



**NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF MECHANICAL ENGINEERING
SECTION OF FLUDIS
LABORATORY OF THERMAL TURBOMACHINES**

**AEROSPACE PROPULSION SIMULATION
SOFTWARE COMPARISON BETWEEN
NPSS AND PROOSIS**

Diploma Thesis
Angelos Mexis

Supervision: N. Aretakis

Athens
February 2020



**NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF MECHANICAL ENGINEERING
SECTION OF FLUDIS
LABORATORY OF THERMAL TURBOMACHINES**

Abstract

**AEROSPACE PROPULSION SIMULATION SOFTWARE COMPARISON
BETWEEN NPSS AND PROOSIS**

Angelos Mexis

Aerospace industry is continuously advancing to new technologies, hence scientists need appropriate tools in their effort to pioneer and push forward the frontiers of technology. PROOSIS and NPSS are two simulation software with vast abilities and a wide area of application, especially in the fields of gas turbine engines and are the dominant programs currently used by large engine manufacturers. NPSS has been developed by NASA and is mainly used by companies in U.S.A., whereas PROOSIS is developed and used in Europe. However, companies commonly operate one of the two programs, whereas there are cases where companies that are part of both consortiums are obliged to use both software simultaneously. This imposes great complications in collaborative projects where each company uses a different modelling tool. Thus, it would be of high interest to find a way to interface these powerful tools and facilitate the development process of aerospace engines. This report serves as the first step in the interfacing process, presents the main differences between the programs as well as an analytic comparison between the simulation results of the two, focusing however in NPSS since PROOSIS is already used as the main tool of the Laboratory of Thermal Turbomachines at NTUA, Athens. An initial overview of NPSS and PROOSIS is presented, followed by individual component and complete engine model generation in an attempt to better understand the modeling philosophy and the programming differences between the two. Then, a comparison of the thermodynamic simulation results will be presented and finally, some ideas on interfacing these programs will be discussed.

ACKNOWLEDGEMENTS

At this point I would like to express my sincere appreciation and gratitude to my supervisor, N. Aretakis, for giving me the opportunity to work on such an interesting topic.

Special thanks also expressed to A. Alexiou for his invaluable advice and guidance over the duration of this diploma thesis, as well as to all the faculty of the Laboratory of Thermal Turbomachines.

Moreover, I would also like to express my deep gratitude for professors K. Mathioudakis and K. C. Giannakoglou for introducing me to the fascinating world of thermal turbomachines and empowering me with valuable knowledge and skills.

Finally, I would like to thank my family and my friends for their support, encouragement and advice throughout my life and during my years of study.

TABLE OF CONTENTS

1	INTRODUCTION.....	1.1
2	SOFTWARE DESCRIPTION.....	2.1
2.1	PROOSIS.....	2.1
2.2	NPSS.....	2.3
3	MODEL GENERATION.....	3.1
3.1	PROOSIS modeling.....	3.1
3.2	NPSS modeling.....	3.4
4	THE NPSS SOLVER, SUBELEMENTS AND PORTS	4.1
4.1	Dependent and Independent variables	4.1
4.1.1	Constraints	4.2
4.2	Elements, Subelements and Sockets	4.3
4.3	Design and Off-Design Variables.....	4.5
4.3.1	InletStart.....	4.5
4.3.2	Compressor	4.5
4.3.3	Turbine.....	4.5
4.3.4	Shaft	4.6
4.3.5	Nozzle	4.6
4.3.6	Turbojet.....	4.7
4.3.7	Turbofan.....	4.10
4.3.8	Turboshaft.....	4.11
4.4	Ports.....	4.13
5	NPSS MAP INTEGRATION IN PROOSIS	5.1
5.1	Compressor R-line map	5.1
5.2	Turbine PR map	5.3
5.3	NPSS map integration process	5.3

6	NPSS FLUID MODEL INTEGRATION IN PROOSIS.....	6.1
7	COMPONENT LEVEL COMPARISON	7.1
7.1	Inlet.....	7.1
7.1.1	PROOSIS	7.1
7.1.2	NPSS	7.4
7.1.3	Results Comparison	7.5
7.2	Ambient.....	7.6
7.2.1	PROOSIS	7.6
7.2.2	NPSS	7.6
7.2.3	Results Comparison	7.6
7.3	Duct	7.8
7.3.1	PROOSIS	7.8
7.3.2	NPSS	7.8
7.3.3	Results Comparison	7.10
7.4	Compressor.....	7.12
7.4.1	PROOSIS	7.12
7.4.2	NPSS	7.12
7.4.3	Results Comparison	7.15
7.4.4	Fan.....	7.16
7.5	Burner	7.16
7.5.1	PROOSIS	7.16
7.5.2	NPSS	7.17
7.5.3	Results Comparison	7.18
7.6	Turbine.....	7.20
7.6.1	PROOSIS	7.20
7.6.2	NPSS	7.21
7.6.3	Results Comparison	7.21
7.7	Nozzle	7.22
7.7.1	PROOSIS	7.22
7.7.2	NPSS	7.23
7.7.3	Nozzle coefficient comparison	7.25
7.7.4	Results Comparison	7.27
8	ENGINE LEVEL COMPARISON.....	8.1

8.1 Turbojet	8.1
8.1.1 PROOSIS	8.1
8.1.2 NPSS	8.2
8.1.3 Results Comparison	8.8
8.2 Turbofan	8.10
8.2.1 PROOSIS	8.10
8.2.2 NPSS	8.11
8.2.3 Results Comparison	8.20
8.3 Turboshaft	8.23
8.3.1 PROOSIS	8.23
8.3.2 NPSS	8.23
8.3.3 Results Comparison	8.25
9 SUMMARY AND FUTURE WORK	9.1
10 ΕΚΤΕΤΑΜΕΝΗ ΠΕΡΙΛΗΨΗ (ΕΛΛΗΝΙΚΑ)	10.1
10.1 Εισαγωγή	10.2
10.2 Περιγραφή Λογισμικόν	10.4
10.2.1 PROOSIS.....	10.4
10.2.2 NPSS.....	10.5
10.3 Δημιουργία Μοντέλων.....	10.7
10.3.1 Μοντελοποίηση στο PROOSIS	10.7
10.3.2 Μοντελοποίηση στο NPSS	10.8
10.4 Ο solver του NPSS, τα subelements και τα ports	10.12
10.4.1 Εξαρτημένες και ανεξάρτητες μεταβλητές	10.12
10.4.2 Περιορισμοί.....	10.13
10.4.3 Elements, Subelements και Sockets	10.13
10.4.4 Μεταβλητές σε Design και Off-Design	10.15
10.4.5 Ports	10.22
10.5 Ενσωμάτωση των χαρτών επιδόσεων του NPSS στο PROOSIS	10.24
10.5.1 Χάρτης συμπιεστή (R-line map)	10.25
10.5.2 Χάρτης στροβίλου (Turbine PR map)	10.25
10.5.3 Διαδικασία ενσωμάτωσης χαρτών	10.26
10.6 Ενσωμάτωση του θερμοδυναμικού μοντέλου του NPSS στο PROOSIS.	10.26

