increase. Below 11km the lower temperature outweighs the higher velocity, leading to an overall lower emission. This explains the discontinuity at $11km$. To understand the full impact of these emissions, the GWP from Figure 5 is multiplied to the NO_X emissions:

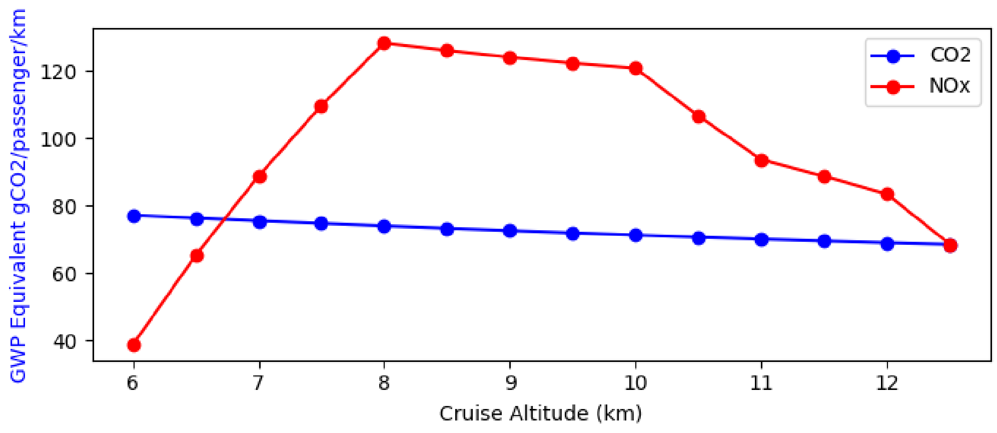


Figure 7: GWP Equivalent $gCO_2/passenger - km$

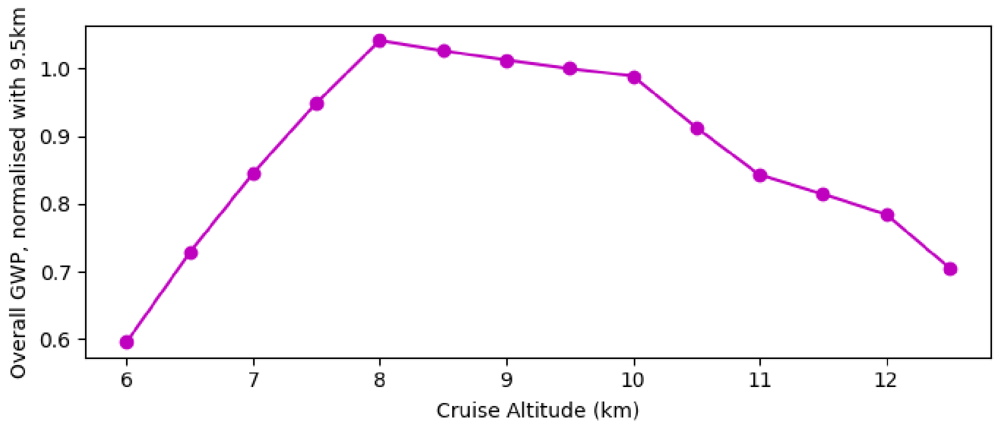


Figure 8: Overall GWP normalised with $h = 9.5km$

The GWP of both emissions are summed up in Figure 8, where it is normalised with the value at $9.5km$. It can be seen that the global warming effects are the highest around $8 - 10km$, dropping rapidly on either side. This is due to the GWP of NO_X (Figure 5) being the more dominating factor. The high levels of emissions at low altitude (and later increasing NO_X past 11km) are outweighed by the negligible relative global warming effects at these altitudes. Ideally the cruise altitude should be very low or very high, but limitations of transonic drag and minimum altitude apply, which will be discussed in 5.1.

4 Effects of Engine Overall Pressure Ratio

OPR from 20 to 55 are investigated, with cruise altitude fixed at 9.5km.

4.1 Fuel Burnt

The cycle efficiency increases as OPR increases, leading to a decreasing overall fuel burnt. This decrease levels off at $OPR = 55$ as seen in Figure 10. The optimum OPR for fuel economy is at the highest OPR possible, limited by current technology (discussed later).

