

JetCAP: Jet Engine Cycle Analysis and Performance tool

Abhijit Venkat P, Aditya Prasad, Gauresh Rangaiyan

SRIP 2024, IIT Gandhinagar

Mentors: Prof. Dilip S. Sundaram, Mr. Ganeshkumar V

Contents

1	Introduction	I
2	Parametric Cycle Analysis (PCA)	I
2.1	What is PCA?	I
2.2	Our Methodology for PCA	I
	<i>atmosphere.py • fuel.py • aircraft.py • compressor.py • nozzle.py • combustor.py • turbine.py • mixer.py • main.py • turbofan.py • turbojet.py • ramjet.py • formulaSheet.py • initialisingComponents.py • set_defaultvalues() • plots.py:</i>	
2.3	What is JetCAP capable of doing?	IV
2.4	JetCAP PCA validation with PARA	V
	<i>Turbojet (Dual-spool) - Ideal Cycle • Turbojet (Dual-spool) - Real Cycle • Turbofan (Dual-spool) - Real Cycle • Turbofan (Dual-spool mixed exhaust) - Real Cycle • Turbofan (Dual-spool with Afterburner) - Real Cycle</i>	
3	Engine Performance Analysis (EPA)	VI
3.1	What is EPA?	VI
3.2	Our methodology for EPA	VI
4	A Final Discussion	VII
5	Acknowledgements	VII
	References	VII

1. Introduction

In the pursuit of advancing aerospace engineering tools, we have developed a sophisticated software program designed to calculate engine performance parameters for aircraft under various flight conditions. This project is inspired by existing tools such as PARA and PERF by Jack D. Mattingly and Gasturb. However, recognising the limitations of PARA and PERF in terms of accuracy, reliability, and the proprietary nature of Gasturb, we aimed to create an improved and accessible solution.

JetCAP offers a comprehensive analysis of different engine cycles, including turbojet, turbofan, and ramjet, with further customisation options such as dual spool configurations and the inclusion of afterburners for turbojet and turbofan engines.

The program calculates key performance metrics, such as specific thrust (ST), thrust-specific fuel consumption (TSFC), fuel-to-air ratio (f), and the thermal and propulsive efficiencies of the engine cycle, among other parameters. The software can also compute the off-design performance.

This tool enhances the precision of engine performance evaluation and is a valuable resource for aerospace research and education.

2. Parametric Cycle Analysis (PCA)

2.1. What is PCA?

Parametric Cycle Analysis determines the performance of an engine for a given set of design choice parameters, such as compressor pressure ratio (π_c), performance efficiencies (e.g. Turbine efficiency), and the physical limits of the system (e.g. Maximum temperature at turbine inlet (T_{t4})).

Using this information and the flight conditions such as Mach number (M_0) and altitude, we can determine important performance parameters such as Specific Thrust (ST), Thrust Specific Fuel Consumption (TSFC), and so on.

2.2. Our Methodology for PCA

We used Python to create the program and various libraries for plotting and GUI. Since all the engine types have similar components, we decided to code every component separately and join the components in such a way that would help us create ramjet, turbojet, and turbofan engines.

We also created options for turbojet and turbofan engines, which could be in single-spool or dual-spool configurations. There is also an option to select afterburner.

In Python, we made classes for each component with appropriate and necessary functions and methods within them. In this section, we shall explain each and every class/file and their role and functions in the overall code.

2.2.1. *atmosphere.py*

The *atmosphere.py* file contains the *atmosphere* class, which stores the atmospheric properties that affect the performance of the turbofan engine. The class is initialised with T_0 (ambient temperature) and P_0 (ambient pressure). These properties are crucial for determining the engine's performance at different altitudes and conditions.

2.2.2. *fuel.py*

The *fuel.py* file contains the *fuel* class, which stores the chemical properties of fuel and air necessary for calculating the engine's performance. The class is initialised with the following parameters:

- γ_c : Specific heat ratio of cold air
- γ_t : Specific heat ratio of hot air
- R_c : Gas constant for cold air
- R_t : Gas constant for hot air
- C_{pc} : Specific heat at constant pressure for cold air
- C_{pt} : Specific heat at constant pressure for hot air
- H_{pr} : The heating value of the fuel

These properties are used in various calculations throughout the engine simulation.

2.2.3. *aircraft.py*

The *aircraft.py* file contains the *aircraft* class, which combines the *Atmosphere* and *Fuel* classes and adds additional properties specific to the aircraft and engine:

- M_0 : Mach number
- α : Bypass ratio of the engine

The class also includes methods to calculate the temperature ratio (τ_r) and pressure ratio (π_r) using the following formulae:

$$\tau_r = 1 + \frac{\gamma_c - 1}{2} M_0^2$$

$$\pi_r = \tau_r^{\frac{\gamma_c}{\gamma_c - 1}}$$

These ratios are essential for determining the changes in temperature and pressure as air moves through the engine.