10.7 Σύγκριση σε επίπεδο συνιστωσών	10.31
10.7.1 Αγωγός εισόδου.....	10.31
10.7.2 Ατμοσφαιρικές συνθήκες	10.32
10.7.3 Αγωγός.....	10.33
10.7.4 Συμπιεστής.....	10.34
10.7.5 Θάλαμος καύσης	10.36
10.7.6 Στρόβιλος.....	10.38
10.7.7 Ακροφύσιο.....	10.39
10.8 Σύγκριση σε επίπεδο κινητήρα	10.42
10.8.1 Turbojet.....	10.42
10.8.2 Turbofan	10.45
10.8.3 Turboshaft.....	10.49
10.9 Ανακεφαλαίωση και μελλοντική εργασία.....	10.52
11 REFERENCES	11.1

TABLE OF FIGURES

Figure 1.1 NPSS applications	1.2
Figure 2.1 Typical PROOSIS interface structure	2.2
Figure 2.2 NPSS interface	2.4
Figure 3.1 Creating a new library in PROOSIS.....	3.2
Figure 3.2 Creating a new schematic in PROOSIS	3.2
Figure 3.3 Schematic of a simple Turbojet model in PROOSIS	3.3
Figure 3.4 Typical layout of the monitor window in PROOSIS	3.3
Figure 3.5 Running a model in NPSS	3.7
Figure 3.6 Running a model in a subfolder and printing results.....	3.7
Figure 4.1 Defining constraints in NPSS	4.2
Figure 4.2 NPSS typical compressor element schematic, [25].....	4.4
Figure 4.3 NPSS typical compressor schematic with map subelement, [25].....	4.4
Figure 5.1 Excerpt of the Fan R-line map file	5.2
Figure 6.1 Comparison of NPSS and PROOSIS fluid model enthalpy.....	6.3
Figure 6.2 NPSS and PROOSIS fluid model enthalpy percentage differences	6.3
Figure 6.3 Comparison of NPSS and PROOSIS fluid model entropy	6.4
Figure 6.4 NPSS and PROOSIS fluid model entropy percentage differences	6.4
Figure 6.5 Comparison of NPSS and PROOSIS fluid model isentropic coefficient	6.5
Figure 6.6 NPSS and PROOSIS fluid model isentropic coefficient percentage differences	6.5
Figure 7.1 Inlet schematic diagram in PROOSIS environment.....	7.2
Figure 7.2 Editing attributes of the Inlet component in PROOSIS	7.2
Figure 7.3 Creating a new experiment in PROOSIS	7.3

Figure 7.4 Setting-up the experiment in PROOSIS	7.3
Figure 7.5 Monitor window in PROOSIS	7.4
Figure 7.6 Ambient schematic diagram in PROOSIS environment.....	7.6
Figure 7.7 Duct schematic diagram in PROOSIS environment	7.8
Figure 7.8 Duct outlet Mach number versus dynamic head loss factor	7.11
Figure 7.9 Duct fractional pressure loss versus dynamic head loss factor	7.11
Figure 7.10 Compressor schematic diagram in PROOSIS environment	7.12
Figure 7.11 Compressor off-design performance in PROOSIS and NPSS.....	7.16
Figure 7.12 Burner schematic diagram in PROOSIS environment.....	7.17
Figure 7.13 Burner exhaust temperature against fuel-to-air ratio.....	7.19
Figure 7.14 Burner exhaust temperature against fuel-to-air-ratio between the allFuel NPSS thermodynamic package and PROOSIS JP4 fluid model	7.20
Figure 7.15 Turbine schematic diagram in PROOSIS environment	7.21
Figure 7.16 Turbine off-design performance in PROOSIS and NPSS	7.22
Figure 7.17 Nozzle schematic diagram in PROOSIS environment	7.23
Figure 7.18 Nozzle performance comparison in PROOSIS and NPSS	7.28
Figure 8.1 Turbojet schematic in PROOSIS environment	8.2
Figure 8.2 Turbojet schematic on NPSS, [26]	8.3
Figure 8.3 PROOSIS-NPSS turbojet cycle percentage differences at design point.....	8.8
Figure 8.4 PROOSIS-NPSS turbojet cycle percentage differences at off-design	8.9
Figure 8.5 PROOSIS-NPSS turbojet main performance parameter comparison.....	8.10
Figure 8.6 Turbofan schematic in PROOSIS environment	8.11
Figure 8.7 Turbofan schematic in NPSS.....	8.12
Figure 8.8 NPSS Turbofan modeling option 1, [26].....	8.18

Figure 8.9 NPSS Turbofan modeling option 2, [26].....	8.19
Figure 8.10 NPSS Turbofan modeling option 3, [26]	8.19
Figure 8.11 PROOSIS-NPSS turbofan cycle percentage differences at design point ..	8.21
Figure 8.12 PROOSIS-NPSS turbofan cycle percentage differences at off-design.....	8.21
Figure 8.13 PROOSIS-NPSS turbofan main performance parameter comparison.....	8.22
Figure 8.14 PROOSIS-NPSS turbofan mechanical speed comparison.....	8.22
Figure 8.15 Turboshaft schematic in PROOSIS environment	8.23
Figure 8.16 PROOSIS-NPSS turboshaft cycle percentage differences at design point	8.27
Figure 8.17 PROOSIS-NPSS turboshaft engine cycle percentage differences at off-design	8.27
Figure 8.18 PROOSIS-NPSS turboshaft main performance parameters comparison ..	8.28

LIST OF TABLES

Table 4.1 Turbojet independent variables at design point	4.7
Table 4.2 Turbojet dependent variables at Design point	4.7
Table 4.3 Turbojet independent variables at Off-Design	4.9
Table 4.4 Turbojet dependent variables at Off-Design.....	4.9
Table 4.5 Turbofan independent variables at Design point	4.10
Table 4.6 Turbofan dependent variables at Design point	4.10
Table 4.7 Turbofan independent variables at Off-Design	4.11
Table 4.8 Turbofan dependent variables at Off-Design	4.11
Table 4.9 Turboshaft independent variables at Design point	4.12
Table 4.10 Turboshaft dependent variables at Design point	4.12
Table 4.11 Turboshaft independent variables at Off-Design.....	4.12
Table 4.12 Turboshaft dependent variables at Off-Design.....	4.12
Table 4.13 Fluid port variables comparison.....	4.13
Table 4.14 Fuel port variables comparison.....	4.14
Table 4.15 Mechanical port variables comparison	4.14
Table 7.1 PROOSIS-NPSS results comparison: Inlet component	7.5
Table 7.2 Ambient component simulation; considered flight conditions.....	7.6
Table 7.3 PROOSIS Ambient component simulation results	7.7
Table 7.4 NPSS Ambient element simulation results.....	7.7
Table 7.5 Input data for Compressor component comparison.....	7.15
Table 7.6 Input data for Burner component comparison	7.18
Table 7.7 PROOSIS-NPSS result comparison: Burner component	7.19

Table 7.8 PROOSIS Turbine component experiment boundaries	7.22
Table 7.9 Nozzle coefficient comparison	7.25
Table 7.10 Input data for Nozzle component comparison.....	7.27
Table 8.1 Turbojet design point input data	8.8
Table 8.2 Turbofan design point input data.....	8.20
Table 8.3 Turbofan design point input data.....	8.26

1 INTRODUCTION

The implementation of new technologies in the aerospace propulsion industry is becoming progressively expensive, as it requires real-size experimental tests and certifications to capture the complex interactions among the multiple components and disciplines present in complex systems such as gas turbine engines.

