भारतीय प्रौद्योगिकी संस्थान जम्मू
INDIAN INSTITUTE OF TECHNOLOGY JAMMU
विद्याधनं सर्वधन प्रधानम्

Compressible Fluid Flow

Title : Design of a supersonic aircraft engine

Team:

Rishika Ranyal (2021ume0225)

Rohan Kumar(2021ume0226)

Vansh Singh(2021ume0234)

Vivek Kumar(2021ume0237)

Department of Mechanical Engineering

Indian Institute of Technology (IIT) Jammu

Abstract:

This work focuses on the design and analysis of turbojet cycles for supersonic aircraft. We have taken a case study of SP-71 operating at a cruising Mach number of 3.2 at an altitude of 80,000 ft. The primary objective is to design an efficient intake diffuser that decelerates supersonic airflow to subsonic speeds before reaching the compressor, ensuring stable operation and acquiring the required thrust. The study involves modelling thermodynamic properties, analysing shock wave behaviour, optimising pressure recovery to meet the demands of high-speed flight, and obtaining the required thrust and exit Mach number.The approach undertaken in this study is primarily theoretical and computational, focusing on the aerodynamic and thermodynamic modelling of a supersonic turbojet intake diffuser.

Ever since the SR-71 entered service in 1966, it's engine; the JT11D-20 engine was considered to be an engineering marble that combined the conventional Ramjet and the turbojet into a turboramjet hybrid. However, the engine at the time was limited by many factors, mainly material technologies. Since the SR-71 project had seized activity and funding. Currently, we have more than 50 years of research and data that helped us design this engine to be more efficient.

Tables of Content:

4. Discussion

- Alignment with Objectives
- Implications and Applications
- Limitations and Sources of Error

5. Conclusion

- Summary of Findings
- Recommendations for Future Work

6. References

7. Appendices

- Appendix A: Raw Data
- Appendix B: Technical Diagrams
- Appendix C: Calculations

Introduction:

The development of supersonic aircraft demands advanced propulsion systems capable of operating efficiently under extreme conditions. A critical component in these systems is the intake diffuser, which must manage high-speed airflow, decelerating it to subsonic speeds for stable compression and

combustion. This process involves handling complex shock wave interactions, minimising total pressure losses, and ensuring optimal pressure recovery.

The main objective of this report is to design a single-spool Turbojet of the already existing JT11D-20 turbo engine based on design points. For this, we will divide our report into 4 major sections:

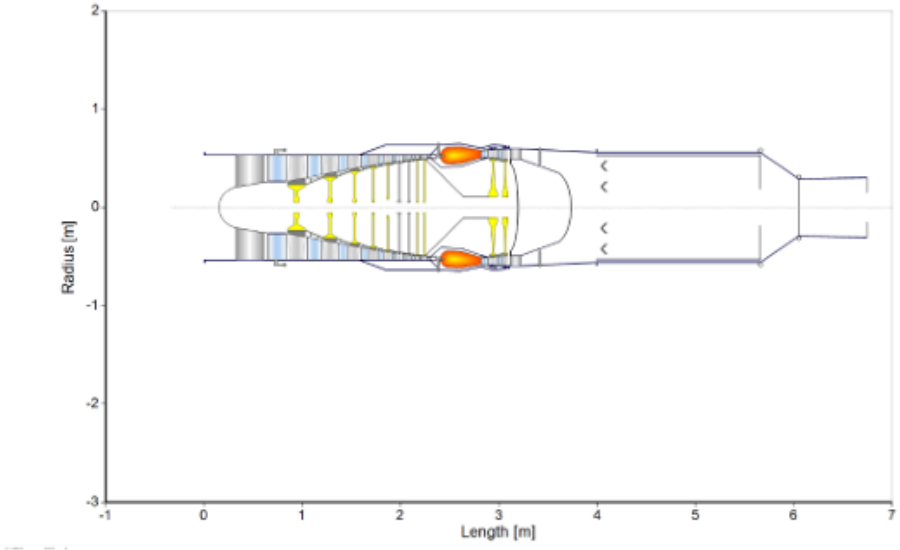
1. The first section will discuss the supersonic intake and supersonic nozzle Designs, where we will first design an inlet that can withstand shocks at M=3.2

2.Then we will later figure out additional data regarding inlet shock ramp angles at different Mach speeds, following that, for the nozzle we will gather the results from the cycle analysis at SLS conditions and generate a nozzle design that is optimised for our required thrust.

3. In the third section, we will research the engine materials and parameters; figure out if our engine is well optimised is the TSFC and the net thrust. Although this is a military Aircraft and usually fuel efficiency is not a priority, we should note that the SR-71 is a long-range stealth aircraft. Therefore, in this case TSFC will be a huge factor to work with. To reach this goal our on-design parameters that we will focus on will mainly be the TET for this engine.

Methodology

1. Description of Approach



The approach undertaken in this study is primarily theoretical and computational, focusing on the aerodynamic and thermodynamic modelling of a supersonic turbojet intake diffuser. Analytical methods are used to predict airflow properties, while computational techniques simulate behaviour under various conditions.