2.2.4. *compressor.py*

The *compressor.py* file contains the *compressor* class, which can be instantiated to create both the fan and the compressor components of the engine. It is initialized with:

- π_c : Pressure ratio
- η_c : Polytropic efficiency
- *aircraft*: An instance of the *Aircraft* class

The class includes methods to calculate the temperature ratio across the compressor (τ_c) and the total properties (PnT) of the flow exiting the component. (Here, PnT stands for the total Pressure & total temperature, respectively). The formulas used are:

$$\tau_c = \pi_c^{\frac{\gamma_c - 1}{\gamma_c \eta_c}}$$

$$P_{t, out} = \pi_c \cdot P_{t, in}$$

$$T_{t, out} = \tau_c \cdot T_{t, in}$$

2.2.5. nozzle.py

The `nozzle.py` file contains the `nozzle` class, which models the nozzle components of the engine, including the diffuser, core nozzle, and fan nozzle. It is initialized with the pressure ratio (π_n) and includes a method to calculate the total properties (PnT) of the gas exiting the nozzle:

$$P_{t, out} = \pi_n \cdot P_{t, in}$$

$$T_{t, out} = \tau_n \cdot T_{t, in}$$

(where τ_n is taken to be 1, assuming adiabatic walls). Since the diffuser and nozzle are just inverted versions of one-another and even follow the same governing equations, the diffuser component is made out of the nozzle class itself, as described above. The pressure ratio across the diffuser is π_d . Similarly, for the fan nozzle, the pressure ratio across it is π_{fn} .

2.2.6. combustor.py

The `combustor.py` file contains the `combustor` class, which models the combustion chamber. It is initialized with:

- π_b : Pressure ratio
- η_b : Burner efficiency
- T_{t4} : Burner total temperature
- `aircraft`: An instance of the `aircraft` class

The class includes methods to calculate the temperature ratio (τ_λ), the total properties (PnT) of the exiting gas, and the fuel-air ratio (f). The formulas used are:

$$\tau_\lambda = \frac{C_{pt} \cdot T_{t4}}{C_{pc} \cdot T_0}$$

$$P_{t, out} = \pi_b \cdot P_{t, in}$$

$$T_{t, out} = T_{t4}$$

$$f_0 = \frac{C_{pt} \cdot T_{t4} - C_{pc} \cdot T_{t3}}{\eta_b \cdot H_{pr} - C_{pt} \cdot T_{t4}} \frac{1}{1 + \alpha}$$

$$f_0 = \frac{h_{t4} - h_{t3}}{\eta_b \cdot H_{pr} - h_{t3}} \frac{1}{1 + \alpha}$$

Iterative Approach to Calculate f :

An iterative approach is used to calculate f , as outlined in the book by Jack Mattingly, using experimental data for gas properties at different temperatures and fuel-air ratios. However, we have applied our own approach since the approach in the book is unclear. The data is available in Appendix L of the supplementary reading material.

There are only 5 columns corresponding to varying values of f . These values are $f = 0$, $f = 0.0169$, $f = 0.0338$, $f = 0.0507$, $f = 0.0676$.

The gas properties (C_{pt} and C_{pc}) vary with temperature and depend on the fuel-air ratio. The values $C_{pt} \cdot T_{t4}$ and $C_{pc} \cdot T_{t3}$ represent the specific enthalpy at those temperatures. For T_{t3} , since fuel has not been added yet, we use the column where $f = 0$.

For T_{t4} , we start with an initial guess for the fuel-air ratio to find the corresponding enthalpy. Using this enthalpy value, we calculate a new f_0 value using the above equation, which lies between two columns f_L and f_R . We express f_0 as a linear combination of f_L and f_R :

$$f_0 = \beta f_L + (1 - \beta) f_R$$

From this equation, we determine β :

$$\beta = \frac{f_0 - f_R}{f_L - f_R}$$

Using the same weight β , we interpolate the enthalpy values h_L and h_R corresponding to f_L , f_R , and T_{t4} :

$$(h_{new})_{t4} = \beta h_L + (1 - \beta) h_R$$

We substitute this interpolated enthalpy back into the equation for f_0 and iterate until convergence is achieved. This iterative process has been verified against the book, yielding results that match well.

2.2.7. turbine.py

The `turbine.py` file contains the `turbine` class, which models the turbine component of the engine. It is initialized with:

- η_m : Mechanical efficiency
- π_t : Polytropic efficiency
- `aircraft`: An instance of the `aircraft` class
- `fan`: An instance of the `compressor` class representing the fan
- `LowCompressor`: An instance of the `compressor` class representing the Low-pressure compressor
- `HighCompressor`: An instance of the `compressor` class representing the High-pressure compressor
- `combustor`: An instance of the `combustor` class
- `Type`: "LPT"/"HPT". This refers to the low and high-pressure turbines, respectively.