Numerical simulation software has provided engineers the ability to create various system models and components, especially of gas turbine engines, and evaluate new concepts early in the design phase. It has also allowed for rapid assessment of operational problems through computational simulations, especially in cases where problems were encountered in conditions that would be difficult to simulate experimentally. These programs have reduced significantly the development and evaluation costs that once were required through repeated testing and re-designing of large scale products [1].

The frantic progress taking place in computational engineering the past years in conjunction with the rapid increase in computing power and parallel computing has rendered these models ever so powerful and has enriched their capabilities, their fidelity and their correlation between simulated and real life operating conditions.

The dominant commercial gas turbine engine simulation software currently used, among others, by large engine manufacturers, is PROOSIS and NPSS.

PROOSIS is mainly used in Europe and is the main tool of the Laboratory of Thermal Turbomachines, whereas NPSS, developed by NASA, is used in U.S.A. as well as in other countries. Considering the fact that NPSS is a new program for the laboratory, this report will emphasize on the abilities and modeling philosophy of NPSS.

NPSS has been used in a large number of projects either as the main tool or not. For example it has been used for Nuclear Thermal Rocket (NTR) simulations [2], which is a technology concerning manned missions on Mars. NPSS has also been used for various space propulsion modeling [3], modeling boundary layer ingestion [4] and optimization of inlet shape design of a N2B hybrid wing body configuration [5]. Moreover, it has been used as a scramjet combustor flow solver [6], where the flow in the combustion chamber is supersonic and for the development of an open rotor cycle model as well [7]. In recent years, extensive studies for the development of hybrid electric propulsion [8],[9] and

electrified aircraft propulsion (EAP) [10] have been conducted using NPSS. New technologies, such as propulsion systems for vertical lift aircraft [11],[12],[13] and electrical power system sizing [14] have been developed using NPSS. Finally, moving towards electrification, electrical ports and elements have been developed [15], whereas battery models will be developed soon.

It now is clear that NPSS is a very strong tool used in a wide field of applications and is always at the forefront of technological advancements, likewise PROOSIS is. However, the majority of companies operate one or even both programs when they are members of both consortiums. This is very critical in collaborative projects where different companies contribute in different parts of an engine, as it is not currently possible for PROOSIS files to be processed on NPSS and vice versa.

This report serves as the first step of a possible interface and attempts to evaluate differences between the two programs as far as calculations and thermodynamic packages are concerned as well as differences in model generation and simulation.



Figure 1.1 NPSS applications

2

SOFTWARE DESCRIPTION

In the present chapter a quick presentation of the software used for the purpose of this project will be given. At first we will give a brief word about the history of the programs. In addition, we will discuss about their abilities, the tools they provide and the fields in which they are currently used. Finally, we will present how each program generates a model of a gas turbine engine and performs a simulation. It is however important, to mention that PROOSIS 3.10.0 and NPSS 2.8 student version were used for the purpose of this comparison.

2.1 PROOSIS

PROOSIS (Propulsion Object Oriented Simulation Software) is a standalone, flexible and extendible object-oriented simulation environment for modelling aeronautical gas turbine engines and other systems (control, electrical, thermal, hydraulic, mechanical, etc.). It was originally developed within the integrated European project VIVACE (Value Improvement through a Virtual Aeronautical Collaborative Enterprise) by a consortium of European universities, research institutes and corporate companies. It is based on EcosimPro, a simulation tool developed by Empresarios Agrupados Internacional S.A., the European Space Agency's preferred tool for rocket propulsion, environmental control and life support systems, among others. PROOSIS has all the capabilities of EcosimPro, plus some additional capabilities required for simulating aeronautical gas turbines, such as performance maps handling, multipoint design tools, design with constraints, etc. In addition, it provides the TURBO toolkit with typical components for modeling any type of gas turbine engine.

More about PROOSIS

PROOSIS is capable of steady state and transient simulations as well as customer deck generation. Different types of calculations can be performed (single or multipoint,

design, off-design, test analysis, sensitivity, parametric and optimization studies, mission analysis, diagnostics, control system design and test, etc.).

PROOSIS is also capable of performing multi-fidelity, multi-disciplinary and distributed simulations. These are greatly facilitated by its open architecture, which allows it to connect to external commercial or company specific tools (Excel, MATLAB, CFD, FEA) and link with codes written in C, C++ and FORTRAN.

These features make PROOSIS a useful tool for all phases of the engine life cycle, from preliminary and detailed design to post-certification and in-service support, and allow it to serve as a common framework in multi-partner collaborative engine projects providing common standards and methodologies.

Lastly, PROOSIS provides a multi-domain simulation platform for the simulation of gas turbines, engine/aircraft systems and power plants.

PROOSIS has an advanced Graphical User Interface and uses a high level, object oriented language (EL) for modelling engine systems and state-of-the-art technologies in areas such as numerical solvers, non-causal modelling of reusable libraries, XML file formats, map handling etc. Using EL, users can also create new components and libraries, or extend the existing ones [16],[17],[18],[19].