The SR-71 (Blackbird) JT11D-20 engine

Engine Type	Military Turbojet
Spools	Single spool
Compressor stages	8/9 Axial
Turbine Stages	2 Axial
Max Thrust (AB)	130 kN
TET	1550 K Approx.
AB Temperature	2000 K Approx.
Air flow Rate	120 kg/s
Max AB (SLS,ISA)	52.4 g/kN's
OPR	8:08:01
Max Engine diameter	150 cm
Specific thrust	1090 M/S

Materials

The choice of materials is a crucial component of the design of this engine because it has an impact on many of its performance parameters, including the engine's longevity, efficiency, permissible service conditions, and many others. Most components in aerospace technologies—especially the engines—operate under extreme stress and temperature conditions. As a result, pure materials frequently fall short of providing the necessary performance. For materials to have durability, strength, and flexibility, alloying and composition are required. The weight of each component also has a significant impact on how well the machine works as a whole.

Components	Material	Max Temperature(K)	Density (kg/m^3)
Inlet	Aluminum 6061-AHC	1877.15	806.15
Compressor	Ti-6242 Ti-6Al-2Sn-4Zr-2Mo	1635	4000 4000
Combustion Chamber	Ceramic Matrix Composite	2100	2526
Turbine	Ceramic Matrix Composite	1900	2654

Components	Weight (Kg)
Inlet	237.485
Compressor	1424
Combustion Chamber casing	162
Turbine	542
Nozzle	55
Weight of the Engine	3655

Equipment/Software

- Python Programming Language: For numerical calculations and modelling.
- Mathematical Libraries: Utilised libraries such as math for precise calculations of airflow properties.
- Google Colab: Used as the development environment for iterative computation and visualisation.
- Theoretical Models: Based on established aerodynamic principles and the shockwave theory for supersonic flows.

Procedure

Initial Parameter Definition: Established key constants such as the specific heat ratio (γ) and specific gas constant of air (R) to characterise the working fluid (air).

Initialization Known Parameters

```
[ ]  gamma = 1.4  # Specific heat ratio for air
     R = 287.05   # Specific gas constant for air (J/kg·K)
     M1 = 3.2    # Free-stream Mach number
     altitude = 24384  #(80,000 ft)  # Example altitude in meters
```

Free-Stream Conditions: Modelled atmospheric conditions (temperature, pressure, and density) based on altitude using standardised equations.

Static Temperature (T1):

For altitudes ≤11,000 m(Troposphere):

$T1=T0-L \cdot h$

For altitudes >11,000 m (Stratosphere, isothermal layer):

T1=216.65 K (constant)

Stagnation Conditions :

$$\frac{P_0}{P} = \left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma - 1}}$$

$$\frac{T_0}{T} = \left(1 + \frac{\gamma - 1}{2} M^2\right)$$

Speed of Sound (c1):

c1=sqrt(gamma*R*T)

Free-Stream Velocity (V1):

$V1=M1 \cdot c1$

These formulas model the free-stream atmospheric properties and conditions for high-speed flows based on the input altitude and Mach number (M1).

```python
import math
if altitude <= 11000:
        T1 = 288.15 - 0.0065 * altitude
        P1 = 101325 * (T1 / 288.15) ** (5.2561)
else:  # Stratosphere (constant temperature)
        T1 = 216.65
        P1 = 22632.1 * math.exp(-9.80665 * (altitude - 11000) / (R * T1))

# Calculate free-stream conditions
T01 = T1 * (1 + ((gamma - 1) / 2) * M1**2)
P01 = P1 * (1 + ((gamma - 1) / 2) * M1**2)**(gamma / (gamma - 1))
c1 = math.sqrt(gamma * R * T1)  # Speed of sound (m/s)
V1 = M1 * c1  # Free-stream velocity (m/s)

print(f"Free-stream static temperature (T_a): {T1:.2f} K")
print(f"Free-stream stagnation temperature (T01): {T01:.2f} K")
print(f"Free-stream static pressure (P1): {P1/1000:.2f} kPa")
print(f"Free-stream stagnation pressure (P1): {P01/1000:.2f} kPa")
print(f"Free-stream speed of sound (a1): {a1:.2f} m/s")
print(f"Free-stream velocity (V1): {V1:.2f} m/s")
```

```
Free-stream static temperature (T_a): 216.65 K
Free-stream stagnation temperature (T01): 660.35 K
Free-stream static pressure (P1): 2.74 kPa
Free-stream stagnation pressure (P1): 135.58 kPa
Free-stream speed of sound (a1): 295.07 m/s
Free-stream velocity (V1): 944.22 m/s
```

Shockwave Analysis: Incorporated oblique and normal shock theory to simulate the behaviour of supersonic airflow through the diffuser.