The class includes methods to calculate the temperature ratio (τ_t), pressure ratio (π_t), and the total properties (PnT) of the exiting gas. The formulae used are complex and involve energy conservation between the turbine and other components.

The temperature ratio (τ_t) depends on the type of turbine. Different formulas are used to find τ_t .

For LPT: The Low-pressure turbine (LPT) is calculated by equating the work output of the Low-pressure turbine with the work input of the fan and the low-pressure compressor:

$$\tau_{LPT} = 1 - \frac{\tau_\lambda \cdot (\gamma - 1) + \alpha \cdot (f - 1)}{\eta_m \cdot (1 + f \cdot (1 + \alpha))}$$

For HPT: The High-pressure turbine (HPT) is calculated by equating the work output of the High-pressure turbine with the work input of the High-pressure compressor:

$$\tau_{HPT} = 1 - \frac{\pi_{LPC} \cdot (\pi_{HPC} - 1)}{\eta_m \cdot \pi_\lambda}$$

$$\pi_t = \tau_t \cdot (\pi_t - 1)$$

$$P_{t, out} = \pi_t \cdot P_{t, in}$$

$$T_{t, out} = \tau_t \cdot T_{t, in}$$

2.2.8. mixer.py

The `mixer` class is used only for Turbofan engines, where the user can choose either mixed flow exhaust or separate flow exhaust. This class encapsulates the detailed calculations needed to determine the temperature and pressure changes across a mixed exhaust turbofan engine's mixer, utilizing input parameters and various thermodynamic relationships.

1. **Initialization:** The mixer class is initialized with various engine components and parameters such as the aircraft, fan, diffuser, compressors, combustor, and turbines.
2. **Calculation of Temperature Ratio (`get_tau_m` method):** This method calculates the temperature ratio (τ_m) based on the bypass ratio (α), mixer pressure ratio (π_m), and several thermodynamic properties and efficiencies. It involves calculating intermediate parameters like $\frac{\dot{m}_{f,CC}}{\dot{m}_c}$, α' , C_{p6A} , R_{6A} , γ_{6A} , and $\frac{T_{t16}}{T_{t6}}$.

3. **Phi Calculation (phi method):** This utility method computes a specific function of Mach number (M) and specific heat ratio (γ).
4. **Mass Flow Parameter Calculation (MFP method):** Another utility method that computes the Mass Flow Parameter for a given specific heat ratio (γ), gas constant (R), and Mach number (M).
5. **Calculation of Mixer Pressure Ratio (get_pi_m method):** This method calculates the mixer pressure ratio (π_m). If a maximum pressure ratio ($\pi_{m,max}$) is specified, it performs detailed calculations involving $\frac{P_{t16}}{P_{t6}}$, M_{16} , ϕ_6 , ϕ_{16} , τ_m , M_{6A} , and $\frac{A_{16}}{A_6}$ to find an ideal pressure ratio and scales it by $\pi_{m,max}$. Otherwise, it defaults π_m to 1 for ideal mixed exhaust turbofans.
6. **Output Pressure and Temperature (get_PnT method):** This method calculates the output total pressure and temperature ($P_{t,out}$, $T_{t,out}$) of the mixer based on input conditions and the previously calculated π_m and τ_m .

2.2.9. main.py

The `main.py` file is the entry point of the simulation. It instantiates all the necessary classes and creates a turbofan object. It then calls the `get_engineDetails` method to print out various engine performance parameters.

2.2.10. turbofan.py

The `turbofan.py` file contains the turbofan class, which combines all the engine components and calculates the engine's performance characteristics. It uses a `formulaSheet` class to calculate various engine performance formulas.

2.2.11. turbojet.py

The `turbojet.py` file contains the turbojet class, which consists of all the engine components and calculates the engine's performance characteristics. Like the `turbofan.py` file, it uses `formulaSheet.py` to calculate all the necessary performance parameters.

2.2.12. ramjet.py

This file is similar to the turbojet and turbofan classes. It combines all the engine components, including the components such as compressor and turbine, (which are not in a ramjet engine though) and are initialised appropriately so that their presence does not make a difference to the ramjet calculations.

2.2.13. formulaSheet.py

The `formulaSheet.py` file is a crucial component of the turbofan engine simulation, containing a class named `formulaSheet` that encapsulates various performance calculations for the engine. This class is initialised with an instance of the engine object, which provides access to all the necessary engine parameters and component properties.