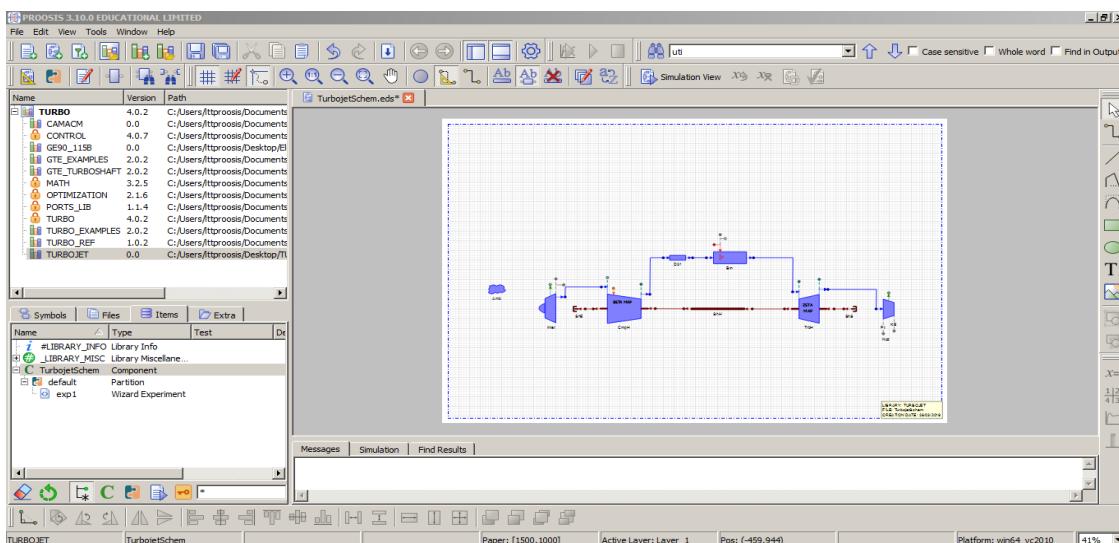


Figure 2.1 Typical PROOSIS interface structure

Any gas turbine engine configuration or system can be created graphically in the new schematic window, using components from the included libraries.

The **Component** contains a mathematical description of the corresponding real-world component such as compressor, turbine, burner, nozzle, shaft, etc. Components communicate with each other via their **Ports**. Ports define the set of variables to be conveyed between the connected components (e.g., mass flow rate, pressure and temperature in a Fluid Port, rotational speed and torque in a Mechanical Port).

Components and Ports are stored in the reusable included libraries and regarding gas turbine engines, they are located in the TURBO library.

Using the existing libraries and its components, a user can generate a model graphically by “dragging-and-dropping” the required component symbols from the included libraries to the schematic window, connecting the components through the appropriate ports and editing their attributes. A model can be a simple compressor component, a sub-assembly or a complete engine with its subsystems. Finally, a model can be consisted of components from different libraries, provided they share the same ports.

2.2 NPSS

NPSS (Numerical Propulsion System Simulation) is an object oriented, multi-physics, engineering design and simulation environment which enables development, collaboration and seamless integration of system models. Primary application areas for NPSS include aerospace systems (i.e. engine performance models for aircraft propulsion), thermodynamic system analysis such as Rankine and Brayton cycles, various rocket propulsion cycles, and industry standardization for model sharing and integration [20],[21]. It was originally developed by NASA Glenn Research Center in collaboration with the U.S. Government, aerospace industry and universities aiming to create an advanced multidisciplinary and multifidelity solver to be as generic as possible [22]. Later, NASA signed over control, maintenance and development of the code to an industrial consortium. Finally, in 2013, the responsibility of the code was transferred solely to Southwest Research Institute (SwRI).

More about NPSS

Since the program is fundamentally a flow-network solver, it has also been applied to a variety of other fluid/thermal subjects such as multi-phase heat transfer systems, refrigeration cycles, variations of common power cycles (i.e. Brayton), and overall vehicle emission analyses.

NPSS provides users with a wide range of solution modes, such as design, off-design, transient, parametric, mission analysis and test data matching.

The interface of the program is conducted through a command window, where various input files and simulations can be launched.

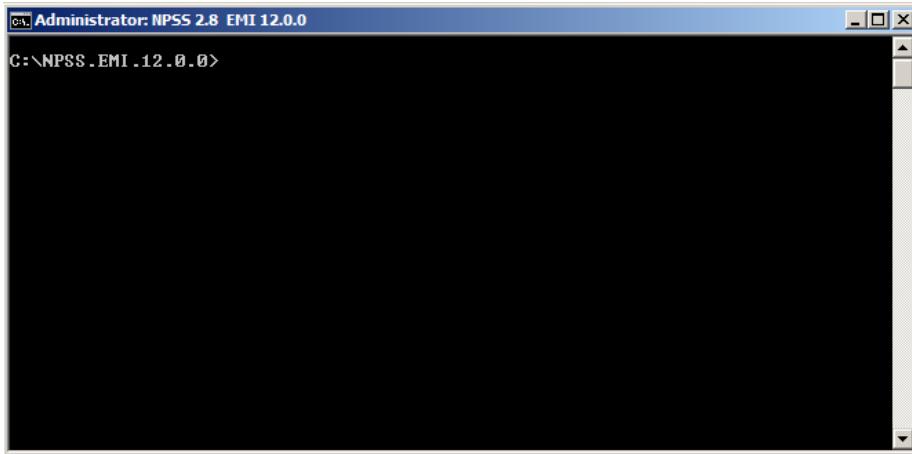


Figure 2.2 NPSS interface

Generation of a model can be completed using any programming text editor that supports coloring and auto-complete features. NPSS is a C++ based tool, which facilitates the readability of a model through most text editors.

Any engine model can be generated using a text editor and the included thermodynamic property databases and standard component libraries of the program. The user can create a gas turbine engine model using components (referred to as “elements”) and defining the technical data which describe their individual performance. Following element specification, the user must link together the elements through the linkPorts commands.

NPSS is flexible, providing access to the code of each element, allowing users to define new elements or even modify/customize the included elements.

Once the model is developed, the user can set up the problem, specify one of the included thermodynamic packages, define the solution goals and constraints and use the auto-solver setup that the code provides or even use custom solver settings.

In simple models, engineers can completely define an entire simulation in a single file. In larger models and more complicated systems, however, it is useful to organize the simulation using a variety of file types, which are then linked together to form a complete simulation [23]. A typical simulation contains the following types of files:

.run: the main file of the problem setup which point to all the other files and specifies the desired thermodynamic package to be used for fluid properties and initializes data output files.

.mdl: contains all the elements and linkages required to represent the desired fluid/thermal system.

.int or *.dll*: contains the engineering methods for each of the elements used in the model (i.e. Compressor, Turbine).

.inc: contains other items needed for organization of the overall simulation, such as an include file for the solver settings.

.fnc: contains user defined functions to perform custom operations.

.view: defines viewers used to extract data and send to formatted output files.

Finally, output data can be displayed on the screen of the command window or stored to an output file designated in the problem setup. The data can be presented in a column or a row view, using specific commands or, most commonly, a view file can be utilized to achieve the desired output data layout.