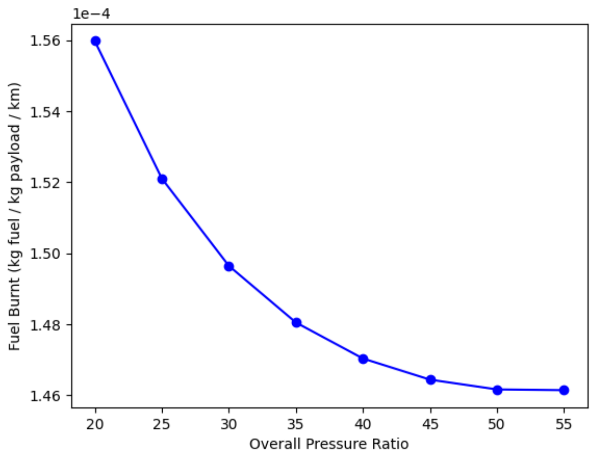


Figure 9: Overall fuel burnt

4.2 Emissions and GWP

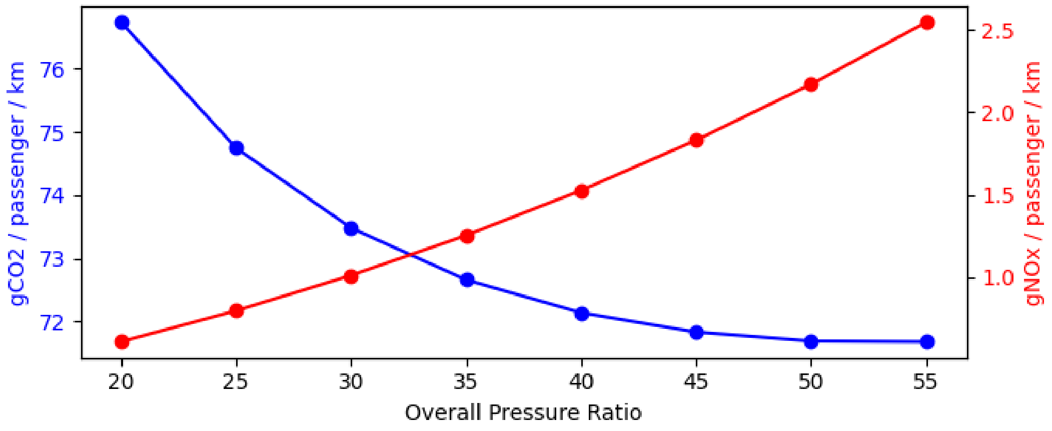


Figure 10: Emissions per passenger-km against OPR

The CO_2 emissions follow the same trend as fuel burnt because EI_{CO_2} is constant at $3088 gCO_2/kgfuel$. The NO_X emissions have an opposite trend which increases as OPR increases. This is a result of a higher compressor exit temperature, hence T_{03} and EI_{NO_X}. The higher EI_{NO_X} exceeds the lower fuel burnt. To understand the full impact of these emissions, a $GWP = 66.8$ at $9.5km$ from Figure 5 is multiplied to NO_X emissions:

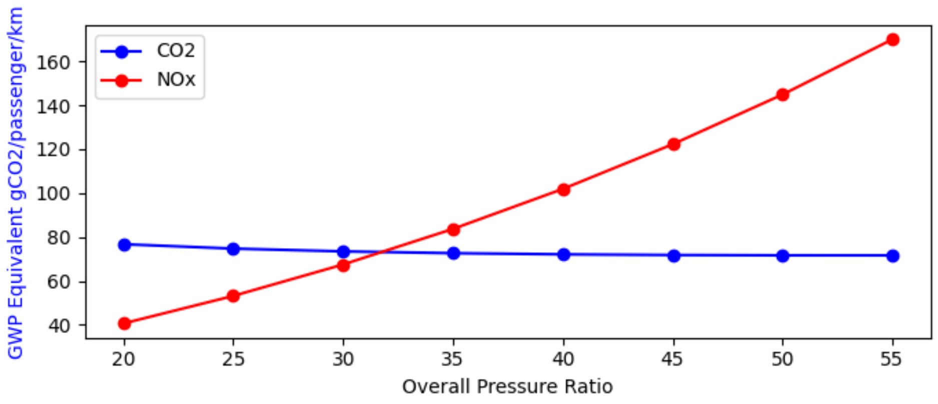


Figure 11: GWP equivalent gCO_2/passenger-km

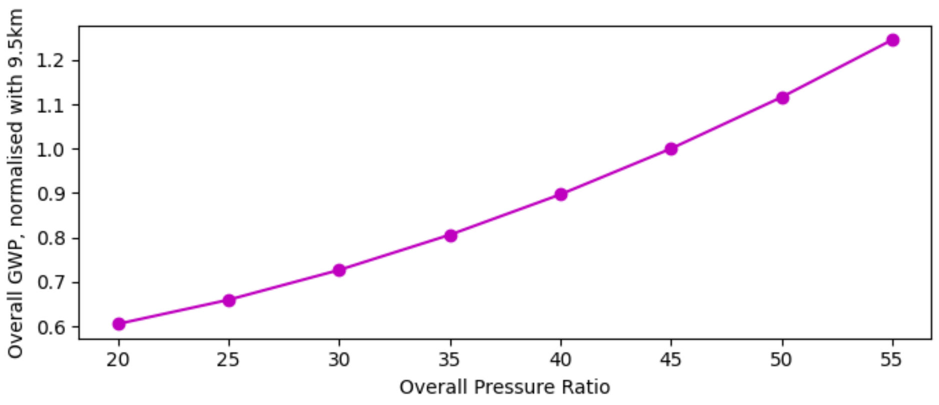


Figure 12: Overall GWP normalised with OPR=45

Combining the global warming effects of both gases shows that the overall effects are dominated by the NO_X at high OPR. In Figure 12 the lower the OPR, the lower the overall global warming impacts the emissions have. This concludes that the optimum OPR for the environment is at the lowest OPR possible. However, this contradicts the optimum OPR for fuel economy, and this will be discussed in the next section. Moreover, the altitude was set at 9.5km, where the GWP of NO_X is significant. Therefore, it is worthwhile to run the same model for all altitudes and OPR.

5 Optimum Operating Points

By running the program for altitudes from $6 - 13.5km$ and OPR from $10 - 55$, contours for the overall global warming potential and fuel burnt are visualised. This allows the best operating points to be identified. This will show what is best for the environment versus what is best in terms fuel burnt and hence cost.

5.1 The Environment

Figure 13 shows that the best operational points for the environment lie in regions of low OPR, at very high or very low altitudes. However, the maximum altitude is limited by the onset of transonic wave drag, which occurs at around $10.8km$ as discussed in section 3.1. Above this altitude the extra emissions due to transonic drag starts to dominate the lower GWP due to altitude. Secondly, the minimum altitude is limited by mountain ranges, for example the Himalayan mountain ranges can reach up to 9km. This lower limit means the GWP at $9km$ is greater than the GWP at the upper limit of $10.8km$. The minimum OPR is limited by the net work output, or thrust, the engine produces. For temperature ratio of $\theta = 6$, the non-dimensionalised specific power drops abruptly below $OPR = 15$. To conclude, the optimum operating condition to minimise overall global warming effects: $\underline{\boldsymbol{OPR = 15 \text{ and } altitude = 11km}}$.