The `formulaSheet` class includes several methods that perform key calculations:

- `get_PnT_from_PtTtM(Pt, Tt, M, gam)`: This method calculates the static pressure (P) and temperature (T) from the total pressure (P_t), total temperature (T_t), Mach number (M), and specific heat ratio (γ). It uses the isentropic flow relations to determine the static conditions from the total conditions. The formulae used are:

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

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

- `get_a(gam, R, T)`: This method calculates the speed of sound (a) using the specific heat ratio (γ), gas constant (R), and temperature (T). The speed of sound is a fundamental property used in various aerodynamic calculations.

$$a = \sqrt{\gamma R T}$$

- `get_v(gam, R, T, M)`: This method computes the velocity (v) of the flow using the Mach number (M), specific heat ratio (γ), gas constant (R), and temperature (T). The velocity is directly proportional to the speed of sound (a), which is calculated from the above method.

$$v = M \cdot \text{get_a}(\gamma, R, T)$$

- `get_Tc_Mc()`: This method computes the Specific Thrust for the core flow of the turbofan engine. It uses the properties of the core nozzle exit and the ambient conditions to calculate the specific thrust generated by the core flow.

$$\frac{T_c}{m_c} = a_0(1+f(1+\alpha)) \left(M_9 \sqrt{\frac{T_9}{T_0}} - \frac{M_0}{(1+f(1+\alpha))} + \sqrt{\frac{T_9}{T_0}} \frac{1}{\gamma_9 M_9} \left(1 - \frac{P_0}{P_9} \right) \right)$$

- `get_Tb_Mb()`: This method computes the Specific Thrust for the bypass flow of the turbofan engine. It uses the properties of the bypass nozzle exit and the ambient conditions to calculate the thrust generated by the bypass flow.

$$\frac{T_b}{m_b} = a_0 \left(M_{19} \sqrt{\frac{T_{19}}{T_0}} - M_0 \right) + \frac{a_{19} \left(1 - \frac{P_0}{P_{19}} \right)}{\gamma_{19} M_{19}}$$

- `get_ST()`: This method calculates the turbofan engine's overall thrust (ST) by combining the thrust from the core and bypass flows. It uses the bypass ratio (α) to weigh the contribution of each flow to the overall specific thrust.

$$ST = \frac{\alpha}{1+\alpha} \cdot \text{get_Tb_Mb}() + \frac{1}{1+\alpha} \cdot \text{get_Tc_Mc}()$$

- `get_TSFC()`: This method calculates the engine's thrust-specific fuel consumption (TSFC). It divides the fuel-air ratio by the overall thrust to determine the engine's efficiency in fuel consumption per unit thrust.

$$TSFC = \frac{\text{get_f}()}{\text{get_ST}()}$$

Here, the overall fuel-air ratio (f) is called from the `get_f()` function, which is defined inside the combustor class object.

The `formulaSheet` class is a central repository for the engine's performance calculations, ensuring that all the necessary formulas are encapsulated and easily accessible for the overall engine simulation.

2.2.14. initialisingComponents.py

This class is designed to initialise and set up the components of an aircraft engine based on the given parameters.

- **__init__ Method:** This is the constructor for the class. It initialises various engine parameters to set up and operate different engine components. The parameters include:
 - `engineType`: Type of the engine (e.g., turbofan, turbojet, ramjet)
 - `AB`: Afterburner flag (Y/N)
 - `T0, P0`: Ambient temperature and pressure
 - `M0`: Mach number
 - `α`: Bypass ratio
 - `γc, γt, γab`: Specific heat ratios for compressor, turbine, and afterburner

- R_c, R_t : Gas constants for compressor and turbine
- C_{pc}, C_{pt}, C_{pab} : Specific heat capacities for compressor, turbine, and afterburner
- T_{t4}, T_{t7} : Turbine inlet temperatures
- H_{pr} : Heat of fuel
- $\eta_b, \eta_m, \eta_{ab}$: Efficiencies for the combustor, mechanical, and afterburner
- $\pi_{d,max}, \pi_f, \pi_c, \pi_{cl}, \pi_{ch}$: Pressure ratios for the diffuser, fan, overall compressor, low-pressure compressor, high-pressure compressor respectively.
- $\pi_b, \pi_{ab}, \pi_n, \pi_{fn}$: Pressure ratios for the combustor, afterburner, nozzle, fan nozzle
- $\text{peta}_f(e_f), \text{peta}_c(e_c), \text{peta}_t(e_t)$: Polytropic efficiencies for fan, compressor, and turbine
- $\frac{P_0}{P_9}, \frac{P_0}{P_9}$: Static pressure ratios at nozzle exits
- `nozzleType`: Type of nozzle (Converging/Converging-Diverging)
- **get_engine Method**: This method sets up the engine components and calculates the necessary parameters.
- **Input Parameter Extraction and Calculation**: Local variables are assigned the values of the input parameters. η_r and π_d are calculated based on the Mach number (M_0).
- **Parameter Dictionary**: A dictionary named “parameters” is created to store all the input parameters for easy access.
- **Component Initialization**:
 - **Atmosphere**: An instance of the atmosphere class is created with T_0 and P_0 .
 - **Fuel**: An instance of the fuel class is created with specific heat ratios, gas constants, specific heat capacities, and heat of fuel.
 - **Aircraft**: An instance of the aircraft class is created with M_0 , α , and previously created atmosphere and fuel instances.
- **Engine Components**:
 - **Fan**: An instance of the compressor class for the fan is created with π_f and $\text{peta}_f(e_f)$.
 - **Diffuser**: An instance of the nozzle class for the diffuser is created with π_d .
 - **Compressors**: Instances for low-pressure and high-pressure compressors are created with π_{cl} , π_{ch} , and $\text{peta}_c(e_c)$.
 - **Combustor**: An instance of the combustor class is created with π_b , η_b , and T_{t4} .
 - **Turbines**: Instances for high-pressure and low-pressure turbines are created with η_m , $\text{peta}_t(e_t)$, and previously created component instances.
 - **Nozzles**: Instances for the primary and fan nozzles are created with π_n and π_{fn} .
 - **Afterburner (Optional)**: If AB is “Y”, an instance of the combustor class for the afterburner is created; otherwise, it’s set to None.