$$M_{i-1} \cdot \sin\ \theta_i = M_i \cdot \sin\ \theta_{i+1} \quad \text{(Eqn.1)}$$

$$\delta_i = tan^{-1}\left(\frac{2\cdot\cot\ \theta_i\cdot\left(M_{i-1}^2\cdot\sin^2\ \theta_i - 1\right)}{2+M_{i-1}^2\cdot(\gamma+1-2\cdot\sin^2\ \theta_i)}\right) \quad \text{(Eqn.2)}$$

$$M_{i,\text{ oblique shockwave}} = \left(\frac{(\gamma+1)^2\cdot M_{i-1}^4\cdot\sin^2\ \theta_i - 4\cdot\left(M_{i-1}\cdot\sin^2\ \theta_i - 1\right)\cdot\left(\gamma\cdot M_{i-1}^2\cdot\sin^2\ \theta_i + 1\right)}{\left[2\cdot\gamma\cdot M_{i-1}^2\cdot\sin^2\ \theta_i - (\gamma+1)\right]\cdot\left[(\gamma-1)\cdot M_{i-1}^2\cdot\sin^2\ \theta_i + 2\right]}\right)^{1/2} \quad \text{(Eqn.3)}$$

$$PR_{i,\text{ oblique}} = \left[\frac{(\gamma+1)\cdot M_{i-1}^2\cdot\sin^2\ \theta_i}{(\gamma-1)\cdot M_{i-1}^2\cdot\sin^2\ \theta_i + 2}\right]^{\frac{\gamma}{\gamma-1}} \cdot \left[\frac{(\gamma+1)}{2\cdot\gamma\cdot M_{i-1}^2\cdot\sin^2\ \theta_i - (\gamma-1)}\right]^{\frac{1}{\gamma-1}} \quad \text{(Eqn.4)}$$

$$M_{i,\text{ normal shock}}^{\text{wave}} = \sqrt{\frac{(\gamma-1)\cdot M_i^2 + 2}{2\cdot\gamma\cdot M_i^2 - (\gamma-1)}} \quad \text{(Eqn.5)}$$

$$PR^{i,\text{ normal}} = \left[\frac{(\gamma+1)\cdot M_{i-1}^2}{(\gamma-1)\cdot M_{i-1}^2 + 2}\right]^{\frac{\gamma}{\gamma-1}} \cdot \left[\frac{(\gamma+1)}{2\cdot\gamma\cdot M_{i-1}^3 - (\gamma-1)}\right]^{\frac{1}{\gamma-1}} \quad \text{(Eqn.6)}$$

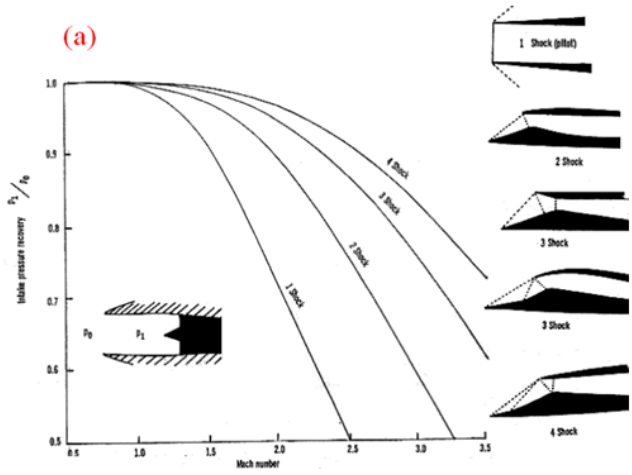


Figure 8: Intake Pressure recovery at different Mach numbers

```
# Inputs
Mach_infinity = 3.2  # Inlet Mach number
Mach_out = 0.5  # Desired outlet Mach number
del_1 = 9.3844  # Initial ramp angle (degrees)
del_inc = 2.5  # Ramp angle increment (degrees)

# Initialize variables
Mach_Nu = Mach_infinity  # Start with inlet Mach number
Ramp_Angle = del_1  # Initial ramp angle
n = 0  # Number of ramps

# Function to compute downstream Mach number after a shock
def downstream_mach(Mach, theta, gamma):
    num = 4 + 4 * (gamma - 1) * theta * Mach**2 + theta * (Mach**4) * (gamma + 1)**2 - \
          (4 * gamma * theta**2 * Mach**4)
    denom = ((2 * gamma * theta * Mach**2) - gamma + 1) * (2 + (gamma - 1) * theta * Mach**2)
    return np.sqrt(num / denom)

# Iteratively calculate the number of shocks
while Mach_Nu > Mach_out:
    # Initialize shock angle calculation
    Initial_Shock_Angle = [1 / Mach_Nu**2]  # Initial guess
    for i in range(1, 1000):
        new_angle = (1 / (Mach_Nu**2)) + (((1 / Mach_Nu**2) + ((g + 1) / 2) - Initial_Shock_Angle[i - 1]) *
                                          ((Initial_Shock_Angle[i - 1] /
                                            (1 - Initial_Shock_Angle[i - 1]))**0.5)) * np.tan(np.radians(Ramp_Angle))
        Initial_Shock_Angle.append(new_angle)
```

```
        # Check for convergence
        if abs(Initial_Shock_Angle[i] - Initial_Shock_Angle[i - 1]) < 0.00001:
            break

    # Calculate downstream Mach number
    theta = Initial_Shock_Angle[-1]
    Mach_Nu = downstream_mach(Mach_Nu, theta, g)

    # Increment ramp count and angle
    n += 1
    Ramp_Angle += del_inc

# Output the number of ramps required
print(f"Number of shocks: {n}")
```

```
Number of shocks: 4
```