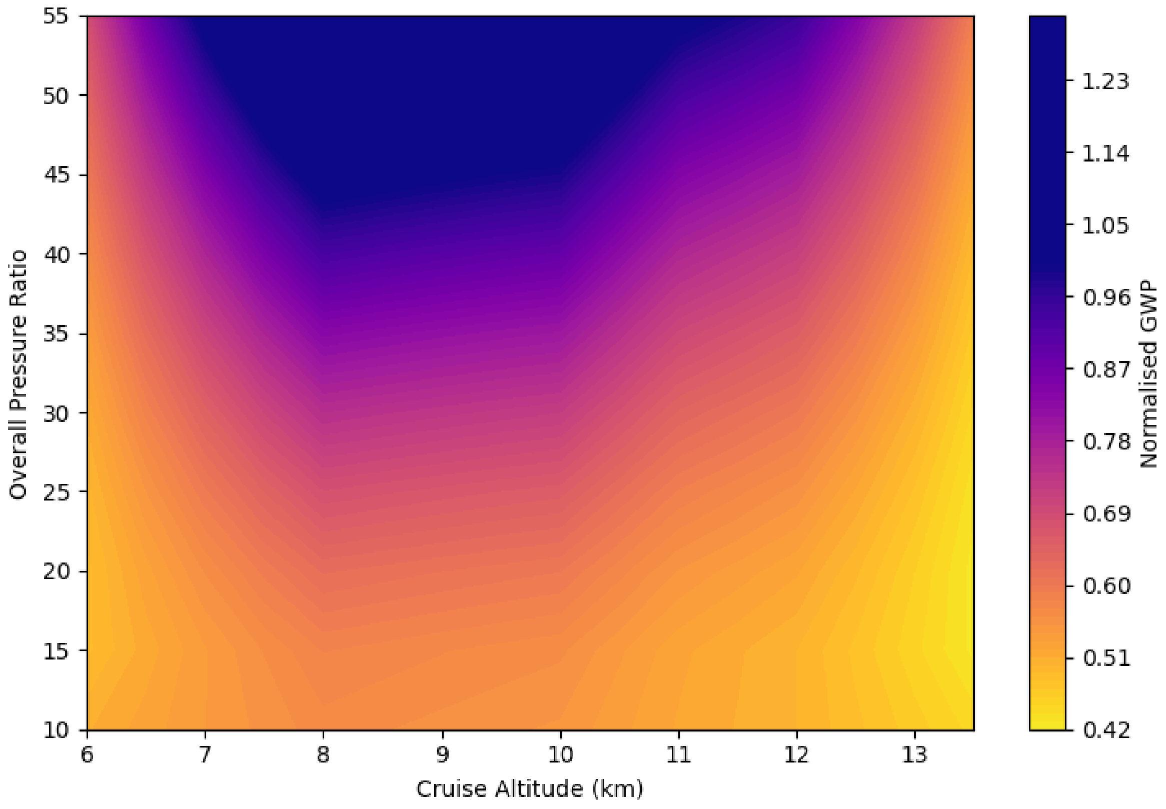


Figure 13: Contour of overall GWP, normalised with $h = 9.5km$, $OPR = 45$

However, for flight paths that do not cross high mountain ranges, it is also worth operating at low altitudes for a similar reduced GWP. This is especially applicable to short flights where the relative fuel burn due to climbing to high altitudes becomes significant.

5.2 The Costs

The operating condition found in the previous section assumes airliners have the environment's best interests in mind. However, fuel burnt is directly related to operating costs. A contour is plotted for the overall fuel burnt in Figure 14. This shows the optimum operating condition for fuel economy is at the highest altitude and OPR possible. The maximum altitude is once again limited by transonic drag, but now the maximum OPR is limited by the material's thermal properties and number of compressor stages required. This places the limit at 55-60.

With this in mind, airliners must compromise between mitigating environmental impacts by flying at low OPR, and minimising costs by flying at high OPR. At the moment the focus is on the latter, hence the development of high OPR engines, but with the right governmental and societal shift airliners could easily fly at lower OPR in favour of the environment. The optimum altitude for environmental and costs is both at the highest altitude allowed by the onset of transonic wave drag.

