MARMARA UNIVERSITY

FACULTY OF ENGINEERING

MODELING THE THERMODYNAMIC CYCLE OF A TURBOJET ENGINE

Bahadır YILDIZ, Fatih ÇINAR, Havva DEMİR, Lara DURAN

GRADUATION PROJECT REPORT

Department of Mechanical Engineering

Supervisor

Doç. Dr. Candeniz SEÇKİN

ISTANBUL, 2024

MARMARA UNIVERSITY
FACULTY OF ENGINEERING

Modeling the Thermodynamic Cycle of a Turbojet Engine

Bahadır YILDIZ, Fatih ÇINAR, Havva DEMİR, Lara DURAN
June, 2024, Istanbul

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE

OF BACHELOR OF SCIENCE

AT MARMARA UNIVERSITY

Signature of Author(s) .....
Department of Mechanical Engineering

Certified By .....
Project Supervisor, Department of Mechanical Engineering

Accepted By.....
Head of the Department of Mechanical Engineering

ACKNOWLEDGEMENT

First, we would like to thank our supervisor Doç. Dr.Candeniz SEÇKİN, for the valuable guidance and advice on preparing this thesis and giving me moral and materialsupport.

June 2024 **Bahadır YILDIZ, Fatih ÇINAR, Havva DEMİR, Lara DURAN**

Table of Contents

ÖZET

Bu tez, turbojet motorunun termodinamik çevriminin modellenmesi üzerine odaklanmaktadır. Araştırmanın temel amacı, turbojet motorlarının çalışma dinamiklerini ve termodinamik verimliliklerini doğru bir şekilde yansıtan bir model geliştirmektir. Bu çalışmada ideal Brayton çevrimi kullanılmış ve Gasturb uygulamasına benzer bir arayüz geliştirilmiştir. Bu modelleme girişimi, mevcut literatürün kapsamlı bir şekilde incelenmesiyle başlamakta ve turbojet motorlarının teorik analizleriyle devam etmektedir. Python programlama dili kullanılarak geliştirilmiş olan bu arayüz, ampirik veriler ve performans metrikleri kullanılarak doğrulanmış ve güvenilirliği sağlanmıştır. Araştırmanın bulguları, turbojet motorlarının verimliliklerinin artırılması ve emisyonlarının azaltılması için önemli yollar sunarak, havacılık mühendisliği alanına hem akademik hem de pratik katkılar sağlamaktadır. Bu tez, daha sürdürülebilir ve ekonomik olarak uygulanabilir havacılık uygulamalarına katkıda bulunma potansiyeline sahiptir.

ABSTRACT

This thesis focuses on the modeling of the thermodynamic cycle of a turbojet engine. The primary aim of this research is to develop a model that accurately reflects the operational dynamics and thermodynamic efficiencies of turbojet engines. The study utilizes the ideal Brayton cycle and includes the development of an interface similar to the Gasturb application. This modeling initiative begins with an extensive review of existing literature and continues with theoretical analyses of turbojet engines. The interface, developed using the Python programming language, is further refined and validated using empirical data and performance metrics. The findings of this research provide significant insights into enhancing efficiency and reducing the emissions of turbojet engines, offering both academic and practical contributions to the field of aerospace engineering. This thesis holds potential for contributing to more sustainable and economically viable aviation practices.

SYMBOLS

T_A: **Ambient Temperature**

P_A: **Ambient Pressure**

A: **Nozzle Area**

C_{PA}**: Specific Heat at Constant Pressure (air)**

C_{PG}: **Specific Heat at Constant Pressure (Combustion Gas)**

γ: **Specific Heat Ratio of Air**

γ_G: **Specific Heat Ratio of Combustion Gases**

R: **Gas Constant**

η_I: **Intake efficiency**

η_C: **Compressor Efficiency**

η_b: **Combustion Efficiency**

η_t: **Turbine Efficiency**

η_N: **Propelling Nozzle Efficiency**

η_M: **Mechanical Efficiency ηM**

c_p: **Specific Heat**

V_A: **Accelerating Air From Zero Velocity**

$V_{1:}$ **Compressor Inlet Velocity**

$P_{1:}$ **Compressor Inlet Pressure**

T_1: **Inlet Temperature**

$c_{p,avg}$: **Average Specific Heat at Constant Pressure**

Q_R: **Fuel Calorific**

f: **Fuel to Air Ratio**

CPR: **Compressor Pressure Ratio**

T: **Thrust**

F_S: **Specific Thrust**

TSFC: **Thrust-Specific Fuel Consumption**

h: **Enthalpy**

s_X: **Entropy of an X condition**

ρ_X: **Density of an X condition**

ν_X: **Specific Volume of an X condition**

η_{ab}: **Afterburner Combustion Efficiency**

$\dot{m}_a$: **Mass Flow Rate**

T_2: **Compressor Outlet Temperature**

T_3: **Turbine Inlet Temperature**

T_4: **Turbine Outlet Temperature**

$T_{4,\text{ with losses}}$: **Turbine Outlet Temperature with Losses**

T_6: **Afterburner Outlet Temperature**

T_8: **Nozzle Outlet Temperature**

W_T: **Turbine Work**

W_C: **Compressor Work**

P_C: **Critical Pressure**

γ_h: **Specific Heat Ratio Combustion Gas**

V_8: **Nozzle Outlet Velocity**

$\dot{m}_8$: **Nozzle Outlet Mass Flow Rate**

P_2: **Compressor Outlet Pressure**

P_3: **Turbine Inlet Pressure**

P_4: **Turbine Outlet Pressure**

P_6: **Afterburner Outlet Pressure**

P_8: **Nozzle Outlet Pressure**

ABBREVIATIONS

CFD: **Computational Fluid Dynamics**

ICAO: **International Civil Aviation Organization**

ISA: **The Standard Atmosphere**

ISO: **International Organization for Standardization**

M: **Mach Number**

FIGURE LIST

1. INTRODUCTION

The relentless pursuit of advancements in aerospace technology, specifically in the optimization of aircraft propulsion systems, underscores the critical importance of engineering endeavors aimed at enhancing efficiency, reliability, and environmental sustainability. This thesis, titled "Modeling the Thermodynamic Cycle of a Turbojet Engine," embarks on an in-depth exploration of the thermodynamic intricacies that govern the operation of turbojet engines. Central to the propulsion systems of modern aviation, turbojet engines operate on the Brayton cycle principles, involving air intake, compression, combustion, expansion, and exhaust processes. The efficiency and performance of these engines are intrinsically linked to the thermodynamic properties and transformations occurring within this cycle. Despite considerable advancements in turbojet technology over the decades, the industry continues to grapple with the challenges of reducing fuel consumption and emissions while enhancing overall performance.

The primary aim of this research is to develop a sophisticated and accurate model that captures the operational dynamics and thermodynamic efficiencies of the turbojet engine. By doing so, it endeavors to shed light on potential optimization strategies that could pave the way for advancements in engine design and operation, thereby contributing significantly to the aerospace engineering field. This modeling initiative is not merely an academic exercise but a critical step towards understanding and improving the environmental footprint of aviation propulsion systems.

Adopting a comprehensive methodology, this thesis begins with an extensive review of the existing literature to consolidate current knowledge and identify gaps in the understanding of turbojet engine thermodynamics. Theoretical analyses based on thermodynamics, fluid mechanics, and heat transfer principles are employed to elucidate the fundamental processes underpinning the Brayton cycle. The heart of this research lies in the development of a mathematical model that accurately reflects the thermodynamic cycle of turbojet engines. This model is further refined and validated through computational simulations, utilizing empirical data and performance metrics to ensure its reliability and accuracy.

The significance of this research extends beyond the academic domain, offering practical insights that could inform future technological developments in aerospace propulsion.

By highlighting pathways for efficiency enhancements and emission reductions, the findings of this thesis have the potential to contribute to more sustainable and economically viable aviation practices, aligning with global efforts to mitigate the environmental impact of air travel.

In summary, "Modeling the Thermodynamic Cycle of a Turbojet Engine" represents a comprehensive endeavor to advance the understanding and optimization of turbojet engines through sophisticated thermodynamic modeling. This research not only enriches the academic landscape of aerospace engineering but also holds promise for influencing the future design, operation, and sustainability of turbojet propulsion systems, marking a significant stride toward achieving a more sustainable aviation industry.

1.1. History

The modeling of the thermodynamic cycle of turbojet engines has a rich history that intersects with the evolution of aerospace engineering, beginning with the pioneering work of Frank Whittle in the 1930s. Whittle's patenting of the first turbojet engine not only catalyzed rapid advancements in aircraft propulsion but also necessitated a deeper understanding of the thermodynamics governing these engines. This era marked the dawn of a technological revolution in air travel, offering unprecedented speeds and altitudes. The core principles laid out by early innovators such as Whittle and von Ohain emphasized the Brayton cycle's critical role in jet engine operation, forming the foundation for all subsequent research and development in jet propulsion technologies. (1)

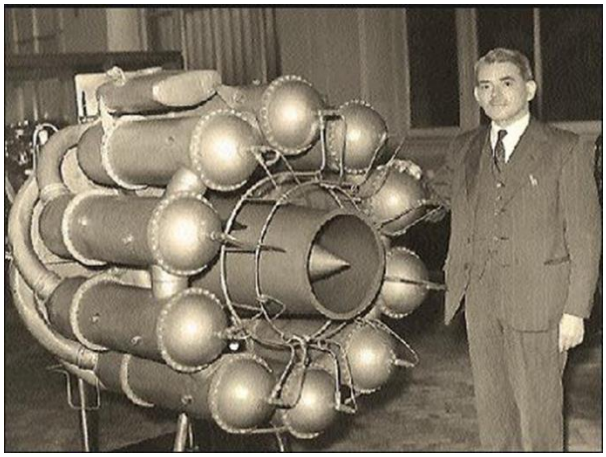


Figure 1. 1: Frank Whittle's first Jet engine.

As the field advanced, the complexity of modeling turbojet engines' thermodynamic cycles increased, paralleling the technological advancements in the engines themselves. The advent of computational fluid dynamics (CFD) in the latter half of the 20th century represented a significant leap forward, enabling more precise simulations of the intricate airflow and combustion processes occurring within engines. This period saw a shift towards optimizing engine designs for enhanced fuel efficiency and reduced emissions, driven by both economic factors, and growing environmental concerns related to aviation's impact. (1)

In recent years, the push for sustainable aviation has further underscored the importance of sophisticated thermodynamic models. These models are now instrumental in exploring the performance of turbojet engines under a variety of conditions, including the use of alternative fuels and the incorporation of innovative engine architectures. The ongoing refinement of these models is crucial for addressing the aviation industry's dual challenges of minimizing its environmental footprint and navigating the economic realities of fluctuating fuel costs and stringent regulatory landscapes. (1)

The narrative of turbojet engine modeling is a testament to the collaborative spirit of the aerospace community, bringing together physicists, chemists, material scientists, and engineers in a shared quest for innovation. As the industry strides towards a more sustainable future, the insights garnered from decades of thermodynamic modeling will continue to play a vital role in shaping the next generation of turbojet engines, ensuring that air travel remains a cornerstone of global connectivity in the 21st century. (1)

1.2. The Purpose and Significance of The Study

In the realm of aerospace engineering, the pursuit of innovation is not just a matter of surpassing the boundaries of speed and altitude but also involves addressing the twin challenges of efficiency and sustainability. At the heart of this quest lies the turbojet engine, a cornerstone of modern aviation that has powered aircraft across skies for decades. The study dedicated to modeling the thermodynamic cycle of such engines stands at the confluence of scientific curiosity and practical necessity. It embodies an effort to push the frontiers of knowledge while grappling with the immediate concerns of our times. (2)

Figure 1. 2: Missiles powered by turbojet engines.

The significance of modeling the thermodynamic cycle cannot be overstated. The operation of a turbojet engine, guided by the principles of the Brayton cycle, involves a complex interplay of air intake, compression, combustion, and exhaust. Each stage of this cycle has profound implications for the engine's performance, fuel consumption, and emissions. In an era where the aviation industry is increasingly under scrutiny for its environmental impact, understanding and optimizing these processes is not merely an academic exercise but a pressing need. This research, therefore, is not just about enhancing the efficiency of these engines; it's about reimagining them for a world that demands greener alternatives. (2)

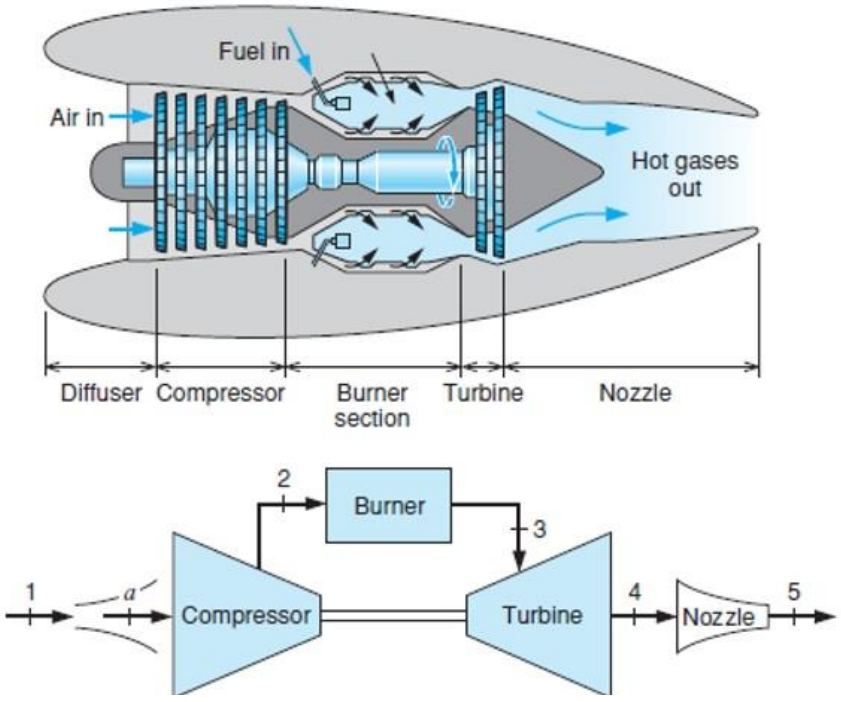


Figure 1. 3: Brayton cycle in Turbojet Engine.

Moreover, the purpose of this study transcends the technical realm. It serves as a bridge between the theoretical underpinnings of thermodynamics and the practical challenges of aerospace engineering. By developing a detailed model of the turbojet engine's thermodynamic cycle, this thesis aims to provide a tool for engineers and designers. Such a tool can not only predict engine performance under a variety of conditions but also suggest pathways for innovation. Whether it's through the adoption of new materials, the refinement of engine components, or the exploration of alternative fuels, the insights gleaned from this research could pave the way for the next generation of turbojet engines. (2)

This endeavor is timely and critical. As the world edges closer to the limits of its fossil fuel reserves and confronts the reality of climate change, the aviation industry faces a dual challenge. It must find ways to cut its greenhouse gas emissions while continuing to meet the growing demand for air travel. In this context, the modeling of the thermodynamic cycle of a turbojet engine is not just an academic pursuit but a vital step toward a more sustainable future. It represents a commitment to the principles of sustainable development, seeking to balance the needs of the present with the well-being of future generations. (2)

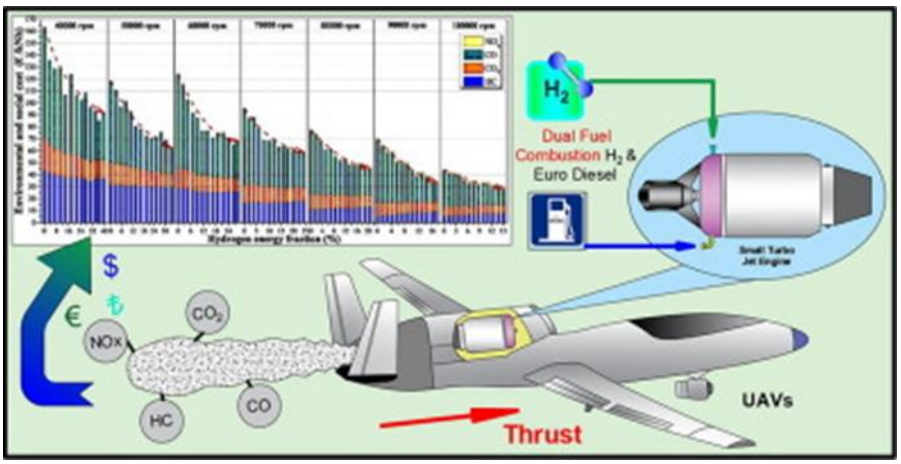


Figure 1. 4: Turbojet engine and sustainability.

In conclusion, the study on modeling the thermodynamic cycle of a turbojet engine is a reflection of the broader challenges and opportunities that lie ahead for the aerospace industry. It is an exploration that is as much about understanding the laws of physics as it is about addressing the economic and environmental concerns of our times. Through this research, we are not just aiming to make incremental improvements to engine efficiency but are striving to contribute to the sustainability of aviation and, by extension, to the sustainability of our planet. (3)

1.3. Operating Principle

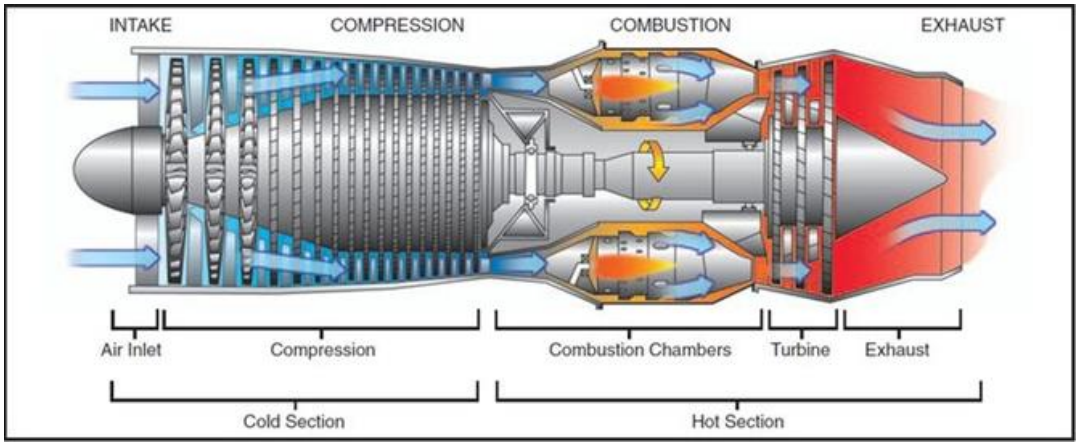


Figure 1. 5: Turbojet engine parts.

A turbojet engine operates based on a relatively straightforward principle that involves four main stages: air intake, compression, combustion, and exhaust.

- **Air Intake:** The process begins with air being drawn into the engine through the intake. This air is essential for the combustion process that occurs later in the cycle.

- **Compression:** Once inside, the air is directed to a compressor where it is compressed to a higher pressure. Compressing the air increases its temperature and energy content, preparing it for efficient combustion. The compressor consists of multiple stages of rotating blades and stationary stators that progressively compress the incoming air.

- **Combustion:** The high-pressure air then moves into the combustion chamber, where it is mixed with fuel (typically aviation kerosene) and ignited. The combustion of the fuel-air mixture significantly increases the temperature and volume of the gases, creating high-energy exhaust gases. This rapid expansion of gases generates thrust and propels the engine (and the aircraft) forward.

- **Exhaust:** Finally, the hot exhaust gases are expelled through the turbine and out of the engine via the nozzle at the back. As the gases pass through the turbine, they cause it to spin. The turbine is connected to the compressor by a shaft, and the energy extracted by the turbine is used to power the compressor. The expulsion of gases through the nozzle at high speed according to Newton's third law of motion ("For every action, there is an

equal and opposite reaction") provides additional thrust that helps propel the aircraft forward. (4)

The efficiency and performance of a turbojet engine are influenced by various factors, including the design of its components, the quality of the fuel, and the operating conditions. The entire process, governed by the principles of the Brayton cycle, showcases the conversion of chemical energy in the fuel into mechanical energy in the form of thrust, enabling aircraft to fly at high speeds and altitudes. (5)

2. METHODOLOGY

2.1. Description Of Main Components

Turbojet engines, as a type of internal combustion engines, adhere to a fundamental principle of converting air into thrust. These engines, commonly used in aircraft, generate all of their thrust by producing a high-speed and high-energy flow of gases. Unlike other types of jet engines, the air entering a turbojet engine passes through the engine core, thus being entirely converted into thrust through an internal combustion process. This process involves a complex series of components and operations. (4)

The basic components of a turbojet engine include the air intake, compressor, combustion chamber, turbine, and exhaust nozzle. The process begins with the air intake, where air is drawn into the engine and then compressed by the compressor. The compressor compresses the airflow, densifying it and increasing its temperature. The compressed air is then directed to the combustion chamber, where fuel is injected and burned. The resulting high-pressure and high-temperature gases cause the turbine to rotate, ultimately generating the energy needed to drive the compressor. (4)

The remaining energy is released as thrust through the exhaust nozzle. The exhaust nozzle typically utilizes this energy to create a high-speed jet flow that provides the thrust. This jet flow creates a fast-moving stream, propelling the aircraft forward. (4)

The performance of turbojet engines is typically optimized for speeds approaching Mach 2. However, at lower speeds, turbofan engines, especially due to their mass, are known

to be more efficient and produce less noise. This constitutes a fundamental difference in performance and application areas between turbojet and turbofan engines. (4)

In the design of turbojet engines, the air intake holds particular importance. The air intake can directly affect the engine's efficiency and the overall performance of the aircraft. Specifically, the design of the air intake should be carefully examined to minimize pressure losses and ensure smooth airflow to the compressor. Irregular or disrupted airflow can cause a surge in the compressor, leading to serious mechanical damage. Therefore, the air intake is a significant focal point in the design of turbojet engines and should be handled with care. (4)

Considering these fundamental principles, turbojet engines have a wide range of applications. Particularly in the aviation sector, turbojet engines are commonly used as power sources for large aircraft. However, at higher speeds and altitudes, the efficiency of turbojet engines decreases, making turbofan engines more preferred. These factors are essential considerations affecting the performance and usage of turbojet engines. (4)

Gas turbines are an integral part of modern industry, particularly in the aviation and energy sectors, where they are widely used. These engines, as a specialized type of internal combustion engines, adopt a fundamental principle of converting air into propulsive force. They typically consist of five main components: inlet, compressor, combustion chamber, turbine, and exhaust nozzle. Their operation is complex and occurs through sequential steps. (4)

Firstly, air enters from the inlet into the turbine and then progresses to the compressor. The compressor increases the air pressure required for the turbine to operate. Then, fuel is injected into the compressed air and ignited in the combustion chamber. As a result of this combustion process, a mixture of high-temperature combustion gases and air is produced. Subsequently, the high-temperature and high-pressure gas enters the turbine and expands until exhaust pressure. During this expansion, a portion of the energy converts to kinetic or rotational energy, producing the mechanical energy that drives the compressor. Simultaneously, another portion of the energy creates thrust in the exhaust nozzle. (6)

Gas turbines have found broad applications in both the aviation and electricity generation sectors. They are particularly vital in aviation for efficiently powering large aircraft. However, continuous research and development efforts focus on the design and operation of

these engines. The aim is to reduce environmental impacts and enhance performance by focusing on factors such as fuel efficiency, operational flexibility, and noise reduction. Thermal and material engineering plays a crucial role in the design and development of gas turbines to achieve these goals. (6)

On the other hand, turbojet engines hold significant importance in the aviation industry. The first centrifugal flow turbojet engine was invented in the 1930s by Frank Whittle and Hans von Ohain. These engines generate thrust by producing a high-speed and high-energy gas flow. However, at low speeds and subsonic velocities, the efficiency of turbojet engines is low, and they exhibit high specific fuel consumption. Additionally, turbojet engines produce high levels of noise, which limits their usage. Over time, more efficient and quieter turboprop and turbofan engines have replaced turbojets. Turbofan engines are quieter and consume less fuel compared to turbojets, making them the preferred choice. Therefore, in the aviation industry, turbofan engines have generally supplanted turbojets. (6)

The difference between shaft power cycles and aircraft gas turbine cycles manifests in the useful power output. In turbojets, thrust is generated in the exhaust nozzle. Gas turbines enable higher speeds and altitudes, thus replacing piston engines. Their inlets, crucial for achieving forward speed, must be considered separately as a component. Inlet design significantly influences engine efficiency and aircraft safety. Minimizing pressure losses and ensuring smooth airflow to the compressor are essential. Irregular or disrupted environmental airflow can cause a surge in the compressor, leading to severe mechanical damage. A gas turbine compressor requires an airflow with an axial Mach number (M) between 0.4-0.5 at the first-stage inlet. While subsonic aircraft typically cruise between 0.8-0.85 (M), supersonic aircraft operate at speeds of M 2-2.5. The engine operates with maximum power and airflow at zero forward speed during takeoff. Therefore, the inlet must handle heavy loads under many operating conditions. In static conditions or at very low forward speeds, the inlet acts as a nozzle, accelerating air from zero velocity (V_A) to compressor inlet velocity (V_1). At normal forward speeds, the air slows from V_A to V_1, and static pressure rises from atmospheric pressure (P_A) to compressor inlet pressure (P_1). This study focuses on modeling a turbojet gas turbine and examines how to develop a gas turbine model as close to reality as possible through theoretical calculations. A model will be created in GasTurb using theoretical calculations. (6)

Primary components of turbojet engines are:

- Air intake
- Compressor
- Combustion chamber
- Turbine
- Nozzle
- Afterburner

1.) **Intake Section:** The air intake serves as the entry point for outside air into the engine, ensuring a smooth flow and aerodynamic efficiency. Despite their apparent simplicity, air intakes are actually quite complex in design.

Air intakes come in two main geometries: converging and diverging. Converging intakes are utilized to accelerate air to approximately 0.5 Mach for engines operating below that speed threshold. Conversely, divergent ducts are employed to decelerate air to 0.5 Mach for engines operating above that speed.

Additionally, air intakes are equipped with heating systems to prevent ice buildup. Ice accumulation can disrupt airflow into the engine, leading to turbulent flow and potential engine stall. Therefore, these heating systems are essential to maintain optimal engine performance and safety. (7)

2.) **Compressor:** The compressor functions by compressing air, thereby increasing its pressure. This process facilitates a greater volume of air entering the engine, playing a pivotal role in preparing the air for the combustion chamber.

Compressors are utilized to compress the incoming airflow by augmenting its kinetic energy. Subsequently, the diffuser slows down the air and converts kinetic energy into potential energy (pressure), consequently enhancing the engine's efficiency. The compressor can either be a radial flow impeller, accelerating the flow towards a diffuser, or an axial flow compressor, expediting the airflow towards a diffuser. Traditionally, both types of blades are constructed from titanium, aluminum, or steel. Titanium is often favored due to its lightweight nature and resistance to corrosion and creep. Additionally, carbon fiber blades, particularly in the GEnx engine, are employed. Metal compressor blades are cast with molten metal, cooled, and then processed into their final shape. (8)

In turbojet engines, the compressor compresses the air by drawing it into the engine, thus amplifying its pressure. This process facilitates the intake of a greater volume of air into the combustion chamber, thereby providing the requisite oxygen for combustion. Furthermore, the compressor elevates the temperature of the air by compressing it, resulting in a more efficient combustion process. Consequently, the compressor enhances the engine's efficiency and contributes to the generation of a more potent thrust force. (8)

3.) Combustion Chamber: By compressing air in the compressor, we increase its pressure, allowing more air to enter the combustion chamber. The combustion chamber is a critical area where air and fuel are mixed and ignited. Here, the flow of air is controlled, and the fuel is properly mixed with the air. This mixture is ignited under high temperature and pressure, resulting in the formation of gas. (9)

The temperature of the gas exiting the combustion chamber is quite high, typically ranging from 1,800 to 2,000°C. However, this high temperature can cause the combustion chamber to melt, which can adversely affect the durability of the engine. Therefore, it is important for the combustion chamber to be strong and durable. (9)

Several measures are taken to increase the strength of the combustion chamber. Firstly, a boundary layer is created between the hot gas inside the combustion chamber and the outside of the chamber, reducing the external temperature of the chamber. Secondly, the combustion chamber is made of a titanium alloy and coated with ceramic. These ceramic coatings reduce the temperature of the combustion chamber, preventing it from melting. (9)

In conclusion, the combustion chamber is of critical importance for ensuring the efficient operation and durability of the engine. Therefore, careful attention must be paid to the design and material selection of the combustion chamber. (9)

4.) Turbine: Gas turbine engines are an important type of internal combustion engines used to convert air into thrust. These engines typically consist of five main components: the inlet, compressor, combustion chamber, turbine, and exhaust nozzle. Their operation is complex and occurs in sequential steps. (9)

In the initial stage of the engine operation, air is drawn in from the outside and directed into the engine from the inlet section. Subsequently, the compressor, used to compress the incoming air, increases the airflow, thereby enhancing the efficiency of the engine. Then, fuel is injected into this compressed air and ignited in the combustion chamber. This combustion process leads to the formation of gas at high temperature and pressure. (9)

In the next step, the high-pressure gases exiting the combustion chamber pass through a component called the turbine. The turbine converts the energy contained in these gases into mechanical energy, generating rotational motion. Through a driveshaft, the turbine drives the compressor, facilitating the intake of fresh air. (9)

After exiting the turbine, the hot gases rapidly expand and are expelled through the exhaust nozzle. During this expansion, some of the energy contained in the gases contributes to the rotation of the turbine. Turbine blades are typically made of nickel-based alloys to withstand high temperatures and pressures. Additionally, the blades are coated with ceramic to prevent melting and are equipped with internal air channels for cooling. (9)

Following these steps, the high-speed gases expelled through the exhaust nozzle are directed by the thrust nozzle, thereby creating the engine's thrust force. In this way, gas turbine engines utilize the fundamental principles of internal combustion engines to provide a powerful and efficient source of propulsion. (9)

5.) Exhaust Nozzle: Gas turbine engines play a significant role in the aviation industry and energy production. These engines are considered a specialized type of internal combustion engines and adopt a fundamental principle of converting air into thrust force. Generally, engines consist of five main components: inlet, compressor, combustion chamber, turbine, and exhaust nozzle. Each of these components plays a specific role in the complex operation of the engine. (9)

Firstly, air is drawn into the engine from the inlet and progresses to the compressor. The compressor increases the pressure of the air by compressing it and prepares it for the combustion chamber. Then, fuel is injected into the compressed air and ignited in the combustion chamber. This combustion process results in the formation of gases under high temperature and pressure. (9)

The high-pressure gases exiting the combustion chamber enter the turbine. The turbine converts the energy of the gases into mechanical energy, creating rotational motion. Subsequently, the hot gases expand rapidly, partially causing the turbine blades to rotate. The turbine blades, connected to the compressor blades via a driveshaft, facilitate the flow of fresh air. (9)

Lastly, the exhaust nozzle controls the speed and direction of the exhaust gases. The exhaust gases typically exit at high temperatures (550–850°C), requiring the exhaust to be made of durable materials such as nickel alloy or titanium alloy. The exhaust nozzle regulates the gas flow to optimize the engine's performance. (9)

In this way, gas turbine engines operate in a complex process and have a wide range of applications in the aviation industry and energy production. (9)

6.) Afterburner: Afterburners are simple fuel injection systems that inject extra fuel into the post-combustion section of an engine for a short period of time to provide additional power. Afterburner-equipped jet engines typically feature a double-wall structure with an air gap between them. The existing hot gases pass through this gap, known as the inner wall of the engine, where they mix with the incoming cold airflow from the front of the engine. This design also prevents the transfer of heat to the aircraft structure. (9)

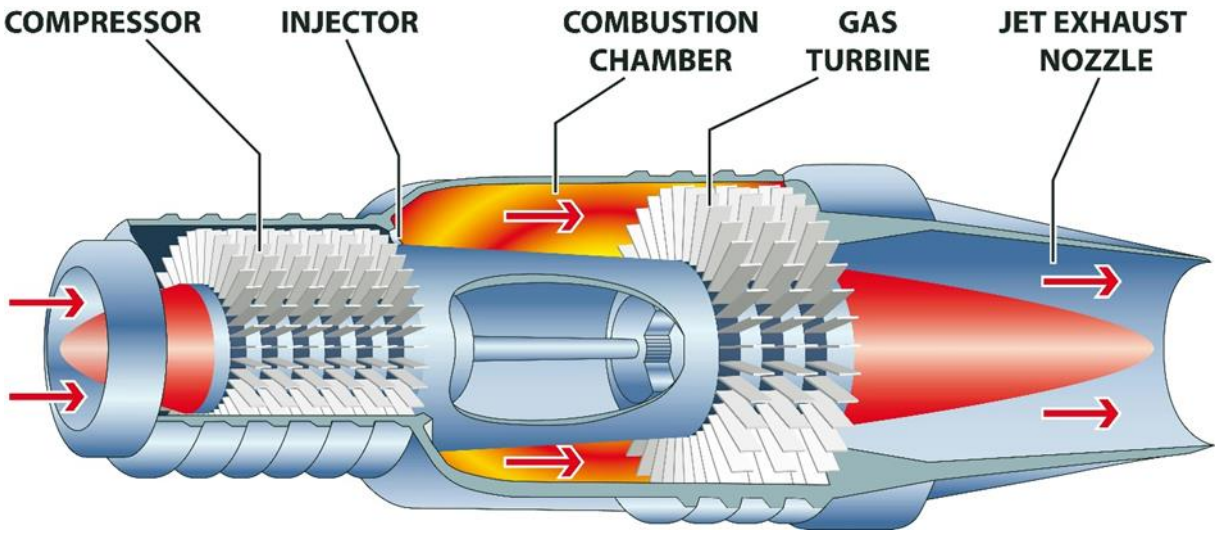


Figure 2. 1: The main components of the turbojet engine.