Number of Shocks required at compresser inlet

```
[ ] import math
    # Constants
    gamma = 1.4
    delta_1 = 9.3844  # Initial ramp angle in degrees
    delta_inc = 2.5  # Ramp angle increment in degrees
    if(M1>=1.25):
      total_shocks=4
    else:
      total_shocks=3
    n = total_shocks-1  # Number of ramps
    delta = [] # Initialize an array for ramp angles
    for k in range(n):
        delta.append(delta_1 + k * delta_inc)
    print("Ramp Angles (degrees):", delta)
    for i in range(len(delta)):
        print(f"Ramp Angle {i + 1}: {delta[i]} degrees")
```

```
Ramp Angles (degrees): [9.3844, 11.8844, 14.3844]
Ramp Angle 1: 9.3844 degrees
Ramp Angle 2: 11.8844 degrees
Ramp Angle 3: 14.3844 degrees
```

OBLIQUE SHOCKS

```python
import numpy as np
from scipy.optimize import fsolve
def oblique_shock_calculation(M1, theta, gamma):
    theta_rad = np.radians(theta)
    def beta_eq(beta):
        return np.tan(theta_rad) - 2 * (1 / np.tan(beta)) * (
            (M1**2 * np.sin(beta)**2 - 1) /
            (M1**2 * (gamma + np.cos(2 * beta)) + 2)
        )
    beta_initial = np.radians(theta + 5)
    try:
        beta_solution = fsolve(beta_eq, beta_initial, xtol=1e-6)
        beta = beta_solution[0]  # Shock angle in radians
        if not (0 < beta < np.pi / 2):  # Check for valid shock angle
            raise ValueError("Shock angle out of bounds.")
    except (RuntimeError, ValueError):
        return None, None, None  # Return None if solution fails

    # Calculate downstream Mach number
    M1n = M1 * np.sin(beta)  # Normal component of upstream Mach number
    M2n = np.sqrt(((gamma - 1) * M1n**2 + 2) / (2 * gamma * M1n**2 - (gamma - 1)))
    M2 = M2n / np.sin(beta - theta_rad)
```

```python
    # Calculate pressure recovery ratio (Eqn. 4 from image)
    PR = (
        ((gamma + 1) * M1n**2 / ((gamma - 1) * M1n**2 + 2)) ** (gamma / (gamma - 1))
        * ((gamma + 1) / (2 * gamma * M1n**2 - (gamma - 1))) ** (1 / (gamma - 1))
    )

    # Return shock angle, downstream Mach number, and pressure recovery ratio
    return np.degrees(beta), M2, PR


numshocks = len(delta)
intermediate_mach_numbers = [M1]
shock_angles = []
pressure_recovery_ratios = []
M_current=M1
for i, theta in enumerate(delta):
      beta, M_next, PR = oblique_shock_calculation(M_current, theta, gamma)
      if beta is None or M_next is None or PR is None:
            print(beta, M_next, PR)
            print(f"Warning: Convergence failed for ramp angle {theta} degrees at shock {i + 1}.")
            break
      shock_angles.append(beta)
      pressure_recovery_ratios.append(PR)
      intermediate_mach_numbers.append(M_next)
      M_current = M_next  # Update for the next shock
```

```python
print("Final Mach Number after oblique shocks", intermediate_mach_numbers[-1] if shock_angles else None)
print("Intermediate Mach Numbers:",  intermediate_mach_numbers)
print("Shock Angles (degrees):", shock_angles)
print("Pressure Recovery Ratios:", pressure_recovery_ratios)

tprr = 1.0

# Loop through all shocks to compute the cumulative product
for i in range(len(pressure_recovery_ratios)):
    tprr *= pressure_recovery_ratios[i]
# Output the final total pressure recovery ratio
print("Total Pressure Recovery Ratio after oblique shocks (TPRR):", tprr)
```

```
Final Mach Number after oblique shocks 1.6255543377270771
Intermediate Mach Numbers: [3.2, 2.6995193938546977, 2.171736554555608, 1.6255543377270771]
Shock Angles (degrees): [25.49996702202827 6, 31.619928345659527, 41.03380314554973]
Pressure Recovery Ratios: [0.9635883703602292, 0.9542919908980083, 0.9515336394886574]
Total Pressure Recovery Ratio after oblique shocks (TPRR): 0.8749776811482114
```