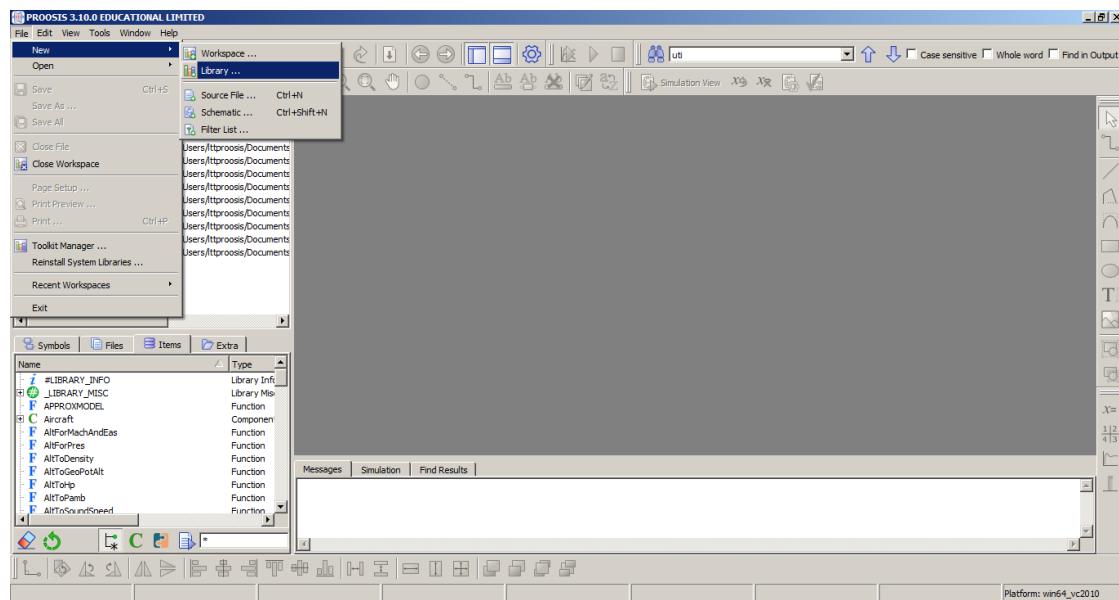
3

MODEL GENERATION

In this chapter a generic methodology of model generation will be given for each program. This will be the standard process to follow in order to create any model simulation, however it is important to note that the two programs have very different modeling structures, however the philosophy is very similar.

3.1 PROOSIS modeling

In order to create any model simulation in the PROOSIS environment, a new library and a new schematic must be at first generated as presented below:



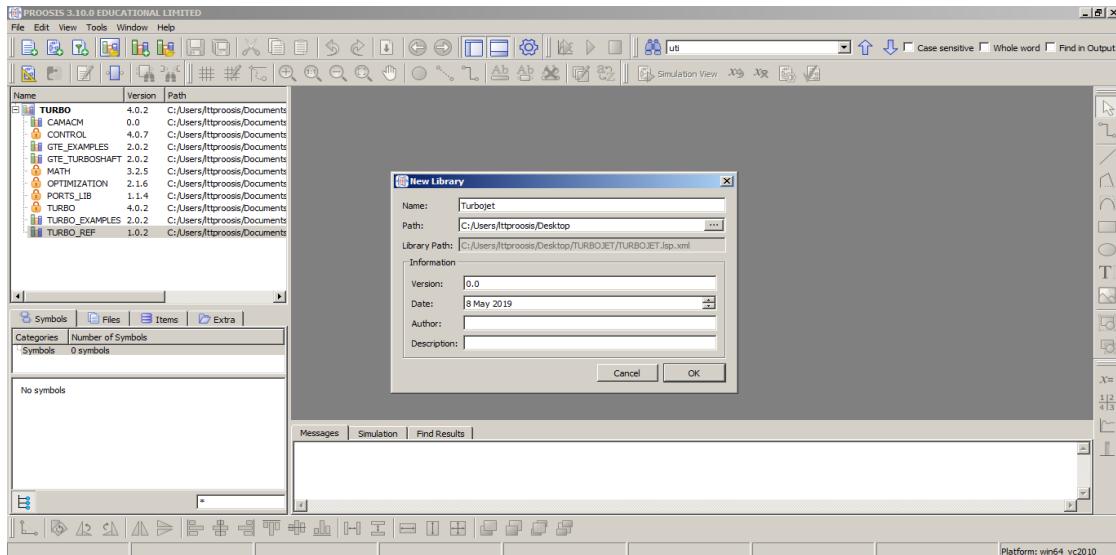


Figure 3.1 Creating a new library in PROOSIS

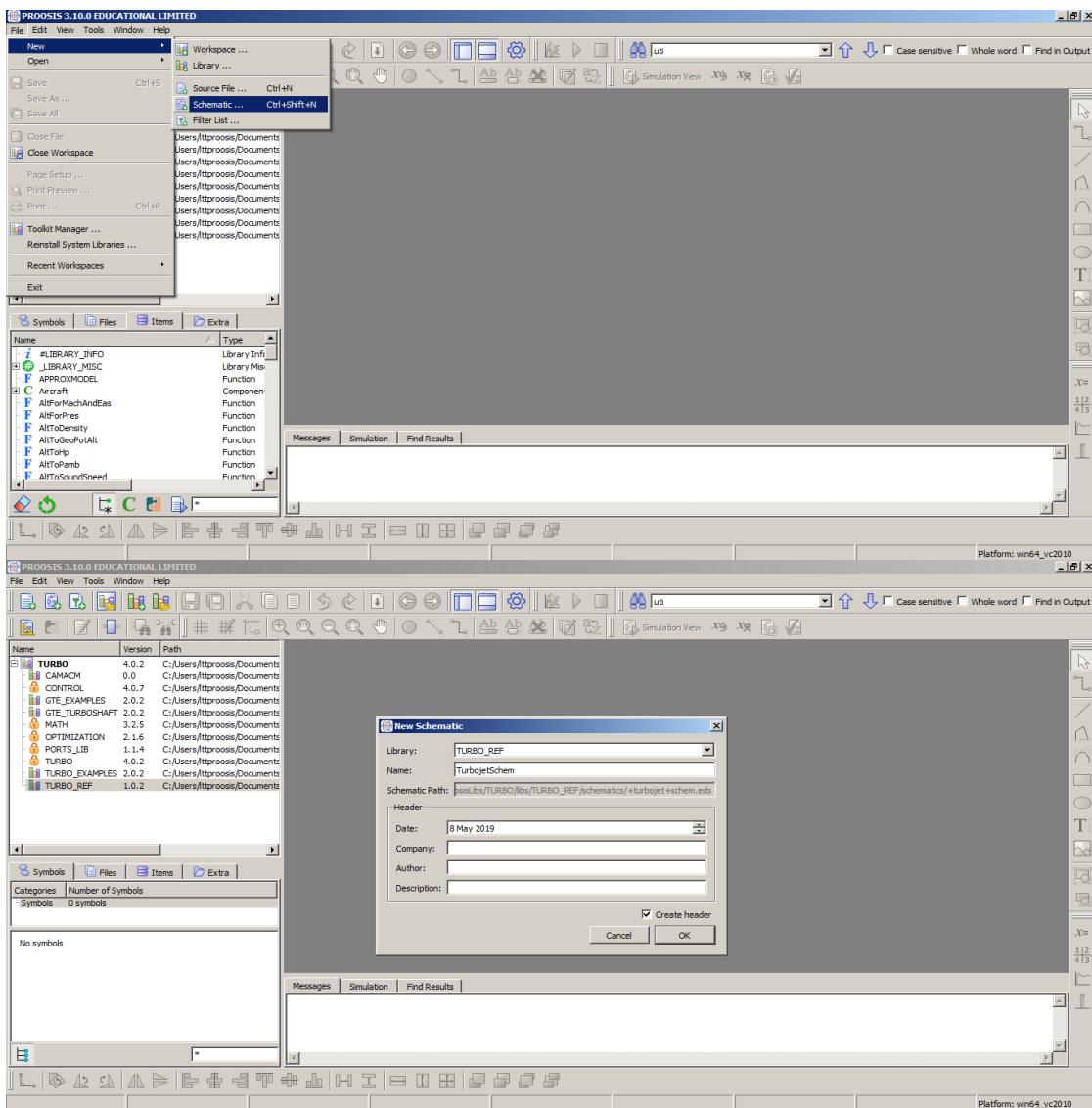


Figure 3.2 Creating a new schematic in PROOSIS

In the new schematic users can easily create a model of any gas turbine engine by dragging-and-dropping the required component symbols from the TURBO library, which simulate different parts of the engine. In addition, the *General* and *Atmosphere* symbols must be inserted as well to include the atmospheric conditions if needed.

Finally, the schematic of a simple Turbojet model is presented in the following picture:

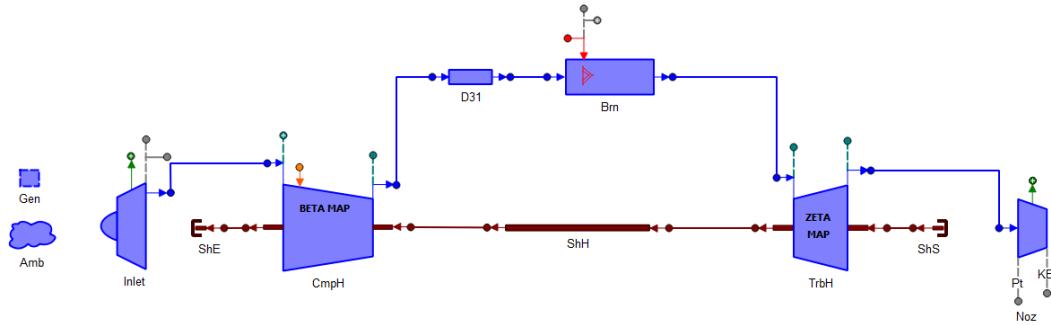


Figure 3.3 Schematic of a simple Turbojet model in PROOSIS

At this point one can edit the attributes of each component simply by right clicking on the desired component and selecting the “edit attributes” option. Next step is to create a new partition and a new experiment, where the required setup for the simulation, takes place. Finally, to run the model, simply click the *Simulate* or the *Simulate in Monitor* option to print the results of the simulation on the monitor window.

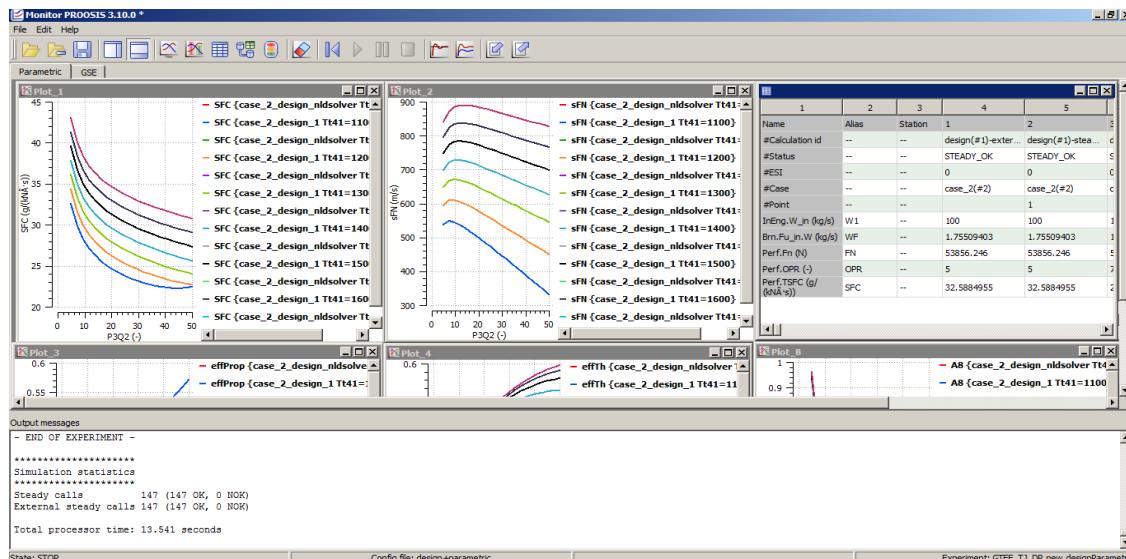


Figure 3.4 Typical layout of the monitor window in PROOSIS

The previous steps of the experiment setup and simulation will be discussed thoroughly in Chapters 7 and 8.

3.2 NPSS modeling

As mentioned above, creating a component and thus a complete engine model in NPSS differs from PROOSIS. Aside from the absence of the graphical interface in NPSS, the coding and instantiation process of a model is different between the software. In this instance the generation and instantiation of a component is accomplished through specific programming commands in a text editor. Finally, compilation and execution of the model is conducted in the command window. The process is fairly simple and is presented below.

A simple Inlet Duct component will be used as an example to describe the generation process in NPSS. Users must follow the steps presented below:

1. Open a new text file in a text editor that supports C++ language.
2. Specify the thermodynamic package desired for the fluid properties, as follows:

```
setThermoPackage("allFuel");
```

Other thermodynamic options include GasTbl, Janaf, CEA, FPT and REFPROP.

3. Create and instantiate the necessary elements:

Firstly, we will need an atmosphere element where the atmospheric conditions are defined and will be input to the Inlet element:

```
Element Ambient Amb {  
    switchMode = "ALDTMN";  
    alt_in = 0.;  
    dTs_in = 0.;  
    MN_in = 0.8;  
}
```

The code requires a very simple syntax in a block as seen above. The statements constituting a block must always be enclosed within curly braces. These blocks represent each component (i.e. Element) of the model.

The first line instantiates the *Ambient* which is of Element type included in the library, named *Amb*. The naming of an element is solely defined by the user, thus it should be unique to distinguish between multiple instances of that particular Element type in a model. For example, in a turbofan engine model the compressor element can be used to model both HPC and LPC or Fan.

The first statement appearing in the block defines which variables will be used to instantiate the element. A number of different options for the switch are available to use. In this case the option *ALDTMN* refers to altitude, dT and Mach number which will define the inlet conditions. Note that every statement must be terminated by a semi-colon.

Furthermore, we will create the *InletStart* element which defines the starting conditions for a flow into a downstream inlet. In this element we will define the name of the ambient component for obtaining reference variables in the first statement of the block and the inlet air mass flow in the second statement.

```
Element InletStart InletStart {
    AmbientName = "Amb";
    W_in = 100.;
}
```

In the next step follows the creation of the *Inlet* element, similarly to the foreseen:

```
Element Inlet InEng {
    PqP_in = 0.995;
}
```

The only statement used in this case will define the input pressure recovery.