2.2. Air Standards

The efficiency and performance of turbojet engines depend on the atmospheric conditions of their operating environment and therefore must conform to specific air standards. These standards are typically based on the international civil aviation authorities' guidelines and widely accepted in the aviation industry. The standard atmosphere represents the average atmospheric conditions at a given altitude; variables such as temperature, pressure, and density are defined as altitude increases from sea level. (10)

The primary atmospheric variables that affect the performance of turbojet engines generally include air pressure, temperature, and density. Air pressure affects the engine's capacity for air intake and compression. Low air pressure can reduce engine efficiency, while high air pressure can contribute to better performance and thrust. Temperature determines the efficiency of the combustion process in the engine's combustion chamber. Lower temperatures can enhance engine performance by providing denser and cooler airflow. Conversely, density determines the engine's ability to compress airflow, with higher density contributing to better performance. (10)

The air standards in which turbojet engines operate are typically based on international standards set for air transportation and aircraft manufacturing. These standards are established to ensure that engines operate reliably under various atmospheric conditions and altitudes. Engine design and performance assessment are usually conducted in accordance with these standards. Compliance ensures that engines maintain their functionality at the highest level in terms of reliability, performance, and efficiency. Therefore, conformity to atmospheric conditions is a fundamental requirement of the aviation industry and should be consistently considered. (10)

The most used air standards are as follows:

1.) International Civil Aviation Organization Standard Atmosphere (ICAO ISA): The Standard Atmosphere (ISA), established by the International Civil Aviation Organization (ICAO), serves as a significant reference point for evaluating the performance of jet engines. ISA defines and standardizes atmospheric conditions within the altitude range from sea level up to 11 kilometers. This model provides a critical tool for understanding the operating environment of jet engines and assessing their performance. The primary purpose of ISA is to determine how atmospheric variables such as air pressure, temperature, and density change with altitude. (10)

The ISA model starts with standard atmospheric conditions at sea level and predicts a change with every 1000 feet (or 304.8 meters) increase in altitude. These changes are essential for understanding how significant variables like temperature, pressure, and density in the atmosphere affect the performance of jet engines. For instance, at higher altitudes, air pressure decreases, temperature typically decreases, and density diminishes. These variations affect the

engine's processes of air intake, compression, and combustion, thereby influencing the overall performance of the engine. (10)

The critical role of the ISA model in determining the performance of jet engines forms the basis for many standards used in the aviation industry. These standards are employed in various fields such as aircraft design, air traffic management, and flight planning. Therefore, the standardization of atmospheric conditions by ISA lays the foundation for safe and efficient flights across the jet engine and aviation industry. (10)

Figure 2. 2: International Civil Aviation Organization Standard Atmosphere (ICAO ISA)

2.) International Standard Atmosphere (ISA): The International Standard Atmosphere (ISA) is an atmospheric model defined by the International Organization for Standardization (ISO) and serves as a crucial reference point for evaluating the performance of jet engines. ISA standardizes the atmosphere within the altitude range from sea level up to 80 kilometers, determining the atmospheric conditions within this range. This model provides a critical tool for understanding the operating environment of jet engines and assessing their performance. The primary objective of ISA is to determine how atmospheric variables such as air pressure, temperature, and density change with altitude.

Like ICAO's ISA, ISO's ISA also considers how variables in the atmosphere change with altitude. Specifically, ISO ISA predicts a change with every 1000 feet (or 304.8 meters) increase in altitude, starting from standard atmospheric conditions at sea level. These changes are crucial for understanding how significant variables such as temperature, pressure, and density in the atmosphere affect the performance of jet engines.

Due to its critical role in determining the performance of jet engines, the ISO ISA model forms the basis for many standards used in the aviation industry. These standards are employed in various fields such as aircraft design, air traffic management, and flight planning. Therefore, the standardization of atmospheric conditions by ISO ISA lays the foundation for safe and efficient flights across the jet engine and aviation industry.

3.) ARINC 653: ARINC 653 is a widely used standard for evaluating aircraft engine performance, providing a crucial reference point for understanding the operating environment of jet engines. This standard standardizes the atmosphere within the altitude range from sea level up to 25 kilometers, determining the atmospheric conditions within this range. ARINC 653 is particularly prevalent in the aviation industry for predicting and optimizing engine performance.

The primary objective of ARINC 653 is to understand the effects of atmospheric variables on jet engine performance. This standard considers how significant variables such as temperature, pressure, and density in the atmosphere change with altitude. Understanding the impact of these variables on engine performance plays a significant role in various areas such as aircraft design, air traffic management, and flight planning.

ARINC 653 forms the basis for many analytical and computational methods used in evaluating and predicting jet engine performance. This standard provides essential information on engine performance to engine manufacturers, airlines, and civil aviation authorities. Therefore, the standardization of atmospheric conditions by ARINC 653 is critically important for ensuring the safe and efficient operation of aircraft engines.

Adhering to appropriate air standards is of critical importance to achieve optimal performance of turbojet engines. These standards directly influence the reliability, efficiency, and performance of engines and play a vital role in the development of safe and efficient flight systems in the aviation industry. Suitable air standards ensure that engines operate stably and reliably under various operating conditions, thus enhancing aircraft safety and efficiency.

The performance of turbojet engines can vary depending on environmental conditions, making compliance with specific air standards essential. To achieve optimized engine performance, atmospheric variables such as air pressure, temperature, and density must be

carefully monitored and evaluated. Adherence to the correct air standards enhances engine reliability while also optimizing fuel efficiency and flight costs.

Furthermore, compliance with appropriate air standards provides an essential reference point for engine design and performance evaluation. Engine manufacturers, airlines, and civil aviation authorities ensure the maintenance of safe and effective flight systems by assessing engine compliance with weather conditions.

In conclusion, adhering to appropriate air standards in turbojet engines enhances their reliability, efficiency, and performance and plays a fundamental role in the development of safe and efficient flight systems in the aviation industry. Therefore, continuous monitoring and ensuring compliance with air standards in the processes of engine design, testing, and operation are of paramount importance.

2.3. Ideal Air Cycle

2.3.1. Basic principles and thermodynamic analysis of the Brayton cycle

For propelling an aircraft forward in the air, a propulsion system needs to generate thrust force. Gas turbine engine stands as the most prevalent propulsion system in modern aircraft. Turbojets, turbofans, and turboprops are just a few variations of gas turbine engines; however, they share some common components. The main parts of every gas turbine engine are the combustion chamber, compressor, and power turbine driving the compressor. All turbine engines operate on similar thermodynamic principles. (11)

Understanding the operation of a propulsion system requires delving into the fundamental principles of gas thermodynamics. Gases exhibit various properties such as gas pressure (P), temperature (T), mass (m), and volume (V), which can be perceived through empirical observation. Systematic scientific inquiry has elucidated the interrelation of these properties, determining their values that characterize the state of the gas. Thermodynamic processes such as heating or compression induce changes in the values of these state variables, following the primitive principles set forth by thermodynamic laws. The work done by a gas and the heat exchanged within it depend on the initial and final states of the gas and the

mechanism employed to affect the state transition. Successive processes may emerge, each accelerating a change in state but ultimately resulting in the gas returning to its original state. Such a sequence of processes is depicted as a cycle, forming the fundamental framework for understanding the operation of the engine. (11)

In summary, comprehending the functioning of a propulsion system necessitates an in-depth exploration of gas thermodynamics, wherein systematic scientific inquiry illuminates the fundamental principles governing the behavior of gases and their application in engine operation. (11)

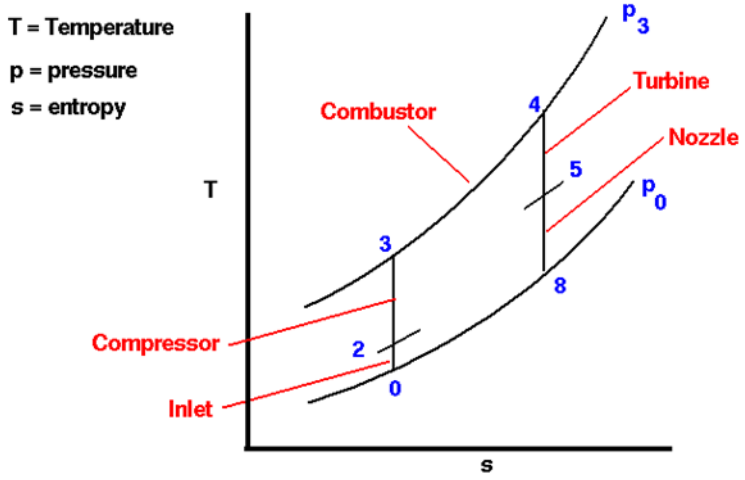


Figure 2. 3: Ideal Brayton cycle T-s diagram.

2.4. Engine Thrust And Efficiency

In the realm of aerospace engineering, the optimization of engine thrust power and efficiency stands as a paramount objective, vital for enhancing both performance and environmental sustainability of aircraft. Recent advancements in engine technology focus on maximizing the thrust-to-weight ratio, a critical factor that influences an aircraft's performance by determining its acceleration capabilities and climb rates. High thrust-to-weight ratios are particularly crucial in combat aircraft, which require rapid maneuverability and short takeoff distances. Research published in the Journal of Aerospace Engineering reveals that innovations in materials and design—such as lighter, more heat-resistant composites and aerodynamically optimized shapes—are pivotal in achieving higher thrust outputs while maintaining or reducing engine weight. (12)

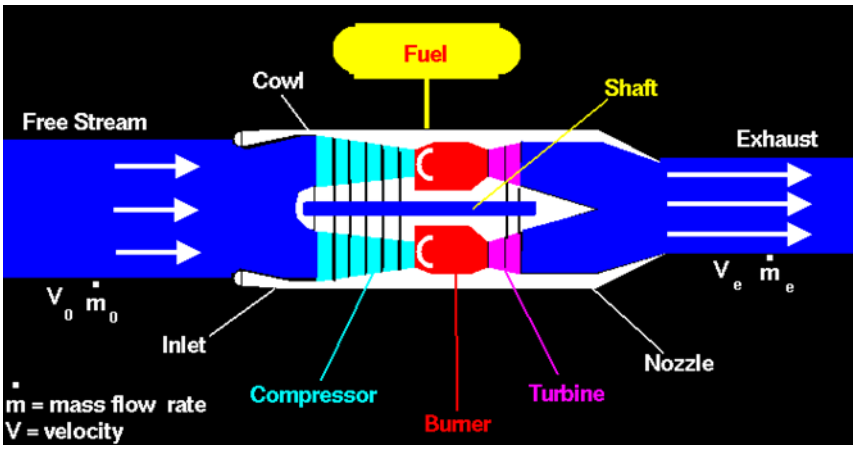


Figure 2. 4: Flow of the stream through the turbojet engine

Furthermore, the efficiency of jet engines is significantly influenced by their operational parameters, including the bypass ratio and the combustion efficiency. Studies in the field, such as those found in the AIAA Journal, have demonstrated that engines with higher bypass ratios tend to exhibit reduced fuel consumption and lower noise levels, making them more suitable for commercial aviation where operational costs and environmental regulations are key concerns. Conversely, military applications often prioritize lower bypass ratios to achieve higher speeds and better thrust for air-to-air and air-to-ground combat, underscoring the need for a balanced approach in engine design that considers the specific requirements of different aircraft. (12)

The interplay between thrust efficiency and the thermodynamic cycle of jet engines is another area of intensive study. Enhancements in the efficiency of components such as compressors and turbines directly affect the overall thermal efficiency of the engine. According to research from MIT, recent developments aim at increasing the pressure and temperature limits of these components, thereby improving the Brayton cycle efficiency and reducing the specific fuel consumption of the engine. This not only leads to better performance but also contributes to the reduction of CO_2 emissions, aligning with global efforts towards more sustainable aviation. (12)

In conclusion, the ongoing research and development in jet engine technology are crucial for the advancement of both military and commercial aviation. The complex balance between thrust power, engine efficiency, and environmental impact continues to drive innovation in this field, promising more efficient, powerful, and eco-friendly aircraft for the

future. Academic contributions are indispensable in pushing the boundaries of what is technically feasible, providing a foundation for future breakthroughs in aerospace technology. (12)

2.5 Thermodynamic Performance Parameters

Turbojet engines are a significant type of engine that compresses air, mixes it with fuel, and then ignites it to produce thrust. Widely used in the aviation industry, these engines enable aircraft to accelerate, travel, and ascend to high altitudes. However, the importance of studying the efficiency, reliability, and environmental impacts of these engines is increasingly recognized. This thesis aims to comprehensively examine and understand the thermodynamic cycle and performance of turbojet engines. (13)

Importance of Thermodynamic Performance:

A detailed analysis of the thermodynamic performance of turbojet engines yields a series of significant outcomes. This analysis provides a comprehensive understanding of the engines' efficiency, energy conversion processes, and effectiveness in generating thrust. Furthermore, while examining the effects on the reliability and durability of the engines, environmental impacts are also evaluated. These analyses offer fundamental insights for the design, development, and optimization of turbojet engines used in the aviation industry. (13)

Thermodynamic Performance Parameters:

Turbojet engines utilize a set of critical parameters to evaluate their thermodynamic performance. These parameters serve as the primary criteria for determining the performance of the engines. Here are some of these parameters: (14)

1) Thermal Efficiency (η): Thermal efficiency (η), which plays a critical role in determining the thermodynamic performance of turbojet engines, indicates how much of the fuel energy input is converted into thrust. High thermal efficiency can result in reduced fuel consumption and lower emissions, thus reducing the environmental impact of engines. Thermal efficiency is a crucial indicator for measuring the effectiveness of the energy conversion process in a turbojet engine, and high thermal efficiency is associated with efficient fuel consumption and low emission levels. (14)

Low thermal efficiency typically leads to increased fuel consumption and, consequently, higher operating costs. Additionally, low efficiency can increase the environmental impact of the engine, particularly in terms of emissions and carbon footprint. Therefore, improving the thermal efficiency of turbojet engines is essential for environmental sustainability and economic efficiency. (14)

2) Mechanical Efficiency (η_m): Mechanical efficiency determines the mechanical power the engine produces in proportion to the thermal energy it receives. This affects the overall efficiency of the engine, taking into account internal mechanical friction and other losses. High mechanical efficiency indicates that more thermal energy is converted into mechanical power, improving overall performance. Low mechanical efficiency indicates that thermal energy causes losses and reduces performance. Understanding and improving mechanical efficiency is important to improve the performance and environmental impact of turbojet engines. Therefore, strategies to increase mechanical efficiency constitute an important area of research in the development of turbojet engines. (14)

3) Maximum Thermal Efficiency (η_{max}): Carnot's theorem defines the highest thermal efficiency for an ideal thermodynamic cycle. This theorem determines the limits of maximum efficiency in thermodynamic systems and provides a guide for engines. For real engines, it is possible to approach the maximum thermal efficiency defined by this theorem; However, in practice this value cannot be fully achieved. (14)

Turbojet engines operate at an efficiency approaching Carnot efficiency. However, Carnot efficiency cannot be fully achieved due to realistic processes within the engine, internal mechanical friction, heat losses and other factors. This is an important factor limiting the practical efficiency of engines. (14)

4) Fuel Air Ratio (F/A): Fuel-air ratio refers to the ratio of the amount of fuel supplied to the engine's combustion chamber and the amount of air entering it. This ratio is a critical parameter of the engine combustion process and directly affects the performance of the engine. The optimal fuel-air ratio represents a balance that provides the best performance. (14)

In turbojet engines, correct adjustment of the fuel-air ratio has a significant impact on the engine's efficiency, thrust and emissions. Correct determination of this ratio allows the engine to minimize fuel consumption while producing maximum thrust. It also plays a critical role in controlling emissions generated during the combustion process. (14)

5) Thrust(T): Gas velocity and flow amount are critical parameters that determine the force produced by the gases exiting the engine. These parameters directly affect the speed and acceleration of the aircraft. (14)

In turbojet engines, accurate control of gas speed and flow rate plays a vital role in determining the performance of the aircraft. Optimizing these parameters helps minimize fuel consumption while maximizing the thrust provided by the engine. Additionally, the correct gas velocity and flow amount increases the aerodynamic stability and maneuverability of the aircraft. (14)

6) Compression Ratio (CPR): The compression ratio expresses the ratio of air pressure absorbed by the compressor to the final pressure. A higher compression ratio ensures higher thermal efficiency and thrust production. (14)

In turbojet engines, proper adjustment of the compression ratio is a critical factor in determining engine performance. A higher compression ratio allows the engine to intake more air, making the combustion process more efficient. Consequently, higher thermal efficiency is achieved, resulting in increased thrust production. (14)

7) Exhaust Gas Temperature (T_8): Exhaust temperature refers to the temperature of the gases formed during the exhaust phase of the engine as a result of combustion. This temperature has a significant impact on the thermal efficiency and performance of the engine.

In turbojet engines, proper control of exhaust temperature is a critical factor for the efficiency and performance of the engine. A high exhaust temperature can reduce the thermal efficiency of the engine and lead to overheating issues. Conversely, a low exhaust temperature can decrease thrust and adversely affect the performance of the engine. (14)

3. CALCULATIONS AND RESULTS

3.1. General Formulazation

$$c_p = a + (b * T) + (c * T * T) + (d * T * T * T) \quad \frac{kJ}{kmol.K} \tag{3.1}$$

The air mass was retrieved from the CoolProp library.

$$c_p(kg) = \frac{c_p(mol)}{m_{air}} \quad \frac{kJ}{kg.K} \tag{3.2}$$

Increase in temperature at the air inlet:

$$\Delta T_1 = \frac{{V_1}^2}{2 \times C_p} \tag{3.3}$$

For the inlet temperature:

$$T_1 = T_A + \frac{{V_1}^2}{2 \times c_p} \tag{3.4}$$

ΔT_1, is the temperature after isentropic compression to P_1. ΔT, can be related to T_1, by introducing an isentropic efficiency η_1 defined as:

$$\frac{P_1}{P_A} = \left(\frac{\Delta T_1}{\Delta T_A}\right)^{\frac{\gamma}{\gamma-1}} \tag{3.5}$$

It follows that:

$$\Delta T_1 - T_A = \eta_1 + \frac{{V_1}^2}{2 \times c_p} \tag{3.6}$$

Thus, η_1 can be interpreted as the portion of the inlet dynamic temperature utilized for isentropic compression in the intake. The intake pressure ratio can subsequently be determined using equation (5):

$$\frac{P_1}{P_A} = \left(1 + \eta_1 + \frac{{V_1}^2}{2 \times c_p \times T_A}\right)^{\frac{\gamma}{\gamma-1}} \tag{3.7}$$

The pressure after intake:

$$P_1 = P_A \times \left(1 + \frac{\gamma - 1}{2} M^2\right)^{\frac{\gamma}{\gamma-1}} \tag{3.8}$$

Using compression ratio pressure at the outlet of the compresssor can be found:

$$P_2 = P_1 \times CPR \tag{3.9}$$

Using same ratio we can find temperature difference at the compressor outlet and inlet (compressor work) and also the temperature at the outlet of the compressor:

$$T_2 - T_1 = \frac{T_1}{\eta_C} \times \left[\left(\frac{P_2}{P_1}^{\frac{\gamma}{\gamma-1}} - 1\right)\right] \tag{3.10}$$

$$T_2 = T_1 + \left[\left(\frac{P_2}{P_1}^{\frac{\gamma}{\gamma-1}} - 1\right)\right] \tag{3.11}$$

The turbine work is:

$$W_T = \frac{W_C}{\eta_M} \tag{3.12}$$

Using the specific heat at constant pressure (air) C_{PA} and, specific heat at constant pressure (combustion gas) C_{PG} the difference through the turbine and the oulet temperature of the turbine can be found as follows:

$$T_3 - T_4 = \frac{C_{PA} \times (T_2 - T_1)}{C_{PG} \times \eta_M} \tag{3.13}$$

Temperature inlet of the turbine accepted as 1450 C°.

$$T_4 = T_3 - \frac{C_{PA} \times (T_2 - T_1)}{C_{PG} \times \eta_M} \tag{3.14}$$

The loss of pressure in the combustion chamber can be denoted as ΔP_B. Then the pressure at the outlet of the combustion chamber can be defined as:

$$P_3 = P_2 \times \left(1 - \frac{\Delta P_B}{P_2}\right) \tag{3.15}$$

The temperature at the turbine outlet with turbine losses:

$$\Delta T_4 = T_3 - \frac{1}{\eta_T} \times (T_3 - T_4) \tag{3.16}$$

Then pressure at the turbine outlet can be calculated:

$$P_4 = P_3 \times \left(\frac{\Delta T_4}{T_3}\right)^{\frac{\gamma_G}{\gamma_G - 1}} \tag{3.17}$$

The nozzle pressure ratio can be defined as:

$$\frac{P_4}{P_A} \tag{3.18}$$

Also we can define critical ratio like: (15)

$$\frac{P_4}{P_C} = \frac{1}{\left[1 - \frac{1}{\eta_N} \times \left(\frac{\gamma_G - 1}{\gamma_G + 1}\right)\right]^{\frac{\gamma_G}{\gamma_G - 1}}} \tag{3.19}$$

To calculate afterburner variables we should define fuel-to-air ratio and afterburner fuel-to-air ratio:

$$f = \frac{C_{Pg}T_3 - C_{Pa}T_1}{\eta_b Q_R - C_{Pg}T_3} \tag{3.20}$$

To find temperature of the turbine outlet with losses we can use following equation:

$$T_{4,with\ losses} = T_3 - \frac{1}{\eta_t}(T_3 - T_4) \tag{3.21}$$

For Operative Afterburner Calculation:

Pressure outlet of the afterburner is equal to pressure outlet of the turbine.

$$\mathrm{P_6} = \mathrm{P_4} \tag{3.22}$$

The temperature outlet of the afterburner taken as 1900 C°.

Critical pressure also can be found as follows:

$$\frac{\mathrm{P_6}}{\mathrm{P_c}} = \frac{1}{\left[1 - \frac{1}{\eta_\mathrm{n}}\left(\frac{\gamma_\mathrm{g} - 1}{\gamma_\mathrm{g} + 1}\right)\right]^{\frac{\gamma_\mathrm{g}}{\gamma_\mathrm{g}-1}}} \tag{3.23}$$

The chocked condition checked with equations 3.24 and 3.25. If 3.24 is valid then noozle chocked if not- 3.25 is valid- then nozzle unchocked.

$$\frac{P_6}{P_A} > \frac{P_6}{P_C} \tag{3.24}$$

$$\frac{P_6}{P_A} < \frac{P_6}{P_C} \tag{3.25}$$

For chocked condition temperature, velocity, and pressure variables can be found as follows:

$$\frac{T_6}{T_8} = \frac{\gamma_h + 1}{2} \tag{3.26}$$

$$\mathrm{V_8} = \sqrt{\mathrm{R}\gamma_\mathrm{g}(\mathrm{T_8})} \tag{3.27}$$

$$P_8 = P_C \tag{3.28}$$

For unchoked condition temperature, velocity, and pressure variables can be found as follows:

$$T_8 = T_6 \tag{3.29}$$

$$P_8 = P_A \tag{3.30}$$

$$V_8 = 2C_{PG}(T_6 - T_8) \tag{3.31}$$

$$\mathrm{f_{ab}} = \frac{(1+\mathrm{f})\left(\mathrm{Cp_g T_6} - \mathrm{Cp_g T_4}\right)}{\eta_{\mathrm{ab}} \mathrm{Q_R} - \mathrm{Cp_g T_6}} \tag{3.32}$$

For Inoperative Afterburner Calculation:

$$P_6 = P_4 \tag{3.33}$$

$$T_6 = T_{4,with\ losses} \tag{3.34}$$

The chocking condition checked again with equations 3.24 and 3.25.

For chocked condition temperature, velocity, and pressure variables can be found as follows:

$$\frac{T_6}{T_8} = \frac{\gamma_h + 1}{2} \tag{3.35}$$

$$\mathrm{V_8} = \sqrt{\mathrm{R\gamma_g(T_8)}} \tag{3.36}$$

$$P_8 = P_C \tag{3.37}$$

For unchoked condition temperature, velocity, and pressure variables can be found as follows:

$$P_8 = P_A \tag{3.38}$$

$$T_8 = T_{4,with\ losses} \tag{3.39}$$

$$V_8 = 2C_{PG}\left(T_{4,with\ losses} - T_8\right) \tag{3.40}$$

$$f_{ab} = 0 \tag{3.41}$$

Density and the mass flow rate outlet of the nozzle can be calculated as follows:

$$\rho = \frac{P_6}{RT_8} \tag{3.42}$$

$$\dot{m}_8 = \rho A V_8 \tag{3.43}$$

Net thrust and specific thrust can be found by using these formulas:

$$T = \dot{m}_8 V_8 - \dot{m}_a V_a \tag{3.44}$$

$$F_S = \frac{T}{\dot{m}_a} \tag{3.45}$$

While the thrust-specific fuel consumption (TSFC) is expressed as: (16)

$$\text{TSFC} = \frac{\dot{m}_8 + \dot{m}_a}{\text{T}} \tag{3.46}$$

Enthalpy of the component's entrace and exit can be fond by this general formula:

$$h = c_p(T_X)T_X \ \ for\ X = 1{,}2 \ldots{,}8 \tag{3.47}$$

Also we can generate a general formula for entropy, density and specific volume:

$$s_{X+1} = s_X + c_p ln\frac{T_{X+1}}{T_X} - Rln\frac{P_{X+1}}{P_X} \tag{3.48}$$

The first entropy of the formula taken from the 'Thermodynamics An Engineering Approach' s appendix. (17)

$$v_X = \frac{1}{\rho_X}\ for\ X = 1{,}2 \ldots{,}8 \tag{3.49}$$

3.2. Explanation of the Code

3.2.1. Introduction

In today's aviation industry, the continuous development of engine technologies plays a critical role in terms of flight performance and fuel efficiency. In this context, accurate

modeling and analysis of thermodynamic cycles of turbojet engines is a fundamental step in engineering design. This report presents a Python application developed for modeling thermodynamic cycles of turbojet engines.

In this study, the user interface was designed via PyQt Designer and the thermodynamic cycle of the turbojet engine was modeled using the Python programming language. The designed application has been used to evaluate the performance of the engine and analyze it under thermodynamic conditions at various operating conditions. The purpose of the developed application is to use a mathematical model and present the results obtained in detail.

Python writing of the mathematical foundations of the thermodynamic cycle of the turbojet engine, the design and use of the application will be explained, how the results are shown in the application and information about how this is done is given.

First, the interface of the application was designed using Qt Designer. The program designed in the Qt Designer program comes out with the .ui extension. This design code was transferred to the Python environment by writing the ui to py conversion code. The data to be received from the user in the main file of the application was captured with the code we wrote, and the connection between the interface and the background was created. Then, the functions required for our calculation (for example, the specific heat at constant pressure function) were created. Subsequently, a calculation code was written using thermodynamic cycles. The results are displayed in the program interface and by exporting to Excel. In addition, the results are shown graphically.

3.2.2. Design of GUI with Qt Designer

The Python application developed for modeling the thermodynamic cycles of turbojet engines was meticulously designed using PyQt Designer. During this design process, various user interface elements offered by Qt Designer were used. Tools like labels, qlineedit, combobox, widgets and checkboxes have been chosen to create every detail of the user interface.

Each interface element has been strategically selected and placed to facilitate users' interaction with the application. This careful selection and deployment process allows users to enter the relevant parameters into the application, select the desired options, and visualize the results of the thermodynamic cycle of the turbojet engine in a clear and precise way. The design phase is shown in ***Figure 3. 1*****Hata! Başvuru kaynağı bulunamadı.**.

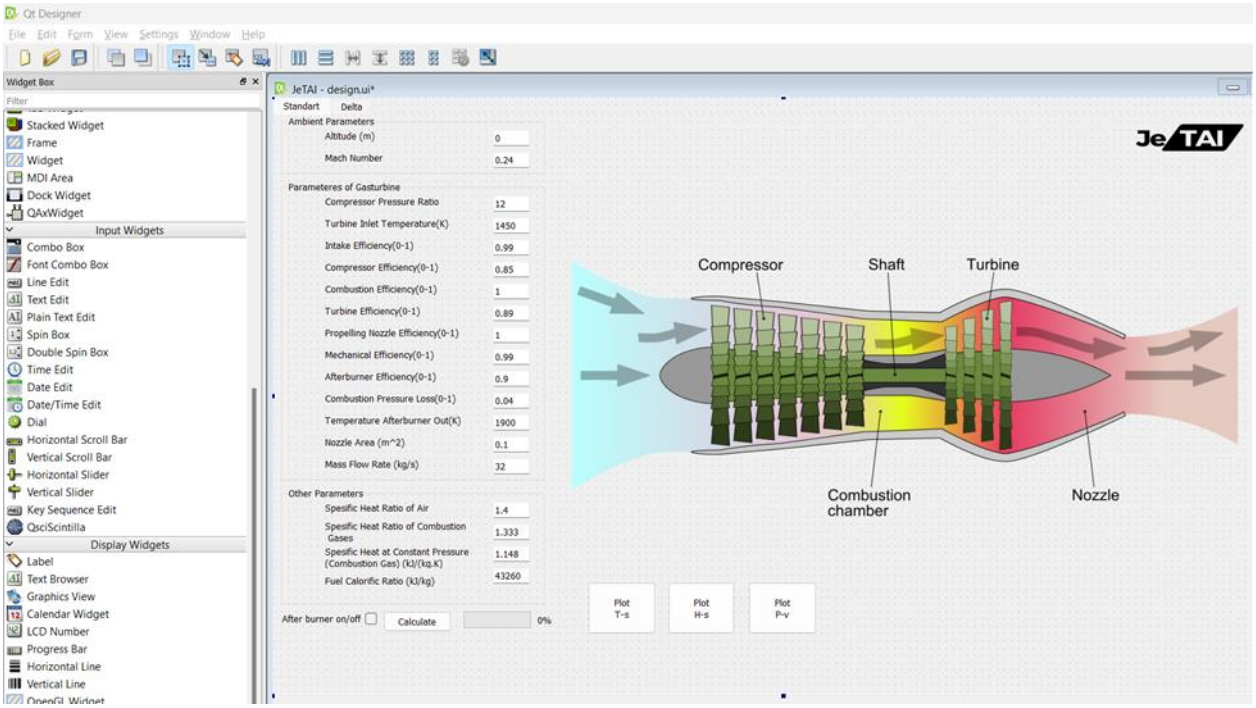


Figure 3. 1: Qt designer interface design phase

Qt designer output appears as ui. In order to use the design, we made in Python, we need to convert it to the py extension. The code we translated this into is shown in ***Figure 3. 2***.

```
from PyQt5 import uic

with open("designn.py","w",encoding="utf-8")as fout:
    uic.compileUi("design.ui",fout)
```

Figure 3. 2: Code used to convert from ui. to py

3.2.3. Auxiliary functions

Auxiliary functions needed to perform our main calculation have been created.