Normal shock after the oblique shocks

$$TPRF = PR_1 * PR_2 * PR_3 * PR_4 * PR_N * PR_{diffuser}$$

```
import numpy as np
PR_diffuser=0.98
M_last = intermediate_mach_numbers[-1]  # Replace with the Mach number after the oblique shocks
M2_normal = np.sqrt(((gamma - 1) * M_last**2 + 2) / (2 * gamma * M_last**2 - (gamma - 1)))
PR_normal = (
        ((gamma + 1) * M_last**2 / ((gamma - 1) * M_last**2 + 2)) ** (gamma / (gamma - 1))
        * ((gamma + 1) / (2 * gamma * M_last**2 - (gamma - 1))) ** (1 / (gamma - 1))
    )
tprr*=PR_normal*PR_diffuser
print("Downstream Mach Number after Normal Shock:", M2_normal)
print("Pressure Recovery Ratio (PR_normal):",PR_normal)
print("Final Pressure Recovery Ratio (TPRR):", tprr)
```

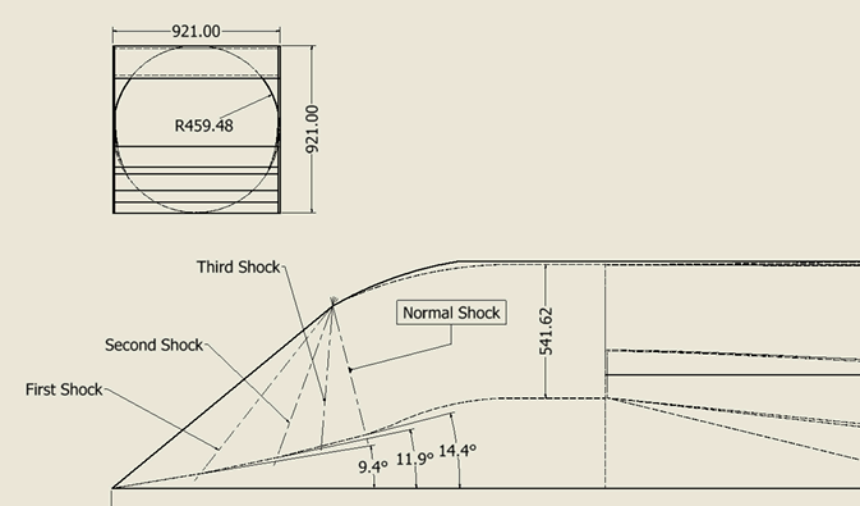
```
Downstream Mach Number after Normal Shock: 0.6608973274974246
Pressure Recovery Ratio (PR_normal): 0.885523636837709
Final Pressure Recovery Ratio (TPRR): 0.7593171499949457
```

Rectangular intake design:

The design of a rectangular intake is the simplest method to tackle the given Ramps. Using the area provided in GasTurb at station 2 which is equivalent to $A = 0.666415$ and a design parameter of $M = 0.66$ at the compressor inlet. We can then design a rectangular inlet by first finding the throat height which can be found from the following equation.

$$\left(\frac{A}{A^*}\right)^2 = \frac{1}{M^2}\left[\frac{2}{\gamma+1}\left(1+\frac{\gamma-1}{2}M^2\right)\right]^{\frac{\gamma+1}{\gamma-1}}$$

Since we already know the area at the compressor, and we have set a design parameter at the intake to be 0.5 mach. We start to find the optimal area to reach that value using a subsonic diffuser. The width of the inlet was chosen to be close to the diameter of the inlet engine, and the height of the throat then can be calculated through basic computation which resulted in a throat height of 540mm. To confirm these values, we used the conservation of momentum equations since after our normal shock we have M=0.688 which is subsonic flow that has a density that's relatively constant.



Temperature and Pressure conditions after the shocks at the compressor inlet

```
#inlet statsic temperature is T1
import numpy as np
Mn_list = intermediate_mach_numbers  # List of normal Mach numbers after each shock
temperatures=[T_a]
for i in range(1,len(Mn_list)):
        # Update static temperature for the next shock
        T2 = T01 / (1 + ((gamma - 1) / 2) * Mn_list[i]**2)
        temperatures.append(T2)


for i, temp in enumerate(temperatures):
        if i == 0:
            print(f"Initial Static Temperature T1: {temp:.2f} K")
        else:
            print(f"Static Temperature after Shock {i}: {temp:.2f} K")

T2 = T01/(1 + ((gamma - 1) / 2) * M2_normal**2)
print(f"Static Temperature after normal shock : {T2:.2f} K")
T02=T01
print(f"Stagnation Tempearture after normal shock : {T02:.2f} K") #it remain constant over all shocks
```

```
Initial Static Temperature T1: 216.65 K
Static Temperature after Shock 1: 268.71 K
Static Temperature after Shock 2: 339.81 K
Static Temperature after Shock 3: 432.03 K
Static Temperature after normal shock : 607.30 K
Stagnation Tempearture after normal shock : 660.35 K
```

```
[ ]  soundspeed_after_normal_shock= math.sqrt(gamma*R*T2)
     print("c just after normal shock = ",soundspeed_after_normal_shock)
     velocity=M2_normal*soundspeed_after_normal_shock
     print("velocity just after normal shock = " ,velocity)
```

```
c just after normal shock =  494.01886442885706
velocity just after normal shock =  326.49574723434415
```