- **Engine Creation**: Depending on the `engineType`, instances of turbofan, turbojet, or ramjet classes are created with the relevant components and parameters and then returned. By following this structure, the `initialisingComponents` class can set up and return a complete engine model based on the specified configuration and operating conditions.

2.2.15. set_defaultvalues()

The function `set_default_values` initializes default values for various engine parameters and conditions. It sets the following:

- **Engine Type**: Default engine type (e.g., “turbofan”).
- **Environmental Conditions**: Default values for altitude, temperature, pressure, and Mach number. It differentiates between

single and multiple variable analysis by setting appropriate ranges and steps.

- **Thermodynamic Properties**: Sets constants such as specific heat capacities (C_{pc}, C_{pt}, C_{pAB}), specific gas constants (R_c, R_t, R_{AB}), and ratios of specific heats ($\text{gam}_c, \text{gam}_t, \text{gam}_{ab}$).
- **Performance Parameters**: Initializes parameters like turbine inlet temperature (T_{t4}), afterburner temperature (T_{t7}), efficiency parameters ($\text{eta}_b, \text{eta}_m, \text{eta}_{ab}$), and pressure ratios (pi_*).
- **Engine Configuration**: Sets default values for the engine configuration, such as mixed flow or separate flow, and nozzle type.

The function `set_parameters` sets the actual parameters for the engine based on user inputs. It performs the following operations:

- **Initialization**: Initializes all parameters to a default value (“-”).
- **Input Handling**: Fetches user inputs and assigns them to the respective parameters, handling both single and multi-variable configurations.
- **Thermodynamic Calculations**: Computes derived thermodynamic properties (e.g., specific gas constants) based on the input parameters.
- **Configuration Checks**: Adjusts parameters based on engine configuration options, such as the presence of an afterburner or the type of flow (mixed or separate).
- **Error Handling**: Provides error messages if the inputs are invalid.

The file defines a class `initialisingParameters`, which is responsible for initialising and managing engine parameters for different types of jet engines, such as turbofan, turbojet, and ramjet. It allows the user to input specifications and modify parameters for engine simulations. The `initialisingComponents` class is imported to provide the engine components and calculations.

2.2.16. plots.py:

The `plots.py` file defines a `plots` class for generating plots related to propulsion engines. Here’s a summary of its components:

- **Initialisation**: The constructor (`__init__`) takes parameters like `engine`, `x_var`, `y_var`, and optional lists for two independent variables.
- **Methods**: Here are the methods for the `plots` class:
 - `show_plots()`: This method determines if a single or two independent variables are used for plotting. It calls either `foo1` or `foo2` accordingly.
 - `foo1(x_axis, x_axis_arr, y_axis)`: Plots a single x-variable against a y-variable. It initialises engine parameters, handles specific engine type restrictions, and generates the plot.
 - `foo2(x_axis, y_axis, ind_var1, ind_var1_arr, ind_var2, ind_var2_arr)`: Handles plots with two independent variables, iterating over their values to generate multiple plots with different parameter combinations. It also manages engine-specific restrictions and configurations.

The file is designed for engine analysis, focusing on plotting performance characteristics like thrust and efficiency based on various engine parameters. For example, for a turbofan engine, one can plot how either TSFC, ST, or f varies with π_c , α , or T_{t4} .

Also, if two independent variables are chosen, they can plot two quantities from TSFC, ST, and f by varying any two of T_{t4} , π_c , or α . This process depends on the engine cycle chosen.