3.2.3.1. Specific Heat at Constant Pressure Function

The specific heat at constant pressure value changes with temperature. In order to find this value, which varies according to temperatures, at different temperatures in the engine, the code function shown in ***Figure 3. 3*** was written.

```
import CoolProp.CoolProp as CoolProp
def cp(T):
    a = 28.11
    b = 0.1967 * 10 ** (-2)
    c = 0.4802 * 10 ** (-5)
    d = -1.966 * 10 ** (-9)
    cp_mol = a + (b * T) + (c * T * T) + (d * T * T * T)  # kJ/kmol.K , error max=0.72 , error avg.=0.33

    m_air = CoolProp.PropsSI("M", "P", 101325, "T", T, "AIR") * 1000  # kg/kmol
    cp_kg = cp_mol / m_air  # kJ/kg.K
    return cp_kg
```

Figure 3. 3: Codes of specific heat function at constant pressure

3.2.3.2. Finding entropy at the relevant temperature

The code of our function to find entropy for the relevant temperature using the thermodynamic properties table is shown in ***Figure 3. 4***.

```python
import pandas as pd

def find_entropy_with_temperature(target_temperature):
    df = pd.read_excel("table.xlsx")
    closest_temps = df['Temperature'].tolist()
    closest_temps.sort()

    lower_temp = max(t for t in closest_temps if t <= target_temperature)
    upper_temp = min(t for t in closest_temps if t >= target_temperature)

    closest_entropies = df[df['Temperature'].isin([lower_temp, upper_temp])].sort_values(by='Temperature')[
        'Entropy'].tolist()

    if len(closest_entropies) > 1:
        upper_entropy = closest_entropies[1]
        lower_entropy = closest_entropies[0]

        x = (upper_entropy - lower_entropy) / (upper_temp - lower_temp) * (
                target_temperature - lower_temp) + lower_entropy
    else:
        x = closest_entropies[0]
    return x
```

Figure 3. 4: Codes of finding entropy at the relevant temperatures

3.2.4. Calculation Codes

The equations in which the thermodynamic cycle is calculated are shown under the heading Calculation. All equations shown under the Calculation heading are written as python code under the calculation function.

The equations in which the thermodynamic cycle is calculated are shown under the heading Calculation. All equations shown under the Calculation heading are written as python code under the calculation function. As an example, a part of the code is shown in ***Figure 3. 5***.

```python
temperature_turbine_outlet = temperature_turbine_inlet - c_pa * (
        temperature_compressor_outlet - temperature_compressor_inlet) / (c_pg * n_m)
pressure_turbine_inlet = pressure_compressor_outlet * (
        1 - combustion_pressure_loss / pressure_compressor_outlet)
temperature_turbine_outlet_with_losses = temperature_turbine_inlet - 1 / n_t * (
        temperature_turbine_inlet - temperature_turbine_outlet)
pressure_turbine_outlet = pressure_turbine_inlet * (
        temperature_turbine_outlet_with_losses / temperature_turbine_inlet) ** (
                                specific_heat_ratio_combustion_gas / (
                                specific_heat_ratio_combustion_gas - 1))
fuel_air_ratio_combustion = (c_pg * temperature_turbine_inlet - c_pa * temperature_compressor_inlet) / (
        n_b * fuel_calorific - c_pg * temperature_turbine_inlet)

p_critical = pressure_turbine_outlet * (1 - 1 / n_n * (
            (specific_heat_ratio_combustion_gas - 1) / (specific_heat_ratio_combustion_gas + 1))) ** (
                        specific_heat_ratio_combustion_gas / (specific_heat_ratio_combustion_gas - 1))
```

Figure 3. 5: Example of calculation codes

3.2.5. Main Function

The design file has been imported into our main file. Our auxiliary functions and calculation function were also imported into our main file and used where necessary. Using the functions shown in ***Figure 3. 6***, operations such as showing error messages, drawing graphs, making calculations, deleting the graph, and saving the graph were performed. While the Klein function is connected to the button used to make calculations on our first screen, the delta function is connected to the button used to make calculations on our second screen.

Figure 3. 6: Main file functions

We connect the buttons in our design codes to the functions in our main file together with the codes in ***Figure 3. 7***.

```
def __init__(self):
    super(MyApp, self).__init__()
    self.setupUi(self)
    self.init_widget()
    self.pushButton.clicked.connect(self.klein)
    self.pushButton_2.clicked.connect(self.delta)
    self.pushButton_3.clicked.connect(self.plot_widget)
    self.pushButton_4.clicked.connect(self.t_s_plot)
    self.pushButton_5.clicked.connect(self.h_s_plot)
    self.pushButton_6.clicked.connect(self.p_v_plot)
    self.pushButton_7.clicked.connect(self.clear)
    self.pushButton_8.clicked.connect(self.save_plot)
    self.matplotlibwidget.canvas.mpl_connect('motion_notify_event', self.hover)
    self.checkBox.stateChanged.connect(self.state)
```

Figure 3. 7: Link codes between main file and design file

The program starts working when the main function file is run.

All codes are written as described before; no values are shown on the main screen of our application at first. When you enter the inputs and press the calculate button, the results in the application are printed on the screen and transferred to the excel file. In addition, whether Afterburner is active or not can be adjusted with the checkbox to the left of the calculate button.

Figure 3. 8 shows the application when it is first opened.

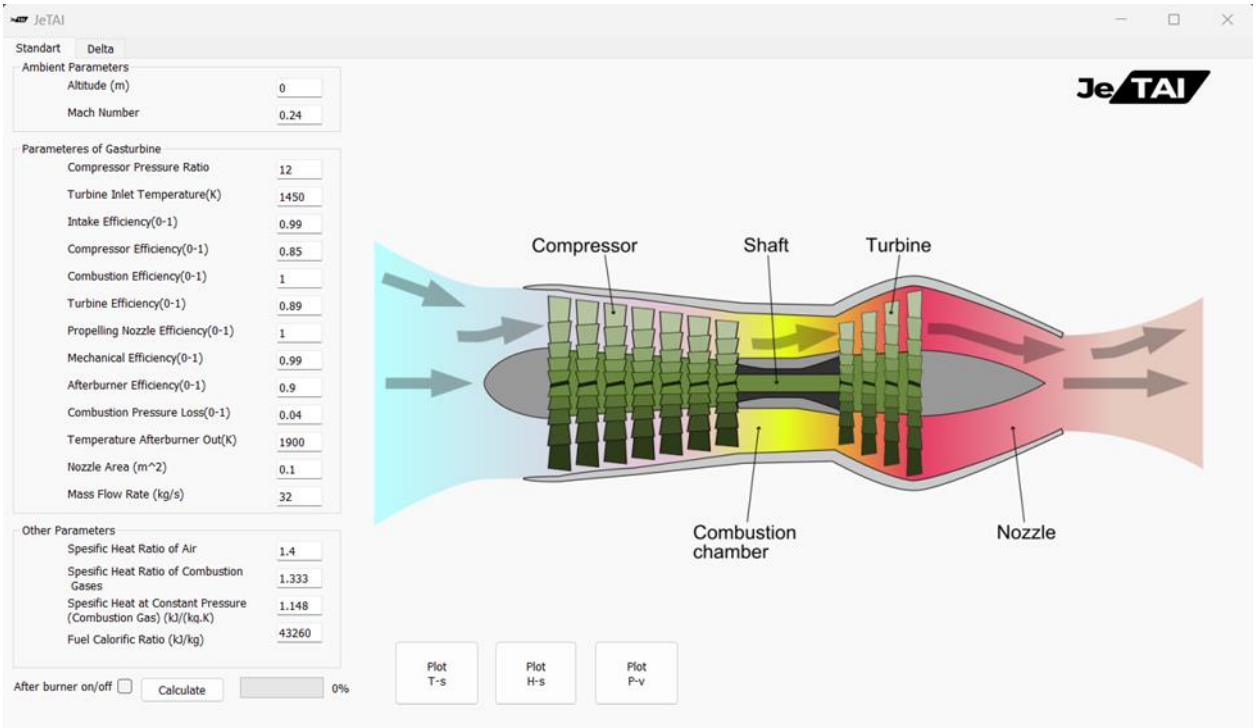


Figure 3. 8: The view of program when it is first opened

When you press the Calculate button, the results are shown on the screen as shown in ***Figure 3. 9***.

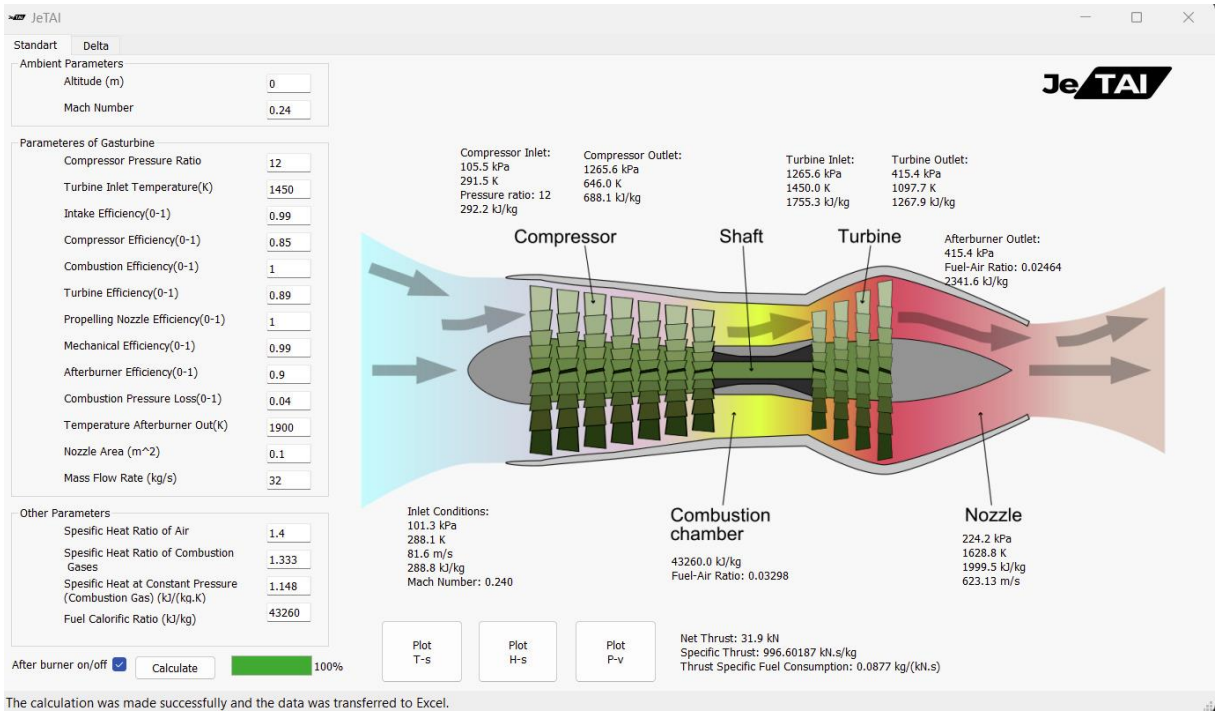


Figure 3. 9: When we press the button, the program screen appears

After pressing the button, the calculated data is transferred to Excel as shown in ***Figure 3. 10***.

	A	B	C	D	E	F	G	H	I
1	**Stations**	**Temperature (K)**	**Pressure (kPa)**	**Enthalpy (kJ/kg)**	**Entropy (kJ/kgK)**		**Velocity m/s**		**Net Thrust (kN)**
2	Ambient	288.15	101.325	288.7767888	1.6615561		81.64787168		31.89125988
3	Compressor Inlet	291.4666045	105.4695935	292.2426102	1.661308885				
4	Compressor Outlet	646.0055568	1265.635121	688.1396342	1.769278363				
5	Turbine Inlet	1450	1265.595121	1755.328569	2.695510232				
6	Turbine Outlet	1097.740326	415.3921531	1267.892675	2.68526042				
7	Afterburner Outlet	1900	415.3921531	2341.605464	3.352438411				
8	Nozzle Outlet	1628.804115	224.2427758	688.1396342	3.339392612		623.133219		
9									

Figure 3. 10: Data exported to Excel

By using the plot buttons on the screen, the graphs shown in ***Figure 3. 11*** can be obtained.

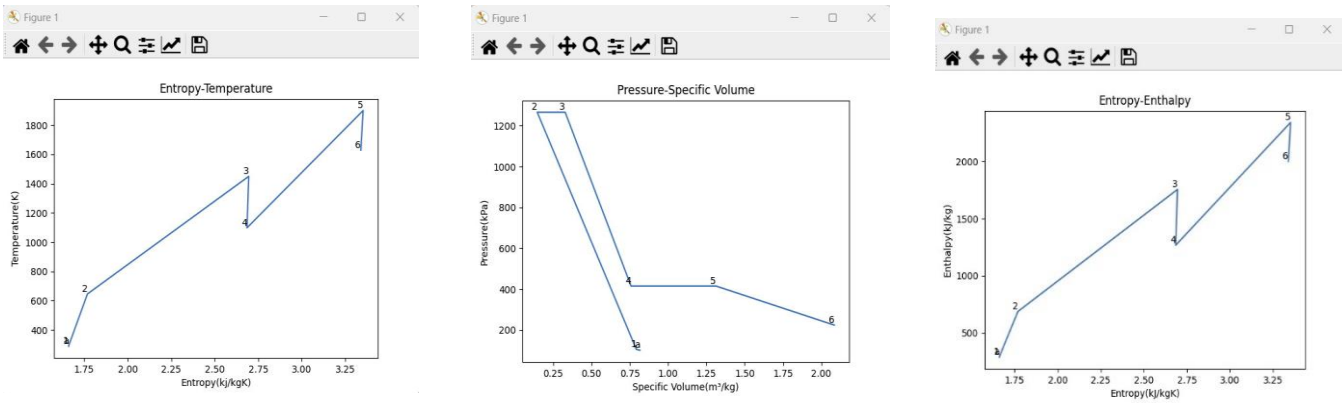


Figure 3. 11: Graphs obtained as a result of the calculation

Calculations can be made for more than one situation on the delta screen, which we call the second screen. For example, we can see the outputs of the engine by increasing the altitude by 500 meters from 0 to 10000 meters. ***Figure 3. 12*** shows the standard version of our screen.

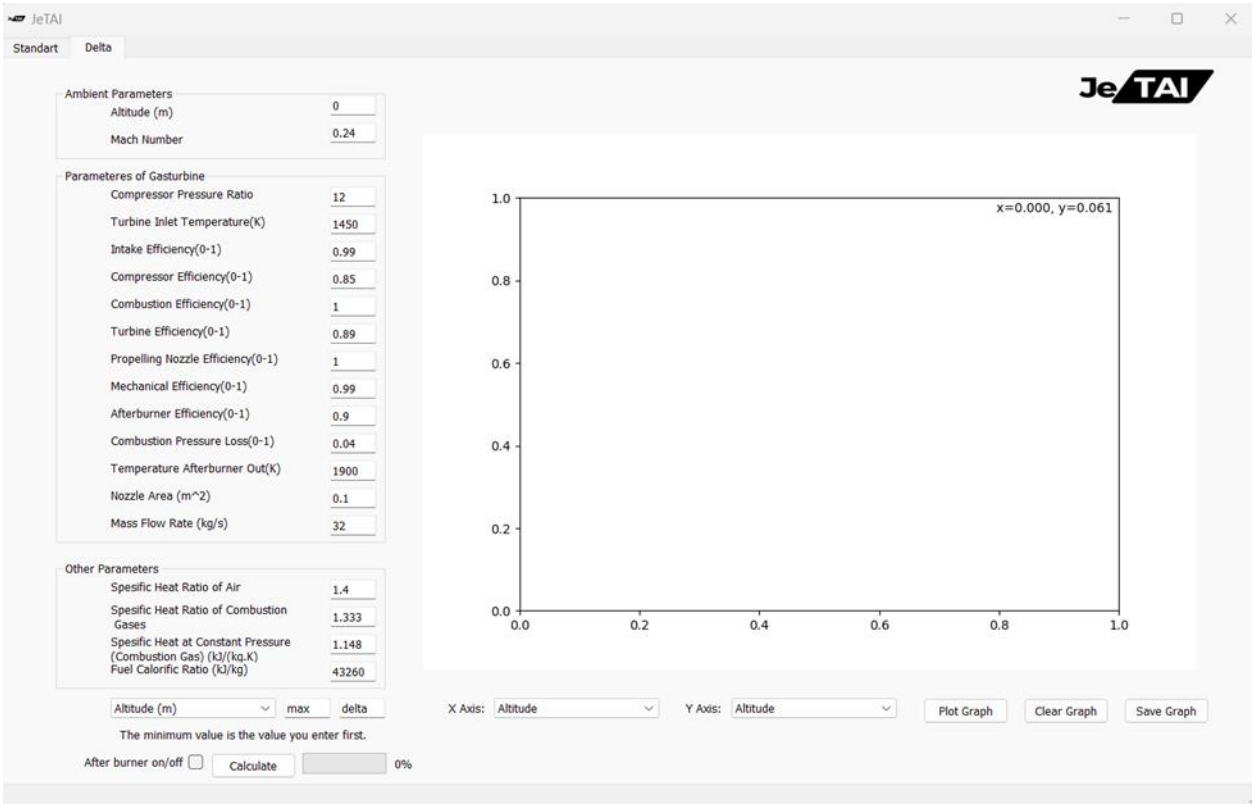


Figure 3. 12: Standard version of delta display

According to the example given, delta is entered as 500, and the maximum altitude is 10,000 meters. When we select the X-axis as altitude and the Y-axis as net thrust and click on the plot button, a graph is plotted as shown in ***Figure 3. 13***.

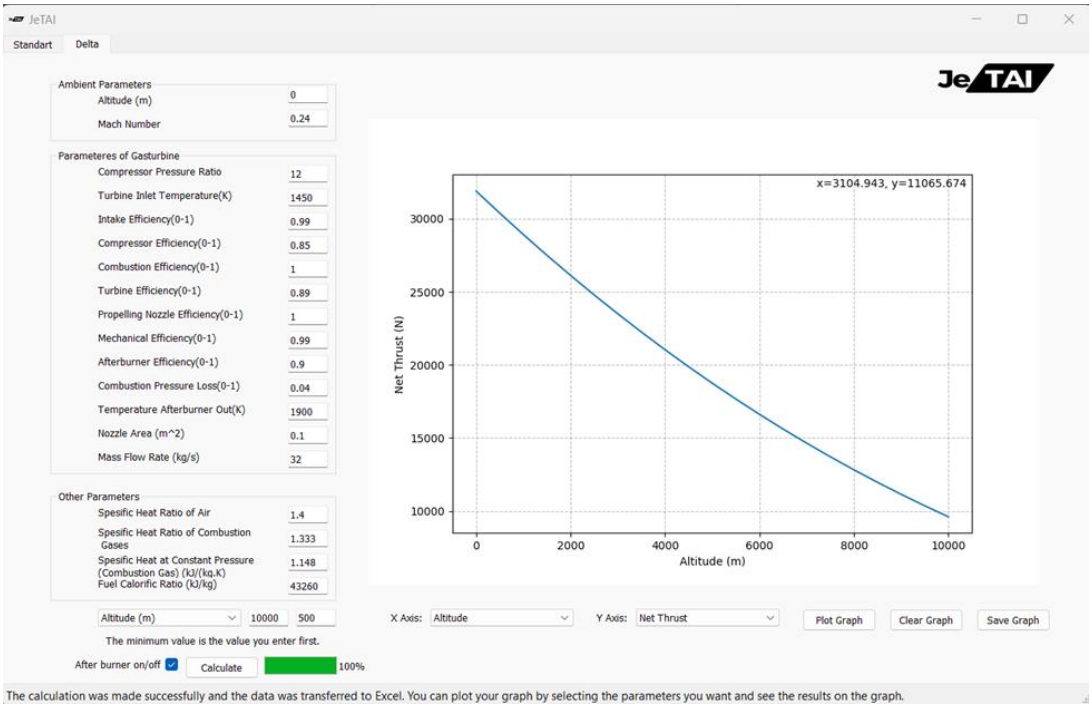


Figure 3. 13: When a transaction is made on the delta screen

In addition, the ability to save the graph drawn on the delta screen using the save button or delete it using the clear button has been added. It is shown in ***Figure 3. 14*** that all output parameters can be selected as X axis and Y axis.

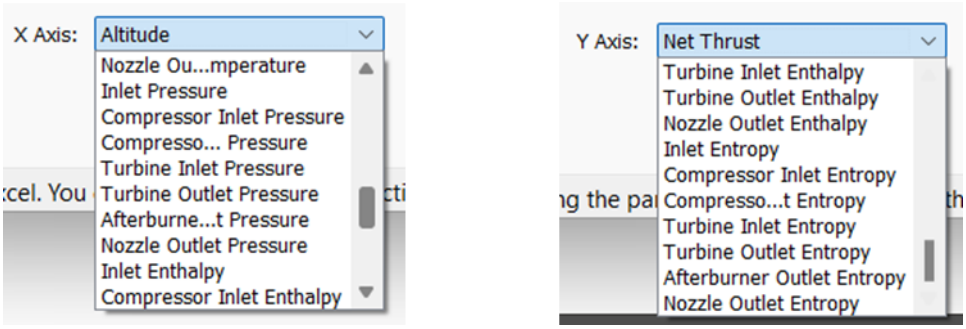


Figure 3. 14: Output selectability on axes

The calculations made on the Delta screen are also transferred to Excel with the calculate button. The appearance of the transferred Excel file is shown in ***Figure 3. 15***.

	A	B	C	D	E	F	G	H	I
1	Altitude (m)	Temperature Ambient (K)	Pressure Ambient (kPa)	Enthalpy Ambient (kJ/kg)	Entropy Ambient (kJ/kgK)	Velocity Ambient m/s	Temperature Compressor Inlet (K)	Temperature Compressor Inlet (K)	Pressure Compressor Inlet (K)
2	0	288.15	101.325	288.7767888	1.6615561	81.64787168	291.4666045	291.4666045	105.4695935
3	500	284.9	95.46073625	285.3843547	1.6501948	81.18611902	288.179197	288.179197	99.36545812
4	1000	281.65	89.87437443	281.9956502	1.6386508	80.72172504	284.8917895	284.8917895	93.55059199
5	1500	278.4	84.55572532	278.6106569	1.626958	80.25464388	281.604382	281.604382	88.01438908
6	2000	275.15	79.4948615	275.2293565	1.61511175	79.78482836	278.3169745	278.3169745	82.74651591
7	2500	271.9	74.68211359	271.8517301	1.6032655	79.31222987	275.0295671	275.0295671	77.73690757
8	3000	268.65	70.10806645	268.477759	1.5912289	78.83679838	271.7421596	271.7421596	72.9757638
9	3500	265.4	65.76355542	265.1074241	1.5789244	78.35848231	268.4547521	268.4547521	68.45354507
10	4000	262.15	61.63966254	261.7407063	1.5666199	77.87722852	265.1673446	265.1673446	64.16096865
11	4500	258.9	57.72771287	258.377586	1.5541559	77.39298219	261.8799372	261.8799372	60.08900476
12	5000	255.65	54.01927069	255.0180438	1.54138015	76.90568681	258.5925297	258.5925297	56.22887262
13	5500	252.4	50.50613584	251.6620599	1.5286044	76.41528404	255.3051222	255.3051222	52.57203665
14	6000	249.15	47.18033995	248.3096143	1.51569095	75.92171367	252.0177147	252.0177147	49.11020255
15	6500	245.9	44.03414281	244.960687	1.5023887	75.42491351	248.7303073	248.7303073	45.8353135
16	7000	242.65	41.06002864	241.6152575	1.48908645	74.92481932	245.4428998	245.4428998	42.73954629
17	7500	239.4	38.25070243	238.2733055	1.4756798	74.42136468	242.1554923	242.1554923	39.81530752
18	8000	236.15	35.5990863	234.9348102	1.46181205	73.91448094	238.8680848	238.8680848	37.05522979
19	8500	232.9	33.09831583	231.5997509	1.4479443	73.40409705	235.5806773	235.5806773	34.45216792
20	9000	229.65	30.74173645	228.2681064	1.4340118	72.89013949	232.2932699	232.2932699	31.99919511
21	9500	226.4	28.52289979	224.9398556	1.4195428	72.37253213	229.0058624	229.0058624	29.68959926
22	10000	223.15	26.4355601	221.614977	1.4050738	71.85119608	225.7184549	225.7184549	27.51687912

Figure 3. 15: Transferring the calculations on the Delta screen to Excel

4. COST ANALYSIS

Conducting a comprehensive cost analysis is crucial for the development and implementation of turbojet engines, as it provides valuable insights into the financial feasibility and potential economic impacts of the project. This analysis involves a detailed examination of both fixed and variable costs, which are taken into account based on specific methods and assumptions used throughout the study. The main components considered in the cost analysis of a turbojet engine include Research and Development (R&D) costs, production costs, operation and maintenance costs, and environmental and regulatory costs.

R&D costs encompass various elements such as design and modeling software expenses, prototype production costs, and testing and verification expenses. These costs are essential for the initial development stages of the turbojet engine, ensuring that the design is viable and meets the required performance criteria. Production costs include material costs, labor costs, and the expenses associated with the equipment and tools used in the manufacturing process. These costs are significant as they directly impact the overall production efficiency and quality of the turbojet engine.

Operation and maintenance costs are also critical components of the cost analysis. These include fuel costs, which are influenced by the engine's efficiency and fuel consumption rates, as well as maintenance and repair costs necessary to ensure the engine's long-term reliability and performance. Additionally, the cost of spare parts must be considered, as these are essential for routine maintenance and unexpected repairs.

Environmental and regulatory costs are increasingly important in today's context, where compliance with stringent environmental regulations is mandatory. These costs include expenses related to emission control technologies and adherence to environmental standards. Such costs are vital for minimizing the environmental impact of the turbojet engine and ensuring that it meets regulatory requirements.

The calculation of these cost components involves explaining their theoretical foundations and the methods used for estimation. For example, fuel consumption rates and associated costs are determined based on engine performance data and projected usage

scenarios. Similarly, material and labor costs are calculated based on current market rates and the specific requirements of the production process.

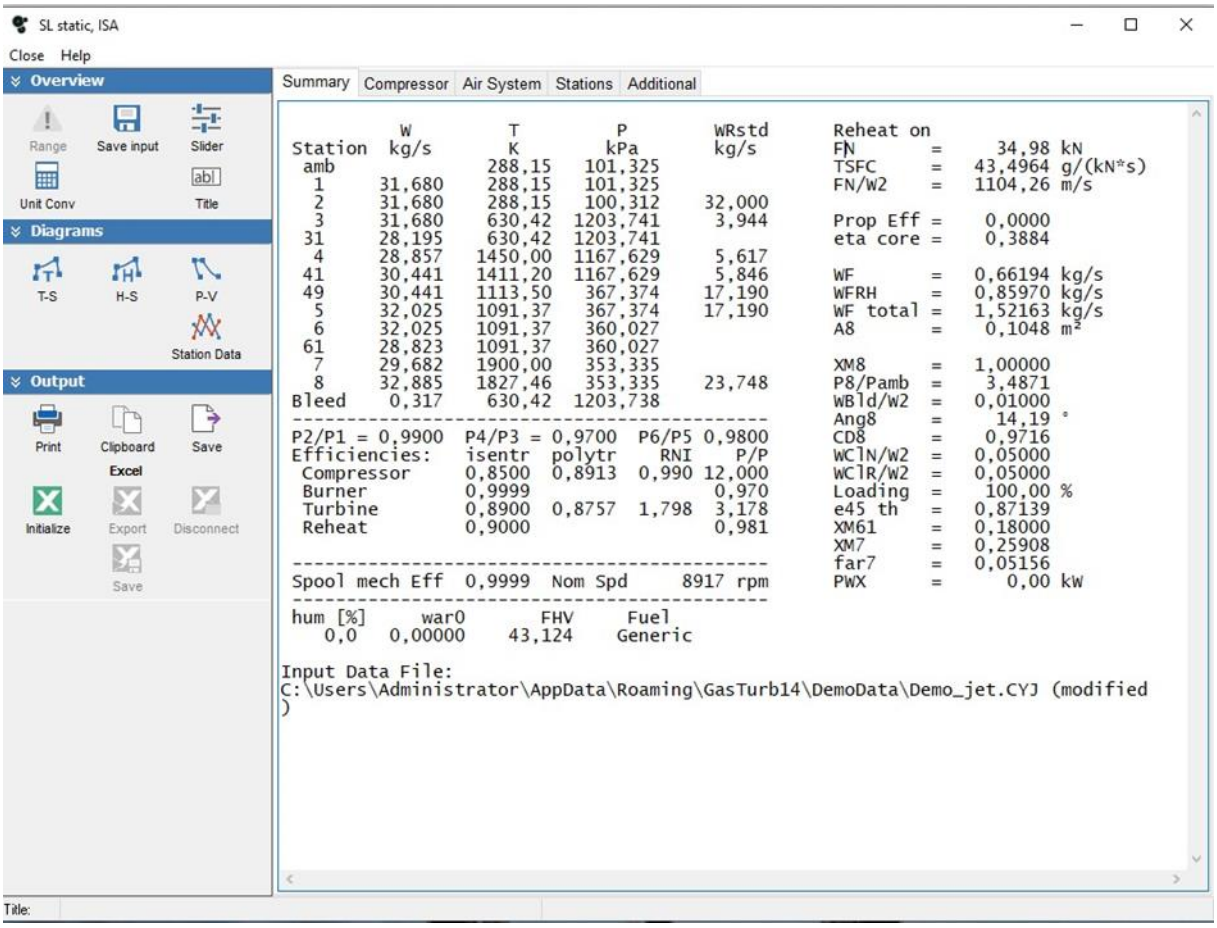
The results of the cost analysis are presented and discussed in terms of their impact on the overall efficiency and sustainability of the turbojet engine. By understanding these costs, it becomes possible to identify areas where cost savings can be achieved and to develop strategies for cost reduction. The general cost analysis provides a summary of the financial implications of developing and implementing turbojet engines, highlighting the importance of managing these costs effectively.