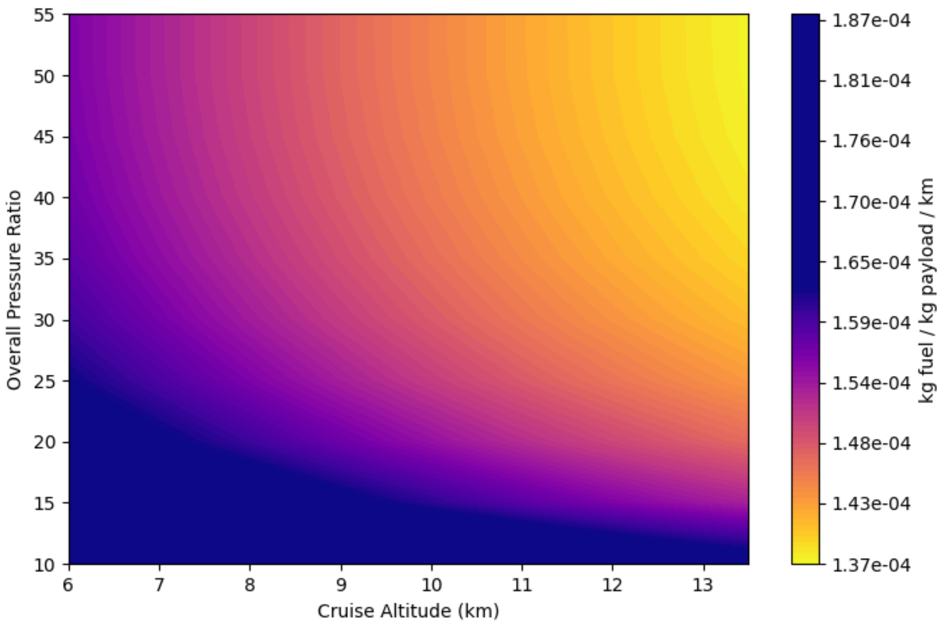


Figure 14: Contour of overall fuel burnt, normalised with h=9.5km, OPR=45

5.3 Other Operating Conditions and Technology

Other conditions such as temperature ratio θ and using 'short hops' for long flights could also be utilised to mitigate global warming effects. The latter was seen briefly in Figure 2a and Figure 3, where shorter journeys reduce fuel burnt and emissions significantly. Technological advancements to improve various efficiencies in the engine and propeller, as well as aerodynamic designs to increase L/D, could also contribute to reducing emissions. Lastly, alternative fuels such as hydrogen and electricity are also means to reduce GWP.

6 Conclusion

A code was developed in Python to analyse the CO_2 and NO_X emissions as well as their relative global warming impacts for various cruise altitudes and OPR. Each flight had a fixed altitude, leading to an increasing Mach and decreasing emissions throughout the flight. Flying at higher altitudes decreased the amount of fuel burnt and emissions. Due to significant GWP of NO_X at mid-range cruise altitudes, the overall GWP for CO_2 and NO_X were lowest outside 8-10km. Flying at higher OPR decreased the amount of fuel burnt but levelled off at 50. CO_2 and NO_X had opposite emission trends with OPR, but NO_X dominated in terms of overall GWP. A contour of overall GWP was plotted for all altitudes and OPR, the optimum operating condition was at $OPR = 15$ and $altitude = 11km$. Maximum and minimum altitudes are limited by transonic wave drag and mountain ranges respectively, and minimum OPR limited by thrust required. A compromise between the environment (low OPR) and cost (high OPR) must be made.

7 Appendix

7.1 Detailed Calculations in Python Iteration

i. Optimum Equivalent Air Speed (EAS) $= V_e^* = \left[\dfrac{mg}{0.5\rho_0 S}\right]^{0.5} \left[\dfrac{K_2}{K_1}\right]^{0.25}$

ii. Actual EAS $= V_e = V_e^* \times \nu$, where $\nu = 1$ is the selected speed ratio

iii. True Air Speed (TAS) $= V = V_e \sqrt{\rho_0/\rho_a}$

iv. Mach Number $M = V/a$, where a is the local speed of sound

v. Jet Mach $M_j^2 = \dfrac{2}{\gamma - 1}\left[(FPR \times p_{02}/p_a)^{(\gamma-1)/\gamma} - 1\right]$

vi. Jet Temperature $\dfrac{T_j}{T_a} = \dfrac{1 + 0.5(\gamma-1)M^2}{1 + 0.5(\gamma-1)M_j^2} FPR^{(\gamma-1)/(\gamma\eta_f)}$

vii. Propulsive Efficiency $\eta_{prop} = 2\left(1 + M_j/M \sqrt{T_j/T_a}\right)^{-1}$

viii. $L/D = \left(\sqrt{K_1 K_2}\,(\nu^2 + 1/\nu^2)\right)^{-1}$

ix. Range Parameter $H = \eta_{prop}\,\eta_{cycle}\,\eta_{tr}\,L/D \times LCV/g$

x. New Weight $w_{end} = w_{start}\, e^{s_{stage}/H}$

xi. $EI_{CO_2} = 3088$, $EI_{NO_X} = 0.011445\, e^{0.006766 \times T_{03}}$, where $T_{03} = T_{02} \times \left(1 + (OPR^{(\gamma-1)/\gamma}/\eta_c)\right)$

xii. Total Emissions for this stage $= EI \times Fuel\ Burnt$

7.2 Full Python Script

NOTE: Change h and OPR in lines 82-85, then comment/uncomment corresponding plots starting at lines 192 (vary h) 264 (vary OPR) 296 (contour)

```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]

```

```python
    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")

        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]
```

```python
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()
```