2.3. What is JetCAP capable of doing?

JetCAP can perform a single-point calculation based on the user inputs for several engine cycles. The engine cycles are as follows:

- Ramjet - Real/Ideal
- Turbojet (Single-Spool) - Real/Ideal
- Turbojet (Dual-Spool) - Real/Ideal
- Turbojet with Afterburner - Real/Ideal
- Turbofan (Dual-Spool) - Real/Ideal
- Turbofan with Mixed Exhaust - Real/Ideal
- Turbofan with Afterburner - Real/Ideal

Apart from these options, the user can select converging only or converging-diverging nozzles for the Turbofan Engines. For Ramjet and Turbojet engines, however, we always use converging-diverging nozzles.

Figure 1. Turbofan PCA input window

Figure 2. Turbofan PCA output window

Additionally, JetCAP can also perform multi-point calculations, where the user can choose to vary one or two independent variables and plot the remaining dependent variables in whichever combination is required.

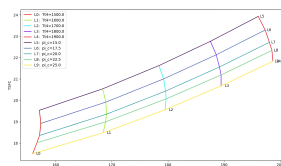


Figure 3. Carpet plot with varying T_{t4} and π_c

2.4. JetCAP PCA validation with PARA

In order to validate the accuracy and reliability of the results produced by JetCAP, we shall compare them to those of PARA. All the important engine cycles will be covered in this comparison. Three of the most important performance parameters, Specific Thrust (ST), Thrust-Specific Fuel Consumption (TSFC), and Fuel-air ratio (f), will be compared.

2.4.1. Turbojet (Dual-spool) - Ideal Cycle

Case1: $M_0 = 1.6$, $alt = 11000m$, $T_{t4} = 1800K$, $\pi_c = 16$, $\pi_{cL} = 3.8$
Case2: $M_0 = 0.8$, $alt = 8000m$, $T_{t4} = 1500K$, $\pi_c = 12$, $\pi_{cL} = 3.8$
Case3: $M_0 = 1.1$, $alt = 13000m$, $T_{t4} = 1650K$, $\pi_c = 18$, $\pi_{cL} = 3.8$

Table 1. Specific Thrust comparison ($\frac{m}{s}$)

	ST_{PARA}	ST_{JetCAP}	% error
Case 1	845.737	900.734	6.10
Case 2	824.423	861.854	4.34
Case 3	880.639	927.420	5.04

Table 2. Fuel-air Ratio comparison

	f_{PARA}	f_{JetCAP}	% error
Case 1	0.02526	0.0336	24.82
Case 2	0.02249	0.0281	19.96
Case 3	0.02431	0.0314	22.57

Table 3. TSFC comparison ($\frac{g}{s.kN}$)

	$TSFC_{PARA}$	$TSFC_{JetCAP}$	% error
Case 1	29.862	37.318	19.97
Case 2	27.282	32.592	16.29
Case 3	27.604	33.808	18.35

2.4.2. Turbojet (Dual-spool) - Real Cycle

Case1: $M_0 = 1.6$, $alt = 11000m$, $T_{t4} = 1800K$, $\pi_c = 16$, $\pi_{cL} = 3.8$
Case2: $M_0 = 0.8$, $alt = 8000m$, $T_{t4} = 1500K$, $\pi_c = 12$, $\pi_{cL} = 3.8$
Case3: $M_0 = 1.1$, $alt = 13000m$, $T_{t4} = 1650K$, $\pi_c = 18$, $\pi_{cL} = 3.8$

Table 4. Specific Thrust comparison ($\frac{m}{s}$)

	ST_{PARA}	ST_{JetCAP}	% error
Case 1	895.97	890.31	0.63
Case 2	828.47	823.12	0.64
Case 3	909.98	904.04	0.65

Table 5. Fuel-air Ratio comparison

	f_{PARA}	f_{JetCAP}	% error
Case 1	0.0352	0.0315	11.74
Case 2	0.03089	0.0269	14.83
Case 3	0.03339	0.0296	12.80

Table 6. TSFC comparison ($\frac{g}{s.kN}$)

	$TSFC_{PARA}$	$TSFC_{JetCAP}$	% error
Case 1	39.282	35.359	11.12
Case 2	37.281	32.674	14.09
Case 3	36.696	32.718	12.15

2.4.3. Turbofan (Dual-spool) - Real Cycle

Case1: $M_0 = 1.6$, $alt = 11000m$, $\alpha = 8$, $T_{t4} = 1800K$, $\pi_c = 24$, $\pi_f = 1.65$, $\pi_{cL} = 3.8$
Case2: $M_0 = 0.8$, $alt = 8000m$, $\alpha = 5$, $T_{t4} = 1500K$, $\pi_c = 18$, $\pi_f = 1.65$, $\pi_{cL} = 3.8$
Case3: $M_0 = 1.1$, $alt = 13000m$, $\alpha = 4$, $T_{t4} = 1650K$, $\pi_c = 12$, $\pi_f = 1.8$, $\pi_{cL} = 2.4$