Furthermore, the analysis includes future research suggestions aimed at improving cost efficiency and reducing the financial burden of turbojet engine projects. An example cost analysis table is provided to illustrate the breakdown of costs, including R&D expenses, production materials, labor, fuel costs, maintenance and repair costs, and emission control device costs, culminating in the total cost. This detailed breakdown allows for a thorough understanding of the financial aspects of turbojet engine projects.

In conclusion, this comprehensive cost analysis, with detailed calculations for each cost component, offers valuable insights into the total cost of the turbojet engine and the relative contribution of each component. By addressing these costs strategically, it is possible to enhance the economic viability and overall success of turbojet engine development and implementation.

5. RESULTS AND DISCUSSION

In this study, the thermodynamic cycle of a turbojet engine was modeled using Python, and the results of this model were compared with those obtained using GasTurb software. The model developed with Python simulated the Brayton cycle of the turbojet engine, taking into account various parameters such as pressure, temperature, and specific heat capacities at different stages of the cycle. The primary outputs obtained from the model include thrust, pressure, and temperature values.



Station	W kg/s	T K	P kPa	WRstd kg/s
amb		288,15	101,325	
1	31,680	288,15	101,325	
2	31,680	288,15	100,312	32,000
3	31,680	630,42	1203,741	3,944
31	28,195	630,42	1203,741	
4	28,857	1450,00	1167,629	5,617
41	30,441	1411,20	1167,629	5,846
49	30,441	1113,50	367,374	17,190
5	32,025	1091,37	367,374	17,190
6	32,025	1091,37	360,027	
61	28,823	1091,37	360,027	
7	29,682	1900,00	353,335	
8	32,885	1827,46	353,335	23,748
Bleed	0,317	630,42	1203,738	

P2/P1 = 0,9900 P4/P3 = 0,9700 P6/P5 0,9800

Efficiencies:	isentr	polytr	RNI	P/P
Compressor	0,8500	0,8913	0,990	12,000
Burner	0,9999			0,970
Turbine	0,8900	0,8757	1,798	3,178
Reheat	0,9000			0,981

Spool mech Eff 0,9999 Nom Spd 8917 rpm

hum [%]	war0	FHV	Fuel
0,0	0,00000	43,124	Generic

Input Data File:
C:\Users\Administrator\AppData\Roaming\GasTurb14\DemoData\Demo_jet.CYJ (modified)

Reheat on

FN	=	34,98 kN
TSFC	=	43,4964 g/(kN*s)
FN/W2	=	1104,26 m/s
Prop Eff	=	0,0000
eta core	=	0,3884
WF	=	0,66194 kg/s
WFRH	=	0,85970 kg/s
WF total	=	1,52163 kg/s
A8	=	0,1048 m²
XM8	=	1,00000
P8/Pamb	=	3,4871
WBld/W2	=	0,01000
Ang8	=	14,19 °
CD8	=	0,9716
WClN/W2	=	0,05000
WClR/W2	=	0,05000
Loading	=	100,00 %
e45 th	=	0,87139
XM61	=	0,18000
XM7	=	0,25908
far7	=	0,05156
PWX	=	0,00 kW

Figure 4. 1: Gasturb values.

The Python model calculated the net thrust of the engine under standard operating conditions as 31.9 kN, while the GasTurb software calculated it as 34.98 kN. The 3.08 kN (approximately 9%) difference in thrust values between the Python model and the GasTurb software demonstrates that both methods can accurately simulate the fundamental performance characteristics of the engine. However, it also indicates that the Python model requires further improvement and calibration of certain parameters.

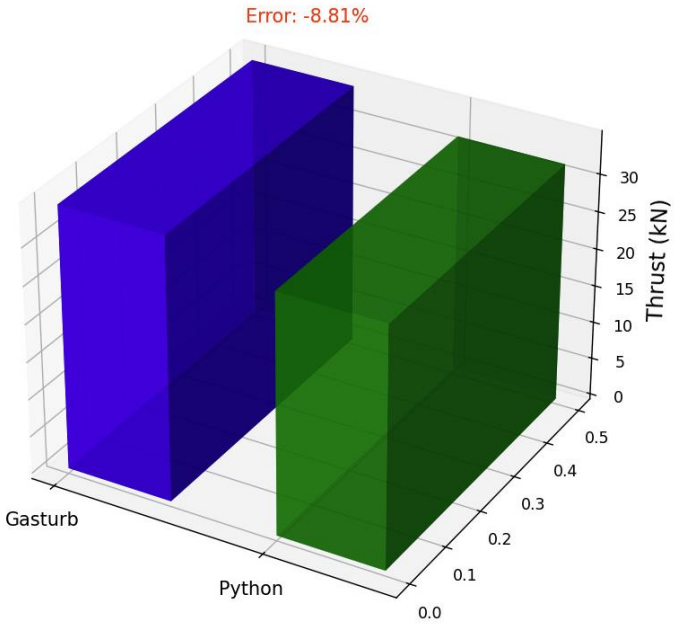


Figure 4. 2: Thrust comparison and error analysis

When comparing pressure values, some discrepancies were observed between the Python model and the GasTurb software. The pressure values for GasTurb were calculated as follows: compressor inlet pressure at 100.312 kPa, compressor outlet pressure at 1203.741 kPa, turbine inlet pressure at 1167.629 kPa, turbine outlet pressure at 367.374 kPa, afterburner outlet pressure at 353.335 kPa, and nozzle outlet pressure at 353.335 kPa. In contrast, the Python model calculated the compressor inlet pressure as 105.470 kPa, compressor outlet pressure as 1265.635 kPa, turbine inlet pressure as 1265.595 kPa, turbine outlet pressure as 415.392 kPa, afterburner outlet pressure as 415.392 kPa, and nozzle outlet pressure as 224.243 kPa.

The discrepancies in pressure values can be attributed to the differences in assumptions and numerical methods between the Python model and the GasTurb software. Critical points such as compressor outlet pressure and turbine outlet pressure significantly affect engine performance and efficiency. The Python model estimated the compressor outlet pressure to be 1265.635 kPa, which is 5.1% higher than the 1203.741 kPa calculated by GasTurb. Similarly, the Python model estimated the turbine outlet pressure to be 415.392 kPa, which is 13.1% higher than the 367.374 kPa calculated by GasTurb. These differences suggest that the Python model predicts higher pressure drops and energy losses.

For the nozzle outlet pressure, the Python model calculated 224.243 kPa, which is 36.5% lower than the 353.335 kPa calculated by GasTurb. This difference indicates that the Python model predicts higher pressure losses in the nozzle, which could affect thrust estimations.

When comparing temperature values, the GasTurb software calculated the temperatures as follows: compressor inlet temperature at 288.15 K, compressor outlet temperature at 630.42 K, turbine inlet temperature at 1411.20 K, turbine outlet temperature at 1091.37 K, and nozzle outlet temperature at 1827.46 K. The Python model, on the other hand, assumed the compressor inlet temperature to be 291.467 K, compressor outlet temperature to be 646.006 K, turbine inlet temperature to be 1450 K, turbine outlet temperature to be 1097.474 K, and nozzle outlet temperature to be 1628.804 K.

The differences in temperature values can also be attributed to the assumptions and calculation methods used in the Python model compared to the GasTurb software. The Python model estimated the compressor outlet temperature to be 646.006 K, which is 2.5% higher than the 630.42 K calculated by GasTurb. For the turbine inlet temperature, the Python model assumed 1450 K, which is 2.7% higher than the 1411.20 K calculated by GasTurb. The turbine outlet temperature in the Python model was 1097.474 K, which is 0.6% higher than the 1091.37 K calculated by GasTurb. The nozzle outlet temperature was calculated as 1628.804 K in the Python model, 10.9% lower than the 1827.46 K calculated by GasTurb. These discrepancies can be explained by the Python model predicting higher temperatures at some stages and lower temperatures at others.

These results demonstrate that our Python model is reliable in simulating the thermodynamic cycle of turbojet engines. The close agreement with GasTurb indicates that the model can be used confidently for further studies and optimization. Additionally, the flexibility and customizability of the Python model provide a wide range of applications for future research. The Python-based model can be expanded and optimized by integrating more complex physical phenomena and advanced simulation techniques.

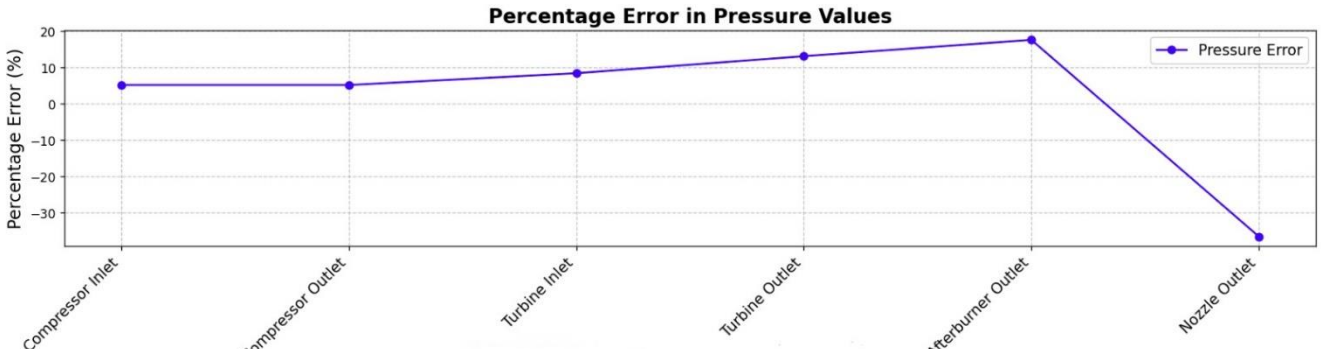


Figure 4. 3: Percentage error of the pressure values in the stations

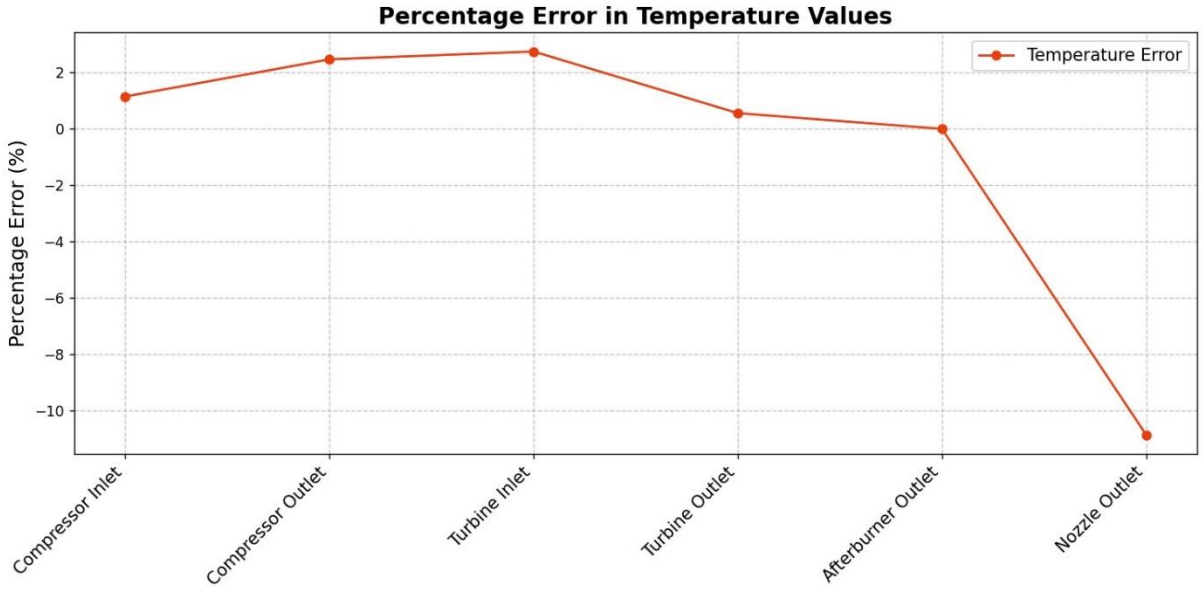


Figure 4. 4: Percentage error in the temperature values in the stations

Differences in nozzle exit pressure and temperature values between the Python model and GasTurb software, despite assuming similar environmental conditions and model assumptions, indicate disparities arising from calculation methods and model intricacies. The Python model computes the nozzle exit pressure as 224.243 kPa (36.54% lower) and the temperature as 1628.804 K (10.88% lower). These distinctions suggest that the Python model overestimates pressure losses and energy transformations within the nozzle, thereby forecasting lower thermal efficiency. Consequently, it underscores the necessity for further calibration and enhancement of the Python model through advanced numerical techniques.

6. CONCLUSION

This study investigates the effectiveness of a Python-based mathematical model developed to simulate the thermodynamic cycle of turbojet engines. The study compares the results obtained from the Python model with those from the GasTurb software to determine the accuracy of the model.

Initially, it was observed that the Python model calculated the net thrust of the engine at standard operating conditions to be 31.9 kN, whereas GasTurb software calculated the thrust to be 34.98 kN under the same conditions. These results indicate a discrepancy of approximately 9% between the thrust predictions of the Python model and GasTurb. The reasons for this difference may stem from variations in the ability of both methods to simulate the fundamental performance characteristics of the engine. However, these results suggest that the Python model can effectively predict the thrust of the turbojet engine, albeit requiring some improvements and calibration.

Upon examining the differences in pressure values, significant disparities between the Python model and GasTurb software were observed, particularly at critical points such as compressor exit pressure and turbine exit pressure. The reasons for these differences may arise from variations in how both methods model the internal dynamics and thermodynamic behavior of the engine.

When comparing temperature values, similarities and differences were observed between the Python model and GasTurb software. Particularly, significant differences were detected at points such as turbine exit temperature and nozzle exit temperature. These differences may stem from variations in the calculation methods and assumptions used by both models.

These findings demonstrate that the Python-based model can accurately simulate the thermodynamic cycle of turbojet engines and produce similar results to GasTurb software. However, further investigation and understanding of the reasons for the differences between the two methods are necessary.

The conclusions of this study provide an important foundation for the design and optimization of more efficient and environmentally friendly turbojet engines. The flexibility and customizability of the Python-based model offer a wide range of applications for future research. The model can be expanded and optimized by integrating more complex physical phenomena and advanced simulation techniques.

7. REFERENCES

1. *Thermodynamics analysis of a turbojet engine integrated with a fuel cell and steam injection for high-speed flight.* **Ji, Zhixing.** 2019, Energy, Cilt 185.

2. *The effect of euro diesel-hydrogen dual fuel combustion on performance and environmental-economic indicators in a small UAV turbojet engine.* **Gürbüz, H., ve diğerleri.** basım yeri bilinmiyor : Fuel, 2021, Cilt 306. 121735.

3. *Modelling of a turbojet gas turbine engine.* **Klein, D. ve Abeykoon, C.** basım yeri bilinmiyor : ITA, IEEE, 2015, September, s. 200-206.

4. *Turbojet engines.* **Avantkar, Gajanan C.** basım yeri bilinmiyor : Gogte Institute of Technology, 2010.

5. **Roux, Elodie.** *Turbofan and turbojet engines: database handbook.* 2007.

6. *"Jet Propulsion: a simple guide to the aerodynamic and thermodynamic design and performance of jet engines".* **Armstrong, Frank ve N., Cumpsty.** [dü.] Cambridge University Press. The Edinburgh Building,Cambridge CB2 2RU : The Aeronautical Journal, 1998, Cilt 102. 1016.

7. *"Design and analysis tool for external-compression supersonic inlets.* **Slater, John.** basım yeri bilinmiyor : 50th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition., 2012.

8. **Saravanamuttoo, ve diğerleri.** *Gas Turbine Theory.* basım yeri bilinmiyor : Pearson Education, 2001.

9. *"Application of an industrial sensor coating system on a Rolls-Royce jet engine for temperature detection.".* **J.P., Feist.** basım yeri bilinmiyor : Journal of Engineering for Gas Turbines and Power, 2013, Cilt 135.1. 012101.

10. **SKYbrary Aviation Safety. [Çevrimiçi] https://skybrary.aero/.**

11. *"Gas turbine engine optimization at conceptual designing.".* Tkachenko, Andrey Y. basım yeri bilinmiyor : MATEC Web of Conferences, EDP Sciences, 2016, Cilt Vol. 77.

12. *Modeling techniques for a computational efficient dynamic turbofan engine model.* Roberts, R. A. ve Eastbourn, S. M. basım yeri bilinmiyor : International Journal of Aerospace Engineering, 2014.

13. *Cuce, Erdem, et al. "An accurate model for photovoltaic (PV) modules to determine electrical characteristics and thermodynamic performance parameters." Energy*

Conversion and Management 146 (2017): 205-216. **Cuce ve Erdem. basım yeri bilinmiyor : nergy Conversion and Management, 2017, Cilt 146, s. 205-216.**

14. ***"Performance evaluation of an experimental turbojet engine.".*** **Ekici, Selçuk. basım yeri bilinmiyor : International Journal of Turbo & Jet-Engines, 2017, Cilt 34.4, s. 365-375.**

15. ***Modelling of a turbojet gas turbine engine. In 2015 internet technologies and applications.*** **D., Klein ve C., Abeykoon. basım yeri bilinmiyor : ITA, 2015, September, s. 200-206.**

16. ***"Off-design modelling of a turbo jet engine with operative afterburner.".*** **Erameh, A. A. , Aburime, E. I. ve Ighodaro, O. O. basım yeri bilinmiyor : Open Journal of Energy Efficiency, 2022, Cilt 11.3, s. 88-107.**

17. ***Thermodynamics: an engineering approach.*** **Cengel, Yunus A., Michael A. Boles, and Mehmet Kanoğlu. New York : McGraw-hill,, 2011, Cilt Vol. 5.**