Having finished now with the generation of all necessary elements it is important to terminate the flow stream originally started at the *InletStart* element as follows:

Element FlowEnd FeAir;

4. The following step is to establish the linkages between the elements as shown:

```
linkPorts( "InletStart.Fl_O",      "InEng.Fl_I",      "F1" );
linkPorts( "InEng.Fl_O",         "FeAir.Fl_I",      "F2" );
```

Each link statement creates a station and connects one output port to one input port. The user can assign the desired name of each station in the third argument. The naming usually follows the numbering of the engine (i.e. output from a compressor would be named F3 or F30). Link ports can be either of fluid type, fuel type or mechanical. Mechanical linkages require the instantiation of a shaft element beforehand which will be discussed in the following chapters.

5. Finally, to execute the solution the following statement must be included:

```
run();
```

6. Results can be either displayed on the screen through the *cout* command or exported into a file. To display results on the screen one has to program the following:

```
cout << "W = " << F1.W << " " << F1.W.units << endl;  

cout << "Ptin = " << F2.Pt << " " << F2.Pt.units << endl;  

cout << "Ttin = " << F2.Tt << " " << F2.Tt.units << endl;
```

Notice that the results are exported by stating the names of the ports the user defined during the element linking.

To facilitate data readability and storage users can export the results into a file using the following commands:

```
OutFileStream results {  
    filename = "res.txt";  
}
```

The block above is similar to element generation presented in previous steps and must therefore be created before the *run* command. In order for the results to be printed to the file instead of the screen one has simply to replace the *cout* command with *results*.

Otherwise, users can utilize the Data Viewers that NPSS provides to simplify formatting and organize outputs. Several types of viewers are provided by the program such as:

- VarDumpViewer (alphabetical list of specified variables)
- CaseColumnViewer (each row is a model variable, each column is a case)
- CaseRowViewer (each column is a model variable, each row is a case)
- PageViewer (single page of values formatted by the user)

In this case a *view* file should be created and instead of the *results* block, users must issue the following command:

```
#include "Results.view"
```

This command should be issued at the beginning of the code, after the declaration of the thermodynamic package.

After the *run* command issue the following

```
rowSheet.update();  
rowSheet.display();
```

Viewers are basically separate files which contain the desired formatting of the output data and are called and used by the main program with the *include* command.

7. Execute the code.

As mentioned earlier, the compilation and the execution of the model is conducted in the command window, using the command *runnpss filename.run*.



Figure 3.5 Running a model in NPSS

Note that the file to be executed must have the *.run* extension and must be located in the same folder that the NPSS program is installed. In case the file is located in a subfolder within the installation folder, one can simply change the path in the command window (i.e. *cd /path/.../*) before execution is performed.

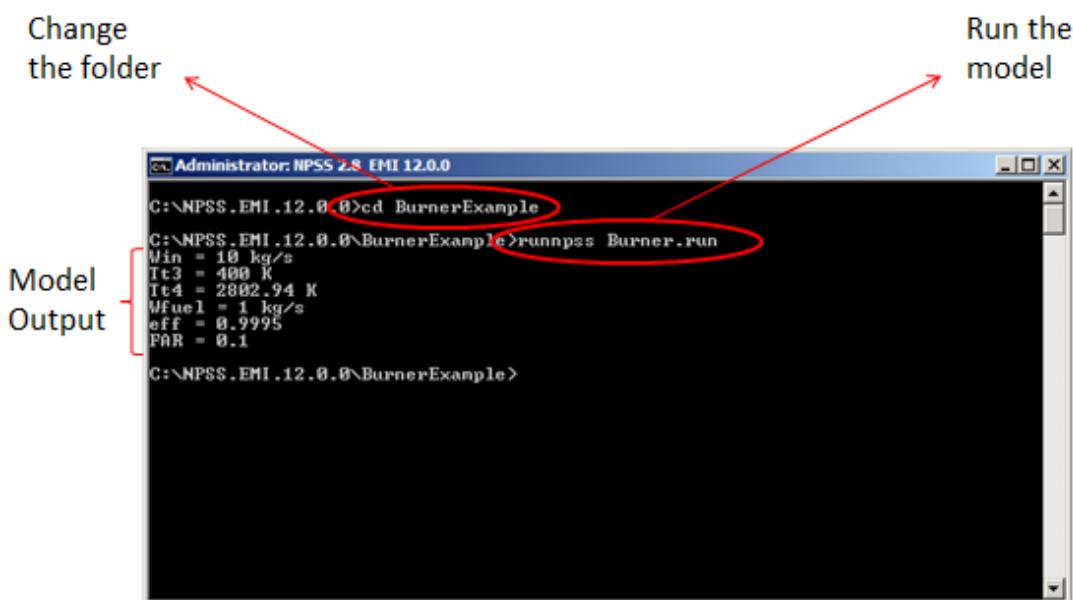


Figure 3.6 Running a model in a subfolder and printing results

In case there is a single or multiple errors in the compilation of the file, it will be displayed on the screen informing users where the error in the code is located. Usually fixing the first error statement is sufficient to compile successfully the model.

More details about the coding of different types of Elements will be discussed in the following chapters.


```

Dependent NetThrust {
    eq_lhs = "Perf.Fn"; //model value
    eq_rhs = "22500"; //target value
}

solver.addIndependent("BrnWfuel"); //add this variable to the solver
solver.addDependent("NetThrust"); //add this variable to the solver

```

Note that the solver must have an equal amount of active Independents and Dependents. These variables will be active as long as the user wishes and can be removed or set inactive with the command *removeIndependent/Dependent*.

4.1.1 Constraints

The NPSS solver has the ability to solve constrained problems using special Dependents called Constraints.

Users simply define the min and max values that some variables can have. These constraints are associated and applied to a Dependent variable. When the min/max limit is reached, Dependent left and right hand solution terms are removed from the Solver and are replaced by those defined by the Constraint. This can be seen in the excerpt of the code presented below:

```

//-----
//          Solver Constraints
//-----

real Fn_req = 10000; // lbf, requested net thrust. Used in solver dependent dep_Fn
real Nmech_max = 10000; // rpm, max allowable shaft speed. Use in off-design constraint.

// Declare a solver dependent variable that targets a specified engine net thrust, Perf.Fn
Dependent dep_Fn {
    eq_lhs = "Perf.Fn"; // model value
    eq_rhs = "Fn_req"; // target value
}

// Declare a solver dependent that limits the shaft speed
Dependent dep_Nmech_max {
    eq_lhs = "ShH.Nmech"; // variable to be constrained
    eq_rhs = "Nmech_max"; // set maximum allowable value
}

// Apply the dep_Nmech_max constraint to the dep_Fn dependent and set the constraint as an upper limit (MAX)
dep_Fn.addConstraint( "dep_Nmech_max", "MAX");

// Note: By default constraints are turned on once they are applied to a dependent.
// Use dep_Fn.useConstraints = FALSE (or TRUE) to turn constraints off/on

```

Figure 4.1 Defining constraints in NPSS

Note that it is possible to apply multiple constraints to specific Dependent (i.e. limit Tt4 and Nmech applied to dep_Fn).