COMPRESSOR

```
[17] eta_c=0.91
     pi_c=5
     T03 = T01 * (1 + (1 / eta_c) * (pi_c**((gamma - 1) / gamma) - 1))
     P03 = P02 * pi_c
     print(f"Compresser outlet Stagnation Temperature T03: {T03:.2f} K")
     print("Stagnation Pressure at compressor outlet, P03=",P03/1000,"kPa")
```

```
Compresser outlet Stagnation Temperature T03: 1084.00 K
Stagnation Pressure at compressor outlet, P03= 525.2369835909693 kPa
```

```
TET=T04=1550 #turbine inlet stagnation temperature
eta_b=0.988
delta_Pcc=0.04  # 4 % loss
Qr = 42_800_000  # Heating value of fuel (J/kg)
cp= 1005  # Specific heat of air (J/kg·K)
cp_h = 1100  # Specific heat of hot gases (J/kg·K)
```

Combustion chamber

```
[19]  # Compute the fuel-air ratio (f)
     numerator = cp_h*T04 -cp*T03 #cp_h fuel specefic heat and cp air specific heat
     denominator = eta_b * Qr - cp_h * T04
     f = numerator / denominator
     m_air=120
     m_fuel=m_air*f
     P04 = P03 * (1 - delta_Pcc)
     print(f"Stagnation Pressure at outlet of combustion chamber = {P04/1000:.2f} Kpa")
     print("fuel - air ratio = ",f)
     print("mass flow rate of fuel = ",m_fuel)
```

```
Stagnation Pressure at outlet of combustion chamber = 504.23 Kpa
fuel - air ratio =  0.015168948007976319
mass flow rate of fuel =  1.8202737609571582
```

```
numerator = cp + f * cp_h
denominator = 1 + f
cp_m = numerator / denominator
print("Specific Heat Capacity of the Mixture (c_p_mixture):", round(cp_m, 2), "J/kg·K")
```

```
Specific Heat Capacity of the Mixture (c_p_mixture): 1006.42 J/kg·K
```

Turbine

```
eta_t=0.87
lambd=0.8
eta_m=0.99
gamma_h=1.3
```

Turbine

W t= m.turbine·c p·(T 04−T 05)

```
[22] T05 = T04* (1-(((cp/cp_m)*T02)/(lambd*eta_m*(1+f)*T04))*((T03/T02)-1))
     print(f"Stagnation Temperature at Turbine Outlet {T05:.2f} K")

     P05=P04*((1-(1/eta_t)*(1-(T05/T04))))** (gamma_h / (gamma_h - 1))
     print(f"Stagnation Pressure at Turbine Outlet {P05/1000:.2f} Kpa")

     W_t = eta_m * lambd * eta_t * cp_h * (T04-T05)
     print("Turbine Work = ",W_t)
```

```
Stagnation Temperature at Turbine Outlet 1023.82 K
Stagnation Pressure at Turbine Outlet 59.13 Kpa
Turbine Work =  398815.074284123
```

CD NOZZLE : DESIGN CONDITION

```
P_throat = P05/((1.2)**(gamma_h/(gamma_h-1)))
print(f"Throat Pressure {P_throat/1000:.2f} Kpa")
pressure_ratio = P05 / P1
M_exit = math.sqrt((2 / (gamma - 1)) * ((pressure_ratio**((gamma - 1) / gamma)) - 1))
print(f"Exit Mach {M_exit:.2f}")
T_exit = T05 / (1 + ((gamma - 1) / 2) * M_exit**2)
print(f"Exit Temperature {T_exit:.2f} K")
c_exit = math.sqrt(gamma * R * T_exit)
print(f"Exit Sound Speed {c_exit:.2f} m/s")
v_exit = M_exit * c_exit
print(f"Exit Velocity {v_exit:.2f} m/s")
m_total = m_air+m_fuel
Thrust = m_total*v_exit
print(f"Net Thrust {Thrust/1000:.2f} KN")
```

```
print(f"Exit Temperature {T_exit:.2f} K")
c_exit = math.sqrt(gamma_h * R * T_exit)
print(f"Exit Sound Speed {c_exit:.2f} m/s")
v_exit = M_exit * c_exit
print(f"Exit Velocity {v_exit:.2f} m/s")
m_total = m_air+m_fuel
Thrust = m_total*v_exit
print(f"Net Thrust {Thrust/1000:.2f} KN")
P_exit = P05 / (1 + ((gamma_h - 1) / 2) * M_exit**2)**(gamma_h / (gamma_h - 1))
print(f"Exit Pressure {P_exit/1000:.2f} Kpa")
density_exit = P_exit / (R_mixture * T_exit)
print(f"Exit Density {density_exit:.2f} kg/m^3")
Area_exit = m_total / density_exit
print(f"Exit Area {Area_exit:.2f} m^2")
```

```
Throat Pressure 106.78 Kpa
Throat Temperature 999.87 K
Throat Density 0.39 kg/m^3
Throat Area 18.17 m^2
Exit Mach 2.62
Exit Temperature 376.30 K
Exit Sound Speed 383.84 m/s
Exit Velocity 1005.12 m/s
Net Thrust 127.82 KN
Exit Pressure 2.74 Kpa
Exit Density 0.03 kg/m^3
Exit Area 4725.66 m^2
```