8. APPENDICES

8.1. Design Codes

```
from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.setWindowModality(QtCore.Qt.NonModal)
        MainWindow.resize(1379, 846)
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("../../../Downloads/Black and
White Monogram Business Logo (2)-PhotoRoom.png-PhotoRoom.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        MainWindow.setWindowIcon(icon)
        MainWindow.setAutoFillBackground(False)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.tabWidget = QtWidgets.QTabWidget(self.centralwidget)
        self.tabWidget.setEnabled(True)
        self.tabWidget.setGeometry(QtCore.QRect(0, 0, 1401, 821))
        self.tabWidget.setObjectName("tabWidget")
        self.tab_1 = QtWidgets.QWidget()
        self.tab_1.setObjectName("tab_1")
        self.groupBox = QtWidgets.QGroupBox(self.tab_1)
        self.groupBox.setGeometry(QtCore.QRect(10, 0, 361, 81))
        self.groupBox.setObjectName("groupBox")
```

```
self.label_2 = QtWidgets.QLabel(self.groupBox)
self.label_2.setGeometry(QtCore.QRect(60, 50, 181, 16))
self.label_2.setObjectName("label_2")
self.label = QtWidgets.QLabel(self.groupBox)
self.label.setGeometry(QtCore.QRect(60, 20, 181, 16))
self.label.setObjectName("label")
self.lineEdit = QtWidgets.QLineEdit(self.groupBox)
self.lineEdit.setGeometry(QtCore.QRect(290, 20, 51, 22))
self.lineEdit.setObjectName("lineEdit")
self.lineEdit_2 = QtWidgets.QLineEdit(self.groupBox)
self.lineEdit_2.setGeometry(QtCore.QRect(290, 50, 51, 22))
self.lineEdit_2.setObjectName("lineEdit_2")
self.label.raise_()
self.lineEdit_2.raise_()
self.lineEdit.raise_()
self.label_2.raise_()
self.groupBox_2 = QtWidgets.QGroupBox(self.tab_1)
self.groupBox_2.setGeometry(QtCore.QRect(10, 90, 361, 411))
self.groupBox_2.setObjectName("groupBox_2")
self.label_8 = QtWidgets.QLabel(self.groupBox_2)
self.label_8.setGeometry(QtCore.QRect(60, 170, 181, 16))
self.label_8.setObjectName("label_8")
self.label_10 = QtWidgets.QLabel(self.groupBox_2)
self.label_10.setGeometry(QtCore.QRect(60, 110, 181, 16))
self.label_10.setObjectName("label_10")
self.label_11 = QtWidgets.QLabel(self.groupBox_2)
self.label_11.setGeometry(QtCore.QRect(60, 230, 181, 16))
self.label_11.setObjectName("label_11")
self.label_13 = QtWidgets.QLabel(self.groupBox_2)
self.label_13.setGeometry(QtCore.QRect(60, 140, 181, 16))
self.label_13.setObjectName("label_13")
self.label_14 = QtWidgets.QLabel(self.groupBox_2)
self.label_14.setGeometry(QtCore.QRect(60, 80, 181, 16))
self.label_14.setObjectName("label_14")
self.label_15 = QtWidgets.QLabel(self.groupBox_2)
self.label_15.setGeometry(QtCore.QRect(60, 200, 191, 16))
self.label_15.setObjectName("label_15")
self.label_16 = QtWidgets.QLabel(self.groupBox_2)
self.label_16.setGeometry(QtCore.QRect(60, 290, 181, 16))
self.label_16.setObjectName("label_16")
self.label_18 = QtWidgets.QLabel(self.groupBox_2)
self.label_18.setGeometry(QtCore.QRect(60, 20, 181, 16))
self.label_18.setObjectName("label_18")
self.label_19 = QtWidgets.QLabel(self.groupBox_2)
self.label_19.setGeometry(QtCore.QRect(60, 50, 181, 16))
self.label_19.setObjectName("label_19")
self.label_20 = QtWidgets.QLabel(self.groupBox_2)
self.label_20.setGeometry(QtCore.QRect(60, 320, 191, 16))
self.label_20.setObjectName("label_20")
self.label_61 = QtWidgets.QLabel(self.groupBox_2)
self.label_61.setGeometry(QtCore.QRect(60, 350, 181, 16))
self.label_61.setObjectName("label_61")
self.label_27 = QtWidgets.QLabel(self.groupBox_2)
self.label_27.setGeometry(QtCore.QRect(60, 260, 181, 16))
self.label_27.setObjectName("label_27")
self.label_62 = QtWidgets.QLabel(self.groupBox_2)
self.label_62.setGeometry(QtCore.QRect(60, 380, 181, 16))
self.label_62.setObjectName("label_62")
```

```
self.lineEdit_6 = QtWidgets.QLineEdit(self.groupBox_2)
self.lineEdit_6.setGeometry(QtCore.QRect(290, 20, 51, 22))
self.lineEdit_6.setObjectName("lineEdit_6")
self.lineEdit_7 = QtWidgets.QLineEdit(self.groupBox_2)
self.lineEdit_7.setGeometry(QtCore.QRect(290, 50, 51, 22))
self.lineEdit_7.setObjectName("lineEdit_7")
self.lineEdit_8 = QtWidgets.QLineEdit(self.groupBox_2)
self.lineEdit_8.setGeometry(QtCore.QRect(290, 80, 51, 22))
self.lineEdit_8.setObjectName("lineEdit_8")
self.lineEdit_9 = QtWidgets.QLineEdit(self.groupBox_2)
self.lineEdit_9.setGeometry(QtCore.QRect(290, 110, 51, 22))
self.lineEdit_9.setObjectName("lineEdit_9")
self.lineEdit_10 = QtWidgets.QLineEdit(self.groupBox_2)
self.lineEdit_10.setGeometry(QtCore.QRect(290, 140, 51, 22))
self.lineEdit_10.setObjectName("lineEdit_10")
self.lineEdit_11 = QtWidgets.QLineEdit(self.groupBox_2)
self.lineEdit_11.setGeometry(QtCore.QRect(290, 170, 51, 22))
self.lineEdit_11.setObjectName("lineEdit_11")
self.lineEdit_12 = QtWidgets.QLineEdit(self.groupBox_2)
self.lineEdit_12.setGeometry(QtCore.QRect(290, 200, 51, 22))
self.lineEdit_12.setObjectName("lineEdit_12")
self.lineEdit_13 = QtWidgets.QLineEdit(self.groupBox_2)
self.lineEdit_13.setGeometry(QtCore.QRect(290, 230, 51, 22))
self.lineEdit_13.setObjectName("lineEdit_13")
self.lineEdit_14 = QtWidgets.QLineEdit(self.groupBox_2)
self.lineEdit_14.setGeometry(QtCore.QRect(290, 260, 51, 22))
self.lineEdit_14.setObjectName("lineEdit_14")
self.lineEdit_15 = QtWidgets.QLineEdit(self.groupBox_2)
self.lineEdit_15.setGeometry(QtCore.QRect(290, 290, 51, 22))
self.lineEdit_15.setObjectName("lineEdit_15")
self.lineEdit_16 = QtWidgets.QLineEdit(self.groupBox_2)
self.lineEdit_16.setGeometry(QtCore.QRect(290, 320, 51, 22))
self.lineEdit_16.setObjectName("lineEdit_16")
self.lineEdit_17 = QtWidgets.QLineEdit(self.groupBox_2)
self.lineEdit_17.setGeometry(QtCore.QRect(290, 350, 51, 22))
self.lineEdit_17.setObjectName("lineEdit_17")
self.lineEdit_18 = QtWidgets.QLineEdit(self.groupBox_2)
self.lineEdit_18.setGeometry(QtCore.QRect(290, 380, 51, 22))
self.lineEdit_18.setObjectName("lineEdit_18")
self.groupBox_3 = QtWidgets.QGroupBox(self.tab_1)
self.groupBox_3.setGeometry(QtCore.QRect(10, 510, 361, 161))
self.groupBox_3.setObjectName("groupBox_3")
self.label_6 = QtWidgets.QLabel(self.groupBox_3)
self.label_6.setGeometry(QtCore.QRect(60, 20, 181, 16))
self.label_6.setObjectName("label_6")
self.lineEdit_19 = QtWidgets.QLineEdit(self.groupBox_3)
self.lineEdit_19.setGeometry(QtCore.QRect(290, 20, 51, 22))
self.lineEdit_19.setObjectName("lineEdit_19")
self.lineEdit_20 = QtWidgets.QLineEdit(self.groupBox_3)
self.lineEdit_20.setGeometry(QtCore.QRect(290, 50, 51, 22))
self.lineEdit_20.setObjectName("lineEdit_20")
self.lineEdit_21 = QtWidgets.QLineEdit(self.groupBox_3)
self.lineEdit_21.setGeometry(QtCore.QRect(290, 80, 51, 22))
self.lineEdit_21.setObjectName("lineEdit_21")
self.lineEdit_22 = QtWidgets.QLineEdit(self.groupBox_3)
self.lineEdit_22.setGeometry(QtCore.QRect(290, 110, 51, 22))
self.lineEdit_22.setObjectName("lineEdit_22")
self.label_60 = QtWidgets.QLabel(self.groupBox_3)
```

```
        self.label_60.setGeometry(QtCore.QRect(60, 80, 211, 31))
        self.label_60.setObjectName("label_60")
        self.label_7 = QtWidgets.QLabel(self.groupBox_3)
        self.label_7.setGeometry(QtCore.QRect(60, 40, 201, 41))
        self.label_7.setObjectName("label_7")
        self.label_48 = QtWidgets.QLabel(self.groupBox_3)
        self.label_48.setGeometry(QtCore.QRect(60, 120, 181, 16))
        self.label_48.setObjectName("label_48")
        self.pushButton = QtWidgets.QPushButton(self.tab_1)
        self.pushButton.setGeometry(QtCore.QRect(150, 680, 93, 28))
        self.pushButton.setObjectName("pushButton")
        self.label_9 = QtWidgets.QLabel(self.tab_1)
        self.label_9.setGeometry(QtCore.QRect(770, 630, 331, 91))
        self.label_9.setText("")
        self.label_9.setObjectName("label_9")
        self.label_25 = QtWidgets.QLabel(self.tab_1)
        self.label_25.setGeometry(QtCore.QRect(760, 550, 151, 61))
        self.label_25.setText("")
        self.label_25.setObjectName("label_25")
        self.label_29 = QtWidgets.QLabel(self.tab_1)
        self.label_29.setGeometry(QtCore.QRect(660, 100, 111, 71))
        self.label_29.setText("")
        self.label_29.setObjectName("label_29")
        self.label_21 = QtWidgets.QLabel(self.tab_1)
        self.label_21.setGeometry(QtCore.QRect(520, 100, 111, 81))
        self.label_21.setText("")
        self.label_21.setObjectName("label_21")
        self.label_28 = QtWidgets.QLabel(self.tab_1)
        self.label_28.setGeometry(QtCore.QRect(1180, -30, 151, 121))
        self.label_28.setText("")
        self.label_28.setPixmap(QtGui.QPixmap("../../../Downloads/Black
and White Monogram Business Logo (2)-PhotoRoom.png-PhotoRoom.png"))
        self.label_28.setScaledContents(True)
        self.label_28.setObjectName("label_28")
        self.label_30 = QtWidgets.QLabel(self.tab_1)
        self.label_30.setGeometry(QtCore.QRect(1010, 100, 91, 81))
        self.label_30.setText("")
        self.label_30.setObjectName("label_30")
        self.label_12 = QtWidgets.QLabel(self.tab_1)
        self.label_12.setGeometry(QtCore.QRect(370, 150, 991, 431))
        self.label_12.setText("")
        self.label_12.setPixmap(QtGui.QPixmap("background.png"))
        self.label_12.setScaledContents(True)
        self.label_12.setObjectName("label_12")
        self.label_22 = QtWidgets.QLabel(self.tab_1)
        self.label_22.setGeometry(QtCore.QRect(460, 490, 131, 131))
        self.label_22.setText("")
        self.label_22.setObjectName("label_22")
        self.label_23 = QtWidgets.QLabel(self.tab_1)
        self.label_23.setGeometry(QtCore.QRect(890, 100, 91, 81))
        self.label_23.setText("")
        self.label_23.setObjectName("label_23")
        self.label_26 = QtWidgets.QLabel(self.tab_1)
        self.label_26.setGeometry(QtCore.QRect(1070, 190, 171, 81))
        self.label_26.setText("")
        self.label_26.setObjectName("label_26")
        self.label_24 = QtWidgets.QLabel(self.tab_1)
        self.label_24.setGeometry(QtCore.QRect(1090, 540, 151, 61))
```

```
self.label_24.setText("")
self.label_24.setObjectName("label_24")
self.pushButton_4 = QtWidgets.QPushButton(self.tab_1)
self.pushButton_4.setGeometry(QtCore.QRect(430, 640, 93, 71))
self.pushButton_4.setObjectName("pushButton_4")
self.pushButton_5 = QtWidgets.QPushButton(self.tab_1)
self.pushButton_5.setGeometry(QtCore.QRect(540, 640, 91, 71))
self.pushButton_5.setObjectName("pushButton_5")
self.pushButton_6 = QtWidgets.QPushButton(self.tab_1)
self.pushButton_6.setGeometry(QtCore.QRect(650, 640, 93, 71))
self.pushButton_6.setObjectName("pushButton_6")
self.progressBar = QtWidgets.QProgressBar(self.tab_1)
self.progressBar.setGeometry(QtCore.QRect(260, 680, 131, 23))
self.progressBar.setProperty("value", 0)
self.progressBar.setObjectName("progressBar")
self.checkBox_2 = QtWidgets.QCheckBox(self.tab_1)
self.checkBox_2.setGeometry(QtCore.QRect(10, 680, 131, 20))
self.checkBox_2.setLayoutDirection(QtCore.Qt.RightToLeft)
self.checkBox_2.setObjectName("checkBox_2")
self.groupBox.raise_()
self.groupBox_2.raise_()
self.groupBox_3.raise_()
self.pushButton.raise_()
self.label_9.raise_()
self.label_25.raise_()
self.label_29.raise_()
self.label_21.raise_()
self.label_30.raise_()
self.label_12.raise_()
self.label_22.raise_()
self.label_23.raise_()
self.label_26.raise_()
self.label_24.raise_()
self.pushButton_4.raise_()
self.pushButton_5.raise_()
self.pushButton_6.raise_()
self.label_28.raise_()
self.progressBar.raise_()
self.checkBox_2.raise_()
self.tabWidget.addTab(self.tab_1, "")
self.tab_2 = QtWidgets.QWidget()
self.tab_2.setObjectName("tab_2")
self.groupBox_4 = QtWidgets.QGroupBox(self.tab_2)
self.groupBox_4.setGeometry(QtCore.QRect(60, 30, 361, 81))
self.groupBox_4.setObjectName("groupBox_4")
self.label_3 = QtWidgets.QLabel(self.groupBox_4)
self.label_3.setGeometry(QtCore.QRect(60, 50, 181, 16))
self.label_3.setObjectName("label_3")
self.label_4 = QtWidgets.QLabel(self.groupBox_4)
self.label_4.setGeometry(QtCore.QRect(60, 20, 181, 16))
self.label_4.setObjectName("label_4")
self.lineEdit_50 = QtWidgets.QLineEdit(self.groupBox_4)
self.lineEdit_50.setGeometry(QtCore.QRect(300, 10, 51, 22))
self.lineEdit_50.setObjectName("lineEdit_50")
self.lineEdit_52 = QtWidgets.QLineEdit(self.groupBox_4)
self.lineEdit_52.setGeometry(QtCore.QRect(300, 40, 51, 22))
self.lineEdit_52.setObjectName("lineEdit_52")
self.groupBox_5 = QtWidgets.QGroupBox(self.tab_2)
```

```
self.groupBox_5.setGeometry(QtCore.QRect(60, 120, 361, 411))
self.groupBox_5.setObjectName("groupBox_5")
self.label_17 = QtWidgets.QLabel(self.groupBox_5)
self.label_17.setGeometry(QtCore.QRect(60, 170, 181, 16))
self.label_17.setObjectName("label_17")
self.label_31 = QtWidgets.QLabel(self.groupBox_5)
self.label_31.setGeometry(QtCore.QRect(60, 110, 181, 16))
self.label_31.setObjectName("label_31")
self.label_32 = QtWidgets.QLabel(self.groupBox_5)
self.label_32.setGeometry(QtCore.QRect(60, 230, 181, 16))
self.label_32.setObjectName("label_32")
self.label_33 = QtWidgets.QLabel(self.groupBox_5)
self.label_33.setGeometry(QtCore.QRect(60, 140, 181, 16))
self.label_33.setObjectName("label_33")
self.label_34 = QtWidgets.QLabel(self.groupBox_5)
self.label_34.setGeometry(QtCore.QRect(60, 80, 181, 16))
self.label_34.setObjectName("label_34")
self.label_35 = QtWidgets.QLabel(self.groupBox_5)
self.label_35.setGeometry(QtCore.QRect(60, 200, 191, 16))
self.label_35.setObjectName("label_35")
self.label_36 = QtWidgets.QLabel(self.groupBox_5)
self.label_36.setGeometry(QtCore.QRect(60, 290, 181, 16))
self.label_36.setObjectName("label_36")
self.label_37 = QtWidgets.QLabel(self.groupBox_5)
self.label_37.setGeometry(QtCore.QRect(60, 20, 181, 16))
self.label_37.setObjectName("label_37")
self.label_38 = QtWidgets.QLabel(self.groupBox_5)
self.label_38.setGeometry(QtCore.QRect(60, 50, 181, 16))
self.label_38.setObjectName("label_38")
self.label_39 = QtWidgets.QLabel(self.groupBox_5)
self.label_39.setGeometry(QtCore.QRect(60, 320, 191, 16))
self.label_39.setObjectName("label_39")
self.label_63 = QtWidgets.QLabel(self.groupBox_5)
self.label_63.setGeometry(QtCore.QRect(60, 350, 181, 16))
self.label_63.setObjectName("label_63")
self.label_40 = QtWidgets.QLabel(self.groupBox_5)
self.label_40.setGeometry(QtCore.QRect(60, 260, 181, 16))
self.label_40.setObjectName("label_40")
self.label_64 = QtWidgets.QLabel(self.groupBox_5)
self.label_64.setGeometry(QtCore.QRect(60, 380, 181, 16))
self.label_64.setObjectName("label_64")
self.lineEdit_53 = QtWidgets.QLineEdit(self.groupBox_5)
self.lineEdit_53.setGeometry(QtCore.QRect(300, 20, 51, 22))
self.lineEdit_53.setObjectName("lineEdit_53")
self.lineEdit_54 = QtWidgets.QLineEdit(self.groupBox_5)
self.lineEdit_54.setGeometry(QtCore.QRect(300, 50, 51, 22))
self.lineEdit_54.setObjectName("lineEdit_54")
self.lineEdit_55 = QtWidgets.QLineEdit(self.groupBox_5)
self.lineEdit_55.setGeometry(QtCore.QRect(300, 80, 51, 22))
self.lineEdit_55.setObjectName("lineEdit_55")
self.lineEdit_56 = QtWidgets.QLineEdit(self.groupBox_5)
self.lineEdit_56.setGeometry(QtCore.QRect(300, 110, 51, 22))
self.lineEdit_56.setObjectName("lineEdit_56")
self.lineEdit_57 = QtWidgets.QLineEdit(self.groupBox_5)
self.lineEdit_57.setGeometry(QtCore.QRect(300, 140, 51, 22))
self.lineEdit_57.setObjectName("lineEdit_57")
self.lineEdit_58 = QtWidgets.QLineEdit(self.groupBox_5)
self.lineEdit_58.setGeometry(QtCore.QRect(300, 170, 51, 22))
```

```
self.lineEdit_58.setObjectName("lineEdit_58")
self.lineEdit_59 = QtWidgets.QLineEdit(self.groupBox_5)
self.lineEdit_59.setGeometry(QtCore.QRect(300, 200, 51, 22))
self.lineEdit_59.setObjectName("lineEdit_59")
self.lineEdit_60 = QtWidgets.QLineEdit(self.groupBox_5)
self.lineEdit_60.setGeometry(QtCore.QRect(300, 230, 51, 22))
self.lineEdit_60.setObjectName("lineEdit_60")
self.lineEdit_61 = QtWidgets.QLineEdit(self.groupBox_5)
self.lineEdit_61.setGeometry(QtCore.QRect(300, 260, 51, 22))
self.lineEdit_61.setObjectName("lineEdit_61")
self.lineEdit_62 = QtWidgets.QLineEdit(self.groupBox_5)
self.lineEdit_62.setGeometry(QtCore.QRect(300, 290, 51, 22))
self.lineEdit_62.setObjectName("lineEdit_62")
self.lineEdit_63 = QtWidgets.QLineEdit(self.groupBox_5)
self.lineEdit_63.setGeometry(QtCore.QRect(300, 320, 51, 22))
self.lineEdit_63.setObjectName("lineEdit_63")
self.lineEdit_64 = QtWidgets.QLineEdit(self.groupBox_5)
self.lineEdit_64.setGeometry(QtCore.QRect(300, 350, 51, 22))
self.lineEdit_64.setObjectName("lineEdit_64")
self.lineEdit_65 = QtWidgets.QLineEdit(self.groupBox_5)
self.lineEdit_65.setGeometry(QtCore.QRect(300, 380, 51, 22))
self.lineEdit_65.setObjectName("lineEdit_65")
self.groupBox_6 = QtWidgets.QGroupBox(self.tab_2)
self.groupBox_6.setGeometry(QtCore.QRect(60, 550, 361, 141))
self.groupBox_6.setObjectName("groupBox_6")
self.label_43 = QtWidgets.QLabel(self.groupBox_6)
self.label_43.setGeometry(QtCore.QRect(60, 110, 181, 16))
self.label_43.setObjectName("label_43")
self.label_44 = QtWidgets.QLabel(self.groupBox_6)
self.label_44.setGeometry(QtCore.QRect(60, 20, 181, 16))
self.label_44.setObjectName("label_44")
self.label_65 = QtWidgets.QLabel(self.groupBox_6)
self.label_65.setGeometry(QtCore.QRect(60, 80, 211, 31))
self.label_65.setObjectName("label_65")
self.label_45 = QtWidgets.QLabel(self.groupBox_6)
self.label_45.setGeometry(QtCore.QRect(60, 40, 201, 41))
self.label_45.setObjectName("label_45")
self.lineEdit_66 = QtWidgets.QLineEdit(self.groupBox_6)
self.lineEdit_66.setGeometry(QtCore.QRect(300, 20, 51, 22))
self.lineEdit_66.setObjectName("lineEdit_66")
self.lineEdit_67 = QtWidgets.QLineEdit(self.groupBox_6)
self.lineEdit_67.setGeometry(QtCore.QRect(300, 50, 51, 22))
self.lineEdit_67.setObjectName("lineEdit_67")
self.lineEdit_68 = QtWidgets.QLineEdit(self.groupBox_6)
self.lineEdit_68.setGeometry(QtCore.QRect(300, 80, 51, 22))
self.lineEdit_68.setObjectName("lineEdit_68")
self.lineEdit_69 = QtWidgets.QLineEdit(self.groupBox_6)
self.lineEdit_69.setGeometry(QtCore.QRect(300, 110, 51, 22))
self.lineEdit_69.setObjectName("lineEdit_69")
self.comboBox = QtWidgets.QComboBox(self.tab_2)
self.comboBox.setGeometry(QtCore.QRect(120, 700, 181, 22))
self.comboBox.setObjectName("comboBox")
self.comboBox.addItem("")
self.comboBox.addItem("")
self.comboBox.addItem("")
self.comboBox.addItem("")
self.comboBox.addItem("")
self.comboBox.addItem("")
```

```
        self.comboBox.addItem("")
        self.comboBox.addItem("")
        self.comboBox.addItem("")
        self.comboBox.addItem("")
        self.comboBox.addItem("")
        self.comboBox.addItem("")
        self.comboBox.addItem("")
        self.comboBox.addItem("")
        self.comboBox.addItem("")
        self.lineEdit_39 = QtWidgets.QLineEdit(self.tab_2)
        self.lineEdit_39.setGeometry(QtCore.QRect(310, 700, 51, 22))
        self.lineEdit_39.setObjectName("lineEdit_39")
        self.lineEdit_40 = QtWidgets.QLineEdit(self.tab_2)
        self.lineEdit_40.setGeometry(QtCore.QRect(370, 700, 51, 22))
        self.lineEdit_40.setObjectName("lineEdit_40")
        self.pushButton_2 = QtWidgets.QPushButton(self.tab_2)
        self.pushButton_2.setGeometry(QtCore.QRect(230, 760, 93, 28))
        self.pushButton_2.setObjectName("pushButton_2")
        self.label_41 = QtWidgets.QLabel(self.tab_2)
        self.label_41.setGeometry(QtCore.QRect(130, 730, 281, 20))
        self.label_41.setObjectName("label_41")
        self.widget = QtWidgets.QWidget(self.tab_2)
        self.widget.setEnabled(True)
        self.widget.setGeometry(QtCore.QRect(440, 60, 891, 631))
        sizePolicy =
QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Maximum,
QtWidgets.QSizePolicy.Maximum)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.widget.sizePolicy().hasHeightForWidth
())
        self.widget.setSizePolicy(sizePolicy)
        self.widget.setMaximumSize(QtCore.QSize(16777215, 16777215))
        self.widget.setObjectName("widget")
        self.progressBar_2 = QtWidgets.QProgressBar(self.tab_2)
        self.progressBar_2.setGeometry(QtCore.QRect(330, 760, 131, 23))
        self.progressBar_2.setProperty("value", 0)
        self.progressBar_2.setObjectName("progressBar_2")
        self.label_42 = QtWidgets.QLabel(self.tab_2)
        self.label_42.setGeometry(QtCore.QRect(1180, -30, 151, 121))
        self.label_42.setText("")
        self.label_42.setPixmap(QtGui.QPixmap("../../../Downloads/Black
and White Monogram Business Logo (2)-PhotoRoom.png-PhotoRoom.png"))
        self.label_42.setScaledContents(True)
        self.label_42.setObjectName("label_42")
        self.pushButton_3 = QtWidgets.QPushButton(self.tab_2)
        self.pushButton_3.setGeometry(QtCore.QRect(1010, 700, 93, 28))
        self.pushButton_3.setObjectName("pushButton_3")
        self.pushButton_8 = QtWidgets.QPushButton(self.tab_2)
        self.pushButton_8.setGeometry(QtCore.QRect(1230, 700, 93, 28))
        self.pushButton_8.setObjectName("pushButton_8")
        self.label_46 = QtWidgets.QLabel(self.tab_2)
        self.label_46.setGeometry(QtCore.QRect(490, 700, 61, 21))
        self.label_46.setObjectName("label_46")
        self.comboBox_7 = QtWidgets.QComboBox(self.tab_2)
        self.comboBox_7.setGeometry(QtCore.QRect(540, 700, 181, 22))
        self.comboBox_7.setObjectName("comboBox_7")
```

```
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.comboBox_7.addItem("")
        self.label_47 = QtWidgets.QLabel(self.tab_2)
        self.label_47.setGeometry(QtCore.QRect(750, 700, 61, 21))
        self.label_47.setObjectName("label_47")
        self.pushButton_7 = QtWidgets.QPushButton(self.tab_2)
        self.pushButton_7.setGeometry(QtCore.QRect(1120, 700, 93, 28))
        self.pushButton_7.setObjectName("pushButton_7")
        self.comboBox_6 = QtWidgets.QComboBox(self.tab_2)
        self.comboBox_6.setGeometry(QtCore.QRect(800, 700, 181, 22))
        self.comboBox_6.setObjectName("comboBox_6")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
```

```
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.comboBox_6.addItem("")
        self.checkBox = QtWidgets.QCheckBox(self.tab_2)
        self.checkBox.setGeometry(QtCore.QRect(90, 760, 131, 20))
        self.checkBox.setLayoutDirection(QtCore.Qt.RightToLeft)
        self.checkBox.setObjectName("checkBox")
        self.widget.raise_()
        self.groupBox_4.raise_()
        self.groupBox_5.raise_()
        self.groupBox_6.raise_()
        self.comboBox.raise_()
        self.lineEdit_39.raise_()
        self.lineEdit_40.raise_()
        self.pushButton_2.raise_()
        self.label_41.raise_()
        self.progressBar_2.raise_()
        self.label_42.raise_()
        self.pushButton_3.raise_()
        self.pushButton_8.raise_()
```

```
        self.label_46.raise_()
        self.comboBox_7.raise_()
        self.label_47.raise_()
        self.pushButton_7.raise_()
        self.comboBox_6.raise_()
        self.checkBox.raise_()
        self.tabWidget.addTab(self.tab_2, "")
        MainWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        self.tabWidget.setCurrentIndex(1)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "JeTAI"))
        self.groupBox.setTitle(_translate("MainWindow", "Ambient
Parameters"))
        self.label_2.setText(_translate("MainWindow", "Mach Number"))
        self.label.setText(_translate("MainWindow", "Altitude (m)"))
        self.lineEdit.setText(_translate("MainWindow", "0"))
        self.lineEdit_2.setText(_translate("MainWindow", "0.24"))
        self.groupBox_2.setTitle(_translate("MainWindow", "Parameteres
of Gasturbine"))
        self.label_8.setText(_translate("MainWindow", "Turbine
Efficiency(0-1)"))
        self.label_10.setText(_translate("MainWindow", "Compressor
Efficiency(0-1)"))
        self.label_11.setText(_translate("MainWindow", "Mechanical
Efficiency(0-1)"))
        self.label_13.setText(_translate("MainWindow", "Combustion
Efficiency(0-1)"))
        self.label_14.setText(_translate("MainWindow", "Intake
Efficiency(0-1)"))
        self.label_15.setText(_translate("MainWindow", "Propelling
Nozzle Efficiency(0-1)"))
        self.label_16.setText(_translate("MainWindow", "Combustion
Pressure Loss(0-1)"))
        self.label_18.setText(_translate("MainWindow", "Compressor
Pressure Ratio"))
        self.label_19.setText(_translate("MainWindow", "Turbine Inlet
Temperature(K)"))
        self.label_20.setText(_translate("MainWindow", "Temperature
Afterburner Out(K)"))
        self.label_61.setText(_translate("MainWindow", "Nozzle Area
(m^2)"))
        self.label_27.setText(_translate("MainWindow", "Afterburner
Efficiency(0-1)"))
        self.label_62.setText(_translate("MainWindow", "Mass Flow Rate
(kg/s)"))
        self.lineEdit_6.setText(_translate("MainWindow", "12"))
        self.lineEdit_7.setText(_translate("MainWindow", "1450"))
        self.lineEdit_8.setText(_translate("MainWindow", "0.99"))
        self.lineEdit_9.setText(_translate("MainWindow", "0.85"))
        self.lineEdit_10.setText(_translate("MainWindow", "1"))
```

```
        self.lineEdit_11.setText(_translate("MainWindow", "0.89"))
        self.lineEdit_12.setText(_translate("MainWindow", "1"))
        self.lineEdit_13.setText(_translate("MainWindow", "0.99"))
        self.lineEdit_14.setText(_translate("MainWindow", "0.9"))
        self.lineEdit_15.setText(_translate("MainWindow", "0.04"))
        self.lineEdit_16.setText(_translate("MainWindow", "1900"))
        self.lineEdit_17.setText(_translate("MainWindow", "0.1"))
        self.lineEdit_18.setText(_translate("MainWindow", "32"))
        self.groupBox_3.setTitle(_translate("MainWindow", "Other
Parameters"))
        self.label_6.setText(_translate("MainWindow", "Spesific Heat
Ratio of Air"))
        self.lineEdit_19.setText(_translate("MainWindow", "1.4"))
        self.lineEdit_20.setText(_translate("MainWindow", "1.333"))
        self.lineEdit_21.setText(_translate("MainWindow", "1.148"))
        self.lineEdit_22.setText(_translate("MainWindow", "43260"))
        self.label_60.setText(_translate("MainWindow", "Spesific Heat
at Constant Pressure\n"
"(Combustion Gas) (kJ/(kg.K)"))
        self.label_7.setText(_translate("MainWindow", "Spesific Heat
Ratio of Combustion\n"
" Gases"))
        self.label_48.setText(_translate("MainWindow", "Fuel Calorific
Ratio (kJ/kg)"))
        self.pushButton.setText(_translate("MainWindow", "Calculate"))
        self.pushButton_4.setText(_translate("MainWindow", "Plot \n"
" T-s "))
        self.pushButton_5.setText(_translate("MainWindow", "Plot \n"
"H-s"))
        self.pushButton_6.setText(_translate("MainWindow", "Plot \n"
"P-v"))
        self.checkBox_2.setText(_translate("MainWindow", "After burner
on/off"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_1),
_translate("MainWindow", "Standart"))
        self.groupBox_4.setTitle(_translate("MainWindow", "Ambient
Parameters"))
        self.label_3.setText(_translate("MainWindow", "Mach Number"))
        self.label_4.setText(_translate("MainWindow", "Altitude (m)"))
        self.lineEdit_50.setText(_translate("MainWindow", "0"))
        self.lineEdit_52.setText(_translate("MainWindow", "0.24"))
        self.groupBox_5.setTitle(_translate("MainWindow", "Parameteres
of Gasturbine"))
        self.label_17.setText(_translate("MainWindow", "Turbine
Efficiency(0-1)"))
        self.label_31.setText(_translate("MainWindow", "Compressor
Efficiency(0-1)"))
        self.label_32.setText(_translate("MainWindow", "Mechanical
Efficiency(0-1)"))
        self.label_33.setText(_translate("MainWindow", "Combustion
Efficiency(0-1)"))
        self.label_34.setText(_translate("MainWindow", "Intake
Efficiency(0-1)"))
        self.label_35.setText(_translate("MainWindow", "Propelling
Nozzle Efficiency(0-1)"))
        self.label_36.setText(_translate("MainWindow", "Combustion
Pressure Loss(0-1)"))
```

```
        self.label_37.setText(_translate("MainWindow", "Compressor
Pressure Ratio"))
        self.label_38.setText(_translate("MainWindow", "Turbine Inlet
Temperature(K)"))
        self.label_39.setText(_translate("MainWindow", "Temperature
Afterburner Out(K)"))
        self.label_63.setText(_translate("MainWindow", "Nozzle Area
(m^2)"))
        self.label_40.setText(_translate("MainWindow", "Afterburner
Efficiency(0-1)"))
        self.label_64.setText(_translate("MainWindow", "Mass Flow Rate
(kg/s)"))
        self.lineEdit_53.setText(_translate("MainWindow", "12"))
        self.lineEdit_54.setText(_translate("MainWindow", "1450"))
        self.lineEdit_55.setText(_translate("MainWindow", "0.99"))
        self.lineEdit_56.setText(_translate("MainWindow", "0.85"))
        self.lineEdit_57.setText(_translate("MainWindow", "1"))
        self.lineEdit_58.setText(_translate("MainWindow", "0.89"))
        self.lineEdit_59.setText(_translate("MainWindow", "1"))
        self.lineEdit_60.setText(_translate("MainWindow", "0.99"))
        self.lineEdit_61.setText(_translate("MainWindow", "0.9"))
        self.lineEdit_62.setText(_translate("MainWindow", "0.04"))
        self.lineEdit_63.setText(_translate("MainWindow", "1900"))
        self.lineEdit_64.setText(_translate("MainWindow", "0.1"))
        self.lineEdit_65.setText(_translate("MainWindow", "32"))
        self.groupBox_6.setTitle(_translate("MainWindow", "Other
Parameters"))
        self.label_43.setText(_translate("MainWindow", "Fuel Calorific
Ratio (kJ/kg)"))
        self.label_44.setText(_translate("MainWindow", "Spesific Heat
Ratio of Air"))
        self.label_65.setText(_translate("MainWindow", "Spesific Heat
at Constant Pressure\n"
"(Combustion Gas) (kJ/(kg.K)"))
        self.label_45.setText(_translate("MainWindow", "Spesific Heat
Ratio of Combustion\n"
" Gases"))
        self.lineEdit_66.setText(_translate("MainWindow", "1.4"))
        self.lineEdit_67.setText(_translate("MainWindow", "1.333"))
        self.lineEdit_68.setText(_translate("MainWindow", "1.148"))
        self.lineEdit_69.setText(_translate("MainWindow", "43260"))
        self.comboBox.setItemText(0, _translate("MainWindow", "Altitude
(m)"))
        self.comboBox.setItemText(1, _translate("MainWindow", "Mach
Number"))
        self.comboBox.setItemText(2, _translate("MainWindow",
"Compressor Pressure Ratio"))
        self.comboBox.setItemText(3, _translate("MainWindow", "Intake
Efficiency(0-1)"))
        self.comboBox.setItemText(4, _translate("MainWindow",
"Compressor Efficiency(0-1)"))
        self.comboBox.setItemText(5, _translate("MainWindow",
"Combustion Efficiency(0-1)"))
        self.comboBox.setItemText(6, _translate("MainWindow", "Turbine
Efficiency(0-1)"))
        self.comboBox.setItemText(7, _translate("MainWindow",
"Propelling Nozzle Efficiency"))
```

```
        self.comboBox.setItemText(8, _translate("MainWindow",
"Mechanical Efficiency"))
        self.comboBox.setItemText(9, _translate("MainWindow",
"Afterburner Efficiency"))
        self.comboBox.setItemText(10, _translate("MainWindow",
"Combustion Pressure Loss(0-1)"))
        self.comboBox.setItemText(11, _translate("MainWindow",
"Temperature Afterburner Out(K)"))
        self.comboBox.setItemText(12, _translate("MainWindow", "Nozzle
Area(m^2)"))
        self.comboBox.setItemText(13, _translate("MainWindow", "Mass
Flow Rate(kg/s)"))
        self.comboBox.setItemText(14, _translate("MainWindow", "Fuel
Calorific Ratio (kJ/kg)"))
        self.lineEdit_39.setText(_translate("MainWindow", "max"))
        self.lineEdit_40.setText(_translate("MainWindow", "delta"))
        self.pushButton_2.setText(_translate("MainWindow",
"Calculate"))
        self.label_41.setText(_translate("MainWindow", "The minimum
value is the value you enter first."))
        self.pushButton_3.setText(_translate("MainWindow", "Plot
Graph"))
        self.pushButton_8.setText(_translate("MainWindow", "Save
Graph"))
        self.label_46.setText(_translate("MainWindow", "X Axis:"))
        self.comboBox_7.setItemText(0, _translate("MainWindow",
"Altitude"))
        self.comboBox_7.setItemText(1, _translate("MainWindow", "Mach
Number"))
        self.comboBox_7.setItemText(2, _translate("MainWindow",
"Compressor Pressure Ratio"))
        self.comboBox_7.setItemText(3, _translate("MainWindow", "Intake
Efficiency(0-1)"))
        self.comboBox_7.setItemText(4, _translate("MainWindow",
"Compressor Efficiency(0-1)"))
        self.comboBox_7.setItemText(5, _translate("MainWindow",
"Combustion Efficiency(0-1)"))
        self.comboBox_7.setItemText(6, _translate("MainWindow",
"Turbine Efficiency(0-1)"))
        self.comboBox_7.setItemText(7, _translate("MainWindow",
"Propelling Nozzle Efficiency(0-1)"))
        self.comboBox_7.setItemText(8, _translate("MainWindow",
"Mechanical Efficiency(0-1)"))
        self.comboBox_7.setItemText(9, _translate("MainWindow",
"Afterburner Efficiency(0-1)"))
        self.comboBox_7.setItemText(10, _translate("MainWindow",
"Combustion Pressure Loss(0-1)"))
        self.comboBox_7.setItemText(11, _translate("MainWindow",
"Temperature Afterburner Out"))
        self.comboBox_7.setItemText(12, _translate("MainWindow",
"Nozzle Area"))
        self.comboBox_7.setItemText(13, _translate("MainWindow", "Mass
Flow Rate"))
        self.comboBox_7.setItemText(14, _translate("MainWindow", "Fuel
Calorific Ratio"))
        self.comboBox_7.setItemText(15, _translate("MainWindow",
"Spesific Thrust"))
```

```
        self.comboBox_7.setItemText(16, _translate("MainWindow", "Net
Thrust"))
        self.comboBox_7.setItemText(17, _translate("MainWindow",
"Velocity Nozzle Out"))
        self.comboBox_7.setItemText(18, _translate("MainWindow", "Inlet
Velocity"))
        self.comboBox_7.setItemText(19, _translate("MainWindow",
"Compressor Inlet Temperature"))
        self.comboBox_7.setItemText(20, _translate("MainWindow",
"Compressor Outlet Temperature"))
        self.comboBox_7.setItemText(21, _translate("MainWindow",
"Turbine Inlet Temperature"))
        self.comboBox_7.setItemText(22, _translate("MainWindow",
"Turbine Outlet Temperature"))
        self.comboBox_7.setItemText(23, _translate("MainWindow",
"Afterburner Outlet Temperature"))
        self.comboBox_7.setItemText(24, _translate("MainWindow",
"Nozzle Outlet Temperature"))
        self.comboBox_7.setItemText(25, _translate("MainWindow", "Inlet
Pressure"))
        self.comboBox_7.setItemText(26, _translate("MainWindow",
"Compressor Inlet Pressure"))
        self.comboBox_7.setItemText(27, _translate("MainWindow",
"Compressor Outlet Pressure"))
        self.comboBox_7.setItemText(28, _translate("MainWindow",
"Turbine Inlet Pressure"))
        self.comboBox_7.setItemText(29, _translate("MainWindow",
"Turbine Outlet Pressure"))
        self.comboBox_7.setItemText(30, _translate("MainWindow",
"Afterburner Outlet Pressure"))
        self.comboBox_7.setItemText(31, _translate("MainWindow",
"Nozzle Outlet Pressure"))
        self.comboBox_7.setItemText(32, _translate("MainWindow", "Inlet
Enthalpy"))
        self.comboBox_7.setItemText(33, _translate("MainWindow",
"Compressor Inlet Enthalpy"))
        self.comboBox_7.setItemText(34, _translate("MainWindow",
"Compressor Outlet Enthalpy"))
        self.comboBox_7.setItemText(35, _translate("MainWindow",
"Turbine Inlet Enthalpy"))
        self.comboBox_7.setItemText(36, _translate("MainWindow",
"Turbine Outlet Enthalpy"))
        self.comboBox_7.setItemText(37, _translate("MainWindow",
"Nozzle Outlet Enthalpy"))
        self.comboBox_7.setItemText(38, _translate("MainWindow", "Inlet
Entropy"))
        self.comboBox_7.setItemText(39, _translate("MainWindow",
"Compressor Inlet Entropy"))
        self.comboBox_7.setItemText(40, _translate("MainWindow",
"Compressor Outlet Entropy"))
        self.comboBox_7.setItemText(41, _translate("MainWindow",
"Turbine Inlet Entropy"))
        self.comboBox_7.setItemText(42, _translate("MainWindow",
"Turbine Outlet Entropy"))
        self.comboBox_7.setItemText(43, _translate("MainWindow",
"Afterburner Outlet Entropy"))
        self.comboBox_7.setItemText(44, _translate("MainWindow",
"Nozzle Outlet Entropy"))
```

```
        self.label_47.setText(_translate("MainWindow", "Y Axis:"))
        self.pushButton_7.setText(_translate("MainWindow", "Clear
Graph"))
        self.comboBox_6.setItemText(0, _translate("MainWindow",
"Altitude"))
        self.comboBox_6.setItemText(1, _translate("MainWindow", "Mach
Number"))
        self.comboBox_6.setItemText(2, _translate("MainWindow",
"Compressor Pressure Ratio"))
        self.comboBox_6.setItemText(3, _translate("MainWindow", "Intake
Efficiency(0-1)"))
        self.comboBox_6.setItemText(4, _translate("MainWindow",
"Compressor Efficiency(0-1)"))
        self.comboBox_6.setItemText(5, _translate("MainWindow",
"Combustion Efficiency(0-1)"))
        self.comboBox_6.setItemText(6, _translate("MainWindow",
"Turbine Efficiency(0-1)"))
        self.comboBox_6.setItemText(7, _translate("MainWindow",
"Propelling Nozzle Efficiency(0-1)"))
        self.comboBox_6.setItemText(8, _translate("MainWindow",
"Mechanical Efficiency(0-1)"))
        self.comboBox_6.setItemText(9, _translate("MainWindow",
"Afterburner Efficiency(0-1)"))
        self.comboBox_6.setItemText(10, _translate("MainWindow",
"Combustion Pressure Loss(0-1)"))
        self.comboBox_6.setItemText(11, _translate("MainWindow",
"Temperature Afterburner Out"))
        self.comboBox_6.setItemText(12, _translate("MainWindow",
"Nozzle Area"))
        self.comboBox_6.setItemText(13, _translate("MainWindow", "Mass
Flow Rate"))
        self.comboBox_6.setItemText(14, _translate("MainWindow", "Fuel
Calorific Ratio"))
        self.comboBox_6.setItemText(15, _translate("MainWindow",
"Spesific Thrust"))
        self.comboBox_6.setItemText(16, _translate("MainWindow", "Net
Thrust"))
        self.comboBox_6.setItemText(17, _translate("MainWindow",
"Velocity Nozzle Out"))
        self.comboBox_6.setItemText(18, _translate("MainWindow", "Inlet
Velocity"))
        self.comboBox_6.setItemText(19, _translate("MainWindow",
"Compressor Inlet Temperature"))
        self.comboBox_6.setItemText(20, _translate("MainWindow",
"Compressor Outlet Temperature"))
        self.comboBox_6.setItemText(21, _translate("MainWindow",
"Turbine Inlet Temperature"))
        self.comboBox_6.setItemText(22, _translate("MainWindow",
"Turbine Outlet Temperature"))
        self.comboBox_6.setItemText(23, _translate("MainWindow",
"Afterburner Outlet Temperature"))
        self.comboBox_6.setItemText(24, _translate("MainWindow",
"Nozzle Outlet Temperature"))
        self.comboBox_6.setItemText(25, _translate("MainWindow", "Inlet
Pressure"))
        self.comboBox_6.setItemText(26, _translate("MainWindow",
"Compressor Inlet Pressure"))
```

```
        self.comboBox_6.setItemText(27, _translate("MainWindow",
"Compressor Outlet Pressure"))
        self.comboBox_6.setItemText(28, _translate("MainWindow",
"Turbine Inlet Pressure"))
        self.comboBox_6.setItemText(29, _translate("MainWindow",
"Turbine Outlet Pressure"))
        self.comboBox_6.setItemText(30, _translate("MainWindow",
"Afterburner Outlet Pressure"))
        self.comboBox_6.setItemText(31, _translate("MainWindow",
"Nozzle Outlet Pressure"))
        self.comboBox_6.setItemText(32, _translate("MainWindow", "Inlet
Enthalpy"))
        self.comboBox_6.setItemText(33, _translate("MainWindow",
"Compressor Inlet Enthalpy"))
        self.comboBox_6.setItemText(34, _translate("MainWindow",
"Compressor Outlet Enthalpy"))
        self.comboBox_6.setItemText(35, _translate("MainWindow",
"Turbine Inlet Enthalpy"))
        self.comboBox_6.setItemText(36, _translate("MainWindow",
"Turbine Outlet Enthalpy"))
        self.comboBox_6.setItemText(37, _translate("MainWindow",
"Nozzle Outlet Enthalpy"))
        self.comboBox_6.setItemText(38, _translate("MainWindow", "Inlet
Entropy"))
        self.comboBox_6.setItemText(39, _translate("MainWindow",
"Compressor Inlet Entropy"))
        self.comboBox_6.setItemText(40, _translate("MainWindow",
"Compressor Outlet Entropy"))
        self.comboBox_6.setItemText(41, _translate("MainWindow",
"Turbine Inlet Entropy"))
        self.comboBox_6.setItemText(42, _translate("MainWindow",
"Turbine Outlet Entropy"))
        self.comboBox_6.setItemText(43, _translate("MainWindow",
"Afterburner Outlet Entropy"))
        self.comboBox_6.setItemText(44, _translate("MainWindow",
"Nozzle Outlet Entropy"))
        self.checkBox.setText(_translate("MainWindow", "After burner
on/off"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_2),
_translate("MainWindow", "Delta"))
```