Table 7. Specific Thrust comparison ($\frac{m}{s}$)

	ST_{PARA}	ST_{JetCAP}	% error
Case 1	100.735	105.325	4.35
Case 2	192.844	197.478	2.34
Case 3	223.122	226.359	1.43

Table 8. Fuel-air Ratio comparison

	f_{PARA}	f_{JetCAP}	% error
Case 1	0.00257	0.0031	17.00
Case 2	0.0041	0.0041	0.00
Case 3	0.00638	0.0064	0.31

Table 9. TSFC comparison ($\frac{g}{s.kN}$)

	$TSFC_{PARA}$	$TSFC_{JetCAP}$	% error
Case 1	31.142	29.772	4.60
Case 2	21.260	20.850	1.96
Case 3	28.574	28.103	1.67

2.4.4. Turbofan (Dual-spool mixed exhaust) - Real Cycle

Case1: $M_0 = 1.6$, $alt = 11000m$, $\alpha = 8$, $T_{t4} = 1800K$, $\pi_c = 24$, $\pi_f = 1.65$, $\pi_{cL} = 3.8$

Case2: $M_0 = 0.8$, $alt = 8000m$, $\alpha = 5$, $T_{t4} = 1500K$, $\pi_c = 18$, $\pi_f = 1.65$, $\pi_{cL} = 3.8$

Case3: $M_0 = 1.1$, $alt = 13000m$, $\alpha = 4$, $T_{t4} = 1650K$, $\pi_c = 12$, $\pi_f = 1.8$, $\pi_{cL} = 2.4$

Table 10. Specific Thrust comparison ($\frac{m}{s}$)

	ST_{PARA}	ST_{JetCAP}	% error
Case 1	165.590	175.945	5.88
Case 2	170.315	173.978	2.10
Case 3	207.295	213.732	3.01

Table 11. Fuel-air Ratio comparison

	f_{PARA}	f_{JetCAP}	% error
Case 1	0.00383	0.00383	0.00
Case 2	0.00320	0.00320	0.00
Case 3	0.00460	0.00460	0.00

Table 12. TSFC comparison ($\frac{g}{s.kN}$)

	$TSFC_{PARA}$	$TSFC_{JetCAP}$	% error
Case 1	23.120	21.503	7.51
Case 2	18.732	18.252	2.62
Case 3	22.175	22.154	0.09

2.4.5. Turbofan (Dual-spool with Afterburner) - Real Cycle

Case1: $M_0 = 1.6$, $alt = 11000m$, $\alpha = 8$, $T_{t4} = 1800K$, $\pi_c = 24$, $\pi_f = 1.65$, $\pi_{cL} = 3.8$, $T_{t7} = 2200K$

Case2: $M_0 = 0.8$, $alt = 8000m$, $\alpha = 5$, $T_{t4} = 1500K$, $\pi_c = 18$, $\pi_f = 1.65$, $\pi_{cL} = 3.8$, $T_{t7} = 2000K$

Case3: $M_0 = 1.1$, $alt = 13000m$, $\alpha = 4$, $T_{t4} = 1650K$, $\pi_c = 12$, $\pi_f = 1.8$, $\pi_{cL} = 2.4$, $T_{t7} = 1800$

Table 13. Specific Thrust comparison ($\frac{m}{s}$)

	ST_{PARA}	ST_{JetCAP}	% error
Case 1	957.975	962.250	0.44
Case 2	704.080	714.242	1.42
Case 3	758.738	767.150	1.09

Table 14. Fuel-air Ratio comparison

	f_{PARA}	f_{JetCAP}	% error
Case 1	0.02824	0.02834	0.00
Case 2	0.02460	0.02489	1.16
Case 3	0.03188	0.03026	5.35

Table 15. TSFC comparison ($\frac{g}{s.kN}$)

	$TSFC_{PARA}$	$TSFC_{JetCAP}$	% error
Case 1	62.612	61.246	2.23
Case 2	76.079	75.921	0.20
Case 3	60.390	61.479	1.77

3. Engine Performance Analysis (EPA)

3.1. What is EPA?

Engine Performance Analysis determines the performance of a given engine at points other than the design point. This is also known as off-design calculation. Before starting the analysis, we must first fix the reference engine. This can be the same engine being used for design-point calculation or PCA. To size our reference engine, the user also has to input the reference mass flow rate. Then, the off-design performance parameters are calculated using the reference engine with the help of some basic equations. The inputs to the off-design calculation are its flight conditions and also the Turbine inlet total temperature (T_{t4}). This T_{t4} is the throttle setting of the engine.

3.2. Our methodology for EPA

EPA requires the assistance of PCA to calculate the performance parameters of the reference engine.