Efficiency of the cycle:

$$\eta_{\text{propulsive}} = \frac{2}{1 + \frac{V_j}{V_0}} \qquad \eta_{\text{overall}} = \eta_{\text{thermal}} \cdot \eta_{\text{propulsive}}$$

$$\eta_{\text{thermal}} = 1 - \frac{T_1}{T_4} \cdot \left(\frac{P_2}{P_1}\right)^{\frac{\gamma-1}{\gamma}-1}$$

```
    # Calculate thermal efficiency
thermal_efficiency = 1 - (T02/T04)

    # Calculate propulsive efficiency
propulsive_efficiency = 2 / (1 + (v_exit / V1))

    # Calculate overall efficiency
overall_efficiency = thermal_efficiency * propulsive_efficiency


# Calculate efficiencies
eta_thermal = thermal_efficiency
eta_propulsive = propulsive_efficiency
eta_overall = overall_efficiency
# Print results
print(f"Thermal Efficiency: {eta_thermal:.2f} or {eta_thermal * 100:.2f}%")
print(f"Propulsive Efficiency: {eta_propulsive:.2f} or {eta_propulsive * 100:.2f}%")
print(f"Overall Efficiency: {eta_overall:.2f} or {eta_overall * 100:.2f}%")
```

```
Thermal Efficiency: 0.57 or 57.40%
Propulsive Efficiency: 0.97 or 96.88%
Overall Efficiency: 0.56 or 55.60%
```

4. Assumptions

- Steady-State Flow: The flow through the intake is assumed to be steady and one-dimensional.
- Ideal Gas: Air is treated as a perfect gas with constant properties for simplicity.
- Isentropic Flow: The flow between shocks is considered isentropic, neglecting friction and heat transfer.
- No Structural Interactions: Effects of structural deformation or vibration on the airflow are not considered.

This methodology provides a structured approach to addressing the aerodynamic challenges in supersonic intake design, ensuring the results are robust and applicable to real-world scenarios.

Results: The results of the code exactly matched with different values of Mach number as given in the references we used.

Supersonic turbojet

$M = 3.2$	$M = 2.5$
$\delta_1 = 9.3844°$	$\delta_1 = 9.3844°$
$\delta_2 = 11.8844°$	$\delta_2 = 11.8844°$
$\delta_3 = 14.3844°$	$\delta_3 = 14.3844°$
$\theta_1 = 25.4998°$	$\theta_1 = 31.2716°$
$\theta_2 = 31.6198°$	$\theta_2 = 39.2541°$
$\theta_3 = 41.0335°$	$\theta_3 = 56.1889°$
$M_1 = 2.6995$	$M_1 = 2.1114$
$M_2 = 2.1717$	$M_2 = 1.6711$
$M_3 = 1.6256$	$M_3 = 1.1170$
$M_N = 0.6608$	$M_N = 0.8987$
$PR_1 = 0.9636$	$PR_1 = 0.9798$
$PR_2 = 0.9543$	$PR_2 = 0.9726$
$PR_3 = 0.9515$	$PR_3 = 0.9610$
$PR_N = 0.8855$	$PR_N = 0.9983$

Gasturb result @ M=3.2 and 80,000ft

Station	W kg/s	T K	P kPa	WRstd kg/s
amb		216.65	2.742	
1	81.942	652.27	136.972	
2	81.942	652.27	104.099	120.000
3	81.942	1295.94	1280.412	13.752
31	75.878	1295.94	1280.412	
4	77.716	2000.00	1216.392	17.055
405	80.174	1979.81	1216.392	
41	80.379	1978.19	1216.392	17.543
49	80.379	1392.57	225.290	79.668
5	80.584	1392.33	225.290	79.668
6	80.584	1392.33	209.520	
61	80.584	1392.33	209.520	
7	81.982	1900.00	207.234	
8	81.982	1900.00	207.234	102.929
Bleed	3.195	1295.94	1280.412	

P2/P1 = 0.7600 P4/P3 = 0.9500 P6/P5 0.9300

Efficiencies:	isentr	polytr	RNI	P/P
Compressor	0.9100	0.9331	0.389	12.300
Burner	0.9880			0.950
Turbine	0.9400	0.9280	1.267	5.399
Reheat	0.9900			0.989