8.2. Main Codes

```
import sys
from typing import List, Any
from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QLabel,
QLineEdit, QPushButton, QMessageBox, QComboBox
from PyQt5 import QtCore, QtWidgets
from PyQt5.QtWidgets import QMainWindow, QApplication, QPushButton,
QWidget, QAction, QTabWidget, QVBoxLayout, QLabel, \
    QHBoxLayout, QGraphicsScene, QGraphicsView, QFileDialog
from pyqtgraph import PlotWidget
from PyQt5.QtCore import QRectF
import random
from constant_spesific_heat import cp
import pandas as pd
```

```
import CoolProp.CoolProp as CoolProp
from calculation import calculation
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as
FigureCanvas
from matplotlib.figure import Figure
import matplotlib.pyplot as plt
import numpy as np
import designn
from matplotlib.figure import Figure
from matplotlib.backends.backend_qt5agg import (NavigationToolbar2QT as
NavigationToolbar)

altitude_append = []
compressor_pressure_ratio_append = []
mach_number_append = []
temperature_a_append = []
pressure_a_append = []
enthalpy_ambient_append = []
velocity_a_append = []
temperature_compressor_inlet_append = []
pressure_compressor_inlet_append = []
enthalpy_compressor_inlet_append = []
temperature_compressor_outlet_append = []
pressure_compressor_outlet_append = []
enthalpy_compressor_outlet_append = []
temperature_turbine_inlet_append = []
pressure_turbine_inlet_append = []
enthalpy_turbine_inlet_append = []
temperature_turbine_outlet_with_losses_append = []
pressure_turbine_outlet_append = []
enthalpy_turbine_outlet_with_losses_append = []
temperature_afterburner_out_append = []
afterburner_outlet_pressure_append = []
enthalpy_afterburner_outlet_append = []
temperature_nozzle_outlet_append = []
pressure_nozzle_outlet_append = []
enthalpy_nozzle_outlet_append = []
velocity_nozzle_outlet_append = []
net_thrust_append = []
fuel_calorific_ratio_append = []
intake_efficiency_append = []
n_c_append = []
n_b_append = []
n_t_append = []
n_n_append = []
n_ab_append = []
n_m_append = []
nozzle_area_append = []
mass_flow_rate_air_append = []
entropy_ambient_append = []
entropy_compressor_inlet_append = []
entropy_compressor_outlet_append = []
entropy_turbine_inlet_append = []
entropy_turbine_outlet_with_losses_append = []
entropy_afterburner_outlet_append = []
entropy_nozzle_outlet_append = []
fuel_air_ratio_combustion_append = []
fuel_air_ratio_afterburner_append = []
```

```
class MatplotlibWidget(QWidget):
    def __init__(self, parent=None):
        super(MatplotlibWidget, self).__init__(parent)
        self.figure, self.ax = plt.subplots()
        self.canvas = FigureCanvas(self.figure)
        self.layoutvertical = QVBoxLayout(self)
        self.layoutvertical.addWidget(self.canvas)


class MyApp(QMainWindow, designn.Ui_MainWindow):

    def __init__(self):
        super(MyApp, self).__init__()
        self.setupUi(self)
        self.init_widget()
        self.pushButton.clicked.connect(self.klein)
        self.pushButton_2.clicked.connect(self.delta)
        self.pushButton_3.clicked.connect(self.plot_widget)
        self.pushButton_4.clicked.connect(self.t_s_plot)
        self.pushButton_5.clicked.connect(self.h_s_plot)
        self.pushButton_6.clicked.connect(self.p_v_plot)
        self.pushButton_7.clicked.connect(self.clear)
        self.pushButton_8.clicked.connect(self.save_plot)
        self.matplotlibwidget.canvas.mpl_connect('motion_notify_event',
self.hover)
        self.checkBox.stateChanged.connect(self.state)

    def state(self, state):
        if state == 2:
            state = 2  # afterburner aktif
        else:
            state = 1  # afterburner aktif değil

    def init_widget(self):
        self.matplotlibwidget = MatplotlibWidget()
        self.layoutvertical = QVBoxLayout(self.widget)
        self.layoutvertical.addWidget(self.matplotlibwidget)

    def plot_widget(self):
        x_axis, x_label =
self.get_selected_axis(self.comboBox_7.currentText())
        y_axis, y_label =
self.get_selected_axis(self.comboBox_6.currentText())
        self.matplotlibwidget.ax.grid(True, linestyle='--',
color='gray', alpha=0.5)
        self.matplotlibwidget.ax.plot(x_axis, y_axis)
        self.matplotlibwidget.ax.set_xlabel(x_label)
        self.matplotlibwidget.ax.set_ylabel(y_label)
        self.matplotlibwidget.canvas.draw()

    def save_plot(self):
        options = QFileDialog.Options()
        filename, _ = QFileDialog.getSaveFileName(self, "Grafiği
Kaydet", "",
                                                  "PNG Files
(*.png);;JPEG Files (*.jpeg);;All Files (*)",
```

```
                                                    options=options)

        if filename:
            self.matplotlibwidget.figure.savefig(filename)

    def get_selected_axis(self, text):
        if text == "Altitude":
            axis = altitude_append
            label = "Altitude (m)"
        elif text == "Mach Number":
            axis = mach_number_append
            label = "Mach Number"
        elif text == "Compressor Pressure Ratio":
            axis = compressor_pressure_ratio_append
            label = "Compressor Pressure Ratio"
        elif text == "Turbine Inlet Temperature":
            axis = temperature_turbine_inlet_append
            label = "Turbine Inlet Temperature(K)"
        elif text == "Intake Efficiency(0-1)":
            axis = intake_efficiency_append
            label = "Intake Efficiency(0-1)"
        elif text == "Compressor Efficiency(0-1)":
            axis = n_c_append
            label = "Compressor Efficiency(0-1)"
        elif text == "Combustion Efficiency(0-1)":
            axis = n_b_append
            label = "Combustion Efficiency(0-1)"
        elif text == "Turbine Efficiency(0-1)":
            axis = n_t_append
            label = "Turbine Efficiency(0-1)"
        elif text == "Propelling Nozzle Efficiency(0-1)":
            axis = n_n_append
            label = "Propelling Nozzle Efficiency(0-1)"
        elif text == "Mechanical Efficiency(0-1)":
            axis = n_m_append
            label = "Mechanical Efficiency(0-1)"
        elif text == "Afterburner Efficiency(0-1)":
            axis = n_ab_append
            label = "Afterburner Efficiency(0-1)"
        elif text == "Combustion Pressure Loss(0-1)":
            axis = temperature_turbine_inlet_append
            label = "Combustion Pressure Loss(0-1)"
        elif text == "Temperature Afterburner Out(K)":
            axis = temperature_afterburner_out_append
            label = "Temperature Afterburner Out"
        elif text == "Nozzle Area":
            axis = nozzle_area_append
            label = "Nozzle Area (m^2)"
        elif text == "Mass Flow Rate":
            axis = mass_flow_rate_air_append
            label = "Mass Flow Rate (kg/s)"
        elif text == "Fuel Calorific Ratio":
            axis = fuel_calorific_ratio_append
            label = "Fuel Calorific Ratio (kJ/kg)"
        elif text == "Net Thrust":
            axis = net_thrust_append
            label = "Net Thrust (N)"
        elif text == "Velocity Nozzle Out":
```

```
    axis = velocity_nozzle_outlet_append
    label = "Velocity Nozzle Out (m/s)"
elif text == "Inlet Velocity":
    axis = velocity_a_append
    label = "Velocity Inlet Velocity (m/s)"
elif text == "Compressor Inlet Temperature":
    axis = temperature_compressor_inlet_append
    label = "Temperature Compressor Inlet (K)"
elif text == "Compressor Outlet Temperature":
    axis = temperature_compressor_outlet_append
    label = "Compressor Outlet Temperature (K)"
elif text == "Turbine Inlet Temperature":
    axis = temperature_turbine_inlet_append
    label = "Turbine Inlet Temperature (K)"
elif text == "Turbine Outlet Temperature":
    axis = temperature_turbine_outlet_with_losses_append
    label = "Turbine Outlet Temperature (K)"
elif text == "Afterburner Outlet Temperature":
    axis = temperature_afterburner_out_append
    label = "Afterburner Outlet Temperature (K)"
elif text == "Nozzle Outlet Temperature":
    axis = temperature_nozzle_outlet_append
    label = "Nozzle Outlet Temperature (K)"

elif text == "Inlet Pressure":
    axis = pressure_a_append
    label = "Pressure of Ambient (kPa)"
elif text == "Compressor Inlet Pressure":
    axis = pressure_compressor_inlet_append
    label = "Compressor Inlet Pressure (kPa)"
elif text == "Compressor Outlet Pressure":
    axis = pressure_compressor_outlet_append
    label = "Compressor Outlet Pressure (kPa)"
elif text == "Turbine Inlet Pressure":
    axis = pressure_turbine_inlet_append
    label = "Turbine Inlet Pressure (kPa)"
elif text == "Turbine Outlet Pressure":
    axis = pressure_turbine_outlet_append
    label = "Turbine Outlet Pressure (kPa)"
elif text == "Afterburner Outlet Pressure":
    axis = afterburner_outlet_pressure_append
    label = "Afterburner Outlet Pressure(kPa)"
elif text == "Nozzle Outlet Pressure":
    axis = pressure_nozzle_outlet_append
    label = "Nozzle Outlet Pressure (kPa)"

elif text == "Inlet Enthalpy":
    axis = enthalpy_ambient_append
    label = "Enhalpy of Ambient (kJ/kg)"
elif text == "Compressor Inlet Enthalpy":
    axis = pressure_compressor_inlet_append
    label = "Compressor Inlet Enthalpy (kJ/kg)"
elif text == "Compressor Outlet Enthalpy":
    axis = pressure_compressor_outlet_append
    label = "Compressor Outlet Enthalpy (kJ/kg)"
elif text == "Turbine Inlet Enthalpy":
    axis = pressure_turbine_inlet_append
    label = "Turbine Inlet Enthalpy (kJ/kg)"
```

```
        elif text == "Turbine Outlet Enthalpy":
            axis = pressure_turbine_outlet_append
            label = "Turbine Outlet Enthalpy (kJ/kg)"
        elif text == "Afterburner Outlet Enthalpy":
            axis = afterburner_outlet_pressure_append
            label = "Afterburner Outlet Enthalpy (kJ/kg)"
        elif text == "Nozzle Outlet Enthalpy":
            axis = pressure_nozzle_outlet_append
            label = "Nozzle Outlet Enthalpy (kJ/kg)"

        elif text == "Inlet Entropy":
            axis = enthalpy_ambient_append
            label = "Entropy of Ambient (kJ/kg)"
        elif text == "Compressor Inlet Entropy":
            axis = pressure_compressor_inlet_append
            label = "Compressor Inlet Entropy (kJ/kg)"
        elif text == "Compressor Outlet Entropy":
            axis = pressure_compressor_outlet_append
            label = "Compressor Outlet Entropy (kJ/kg)"
        elif text == "Turbine Inlet Entropy":
            axis = pressure_turbine_inlet_append
            label = "Turbine Inlet Entropy (kJ/kg)"
        elif text == "Turbine Outlet Entropy":
            axis = pressure_turbine_outlet_append
            label = "Turbine Outlet Entropy (kJ/kg)"
        elif text == "Afterburner Outlet  Entropy":
            axis = afterburner_outlet_pressure_append
            label = "Afterburner Outlet  Entropy (kJ/kg)"
        elif text == "Nozzle Outlet Entropy":
            axis = pressure_nozzle_outlet_append
            label = "Nozzle Outlet  Entropy (kJ/kg)"
        else:
            axis = None
            label = None
        return axis, label

    def hover(self, event):
        if event.inaxes == self.matplotlibwidget.ax:
            x, y = event.xdata, event.ydata
            tooltip_text = f"x={x:.3f}, y={y:.3f}"
            self.show_tooltip(tooltip_text)

    def show_error_message(self, message):
        error_dialog = QMessageBox()
        error_dialog.setIcon(QMessageBox.Critical)
        error_dialog.setWindowTitle("ERROR!")
        error_dialog.setText("Something went wrong!")
        error_dialog.setInformativeText(message)
        error_dialog.exec_()

    def show_tooltip(self, text):
        if hasattr(self, 'tooltip'):
            self.tooltip.remove()
        self.tooltip = self.matplotlibwidget.ax.text(0.99, 0.99, text,
ha='right', va='top', fontsize=10,
                                                     color='black',
transform=self.matplotlibwidget.ax.transAxes)
        self.matplotlibwidget.canvas.draw()
```

```
    def clear(self):
        self.matplotlibwidget.ax.clear()

    def t_s_plot(self):
        altitude = float(self.lineEdit.text())
        mach_number = float(self.lineEdit_2.text())
        compressor_pressure_ratio = float(self.lineEdit_6.text())
        temperature_turbine_inlet = float(self.lineEdit_7.text())  # K
        intake_efficiency = float(self.lineEdit_8.text())
        n_c = float(self.lineEdit_9.text())
        n_b = float(self.lineEdit_10.text())
        n_t = float(self.lineEdit_11.text())
        n_n = float(self.lineEdit_12.text())
        n_m = float(self.lineEdit_13.text())
        n_ab = float(self.lineEdit_14.text())
        combustion_pressure_loss = float(self.lineEdit_15.text())
        nozzle_area = float(self.lineEdit_17.text())  # m^2
        mass_flow_rate_air = float(self.lineEdit_18.text())  # kg/s
        specific_heat_ratio_air = float(self.lineEdit_19.text())
        specific_heat_ratio_combustion_gas =
float(self.lineEdit_20.text())
        c_pa = 1.005
        c_pg = float(self.lineEdit_21.text())
        gas_constant = 0.287  # kJ/(kg*K)
        fuel_calorific = float(self.lineEdit_22.text())  # kJ/kg
        state = self.checkBox_2.isChecked()
        if state == True:
            temperature_afterburner_out =
float(self.lineEdit_16.text())
        else:
            temperature_afterburner_out = 0
            pressure_afterburner_outlet = 0

        pressure_a, temperature_a, velocity_a, pressure_turbine_outlet,
pressure_compressor_inlet, \
        temperature_compressor_inlet, intake_pressure_ratio,
pressure_compressor_outlet, temperature_compressor_outlet, \
        temperature_turbine_outlet,
temperature_turbine_outlet_with_losses, pressure_turbine_inlet,
fuel_air_ratio_combustion, \
        pressure_afterburner_outlet, fuel_air_ratio_afterburner,
temperature_nozzle_outlet, pressure_nozzle_outlet, \
        velocity_nozzle_outlet, massflow_nozzle_out, net_thrust,
specific_thrust, thrust_spesific_fuel_consumption, \
        enthalpy_ambient, enthalpy_compressor_inlet,
enthalpy_compressor_outlet, enthalpy_turbine_inlet, \
        enthalpy_turbine_outlet_with_losses,
enthalpy_afterburner_outlet, enthalpy_nozzle_outlet, entropy_ambient, \
        entropy_compressor_inlet, entropy_compressor_outlet,
entropy_turbine_inlet, entropy_turbine_outlet_with_losses, \
        entropy_afterburner_outlet, entropy_nozzle_outlet,
specific_volume_ambient, specific_volume_compressor_inlet,
specific_volume_compressor_outlet, \
        specific_volume_turbine_inlet,
specific_volume_turbine_outlet_with_losses,
specific_volume_afterburnet_outlet, specific_volume_nozzle_outlet =
calculation(
```

```
            state,
            altitude, mach_number, compressor_pressure_ratio,
temperature_turbine_inlet, intake_efficiency,
            n_c,
            n_b,
            n_t, n_n, n_m, n_ab, combustion_pressure_loss,
temperature_afterburner_out, nozzle_area,
            mass_flow_rate_air,
            specific_heat_ratio_air,
specific_heat_ratio_combustion_gas, c_pg, fuel_calorific)
        if state == True:

            entropy_append = [entropy_ambient,
entropy_compressor_inlet, entropy_compressor_outlet,
                              entropy_turbine_inlet,
                              entropy_turbine_outlet_with_losses,
entropy_afterburner_outlet, entropy_nozzle_outlet]
            temperature_append = [temperature_a,
temperature_compressor_inlet, temperature_compressor_outlet,
                                  temperature_turbine_inlet,
temperature_turbine_outlet_with_losses,
                                  temperature_afterburner_out,
temperature_nozzle_outlet]
            stations = ['a', '1', '2', '3', '4', '5', '6']
        else:
            entropy_append = [entropy_ambient,
entropy_compressor_inlet, entropy_compressor_outlet,
                              entropy_turbine_inlet,
                              entropy_turbine_outlet_with_losses,
entropy_nozzle_outlet]
            temperature_append = [temperature_a,
temperature_compressor_inlet, temperature_compressor_outlet,
                                  temperature_turbine_inlet,
temperature_turbine_outlet_with_losses,
                                  temperature_nozzle_outlet]
            stations = ['a', '1', '2', '3', '4', '5']

        for i, txt in enumerate(stations):
            plt.text(entropy_append[i], temperature_append[i], txt,
fontsize=10, ha='right', va='bottom')
        plt.plot(entropy_append, temperature_append)
        plt.title('Entropy-Temperature')
        plt.xlabel('Entropy(kj/kgK)')
        plt.ylabel('Temperature(K)')
        plt.show()

    def h_s_plot(self):
        altitude = float(self.lineEdit.text())
        mach_number = float(self.lineEdit_2.text())
        compressor_pressure_ratio = float(self.lineEdit_6.text())
        temperature_turbine_inlet = float(self.lineEdit_7.text())  # K
        intake_efficiency = float(self.lineEdit_8.text())
        n_c = float(self.lineEdit_9.text())
        n_b = float(self.lineEdit_10.text())
        n_t = float(self.lineEdit_11.text())
        n_n = float(self.lineEdit_12.text())
        n_m = float(self.lineEdit_13.text())
        n_ab = float(self.lineEdit_14.text())
```

```
        combustion_pressure_loss = float(self.lineEdit_15.text())
        temperature_afterburner_out = float(self.lineEdit_16.text())
        nozzle_area = float(self.lineEdit_17.text())  # m^2
        mass_flow_rate_air = float(self.lineEdit_18.text())  # kg/s
        specific_heat_ratio_air = float(self.lineEdit_19.text())
        specific_heat_ratio_combustion_gas =
float(self.lineEdit_20.text())
        c_pa = 1.005
        c_pg = float(self.lineEdit_21.text())
        gas_constant = 0.287  # kJ/(kg*K)
        fuel_calorific = float(self.lineEdit_22.text())  # kJ/kg
        state = self.checkBox_2.isChecked()
        if state == True:
            temperature_afterburner_out =
float(self.lineEdit_16.text())
        else:
            temperature_afterburner_out = 0
            pressure_afterburner_outlet = 0
        pressure_a, temperature_a, velocity_a, pressure_turbine_outlet,
pressure_compressor_inlet, \
        temperature_compressor_inlet, intake_pressure_ratio,
pressure_compressor_outlet, temperature_compressor_outlet, \
        temperature_turbine_outlet,
temperature_turbine_outlet_with_losses, pressure_turbine_inlet,
fuel_air_ratio_combustion, \
        pressure_afterburner_outlet, fuel_air_ratio_afterburner,
temperature_nozzle_outlet, pressure_nozzle_outlet, \
        velocity_nozzle_outlet, massflow_nozzle_out, net_thrust,
specific_thrust, thrust_spesific_fuel_consumption, \
        enthalpy_ambient, enthalpy_compressor_inlet,
enthalpy_compressor_outlet, enthalpy_turbine_inlet, \
        enthalpy_turbine_outlet_with_losses,
enthalpy_afterburner_outlet, enthalpy_nozzle_outlet, entropy_ambient, \
        entropy_compressor_inlet, entropy_compressor_outlet,
entropy_turbine_inlet, entropy_turbine_outlet_with_losses, \
        entropy_afterburner_outlet, entropy_nozzle_outlet,
specific_volume_ambient, specific_volume_compressor_inlet,
specific_volume_compressor_outlet, \
        specific_volume_turbine_inlet,
specific_volume_turbine_outlet_with_losses,
specific_volume_afterburnet_outlet, specific_volume_nozzle_outlet =
calculation(
            state,
            altitude, mach_number, compressor_pressure_ratio,
temperature_turbine_inlet, intake_efficiency,
            n_c,
            n_b,
            n_t, n_n, n_m, n_ab, combustion_pressure_loss,
temperature_afterburner_out, nozzle_area,
            mass_flow_rate_air,
            specific_heat_ratio_air,
specific_heat_ratio_combustion_gas, c_pg, fuel_calorific)
        if state == True:
            enthalpy_append = [enthalpy_ambient,
enthalpy_compressor_inlet, enthalpy_compressor_outlet,
                               enthalpy_turbine_inlet,
enthalpy_turbine_outlet_with_losses, enthalpy_afterburner_outlet,
                               enthalpy_nozzle_outlet]
```

```
            entropy_append = [entropy_ambient,
entropy_compressor_inlet, entropy_compressor_outlet,
                              entropy_turbine_inlet,
                              entropy_turbine_outlet_with_losses,
entropy_afterburner_outlet, entropy_nozzle_outlet]

            stations = ['a', '1', '2', '3', '4', '5', '6']
        else:
            enthalpy_append = [enthalpy_ambient,
enthalpy_compressor_inlet, enthalpy_compressor_outlet,
                               enthalpy_turbine_inlet,
enthalpy_turbine_outlet_with_losses,
                               enthalpy_nozzle_outlet]
            entropy_append = [entropy_ambient,
entropy_compressor_inlet, entropy_compressor_outlet,
                              entropy_turbine_inlet,
                              entropy_turbine_outlet_with_losses,
entropy_nozzle_outlet]

            stations = ['a', '1', '2', '3', '4', '5']

        for i, txt in enumerate(stations):
            plt.text(entropy_append[i], enthalpy_append[i], txt,
fontsize=10, ha='right', va='bottom')
        plt.title('Entropy-Enthalpy')
        plt.plot(entropy_append, enthalpy_append)
        plt.xlabel('Entropy(kJ/kgK)')
        plt.ylabel('Enthalpy(kJ/kg)')
        plt.show()

    def p_v_plot(self):
        altitude = float(self.lineEdit.text())
        mach_number = float(self.lineEdit_2.text())
        compressor_pressure_ratio = float(self.lineEdit_6.text())
        temperature_turbine_inlet = float(self.lineEdit_7.text())  # K
        intake_efficiency = float(self.lineEdit_8.text())
        n_c = float(self.lineEdit_9.text())
        n_b = float(self.lineEdit_10.text())
        n_t = float(self.lineEdit_11.text())
        n_n = float(self.lineEdit_12.text())
        n_m = float(self.lineEdit_13.text())
        n_ab = float(self.lineEdit_14.text())
        combustion_pressure_loss = float(self.lineEdit_15.text())
        temperature_afterburner_out = float(self.lineEdit_16.text())
        nozzle_area = float(self.lineEdit_17.text())  # m^2
        mass_flow_rate_air = float(self.lineEdit_18.text())  # kg/s
        specific_heat_ratio_air = float(self.lineEdit_19.text())
        specific_heat_ratio_combustion_gas =
float(self.lineEdit_20.text())
        c_pa = 1.005
        c_pg = float(self.lineEdit_21.text())
        gas_constant = 0.287  # kJ/(kg*K)
        fuel_calorific = float(self.lineEdit_22.text())  # kJ/kg
        state = self.checkBox_2.isChecked()
        if state == True:
            temperature_afterburner_out =
float(self.lineEdit_16.text())
        else:
```

```
            temperature_afterburner_out = 0
            pressure_afterburner_outlet = 0
        pressure_a, temperature_a, velocity_a, pressure_turbine_outlet,
pressure_compressor_inlet, \
        temperature_compressor_inlet, intake_pressure_ratio,
pressure_compressor_outlet, temperature_compressor_outlet, \
        temperature_turbine_outlet,
temperature_turbine_outlet_with_losses, pressure_turbine_inlet,
fuel_air_ratio_combustion, \
        pressure_afterburner_outlet, fuel_air_ratio_afterburner,
temperature_nozzle_outlet, pressure_nozzle_outlet, \
        velocity_nozzle_outlet, massflow_nozzle_out, net_thrust,
specific_thrust, thrust_spesific_fuel_consumption, \
        enthalpy_ambient, enthalpy_compressor_inlet,
enthalpy_compressor_outlet, enthalpy_turbine_inlet, \
        enthalpy_turbine_outlet_with_losses,
enthalpy_afterburner_outlet, enthalpy_nozzle_outlet, entropy_ambient, \
        entropy_compressor_inlet, entropy_compressor_outlet,
entropy_turbine_inlet, entropy_turbine_outlet_with_losses, \
        entropy_afterburner_outlet, entropy_nozzle_outlet,
specific_volume_ambient, specific_volume_compressor_inlet,
specific_volume_compressor_outlet, \
        specific_volume_turbine_inlet,
specific_volume_turbine_outlet_with_losses,
specific_volume_afterburnet_outlet, specific_volume_nozzle_outlet =
calculation(
            state,
            altitude, mach_number, compressor_pressure_ratio,
temperature_turbine_inlet, intake_efficiency,
            n_c,
            n_b,
            n_t, n_n, n_m, n_ab, combustion_pressure_loss,
temperature_afterburner_out, nozzle_area,
            mass_flow_rate_air,
            specific_heat_ratio_air,
specific_heat_ratio_combustion_gas, c_pg, fuel_calorific)
        if state == True:
            pressure_append = [pressure_a, pressure_compressor_inlet,
pressure_compressor_outlet,
                               pressure_turbine_inlet,
                               pressure_turbine_outlet,
pressure_afterburner_outlet, pressure_nozzle_outlet]
            specific_volume_append = [specific_volume_ambient,
specific_volume_compressor_inlet,

specific_volume_compressor_outlet,
                                      specific_volume_turbine_inlet,

specific_volume_turbine_outlet_with_losses,
specific_volume_afterburnet_outlet,
                                      specific_volume_nozzle_outlet]
            stations = ['a', '1', '2', '3', '4', '5', '6']
        else:
            pressure_append = [pressure_a, pressure_compressor_inlet,
pressure_compressor_outlet,
                               pressure_turbine_inlet,
                               pressure_turbine_outlet,
pressure_nozzle_outlet]
```

```
            specific_volume_append = [specific_volume_ambient,
specific_volume_compressor_inlet,

specific_volume_compressor_outlet,
                                      specific_volume_turbine_inlet,

specific_volume_turbine_outlet_with_losses,
                                      specific_volume_nozzle_outlet]
            stations = ['a', '1', '2', '3', '4', '5']

        for i, txt in enumerate(stations):
            plt.text(specific_volume_append[i], pressure_append[i],
txt, fontsize=10, ha='right', va='bottom')
        plt.title('Pressure-Specific Volume')
        plt.plot(specific_volume_append, pressure_append)
        plt.ylabel('Pressure(kPa)')
        plt.xlabel('Specific Volume(m³/kg)')
        plt.show()

    def klein(self):
        try:
            for i in range(101):
                self.progressBar.setValue(i)
                QApplication.processEvents()
            altitude = float(self.lineEdit.text())
            mach_number = float(self.lineEdit_2.text())
            compressor_pressure_ratio = float(self.lineEdit_6.text())
            temperature_turbine_inlet = float(self.lineEdit_7.text())
# K
            intake_efficiency = float(self.lineEdit_8.text())
            n_c = float(self.lineEdit_9.text())
            n_b = float(self.lineEdit_10.text())
            n_t = float(self.lineEdit_11.text())
            n_n = float(self.lineEdit_12.text())
            n_m = float(self.lineEdit_13.text())
            n_ab = float(self.lineEdit_14.text())
            combustion_pressure_loss = float(self.lineEdit_15.text())
            temperature_afterburner_out =
float(self.lineEdit_16.text())
            nozzle_area = float(self.lineEdit_17.text())  # m^2
            mass_flow_rate_air = float(self.lineEdit_18.text())  # kg/s
            specific_heat_ratio_air = float(self.lineEdit_19.text())
            specific_heat_ratio_combustion_gas =
float(self.lineEdit_20.text())
            c_pa = 1.005
            c_pg = float(self.lineEdit_21.text())
            gas_constant = 0.287  # kJ/(kg*K)
            fuel_calorific = float(self.lineEdit_22.text())  # kJ/kg
            state = self.checkBox_2.isChecked()
            pressure_a, temperature_a, velocity_a,
pressure_turbine_outlet, pressure_compressor_inlet, \
            temperature_compressor_inlet, intake_pressure_ratio,
pressure_compressor_outlet, temperature_compressor_outlet, \
            temperature_turbine_outlet,
temperature_turbine_outlet_with_losses, pressure_turbine_inlet,
fuel_air_ratio_combustion, \
            pressure_afterburner_outlet, fuel_air_ratio_afterburner,
temperature_nozzle_outlet, pressure_nozzle_outlet, \
```

```
            velocity_nozzle_outlet, massflow_nozzle_out, net_thrust,
specific_thrust, thrust_spesific_fuel_consumption, \
            enthalpy_ambient, enthalpy_compressor_inlet,
enthalpy_compressor_outlet, enthalpy_turbine_inlet, \
            enthalpy_turbine_outlet_with_losses,
enthalpy_afterburner_outlet, enthalpy_nozzle_outlet, entropy_ambient, \
            entropy_compressor_inlet, entropy_compressor_outlet,
entropy_turbine_inlet, entropy_turbine_outlet_with_losses, \
            entropy_afterburner_outlet, entropy_nozzle_outlet,
specific_volume_ambient, specific_volume_compressor_inlet,
specific_volume_compressor_outlet, \
            specific_volume_turbine_inlet,
specific_volume_turbine_outlet_with_losses,
specific_volume_afterburnet_outlet, specific_volume_nozzle_outlet =
calculation(
                state,
                altitude, mach_number, compressor_pressure_ratio,
temperature_turbine_inlet, intake_efficiency,
                n_c,
                n_b,
                n_t, n_n, n_m, n_ab, combustion_pressure_loss,
temperature_afterburner_out, nozzle_area,
                mass_flow_rate_air,
                specific_heat_ratio_air,
specific_heat_ratio_combustion_gas, c_pg, fuel_calorific)

            # printing on window
            self.label_22.setText(
                f"Inlet Conditions:\n{pressure_a:.1f}
kPa\n{temperature_a:.1f} K\n{velocity_a:.1f}
m/s\n{enthalpy_ambient:.1f} kJ/kg\nMach Number: {mach_number:.3f}")
            self.label_21.setText(
                f"Compressor Inlet:\n{pressure_compressor_inlet:.1f}
kPa\n{temperature_compressor_inlet:.1f} K\nPressure ratio:
{compressor_pressure_ratio:.0f}\n{enthalpy_compressor_inlet:.1f}
kJ/kg")
            self.label_29.setText(
                f"Compressor Outlet:\n{pressure_compressor_outlet:.1f}
kPa\n{temperature_compressor_outlet:.1f}
K\n{enthalpy_compressor_outlet:.1f} kJ/kg")
            self.label_25.setText(f"{fuel_calorific:.1f} kJ/kg\nFuel-
Air Ratio: {fuel_air_ratio_combustion:.5f}")
            self.label_23.setText(
                f"Turbine Inlet: \n{pressure_turbine_inlet:.1f}
kPa\n{temperature_turbine_inlet:.1f} K\n{enthalpy_turbine_inlet:.1f}
kJ/kg")
            self.label_30.setText(
                f"Turbine Outlet: \n{pressure_turbine_outlet:.1f}
kPa\n{temperature_turbine_outlet_with_losses:.1f}
K\n{enthalpy_turbine_outlet_with_losses:.1f} kJ/kg")
            state = self.checkBox_2.isChecked()
            if state == True:
                self.label_26.setText(
                    f"Afterburner Outlet:
\n{pressure_afterburner_outlet:.1f} kPa\nFuel-Air Ratio:
{fuel_air_ratio_afterburner:.5f}\n{enthalpy_afterburner_outlet:.1f}
kJ/kg")
```

```
            else:
                self.label_26.setText("")

            self.label_25.setText(f"{fuel_calorific:.1f} kJ/kg\nFuel-
Air Ratio: {fuel_air_ratio_combustion:.5f}")
            self.label_24.setText(
                f"{pressure_nozzle_outlet:.1f}
kPa\n{temperature_nozzle_outlet:.1f} K\n{enthalpy_nozzle_outlet:.1f}
kJ/kg\n{velocity_nozzle_outlet:.2f} m/s")
            self.label_9.setText(
                f"Net Thrust: {net_thrust / 1000:.1f} kN\nSpecific
Thrust: {specific_thrust:.5f} kN.s/kg\nThrust Specific Fuel
Consumption: {thrust_spesific_fuel_consumption:.4f} kg/(kN.s)")
            veri_setleri = [
                {"Stations": "Ambient", 'Temperature (K)':
temperature_a, 'Pressure (kPa)': pressure_a,
                 'Enthalpy (kJ/kg)': enthalpy_ambient, "Entropy
(kJ/kgK)": entropy_ambient, "Velocity m/s": velocity_a,
                 "Net Thrust (kN)": net_thrust / 1000},
                {"Stations": "Compressor Inlet", 'Temperature (K)':
temperature_compressor_inlet,
                 'Pressure (kPa)': pressure_compressor_inlet, 'Enthalpy
(kJ/kg)': enthalpy_compressor_inlet,
                 "Entropy (kJ/kgK)": entropy_compressor_inlet},
                {"Stations": "Compressor Outlet", 'Temperature (K)':
temperature_compressor_outlet,
                 'Pressure (kPa)': pressure_compressor_outlet,
'Enthalpy (kJ/kg)': enthalpy_compressor_outlet,
                 "Entropy (kJ/kgK)": entropy_compressor_outlet},
                {"Stations": "Turbine Inlet", 'Temperature (K)':
temperature_turbine_inlet,
                 'Pressure (kPa)': pressure_turbine_inlet, 'Enthalpy
(kJ/kg)': enthalpy_turbine_inlet,
                 "Entropy (kJ/kgK)": entropy_turbine_inlet},
                {"Stations": "Turbine Outlet", 'Temperature (K)':
temperature_turbine_outlet_with_losses,
                 'Pressure (kPa)': pressure_turbine_outlet, 'Enthalpy
(kJ/kg)': enthalpy_turbine_outlet_with_losses,
                 "Entropy (kJ/kgK)":
entropy_turbine_outlet_with_losses},
                {"Stations": "Afterburner Outlet", 'Temperature (K)':
temperature_afterburner_out,
                 'Pressure (kPa)': pressure_afterburner_outlet,
'Enthalpy (kJ/kg)': enthalpy_afterburner_outlet,
                 "Entropy (kJ/kgK)": entropy_afterburner_outlet},
                {"Stations": "Nozzle Outlet", 'Temperature (K)':
temperature_nozzle_outlet,
                 'Pressure (kPa)': pressure_nozzle_outlet, 'Enthalpy
(kJ/kg)': enthalpy_compressor_outlet,
                 "Entropy (kJ/kgK)": entropy_nozzle_outlet,
                 "Velocity m/s": velocity_nozzle_outlet},
            ]
            df = pd.DataFrame(veri_setleri)
            basliklar = ['Stations', 'Temperature (K)', 'Pressure
(kPa)', 'Enthalpy (kJ/kg)', "Entropy (kJ/kgK)", "",
                         "Velocity m/s", "",
                         "Net Thrust (kN)"]
            df = pd.DataFrame(veri_setleri, columns=basliklar)
```

```
            df.to_excel('veriler.xlsx', index=False)
            self.statusbar.showMessage("The calculation was made
successfully and the data was transferred to Excel.")

        except Exception as e:

            self.show_error_message(str(e))

    def delta(self):
        try:
            for i in range(101):
                self.progressBar_2.setValue(i)
                QApplication.processEvents()
            veri_setleri = []
            altitude = float(self.lineEdit_50.text())
            mach_number = float(self.lineEdit_52.text())
            compressor_pressure_ratio = float(self.lineEdit_53.text())
            temperature_turbine_inlet = float(self.lineEdit_54.text())
# K
            intake_efficiency = float(self.lineEdit_55.text())
            n_c = float(self.lineEdit_56.text())
            n_b = float(self.lineEdit_57.text())
            n_t = float(self.lineEdit_58.text())
            n_n = float(self.lineEdit_59.text())
            n_m = float(self.lineEdit_60.text())
            n_ab = float(self.lineEdit_61.text())
            combustion_pressure_loss = float(self.lineEdit_62.text())
            state = self.checkBox.isChecked()
            if state == True:
                temperature_afterburner_out =
float(self.lineEdit_16.text())
            else:
                temperature_afterburner_out = 0
                pressure_afterburner_outlet = 0
            nozzle_area = float(self.lineEdit_64.text())  # m^2
            mass_flow_rate_air = float(self.lineEdit_65.text())  # kg/s
            specific_heat_ratio_air = float(self.lineEdit_66.text())
            specific_heat_ratio_combustion_gas =
float(self.lineEdit_67.text())
            c_pa = 1.005
            c_pg = float(self.lineEdit_68.text())
            gas_constant = 0.287  # kJ/(kg*K)
            fuel_calorific = float(self.lineEdit_69.text())  # kJ/kg
            delta = float(self.lineEdit_40.text())
            max_value = float(self.lineEdit_39.text())
            altitude_append.clear()
            net_thrust_append.clear()
            compressor_pressure_ratio_append.clear()
            mach_number_append.clear()
            temperature_a_append.clear()
            pressure_a_append.clear()
            enthalpy_ambient_append.clear()
            velocity_a_append.clear()
            temperature_compressor_inlet_append.clear()
            pressure_compressor_inlet_append.clear()
            enthalpy_compressor_inlet_append.clear()
            temperature_compressor_outlet_append.clear()
            pressure_compressor_outlet_append.clear()
```

```
            enthalpy_compressor_outlet_append.clear()
            temperature_turbine_inlet_append.clear()
            pressure_turbine_inlet_append.clear()
            enthalpy_turbine_inlet_append.clear()
            temperature_turbine_outlet_with_losses_append.clear()
            pressure_turbine_outlet_append.clear()
            enthalpy_turbine_outlet_with_losses_append.clear()
            temperature_afterburner_out_append.clear()
            afterburner_outlet_pressure_append.clear()
            enthalpy_afterburner_outlet_append.clear()
            temperature_nozzle_outlet_append.clear()
            pressure_nozzle_outlet_append.clear()
            enthalpy_nozzle_outlet_append.clear()
            velocity_nozzle_outlet_append.clear()
            fuel_calorific_ratio_append.clear()
            intake_efficiency_append.clear()
            n_c_append.clear()
            n_b_append.clear()
            n_t_append.clear()
            n_n_append.clear()
            n_ab_append.clear()
            n_m_append.clear()
            nozzle_area_append.clear()
            mass_flow_rate_air_append.clear()
            entropy_ambient_append.clear()
            entropy_compressor_inlet_append.clear()
            entropy_compressor_outlet_append.clear()
            entropy_turbine_inlet_append.clear()
            entropy_turbine_outlet_with_losses_append.clear()
            entropy_afterburner_outlet_append.clear()
            entropy_nozzle_outlet_append.clear()
            fuel_air_ratio_combustion_append.clear()
            fuel_air_ratio_afterburner_append.clear()

            if self.comboBox.currentText() == "Altitude (m)":
                current_value = altitude
            elif self.comboBox.currentText() == "Mach Number":
                current_value = mach_number
            elif self.comboBox.currentText() == "Compressor Pressure
Ratio":
                current_value = compressor_pressure_ratio
            elif self.comboBox.currentText() == "Turbine Inlet
Temperature(K)":
                current_value = temperature_turbine_inlet
            elif self.comboBox.currentText() == "Intake Efficiency(0-
1)":
                current_value = intake_efficiency
            elif self.comboBox.currentText() == "Compressor
Efficiency(0-1)":
                current_value = n_c
            elif self.comboBox.currentText() == "Combustion
Efficiency(0-1)":
                current_value = n_b