The EPA testing class is designed to perform off-design point analysis of various types of aircraft engines such as turbofans, turbojets, and ramjets. The methodology involves comparing a reference engine's performance with a test engine under different operating conditions. Below is a summary of the key steps and calculations involved:

1. Initialization:

- The reference engine parameters and mass flow rate are initialized.
- The environmental and engine parameters for the test engine are set. (User inputs)

2. Mass Flow Parameter (MFP) Calculation:

- The MFP is calculated using the equation:

$$MFP = \sqrt{\frac{\gamma}{R}} M \left(1 + \frac{\gamma - 1}{2} M^2 \right)^{-\frac{\gamma + 1}{2(\gamma - 1)}}$$

3. Reference Engine Details:

- The thrust, specific thrust, and TSFC (Thrust Specific Fuel Consumption) of the reference engine are determined using PCA.

4. Off-Design Analysis:

- The test engine's environmental and operational parameters (such as Mach number, temperature, and pressure) are set.
- The iteration scheme for calculating parameters such as τ_f , τ_{ch} , π_{ch} , and π_f for turbofan engines is implemented.
- For turbojet and ramjet engines, corresponding τ and π parameters are calculated directly.

5. Parameter Iteration for Turbofans:

- Initialize τ_f , $\pi_t L$, and $\tau_t L$ using reference parameters.

- Iteratively update τ_{ch} , π_{ch} , π_f , M_{19} , M_9 , MFP_{M19} , MFP_{M9} , and MFP_{M9R} until convergence.
- Calculate parameters such as bypass ratio (α) and component pressure ratios.

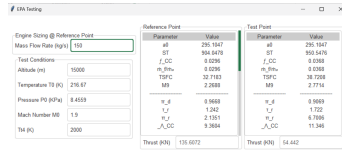
6. Final Calculations:

- Calculate the mass flow rate \dot{m} for the test engine.
- Determine the thrust of the test engine using specific thrust and mass flow rate.

7. Engine Component Initialization:

- Use PCA (Principal Component Analysis) to initialize the components of the test engine based on the calculated parameters.
- The method returns the initialized test engine and its mass flow rate.

The described methodology allows for the performance analysis of different types of engines by iteratively adjusting and comparing key parameters from a reference engine to a test engine under specified off-design conditions.



Engine Setting @ Reference Point		Reference Point		Test Point	
Parameter	Value	Parameter	Value	Parameter	Value
Engine Setting @ Reference Point	100	AR	200.1607	AR	200.1607
Mass Flow Rate (kg/s)	100	ST	904.9479	ST	904.9479
Test Conditions		f_{DC}	0.0206	f_{DC}	0.0206
Altitude (m)	10000	$m_{f,DC}$	0.0206	$m_{f,DC}$	0.0206
Temperature T0 (K)	216.67	T0/C	22.7163	T0/C	22.7163
Pressure P0 (kPa)	8.4333	MP	2.2083	MP	2.2083
Mach Number M0	1.5	π_c	0.9889	π_c	0.9889
Tot (K)	2000	π_f	1.242	π_f	1.242
		π_{ch}	2.1931	π_{ch}	6.7080
		$\pi_{ch,DC}$	9.3844	$\pi_{ch,DC}$	11.340
		Thrust (kN)	135.0772	Thrust (kN)	14.442

Figure 4. EPA window

4. A Final Discussion

As it was clear from the comparison of the PCA results between PARA and JetCAP, the ideal cycles have a significant percentage of error. In the book, the author has made a small assumption, which states that:

$$\dot{m}_0 + \dot{m}_f \approx \dot{m}_0$$

When we apply this assumption to JetCAP, the results match. However, in order to maintain the highest accuracy possible, we have ignored this assumption.

Another source of error could be in the Fuel-air ratio (f). This is because it is being calculated iteratively using experimental data. We have applied our own iteration scheme for the type of data that we had. We validated this scheme using some textbook examples. However, it does not always match the results produced by PARA.

The EPA option in JetCAP has not been validated due to the lack of availability of reliable and similar software tools. It will be improved in the future versions to come.

5. Acknowledgements

We extend our sincere gratitude to Professor Dilip Srinivas Sundaram and Mr. Ganeshkumar V for their invaluable support and mentorship throughout this project. Their guidance, expertise, and encouragement have been instrumental in the successful completion of our work. We are deeply appreciative of their commitment to our academic and professional growth.

References

- [1] D. T. P. Jack D. Mattingly William H. Heiser, *Aircraft Engine Design*, 2nd. Reston, VA, USA: American Institute of Aeronautics and Astronautics, 2002.
- [2] J. D. Mattingly, *Elements of Gas Turbine Propulsion*, 1st. Reston, VA, USA: American Institute of Aeronautics and Astronautics, 2005.