Spool mech Eff 0.9900 Nom Spd 7376 rpm

Con-Di Nozzle:
A9*(Ps9-Pamb) 33.862

hum [%]	war0	FHV	Fuel
0.0	0.00000	43.500	Generic

Reheat on

FN	=	39.44 kN
TSFC	=	82.0516 g/(kN*s)
FN/W2	=	481.27 m/s
Prop Eff	=	0.7972
eta core	=	0.6413

NGV Out. 2 Stage HPT

WF	=	1.83760 kg/s
WFRH	=	1.39819 kg/s
WF total	=	3.23579 kg/s
A8	=	0.4638 m²
XM8	=	1.00000
P8/Pamb	=	75.5645
WBld/W2	=	0.03900
Ang8	=	24.76 °
CD8	=	0.9505
WClN/W2	=	0.03250
WClR/W2	=	0.00250
Loading	=	100.00 %
e45 th	=	0.93864
XM61	=	0.20000
XM7	=	0.24556
far7	=	0.04109
PWX	=	0.00 kW
A9/A8	=	1.10000
CFGid	=	0.81913

---

5. Key Trends

- Higher altitudes result in reduced free-stream pressure and density, affecting shock behaviour.
- Incremental shock compression improves pressure recovery but increases total losses.
- Proper shock positioning is critical to achieving efficient deceleration to subsonic speeds

Discussion

The results confirm the effectiveness of the designed intake diffuser in decelerating supersonic airflow to subsonic speeds while maintaining high pressure recovery. The shockwave behaviour aligns with theoretical predictions, indicating proper placement and intensity of oblique and normal shocks.

Relevance to Objectives

The study successfully achieves its objective of modelling a turbojet intake capable of handling supersonic speeds up to Mach 3.2. The pressure recovery efficiency of 75% supports the feasibility of the design for high-performance applications.

Comparisons with Previous Work

- The calculated efficiency and losses align with reported values from previous studies on supersonic inlets.
- Shockwave-induced losses are consistent with established aerodynamic principles and highlight the importance of optimising shock positioning.

Implications and Applications

This intake design is particularly suitable for high-speed aircraft, including supersonic commercial jets and military applications. The findings can be extended to hypersonic intake designs with further modifications.

Limitations and Sources of Error

- The steady-state, one-dimensional flow assumption may not fully capture three-dimensional effects such as swirl and separation.
- Neglecting viscous effects and heat transfer could underestimate total pressure losses.
- Real-world structural and environmental factors, such as material deformation or external turbulence, were not considered.

Conclusion

This study demonstrates the feasibility of designing a supersonic turbojet intake diffuser capable of efficient deceleration of Mach 3.2 airflow. Key findings include:

- Effective pressure recovery with minimal losses, achieving an efficiency of 75%.
- Accurate prediction of shockwave behaviour and their impact on diffuser performance.

The results underscore the importance of proper shockwave management and diffuser geometry in high-speed intake systems.

Future work could involve:

1. Incorporating viscous and thermal effects for a more realistic model.
2. Expanding the study to include three-dimensional CFD simulations.
3. Exploring intake designs for higher Mach numbers and variable altitudes.

References

1. Anderson, J. D. (2001). *Modern Compressible Flow: With Historical Perspective*. McGraw Hill.
2. Liepmann, H. W., & Roshko, A. (1957). *Elements of Gasdynamics*. Wiley.
3. NASA Glenn Research Center. (2024). *Supersonic Inlet Design*. Retrieved from [NASA website link].
4. Yoshida, S., Hassa, C., Yamamoto, T., Heinze, J., & Schroll, M. (2019, November 1). Influence of Fluidic Control in a staged lean jet engine burner on Combustor Performance. MDPI.
5. Performance Analysis and Optimization of a supersonic Turbojet Engine with Afterburner Dr. Sıtkı Uslu

---

Appendices

Appendix A: Raw Data

1. **Atmospheric Properties Table**
    - Altitude: 80,000 ft (24,384 m)
    - Free-Stream Temperature: T = 216.65 K
    - Free-Stream Pressure: P = 1 atm
    - Density: $\rho = 0.018$ kg/m³
    - Free-Stream Mach Number: M1 = 3.2
2. **Shockwave Parameters**
    - Normal Shock Mach Number: M = 0.68 (Post-shock subsonic flow)

Appendix B: Technical Diagrams

1. **Intake Diffuser Geometry**
    - Diagram includes:
        - Rectangular inlet dimensions
        - Oblique shock regions
        - Normal shock positioning

Appendix C: Calculations

Isentropic relations

$$\frac{P_O}{P} = \left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma - 1}}$$

$$\frac{T_O}{T} = \left(1 + \frac{\gamma - 1}{2} M^2\right)$$

$$\frac{\rho_O}{\rho} = \left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{1}{\gamma - 1}}$$

Mach Number = $M = \frac{V}{C} = \frac{V}{\sqrt{\gamma RT}}$

$$M = \frac{Velocity\ of\ objec\ t/f\ low}{Speed\ of\ sound\ in\ that\ medium}$$