            elif self.comboBox.currentText() == "Turbine Efficiency(0-
1)":
                current_value = n_t
            elif self.comboBox.currentText() == "Propelling Nozzle
Efficiency(0-1)":
```

```
                current_value = n_n
            elif self.comboBox.currentText() == "Mechanical
Efficiency(0-1)":
                current_value = n_m
            elif self.comboBox.currentText() == "Afterburner
Efficiency(0-1)":
                current_value = n_ab
            elif self.comboBox.currentText() == "Combustion Pressure
Loss(0-1)":
                current_value = combustion_pressure_loss
            elif self.comboBox.currentText() == "Temperature
Afterburner Out(K)":
                current_value = temperature_afterburner_out
            elif self.comboBox.currentText() == "Nozzle Area (m^2)":
                current_value = nozzle_area
            elif self.comboBox.currentText() == "Mass Flow Rate
(kg/s)":
                current_value = mass_flow_rate_air

            while current_value <= max_value:

                if self.comboBox.currentText() == "Altitude (m)":
                    altitude = current_value

                    pressure_a, temperature_a, velocity_a,
pressure_turbine_outlet, pressure_compressor_inlet, \
                    temperature_compressor_inlet,
intake_pressure_ratio, pressure_compressor_outlet,
temperature_compressor_outlet, \
                    temperature_turbine_outlet,
temperature_turbine_outlet_with_losses, pressure_turbine_inlet,
fuel_air_ratio_combustion, \
                    pressure_afterburner_outlet,
fuel_air_ratio_afterburner, temperature_nozzle_outlet,
pressure_nozzle_outlet, \
                    velocity_nozzle_outlet, massflow_nozzle_out,
net_thrust, specific_thrust, thrust_spesific_fuel_consumption, \
                    enthalpy_ambient, enthalpy_compressor_inlet,
enthalpy_compressor_outlet, enthalpy_turbine_inlet, \
                    enthalpy_turbine_outlet_with_losses,
enthalpy_afterburner_outlet, enthalpy_nozzle_outlet, entropy_ambient, \
                    entropy_compressor_inlet,
entropy_compressor_outlet, entropy_turbine_inlet,
entropy_turbine_outlet_with_losses, \
                    entropy_afterburner_outlet, entropy_nozzle_outlet,
specific_volume_ambient, specific_volume_compressor_inlet,
specific_volume_compressor_outlet, \
                    specific_volume_turbine_inlet,
specific_volume_turbine_outlet_with_losses,
specific_volume_afterburnet_outlet, specific_volume_nozzle_outlet =
calculation(
                        state,
                        altitude, mach_number,
compressor_pressure_ratio, temperature_turbine_inlet,
intake_efficiency,
                        n_c,
                        n_b,
```

```
                        n_t, n_n, n_m, n_ab, combustion_pressure_loss,
temperature_afterburner_out, nozzle_area,
                        mass_flow_rate_air,
                        specific_heat_ratio_air,
specific_heat_ratio_combustion_gas, c_pg, fuel_calorific)

                elif self.comboBox.currentText() == "Mach Number":
                    mach_number = current_value
                    pressure_a, temperature_a, velocity_a,
pressure_turbine_outlet, pressure_compressor_inlet, \
                    temperature_compressor_inlet,
intake_pressure_ratio, pressure_compressor_outlet,
temperature_compressor_outlet, \
                    temperature_turbine_outlet,
temperature_turbine_outlet_with_losses, pressure_turbine_inlet,
fuel_air_ratio_combustion, \
                    pressure_afterburner_outlet,
fuel_air_ratio_afterburner, temperature_nozzle_outlet,
pressure_nozzle_outlet, \
                    velocity_nozzle_outlet, massflow_nozzle_out,
net_thrust, specific_thrust, thrust_spesific_fuel_consumption, \
                    enthalpy_ambient, enthalpy_compressor_inlet,
enthalpy_compressor_outlet, enthalpy_turbine_inlet, \
                    enthalpy_turbine_outlet_with_losses,
enthalpy_afterburner_outlet, enthalpy_nozzle_outlet, entropy_ambient, \
                    entropy_compressor_inlet,
entropy_compressor_outlet, entropy_turbine_inlet,
entropy_turbine_outlet_with_losses, \
                    entropy_afterburner_outlet, entropy_nozzle_outlet,
specific_volume_ambient, specific_volume_compressor_inlet,
specific_volume_compressor_outlet, \
                    specific_volume_turbine_inlet,
specific_volume_turbine_outlet_with_losses,
specific_volume_afterburnet_outlet, specific_volume_nozzle_outlet =
calculation(
                        state,
                        altitude, mach_number,
compressor_pressure_ratio, temperature_turbine_inlet,
intake_efficiency,
                        n_c,
                        n_b,
                        n_t, n_n, n_m, n_ab, combustion_pressure_loss,
temperature_afterburner_out, nozzle_area,
                        mass_flow_rate_air,
                        specific_heat_ratio_air,
specific_heat_ratio_combustion_gas, c_pg, fuel_calorific)
                elif self.comboBox.currentText() == "Compressor
Pressure Ratio":
                    compressor_pressure_ratio = current_value
                    pressure_a, temperature_a, velocity_a,
pressure_turbine_outlet, pressure_compressor_inlet, \
                    temperature_compressor_inlet,
intake_pressure_ratio, pressure_compressor_outlet,
temperature_compressor_outlet, \
                    temperature_turbine_outlet,
temperature_turbine_outlet_with_losses, pressure_turbine_inlet,
fuel_air_ratio_combustion, \
```

```
                    pressure_afterburner_outlet,
fuel_air_ratio_afterburner, temperature_nozzle_outlet,
pressure_nozzle_outlet, \
                    velocity_nozzle_outlet, massflow_nozzle_out,
net_thrust, specific_thrust, thrust_spesific_fuel_consumption, \
                    enthalpy_ambient, enthalpy_compressor_inlet,
enthalpy_compressor_outlet, enthalpy_turbine_inlet, \
                    enthalpy_turbine_outlet_with_losses,
enthalpy_afterburner_outlet, enthalpy_nozzle_outlet, entropy_ambient, \
                    entropy_compressor_inlet,
entropy_compressor_outlet, entropy_turbine_inlet,
entropy_turbine_outlet_with_losses, \
                    entropy_afterburner_outlet, entropy_nozzle_outlet,
specific_volume_ambient, specific_volume_compressor_inlet,
specific_volume_compressor_outlet, \
                    specific_volume_turbine_inlet,
specific_volume_turbine_outlet_with_losses,
specific_volume_afterburnet_outlet, specific_volume_nozzle_outlet =
calculation(
                        state,
                        altitude, mach_number,
compressor_pressure_ratio, temperature_turbine_inlet,
intake_efficiency,
                        n_c,
                        n_b,
                        n_t, n_n, n_m, n_ab, combustion_pressure_loss,
temperature_afterburner_out, nozzle_area,
                        mass_flow_rate_air,
                        specific_heat_ratio_air,
specific_heat_ratio_combustion_gas, c_pg, fuel_calorific)
                elif self.comboBox.currentText() == "Turbine Inlet
Temperature(K)":
                    temperature_turbine_inlet = current_value
                    pressure_a, temperature_a, velocity_a,
pressure_turbine_outlet, pressure_compressor_inlet, \
                    temperature_compressor_inlet,
intake_pressure_ratio, pressure_compressor_outlet,
temperature_compressor_outlet, \
                    temperature_turbine_outlet,
temperature_turbine_outlet_with_losses, pressure_turbine_inlet,
fuel_air_ratio_combustion, \
                    pressure_afterburner_outlet,
fuel_air_ratio_afterburner, temperature_nozzle_outlet,
pressure_nozzle_outlet, \
                    velocity_nozzle_outlet, massflow_nozzle_out,
net_thrust, specific_thrust, thrust_spesific_fuel_consumption, \
                    enthalpy_ambient, enthalpy_compressor_inlet,
enthalpy_compressor_outlet, enthalpy_turbine_inlet, \
                    enthalpy_turbine_outlet_with_losses,
enthalpy_afterburner_outlet, enthalpy_nozzle_outlet, entropy_ambient, \
                    entropy_compressor_inlet,
entropy_compressor_outlet, entropy_turbine_inlet,
entropy_turbine_outlet_with_losses, \
                    entropy_afterburner_outlet, entropy_nozzle_outlet,
specific_volume_ambient, specific_volume_compressor_inlet,
specific_volume_compressor_outlet, \
                    specific_volume_turbine_inlet,
specific_volume_turbine_outlet_with_losses,
```

```
specific_volume_afterburnet_outlet, specific_volume_nozzle_outlet =
calculation(
                        state,
                        altitude, mach_number,
compressor_pressure_ratio, temperature_turbine_inlet,
intake_efficiency,
                        n_c,
                        n_b,
                        n_t, n_n, n_m, n_ab, combustion_pressure_loss,
temperature_afterburner_out, nozzle_area,
                        mass_flow_rate_air,
                        specific_heat_ratio_air,
specific_heat_ratio_combustion_gas, c_pg, fuel_calorific)
                elif self.comboBox.currentText() == "Intake
Efficiency(0-1)":
                    intake_efficiency = current_value
                    pressure_a, temperature_a, velocity_a,
pressure_turbine_outlet, pressure_compressor_inlet, \
                    temperature_compressor_inlet,
intake_pressure_ratio, pressure_compressor_outlet,
temperature_compressor_outlet, \
                    temperature_turbine_outlet,
temperature_turbine_outlet_with_losses, pressure_turbine_inlet,
fuel_air_ratio_combustion, \
                    pressure_afterburner_outlet,
fuel_air_ratio_afterburner, temperature_nozzle_outlet,
pressure_nozzle_outlet, \
                    velocity_nozzle_outlet, massflow_nozzle_out,
net_thrust, specific_thrust, thrust_spesific_fuel_consumption, \
                    enthalpy_ambient, enthalpy_compressor_inlet,
enthalpy_compressor_outlet, enthalpy_turbine_inlet, \
                    enthalpy_turbine_outlet_with_losses,
enthalpy_afterburner_outlet, enthalpy_nozzle_outlet, entropy_ambient, \
                    entropy_compressor_inlet,
entropy_compressor_outlet, entropy_turbine_inlet,
entropy_turbine_outlet_with_losses, \
                    entropy_afterburner_outlet, entropy_nozzle_outlet,
specific_volume_ambient, specific_volume_compressor_inlet,
specific_volume_compressor_outlet, \
                    specific_volume_turbine_inlet,
specific_volume_turbine_outlet_with_losses,
specific_volume_afterburnet_outlet, specific_volume_nozzle_outlet =
calculation(
                        state,
                        altitude, mach_number,
compressor_pressure_ratio, temperature_turbine_inlet,
intake_efficiency,
                        n_c,
                        n_b,
                        n_t, n_n, n_m, n_ab, combustion_pressure_loss,
temperature_afterburner_out, nozzle_area,
                        mass_flow_rate_air,
                        specific_heat_ratio_air,
specific_heat_ratio_combustion_gas, c_pg, fuel_calorific)
                elif self.comboBox.currentText() == "Compressor
Efficiency(0-1)":
                    n_c = current_value
```

```
                    pressure_a, temperature_a, velocity_a,
pressure_turbine_outlet, pressure_compressor_inlet, \
                    temperature_compressor_inlet,
intake_pressure_ratio, pressure_compressor_outlet,
temperature_compressor_outlet, \
                    temperature_turbine_outlet,
temperature_turbine_outlet_with_losses, pressure_turbine_inlet,
fuel_air_ratio_combustion, \
                    pressure_afterburner_outlet,
fuel_air_ratio_afterburner, temperature_nozzle_outlet,
pressure_nozzle_outlet, \
                    velocity_nozzle_outlet, massflow_nozzle_out,
net_thrust, specific_thrust, thrust_spesific_fuel_consumption, \
                    enthalpy_ambient, enthalpy_compressor_inlet,
enthalpy_compressor_outlet, enthalpy_turbine_inlet, \
                    enthalpy_turbine_outlet_with_losses,
enthalpy_afterburner_outlet, enthalpy_nozzle_outlet, entropy_ambient, \
                    entropy_compressor_inlet,
entropy_compressor_outlet, entropy_turbine_inlet,
entropy_turbine_outlet_with_losses, \
                    entropy_afterburner_outlet, entropy_nozzle_outlet,
specific_volume_ambient, specific_volume_compressor_inlet,
specific_volume_compressor_outlet, \
                    specific_volume_turbine_inlet,
specific_volume_turbine_outlet_with_losses,
specific_volume_afterburnet_outlet, specific_volume_nozzle_outlet =
calculation(
                        state,
                        altitude, mach_number,
compressor_pressure_ratio, temperature_turbine_inlet,
intake_efficiency,
                        n_c,
                        n_b,
                        n_t, n_n, n_m, n_ab, combustion_pressure_loss,
temperature_afterburner_out, nozzle_area,
                        mass_flow_rate_air,
                        specific_heat_ratio_air,
specific_heat_ratio_combustion_gas, c_pg, fuel_calorific)
                elif self.comboBox.currentText() == "Combustion
Efficiency(0-1)":
                    n_b = current_value
                    pressure_a, temperature_a, velocity_a,
pressure_turbine_outlet, pressure_compressor_inlet, \
                    temperature_compressor_inlet,
intake_pressure_ratio, pressure_compressor_outlet,
temperature_compressor_outlet, \
                    temperature_turbine_outlet,
temperature_turbine_outlet_with_losses, pressure_turbine_inlet,
fuel_air_ratio_combustion, \
                    pressure_afterburner_outlet,
fuel_air_ratio_afterburner, temperature_nozzle_outlet,
pressure_nozzle_outlet, \
                    velocity_nozzle_outlet, massflow_nozzle_out,
net_thrust, specific_thrust, thrust_spesific_fuel_consumption, \
                    enthalpy_ambient, enthalpy_compressor_inlet,
enthalpy_compressor_outlet, enthalpy_turbine_inlet, \
                    enthalpy_turbine_outlet_with_losses,
enthalpy_afterburner_outlet, enthalpy_nozzle_outlet, entropy_ambient, \
```

```
                    entropy_compressor_inlet,
entropy_compressor_outlet, entropy_turbine_inlet,
entropy_turbine_outlet_with_losses, \
                    entropy_afterburner_outlet, entropy_nozzle_outlet,
specific_volume_ambient, specific_volume_compressor_inlet,
specific_volume_compressor_outlet, \
                    specific_volume_turbine_inlet,
specific_volume_turbine_outlet_with_losses,
specific_volume_afterburnet_outlet, specific_volume_nozzle_outlet =
calculation(
                        state,
                        altitude, mach_number,
compressor_pressure_ratio, temperature_turbine_inlet,
intake_efficiency,
                        n_c,
                        n_b,
                        n_t, n_n, n_m, n_ab, combustion_pressure_loss,
temperature_afterburner_out, nozzle_area,
                        mass_flow_rate_air,
                        specific_heat_ratio_air,
specific_heat_ratio_combustion_gas, c_pg, fuel_calorific)
                elif self.comboBox.currentText() == "Turbine
Efficiency(0-1)":
                    n_t = current_value
                    pressure_a, temperature_a, velocity_a,
pressure_turbine_outlet, pressure_compressor_inlet, \
                    temperature_compressor_inlet,
intake_pressure_ratio, pressure_compressor_outlet,
temperature_compressor_outlet, \
                    temperature_turbine_outlet,
temperature_turbine_outlet_with_losses, pressure_turbine_inlet,
fuel_air_ratio_combustion, \
                    pressure_afterburner_outlet,
fuel_air_ratio_afterburner, temperature_nozzle_outlet,
pressure_nozzle_outlet, \
                    velocity_nozzle_outlet, massflow_nozzle_out,
net_thrust, specific_thrust, thrust_spesific_fuel_consumption, \
                    enthalpy_ambient, enthalpy_compressor_inlet,
enthalpy_compressor_outlet, enthalpy_turbine_inlet, \
                    enthalpy_turbine_outlet_with_losses,
enthalpy_afterburner_outlet, enthalpy_nozzle_outlet, entropy_ambient, \
                    entropy_compressor_inlet,
entropy_compressor_outlet, entropy_turbine_inlet,
entropy_turbine_outlet_with_losses, \
                    entropy_afterburner_outlet, entropy_nozzle_outlet,
specific_volume_ambient, specific_volume_compressor_inlet,
specific_volume_compressor_outlet, \
                    specific_volume_turbine_inlet,
specific_volume_turbine_outlet_with_losses,
specific_volume_afterburnet_outlet, specific_volume_nozzle_outlet =
calculation(
                        state,
                        altitude, mach_number,
compressor_pressure_ratio, temperature_turbine_inlet,
intake_efficiency,
                        n_c,
                        n_b,
```

```
                        n_t, n_n, n_m, n_ab, combustion_pressure_loss,
temperature_afterburner_out, nozzle_area,
                        mass_flow_rate_air,
                        specific_heat_ratio_air,
specific_heat_ratio_combustion_gas, c_pg, fuel_calorific)
                elif self.comboBox.currentText() == "Propelling Nozzle
Efficiency(0-1)":
                    n_n = current_value
                    pressure_a, temperature_a, velocity_a,
pressure_turbine_outlet, pressure_compressor_inlet, \
                    temperature_compressor_inlet,
intake_pressure_ratio, pressure_compressor_outlet,
temperature_compressor_outlet, \
                    temperature_turbine_outlet,
temperature_turbine_outlet_with_losses, pressure_turbine_inlet,
fuel_air_ratio_combustion, \
                    pressure_afterburner_outlet,
fuel_air_ratio_afterburner, temperature_nozzle_outlet,
pressure_nozzle_outlet, \
                    velocity_nozzle_outlet, massflow_nozzle_out,
net_thrust, specific_thrust, thrust_spesific_fuel_consumption, \
                    enthalpy_ambient, enthalpy_compressor_inlet,
enthalpy_compressor_outlet, enthalpy_turbine_inlet, \
                    enthalpy_turbine_outlet_with_losses,
enthalpy_afterburner_outlet, enthalpy_nozzle_outlet, entropy_ambient, \
                    entropy_compressor_inlet,
entropy_compressor_outlet, entropy_turbine_inlet,
entropy_turbine_outlet_with_losses, \
                    entropy_afterburner_outlet, entropy_nozzle_outlet,
specific_volume_ambient, specific_volume_compressor_inlet,
specific_volume_compressor_outlet, \
                    specific_volume_turbine_inlet,
specific_volume_turbine_outlet_with_losses,
specific_volume_afterburnet_outlet, specific_volume_nozzle_outlet =
calculation(
                        state,
                        altitude, mach_number,
compressor_pressure_ratio, temperature_turbine_inlet,
intake_efficiency,
                        n_c,
                        n_b,
                        n_t, n_n, n_m, n_ab, combustion_pressure_loss,
temperature_afterburner_out, nozzle_area,
                        mass_flow_rate_air,
                        specific_heat_ratio_air,
specific_heat_ratio_combustion_gas, c_pg, fuel_calorific)
                elif self.comboBox.currentText() == "Mechanical
Efficiency(0-1)":
                    n_m = current_value
                    pressure_a, temperature_a, velocity_a,
pressure_turbine_outlet, pressure_compressor_inlet, \
                    temperature_compressor_inlet,
intake_pressure_ratio, pressure_compressor_outlet,
temperature_compressor_outlet, \
                    temperature_turbine_outlet,
temperature_turbine_outlet_with_losses, pressure_turbine_inlet,
fuel_air_ratio_combustion, \
```

```
                    pressure_afterburner_outlet,
fuel_air_ratio_afterburner, temperature_nozzle_outlet,
pressure_nozzle_outlet, \
                    velocity_nozzle_outlet, massflow_nozzle_out,
net_thrust, specific_thrust, thrust_spesific_fuel_consumption, \
                    enthalpy_ambient, enthalpy_compressor_inlet,
enthalpy_compressor_outlet, enthalpy_turbine_inlet, \
                    enthalpy_turbine_outlet_with_losses,
enthalpy_afterburner_outlet, enthalpy_nozzle_outlet, entropy_ambient, \
                    entropy_compressor_inlet,
entropy_compressor_outlet, entropy_turbine_inlet,
entropy_turbine_outlet_with_losses, \
                    entropy_afterburner_outlet, entropy_nozzle_outlet,
specific_volume_ambient, specific_volume_compressor_inlet,
specific_volume_compressor_outlet, \
                    specific_volume_turbine_inlet,
specific_volume_turbine_outlet_with_losses,
specific_volume_afterburnet_outlet, specific_volume_nozzle_outlet =
calculation(
                        state,
                        altitude, mach_number,
compressor_pressure_ratio, temperature_turbine_inlet,
intake_efficiency,
                        n_c,
                        n_b,
                        n_t, n_n, n_m, n_ab, combustion_pressure_loss,
temperature_afterburner_out, nozzle_area,
                        mass_flow_rate_air,
                        specific_heat_ratio_air,
specific_heat_ratio_combustion_gas, c_pg, fuel_calorific)
                elif self.comboBox.currentText() == "Afterburner
Efficiency(0-1)":
                    n_ab = current_value
                    pressure_a, temperature_a, velocity_a,
pressure_turbine_outlet, pressure_compressor_inlet, \
                    temperature_compressor_inlet,
intake_pressure_ratio, pressure_compressor_outlet,
temperature_compressor_outlet, \
                    temperature_turbine_outlet,
temperature_turbine_outlet_with_losses, pressure_turbine_inlet,
fuel_air_ratio_combustion, \
                    pressure_afterburner_outlet,
fuel_air_ratio_afterburner, temperature_nozzle_outlet,
pressure_nozzle_outlet, \
                    velocity_nozzle_outlet, massflow_nozzle_out,
net_thrust, specific_thrust, thrust_spesific_fuel_consumption, \
                    enthalpy_ambient, enthalpy_compressor_inlet,
enthalpy_compressor_outlet, enthalpy_turbine_inlet, \
                    enthalpy_turbine_outlet_with_losses,
enthalpy_afterburner_outlet, enthalpy_nozzle_outlet, entropy_ambient, \
                    entropy_compressor_inlet,
entropy_compressor_outlet, entropy_turbine_inlet,
entropy_turbine_outlet_with_losses, \
                    entropy_afterburner_outlet, entropy_nozzle_outlet,
specific_volume_ambient, specific_volume_compressor_inlet,
specific_volume_compressor_outlet, \
                    specific_volume_turbine_inlet,
specific_volume_turbine_outlet_with_losses,
```

```
specific_volume_afterburnet_outlet, specific_volume_nozzle_outlet =
calculation(
                        state,
                        altitude, mach_number,
compressor_pressure_ratio, temperature_turbine_inlet,
intake_efficiency,
                        n_c,
                        n_b,
                        n_t, n_n, n_m, n_ab, combustion_pressure_loss,
temperature_afterburner_out, nozzle_area,
                        mass_flow_rate_air,
                        specific_heat_ratio_air,
specific_heat_ratio_combustion_gas, c_pg, fuel_calorific)
                elif self.comboBox.currentText() == "Combustion
Pressure Loss(0-1)":
                    combustion_pressure_loss = current_value
                    pressure_a, temperature_a, velocity_a,
pressure_turbine_outlet, pressure_compressor_inlet, \
                    temperature_compressor_inlet,
intake_pressure_ratio, pressure_compressor_outlet,
temperature_compressor_outlet, \
                    temperature_turbine_outlet,
temperature_turbine_outlet_with_losses, pressure_turbine_inlet,
fuel_air_ratio_combustion, \
                    pressure_afterburner_outlet,
fuel_air_ratio_afterburner, temperature_nozzle_outlet,
pressure_nozzle_outlet, \
                    velocity_nozzle_outlet, massflow_nozzle_out,
net_thrust, specific_thrust, thrust_spesific_fuel_consumption, \
                    enthalpy_ambient, enthalpy_compressor_inlet,
enthalpy_compressor_outlet, enthalpy_turbine_inlet, \
                    enthalpy_turbine_outlet_with_losses,
enthalpy_afterburner_outlet, enthalpy_nozzle_outlet, entropy_ambient, \
                    entropy_compressor_inlet,
entropy_compressor_outlet, entropy_turbine_inlet,
entropy_turbine_outlet_with_losses, \
                    entropy_afterburner_outlet, entropy_nozzle_outlet,
specific_volume_ambient, specific_volume_compressor_inlet,
specific_volume_compressor_outlet, \
                    specific_volume_turbine_inlet,
specific_volume_turbine_outlet_with_losses,
specific_volume_afterburnet_outlet, specific_volume_nozzle_outlet =
calculation(
                        state,
                        altitude, mach_number,
compressor_pressure_ratio, temperature_turbine_inlet,
intake_efficiency,
                        n_c,
                        n_b,
                        n_t, n_n, n_m, n_ab, combustion_pressure_loss,
temperature_afterburner_out, nozzle_area,
                        mass_flow_rate_air,
                        specific_heat_ratio_air,
specific_heat_ratio_combustion_gas, c_pg, fuel_calorific)
                elif self.comboBox.currentText() == "Temperature
Afterburner Out(K)":
                    temperature_afterburner_out = current_value
```

```
                    pressure_a, temperature_a, velocity_a,
pressure_turbine_outlet, pressure_compressor_inlet, \
                    temperature_compressor_inlet,
intake_pressure_ratio, pressure_compressor_outlet,
temperature_compressor_outlet, \
                    temperature_turbine_outlet,
temperature_turbine_outlet_with_losses, pressure_turbine_inlet,
fuel_air_ratio_combustion, \
                    pressure_afterburner_outlet,
fuel_air_ratio_afterburner, temperature_nozzle_outlet,
pressure_nozzle_outlet, \
                    velocity_nozzle_outlet, massflow_nozzle_out,
net_thrust, specific_thrust, thrust_spesific_fuel_consumption, \
                    enthalpy_ambient, enthalpy_compressor_inlet,
enthalpy_compressor_outlet, enthalpy_turbine_inlet, \
                    enthalpy_turbine_outlet_with_losses,
enthalpy_afterburner_outlet, enthalpy_nozzle_outlet, entropy_ambient, \
                    entropy_compressor_inlet,
entropy_compressor_outlet, entropy_turbine_inlet,
entropy_turbine_outlet_with_losses, \
                    entropy_afterburner_outlet, entropy_nozzle_outlet,
specific_volume_ambient, specific_volume_compressor_inlet,
specific_volume_compressor_outlet, \
                    specific_volume_turbine_inlet,
specific_volume_turbine_outlet_with_losses,
specific_volume_afterburnet_outlet, specific_volume_nozzle_outlet =
calculation(
                        state,
                        altitude, mach_number,
compressor_pressure_ratio, temperature_turbine_inlet,
intake_efficiency,
                        n_c,
                        n_b,
                        n_t, n_n, n_m, n_ab, combustion_pressure_loss,
temperature_afterburner_out, nozzle_area,
                        mass_flow_rate_air,
                        specific_heat_ratio_air,
specific_heat_ratio_combustion_gas, c_pg, fuel_calorific)
                elif self.comboBox.currentText() == "Nozzle Area
(m^2)":
                    nozzle_area = current_value
                    pressure_a, temperature_a, velocity_a,
pressure_turbine_outlet, pressure_compressor_inlet, \
                    temperature_compressor_inlet,
intake_pressure_ratio, pressure_compressor_outlet,
temperature_compressor_outlet, \
                    temperature_turbine_outlet,
temperature_turbine_outlet_with_losses, pressure_turbine_inlet,
fuel_air_ratio_combustion, \
                    pressure_afterburner_outlet,
fuel_air_ratio_afterburner, temperature_nozzle_outlet,
pressure_nozzle_outlet, \
                    velocity_nozzle_outlet, massflow_nozzle_out,
net_thrust, specific_thrust, thrust_spesific_fuel_consumption, \
                    enthalpy_ambient, enthalpy_compressor_inlet,
enthalpy_compressor_outlet, enthalpy_turbine_inlet, \
                    enthalpy_turbine_outlet_with_losses,
enthalpy_afterburner_outlet, enthalpy_nozzle_outlet, entropy_ambient, \
```

```
                    entropy_compressor_inlet,
entropy_compressor_outlet, entropy_turbine_inlet,
entropy_turbine_outlet_with_losses, \
                    entropy_afterburner_outlet, entropy_nozzle_outlet,
specific_volume_ambient, specific_volume_compressor_inlet,
specific_volume_compressor_outlet, \
                    specific_volume_turbine_inlet,
specific_volume_turbine_outlet_with_losses,
specific_volume_afterburnet_outlet, specific_volume_nozzle_outlet =
calculation(
                        state,
                        altitude, mach_number,
compressor_pressure_ratio, temperature_turbine_inlet,
intake_efficiency,
                        n_c,
                        n_b,
                        n_t, n_n, n_m, n_ab, combustion_pressure_loss,
temperature_afterburner_out, nozzle_area,
                        mass_flow_rate_air,
                        specific_heat_ratio_air,
specific_heat_ratio_combustion_gas, c_pg, fuel_calorific)
                elif self.comboBox.currentText() == "Mass Flow Rate
(kg/s)":
                    mass_flow_rate_air = current_value
                    pressure_a, temperature_a, velocity_a,
pressure_turbine_outlet, pressure_compressor_inlet, \
                    temperature_compressor_inlet,
intake_pressure_ratio, pressure_compressor_outlet,
temperature_compressor_outlet, \
                    temperature_turbine_outlet,
temperature_turbine_outlet_with_losses, pressure_turbine_inlet,
fuel_air_ratio_combustion, \
                    pressure_afterburner_outlet,
fuel_air_ratio_afterburner, temperature_nozzle_outlet,
pressure_nozzle_outlet, \
                    velocity_nozzle_outlet, massflow_nozzle_out,
net_thrust, specific_thrust, thrust_spesific_fuel_consumption, \
                    enthalpy_ambient, enthalpy_compressor_inlet,
enthalpy_compressor_outlet, enthalpy_turbine_inlet, \
                    enthalpy_turbine_outlet_with_losses,
enthalpy_afterburner_outlet, enthalpy_nozzle_outlet, entropy_ambient, \
                    entropy_compressor_inlet,
entropy_compressor_outlet, entropy_turbine_inlet,
entropy_turbine_outlet_with_losses, \
                    entropy_afterburner_outlet, entropy_nozzle_outlet,
specific_volume_ambient, specific_volume_compressor_inlet,
specific_volume_compressor_outlet, \
                    specific_volume_turbine_inlet,
specific_volume_turbine_outlet_with_losses,
specific_volume_afterburnet_outlet, specific_volume_nozzle_outlet =
calculation(
                        state,
                        altitude, mach_number,
compressor_pressure_ratio, temperature_turbine_inlet,
intake_efficiency,
                        n_c,
                        n_b,
```

```
                        n_t, n_n, n_m, n_ab, combustion_pressure_loss,
temperature_afterburner_out, nozzle_area,
                        mass_flow_rate_air,
                        specific_heat_ratio_air,
specific_heat_ratio_combustion_gas, c_pg, fuel_calorific)

                current_value += delta

                veri_setleri.append({
                    "Altitude (m)": altitude,
                    "Mach Number": mach_number,
                    'Temperature Ambient (K)': temperature_a,
                    'Pressure Ambient (kPa)': pressure_a,
                    'Enthalpy Ambient (kJ/kg)': enthalpy_ambient,
                    "Entropy Ambient (kJ/kgK)": entropy_ambient,
                    "Velocity Ambient m/s": velocity_a,
                    "Temperature Compressor Inlet (K)":
temperature_compressor_inlet,
                    "Pressure Compressor Inlet (K)":
pressure_compressor_inlet,
                    "Enthaply Compressor Inlet (kJ/kg)":
enthalpy_compressor_inlet,
                    "Entropy Compressor Inlet  (kJ/kgK)":
entropy_compressor_inlet,
                    "Temperature Compressor Outlet(K)":
temperature_compressor_outlet,
                    "Pressure Compressor Outlet (kPa)":
pressure_compressor_outlet,
                    "Enthalpy Compressor Outlet (kJ/kg)":
enthalpy_compressor_outlet,
                    "Entropy Compressor Outlet (kJ/kgK)":
entropy_compressor_outlet,
                    "Temperature Turbine Inlet (K)":
temperature_turbine_inlet,
                    "Pressure Turbine Inlet (kPa)":
pressure_turbine_inlet,
                    "Enthalpy Turbine Inlet (kJ/kg)":
enthalpy_turbine_inlet,
                    "Entropy Turbine Inlet  (kJ/kgK)":
entropy_turbine_inlet,
                    "Temperature Turbine Outlet (K)":
temperature_turbine_outlet_with_losses,
                    "Pressure Turbine Outlet (kPa)":
pressure_turbine_outlet,
                    "Enthalpy Turbine Outlet (kJ/kg)":
enthalpy_turbine_outlet_with_losses,
                    "Entropy Turbine outlet (kJ/kgK)":
entropy_turbine_outlet_with_losses,
                    "Temperature Afterburner Outlet (K)":
temperature_afterburner_out,
                    "Pressure Afterburner Outlet (kPa)":
pressure_afterburner_outlet,
                    "Enthalpy Afterburner Outlet (kJ/kg)":
enthalpy_afterburner_outlet,
                    "Entropy AFterburner Outlet (kJ/kgK)":
entropy_afterburner_outlet,
                    "Temperature Nozzle Outlet (K)":
temperature_nozzle_outlet,
```

```
                    "Pressure Nozzle Outlet (kPa)":
pressure_nozzle_outlet,
                    "Enthalpy Nozzle Outlet (kJ/kg)":
enthalpy_nozzle_outlet,
                    "Entropy Nozzle Outlet (kJ/kgK)":
entropy_nozzle_outlet,
                    "Velocity Nozzle Outlet (m/s)":
velocity_nozzle_outlet,
                    "Net Thrust (kN)": net_thrust / 1000
                })
                df = pd.DataFrame(veri_setleri)
                basliklar = ["Altitude (m)", "Mach Number",
'Temperature Ambient (K)', 'Pressure Ambient (kPa)',
                             'Enthalpy Ambient (kJ/kg)', "Entropy
Ambient (kJ/kgK)", "Velocity Ambient m/s",
                             "Temperature Compressor Inlet (K)",
                             "Temperature Compressor Inlet (K)",
"Pressure Compressor Inlet (K)",
                             "Enthaply Compressor Inlet (kJ/kg)",
"Entropy Compressor Inlet  (kJ/kgK)",
                             "Temperature Compressor Outlet(K)"
                    , "Pressure Compressor Outlet (kPa)", "Enthalpy
Compressor Outlet (kJ/kg)",
                             "Entropy Compressor Outlet (kJ/kgK)",
                             "Temperature Turbine Inlet (K)",
                             "Pressure Turbine Inlet (kPa)", "Enthalpy
Turbine Inlet (kJ/kg)",
                             "Entropy Turbine Inlet  (kJ/kgK)",
                             "Temperature Turbine Outlet (K)"
                    , "Pressure Turbine Outlet (kPa)", "Enthalpy
Turbine Outlet (kJ/kg)",
                             "Entropy Turbine outlet (kJ/kgK)"
                    , "Temperature Afterburner Outlet (K)", "Pressure
Afterburner Outlet (kPa)"
                    , "Enthalpy Afterburner Outlet (kJ/kg)", "Entropy
AFterburner Outlet (kJ/kgK)",
                             "Temperature Nozzle Outlet (K)"
                    , "Pressure Nozzle Outlet (kPa)", "Enthalpy Nozzle
Outlet (kJ/kg)", "Entropy Nozzle Outlet (kJ/kgK)"
                    , "Velocity Nozzle Outlet (m/s)", "Net Thrust
(kN)"]
                df = pd.DataFrame(veri_setleri, columns=basliklar)
                df.to_excel('delta-veriler.xlsx', index=False)
                self.statusbar.showMessage(
                    "The calculation was made successfully and the data
was transferred to Excel. You can plot your graph by selecting the
parameters you want and see the results on the graph.")
                altitude_append.append(altitude)
                net_thrust_append.append(net_thrust)
                mach_number_append.append(mach_number)
                temperature_a_append.append(temperature_a)
                pressure_a_append.append(pressure_a)
                enthalpy_ambient_append.append(enthalpy_ambient)
                velocity_a_append.append(velocity_a)

temperature_compressor_inlet_append.append(temperature_compressor_inlet
)
```

```
pressure_compressor_inlet_append.append(pressure_compressor_inlet)

enthalpy_compressor_inlet_append.append(enthalpy_compressor_inlet)

temperature_compressor_outlet_append.append(temperature_compressor_outl
et)

pressure_compressor_outlet_append.append(pressure_compressor_outlet)

enthalpy_compressor_outlet_append.append(enthalpy_compressor_outlet)

temperature_turbine_inlet_append.append(temperature_turbine_inlet)

pressure_turbine_inlet_append.append(pressure_turbine_inlet)

enthalpy_turbine_inlet_append.append(enthalpy_turbine_inlet)

temperature_turbine_outlet_with_losses_append.append(temperature_turbin
e_outlet_with_losses)

pressure_turbine_outlet_append.append(pressure_turbine_outlet)

enthalpy_turbine_outlet_with_losses_append.append(enthalpy_turbine_outl
et_with_losses)

temperature_afterburner_out_append.append(temperature_afterburner_out)

afterburner_outlet_pressure_append.append(pressure_afterburner_outlet)

enthalpy_afterburner_outlet_append.append(enthalpy_afterburner_outlet)

temperature_nozzle_outlet_append.append(temperature_nozzle_outlet)

pressure_nozzle_outlet_append.append(pressure_nozzle_outlet)

enthalpy_nozzle_outlet_append.append(enthalpy_nozzle_outlet)

velocity_nozzle_outlet_append.append(velocity_nozzle_outlet)

compressor_pressure_ratio_append.append(compressor_pressure_ratio)
                intake_efficiency_append.append(intake_efficiency)
                n_c_append.append(n_c)
                n_b_append.append(n_b)
                n_t_append.append(n_t)
                n_n_append.append(n_n)
                n_ab_append.append(n_ab)
                n_m_append.append(n_m)
                nozzle_area_append.append(nozzle_area)
                mass_flow_rate_air_append.append(mass_flow_rate_air)
                entropy_ambient_append.append(entropy_ambient)

entropy_compressor_inlet_append.append(entropy_compressor_inlet)

entropy_compressor_outlet_append.append(entropy_compressor_outlet)

entropy_turbine_inlet_append.append(entropy_turbine_inlet)
```

```
entropy_turbine_outlet_with_losses_append.append(entropy_turbine_outlet
_with_losses)

entropy_afterburner_outlet_append.append(entropy_afterburner_outlet)

entropy_nozzle_outlet_append.append(entropy_nozzle_outlet)
                fuel_calorific_ratio_append.append(fuel_calorific)

fuel_air_ratio_combustion_append.append(fuel_air_ratio_combustion)

fuel_air_ratio_afterburner_append.append(fuel_air_ratio_afterburner)



        except Exception as e:

            self.show_error_message(str(e))


if __name__ == "__main__":
    app = QApplication(sys.argv)
    Windows = MyApp()
    Windows.show()

sys.exit(app.exec_())
```

8.3. Calculation Codes

```
from constant_spesific_heat import cp
import CoolProp.CoolProp as CoolProp
from find_entropy_with_temperature import find_entropy_with_temperature
import math


def calculation(state, altitude, mach_number,
compressor_pressure_ratio, temperature_turbine_inlet,
intake_efficiency, n_c,
                n_b,
                n_t, n_n, n_m, n_ab, combustion_pressure_loss,
temperature_afterburner_out, nozzle_area,
                mass_flow_rate_air,
                specific_heat_ratio_air,
specific_heat_ratio_combustion_gas, c_pg, fuel_calorific):
    if altitude < 11000:
        temperature_a = 288.15 - 0.0065 * altitude
        density_a = 1.225 * (1 - 22.558 * 10 ** -6 * altitude) **
4.2559
        pressure_a = 101.325 * (1 - 22.558 * 10 ** -6 * altitude) **
5.2559
        m_air = CoolProp.PropsSI("M", "P", pressure_a * 1000, "T",
temperature_a, "AIR")
        speed_of_sound = (specific_heat_ratio_air * 8.31 *
temperature_a / m_air) ** (1 / 2)
        velocity_a = mach_number * speed_of_sound
```

```
    e = math.e
    c_pa = 1.005
    gas_constant = 0.287  # kJ/(kg*K)
    temperature_compressor_inlet = temperature_a + velocity_a ** 2 /
(2000 * c_pa)
    intake_pressure_ratio = (1 + intake_efficiency * velocity_a ** 2 /
(2000 * c_pa * temperature_a)) ** (
            specific_heat_ratio_air / (specific_heat_ratio_air - 1))
    pressure_compressor_inlet = pressure_a * (1 +
((specific_heat_ratio_air - 1) / 2) * mach_number ** 2) ** (
            specific_heat_ratio_air / (specific_heat_ratio_air - 1))
    pressure_compressor_outlet = compressor_pressure_ratio *
pressure_compressor_inlet
    temperature_compressor_outlet = temperature_compressor_inlet +
temperature_compressor_inlet / n_c * (
            (pressure_compressor_outlet / pressure_compressor_inlet) **
(
            (specific_heat_ratio_air - 1) / specific_heat_ratio_air) -
1)
    temperature_turbine_outlet = temperature_turbine_inlet - c_pa * (
            temperature_compressor_outlet -
temperature_compressor_inlet) / (c_pg * n_m)
    pressure_turbine_inlet = pressure_compressor_outlet * (
            1 - combustion_pressure_loss / pressure_compressor_outlet)
    temperature_turbine_outlet_with_losses = temperature_turbine_inlet
- 1 / n_t * (
            temperature_turbine_inlet - temperature_turbine_outlet)
    pressure_turbine_outlet = pressure_turbine_inlet * (
            temperature_turbine_outlet_with_losses /
temperature_turbine_inlet) ** (

specific_heat_ratio_combustion_gas / (

specific_heat_ratio_combustion_gas - 1))
    fuel_air_ratio_combustion = (c_pg * temperature_turbine_inlet -
c_pa * temperature_compressor_inlet) / (
            n_b * fuel_calorific - c_pg * temperature_turbine_inlet)

    p_critical = pressure_turbine_outlet * (1 - 1 / n_n * (
                (specific_heat_ratio_combustion_gas - 1) /
(specific_heat_ratio_combustion_gas + 1))) ** (
                               specific_heat_ratio_combustion_gas /
(specific_heat_ratio_combustion_gas - 1))
    if state == True :
        print("afterburner aktif")
        pressure_afterburner_outlet = pressure_turbine_outlet

        if pressure_afterburner_outlet/pressure_a >
pressure_afterburner_outlet/p_critical:
            print("the nozzle is choked")
            temperature_nozzle_outlet = (2 /
(specific_heat_ratio_combustion_gas + 1)) * temperature_afterburner_out
            velocity_nozzle_outlet = (gas_constant *
specific_heat_ratio_combustion_gas * temperature_nozzle_outlet)
            pressure_nozzle_outlet = p_critical
        else:
            print("the nozzle unchoked")
            pressure_nozzle_outlet = pressure_a
```

```
            temperature_nozzle_outlet = temperature_afterburner_out
            velocity_nozzle_outlet =
(2*c_pg*(temperature_afterburner_out-temperature_nozzle_outlet))

        fuel_air_ratio_afterburner = (1 + fuel_air_ratio_combustion) *
(
                c_pg * temperature_afterburner_out - c_pg *
temperature_turbine_outlet) / (
                                                n_ab * fuel_calorific -
c_pg * temperature_afterburner_out)
    elif state == False:
        print("afterburner aktif değil")
        pressure_afterburner_outlet = pressure_turbine_outlet
        temperature_afterburner_out =
temperature_turbine_outlet_with_losses

        if pressure_afterburner_outlet / pressure_a >
pressure_afterburner_outlet / p_critical:
            print("the nozzle is choking")
            temperature_nozzle_outlet = (2 /
(specific_heat_ratio_combustion_gas + 1)) *
temperature_turbine_outlet_with_losses
            velocity_nozzle_outlet = (gas_constant *
specific_heat_ratio_combustion_gas * temperature_nozzle_outlet)
            pressure_nozzle_outlet = p_critical
        else:
            print("the nozzle unchoked")
            pressure_nozzle_outlet = pressure_a
            temperature_nozzle_outlet =
temperature_turbine_outlet_with_losses
            velocity_nozzle_outlet = (2 * c_pg *
(temperature_turbine_outlet_with_losses - temperature_nozzle_outlet))
        fuel_air_ratio_afterburner = 0


    #pressure_nozzle_outlet = pressure_afterburner_outlet * (
            #(1 + ((specific_heat_ratio_air - 1) / 2) * mach_number **
2) ** (
            #specific_heat_ratio_air / (specific_heat_ratio_air - 1)))
** -1

    density_nozzle_outlet = pressure_afterburner_outlet / (gas_constant
* temperature_nozzle_outlet)
    massflow_nozzle_out = density_nozzle_outlet * nozzle_area *
velocity_nozzle_outlet

    net_thrusht_v2 = mass_flow_rate_air * ((
                                                1 +
fuel_air_ratio_combustion + fuel_air_ratio_afterburner) *
velocity_nozzle_outlet - velocity_a) + nozzle_area * (
                            pressure_nozzle_outlet - pressure_a)

    net_thrust = massflow_nozzle_out * velocity_nozzle_outlet -
mass_flow_rate_air * velocity_a
    specific_thrust = net_thrust / mass_flow_rate_air
    thrust_spesific_fuel_consumption = (mass_flow_rate_air +
massflow_nozzle_out) / specific_thrust
```

```
    ###
    enthalpy_ambient = cp(temperature_a) * temperature_a
    enthalpy_compressor_inlet = cp(temperature_compressor_inlet) *
temperature_compressor_inlet
    enthalpy_compressor_outlet = cp(temperature_compressor_outlet) *
temperature_compressor_outlet
    enthalpy_turbine_inlet = cp(temperature_turbine_inlet) *
temperature_turbine_inlet
    enthalpy_turbine_outlet_with_losses = cp(
        temperature_turbine_outlet_with_losses) *
temperature_turbine_outlet_with_losses
    enthalpy_afterburner_outlet = cp(temperature_afterburner_out) *
temperature_afterburner_out
    enthalpy_nozzle_outlet = cp(temperature_nozzle_outlet) *
temperature_nozzle_outlet

    entropy_ambient = find_entropy_with_temperature(temperature_a)
    entropy_compressor_inlet = entropy_ambient + cp((entropy_ambient +
temperature_compressor_inlet) / 2) * math.log(
        temperature_compressor_inlet / temperature_a, e) - gas_constant
* math.log(
        pressure_compressor_inlet / pressure_a, e)
    entropy_compressor_outlet = entropy_compressor_inlet + cp(
        (temperature_compressor_inlet + temperature_compressor_outlet)
/ 2) * math.log(
        temperature_compressor_outlet / temperature_compressor_inlet,
e) - gas_constant * math.log(
        pressure_compressor_outlet / pressure_compressor_inlet, e)
    entropy_turbine_inlet = entropy_compressor_outlet + cp(
        (temperature_turbine_inlet + temperature_compressor_outlet) /
2) * math.log(
        temperature_turbine_inlet / temperature_compressor_outlet, e) -
gas_constant * math.log(
        pressure_turbine_inlet / pressure_compressor_outlet, e)
    entropy_turbine_outlet_with_losses = entropy_turbine_inlet + cp(
        (temperature_turbine_inlet +
temperature_turbine_outlet_with_losses) / 2) * math.log(
        temperature_turbine_outlet_with_losses /
temperature_turbine_inlet, e) - gas_constant * math.log(
        pressure_turbine_outlet / pressure_turbine_inlet, e)
    entropy_afterburner_outlet = entropy_turbine_outlet_with_losses +
cp(
        (temperature_afterburner_out +
temperature_turbine_outlet_with_losses) / 2) * math.log(
        temperature_afterburner_out /
temperature_turbine_outlet_with_losses, e) - gas_constant * math.log(
        pressure_afterburner_outlet / pressure_turbine_outlet, e)
    entropy_nozzle_outlet = entropy_afterburner_outlet + cp(
        (temperature_afterburner_out + temperature_nozzle_outlet) / 2)
* math.log(
        temperature_nozzle_outlet / temperature_afterburner_out, e) -
gas_constant * math.log(
        pressure_nozzle_outlet / pressure_afterburner_outlet, e)

    density_compressor_inlet = pressure_compressor_inlet /
(gas_constant * temperature_compressor_inlet)
```

```
    density_compressor_outlet = pressure_compressor_outlet /
(gas_constant * temperature_compressor_outlet)
    density_turbine_inlet = pressure_turbine_inlet / (gas_constant *
temperature_turbine_inlet)
    density_turbine_outlet_with_losses = pressure_turbine_outlet / (
            gas_constant * temperature_turbine_outlet_with_losses)
    density_afterburner_outlet = pressure_afterburner_outlet /
(gas_constant * temperature_afterburner_out)
    density_nozzle_outlet = pressure_nozzle_outlet / (gas_constant *
temperature_nozzle_outlet)

    specific_volume_ambient = 1 / density_a
    specific_volume_compressor_inlet = 1 / density_compressor_inlet
    specific_volume_compressor_outlet = 1 / density_compressor_outlet
    specific_volume_turbine_inlet = 1 / density_turbine_inlet
    specific_volume_turbine_outlet_with_losses = 1 /
density_turbine_outlet_with_losses
    specific_volume_afterburnet_outlet = 1 / density_afterburner_outlet
    specific_volume_nozzle_outlet = 1 / density_nozzle_outlet

    parameters = pressure_a, temperature_a, velocity_a,
pressure_turbine_outlet, pressure_compressor_inlet,
temperature_compressor_inlet, intake_pressure_ratio,
pressure_compressor_outlet, temperature_compressor_outlet,
temperature_turbine_outlet, temperature_turbine_outlet_with_losses,
pressure_turbine_inlet, fuel_air_ratio_combustion,
pressure_afterburner_outlet, fuel_air_ratio_afterburner,
temperature_nozzle_outlet, pressure_nozzle_outlet,
velocity_nozzle_outlet, massflow_nozzle_out, net_thrust,
specific_thrust, thrust_spesific_fuel_consumption, enthalpy_ambient,
enthalpy_compressor_inlet, enthalpy_compressor_outlet,
enthalpy_turbine_inlet, enthalpy_turbine_outlet_with_losses,
enthalpy_afterburner_outlet, enthalpy_nozzle_outlet

    return pressure_a, temperature_a, velocity_a,
pressure_turbine_outlet, pressure_compressor_inlet, \
           temperature_compressor_inlet, intake_pressure_ratio,
pressure_compressor_outlet, temperature_compressor_outlet, \
           temperature_turbine_outlet,
temperature_turbine_outlet_with_losses, pressure_turbine_inlet,
fuel_air_ratio_combustion, \
           pressure_afterburner_outlet, fuel_air_ratio_afterburner,
temperature_nozzle_outlet, pressure_nozzle_outlet, \
           velocity_nozzle_outlet, massflow_nozzle_out, net_thrust,
specific_thrust, thrust_spesific_fuel_consumption, \
           enthalpy_ambient, enthalpy_compressor_inlet,
enthalpy_compressor_outlet, enthalpy_turbine_inlet, \
           enthalpy_turbine_outlet_with_losses,
enthalpy_afterburner_outlet, enthalpy_nozzle_outlet, entropy_ambient, \
           entropy_compressor_inlet, entropy_compressor_outlet,
entropy_turbine_inlet, entropy_turbine_outlet_with_losses, \
           entropy_afterburner_outlet, entropy_nozzle_outlet,
specific_volume_ambient, specific_volume_compressor_inlet,
specific_volume_compressor_outlet, \
           specific_volume_turbine_inlet,
specific_volume_turbine_outlet_with_losses,
specific_volume_afterburnet_outlet, specific_volume_nozzle_outlet
```