4.2 Elements, Subelements and Sockets

Elements, subelements and sockets are extensively used during the programming of the NPSS models.

Elements are the building blocks of the model and provide a way to model the top level calculations of a component. Elements contain *variables*, also called *attributes*. These represent quantities such as physical characteristics, scale factors and gas properties. Some element variables may be *option variables*, which can be assigned the “DESIGN” and “OFFDESIGN” values and along with the switches are typically used to control the behavior of the element [p4 NPSS user’s guide]. As stated in the first chapter these elements were created in order to be as generic as possible. This is facilitated through the implementation of Subelements.

Subelements enhance the Elements so that they are more detailed in their performance analysis (i.e. compressor performance maps) and can be used to model different types of compressors. This is the reason why there is only one Compressor and one Turbine element to simulate all different types of compressors, fans and turbines respectively. Consequently, the program uses the same Compressor element to model the High and Low pressure compressors; however, the distinction between the two is made possible with different subelements, in this case with different performance map files. These Subelements contain calculations, tables and functions that are specific to a certain type of compressor or turbine. Finally, it is important to note that a subelement can be a map or even a file containing the results of a CFD analysis.

This architecture of a generic Element code with the Subelements is of great importance, as it gives flexibility and allows for customization on a component as well as it minimizes the code and divides it in different files. For example, every compressor’s performance is calculated using the fluid inlet conditions and a known efficiency and pressure ratio, which allows every compressor to use the same source code; however, off-design efficiency and pressure ratio result from a Subelement. In addition, in most cases where custom calculations or modifications are needed, they can be implemented through the use of a subelement. This architecture reduces the need to access the code of the element and provides a way to customize the calculations independently, instead of having everything in a single file and trying to locate the specific segment that needs modification each time.

Sockets are essentially channels that facilitate communication between Elements and Subelements.

In the following image readers can see the typical compressor element with all ports and sockets available:

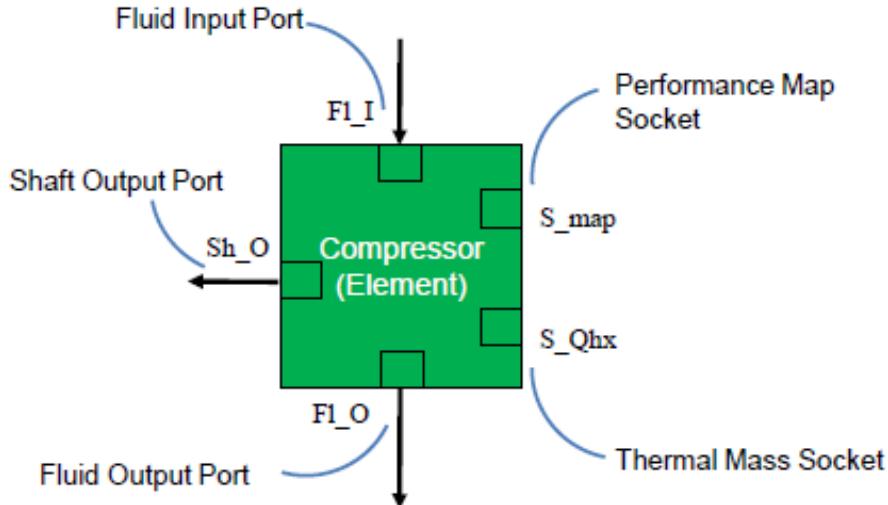


Figure 4.2 NPSS typical compressor element schematic, [25]

At this point it would be important to present the full schematic of a typical compressor element with an RlineMap subelement plugged in the proper socket. The map subelement contains all necessary tables and functions of the specific compressor.

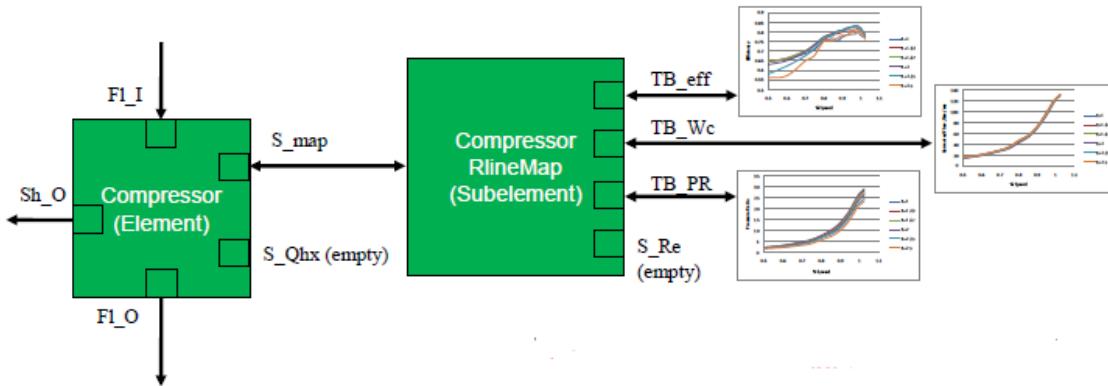


Figure 4.3 NPSS typical compressor schematic with map subelement, [25]

The RlineMap subelement plugs into the Compressor $S_{_map}$ socket and reads eff, Wc and PR maps, calculates maps scalars during design mode and finally sets effBase, PRbase, Wcbase, SMW (stall margin at constant flow) and SMN (stall margin at constant speed) in the parent compressor element.

When the map subelement is plugged into a compressor element the default solver Independent is *ind_RlineMap*, which varies the RlineMap value during off-design whereas the default Dependent is *dep_errWc*, which ensures flow continuity during the off-design mode.

4.3 Design and Off-Design Variables

In this paragraph a quick presentation of the default NPSS dependent and independent variables of the various components and engines will be presented [25].

4.3.1 InletStart

The InletStart element initiates the flow and sets the inlet pressure and temperature of the engine. As already shown in the previous chapter, this element grabs flight conditions from the Ambient element that is specified by the user.

- During Design mode: user inputs W_in (air flow rate)
- During Off-Design mode: the solver varies the air flow rate

4.3.2 Compressor

The following refer to the standard Compressor element and CompressorRlineMap subelement.

- During Design mode
 - User inputs PRdes and effDes
 - The S_map socket (Compressor map subelement) uses the provided maps, PRdes, effDes and Wc to calculate map scalars
- During Off-Design mode
 - Nc (corrected speed), Rline (map parameter) and alpha (stator vane angle) are used to read compressor performance from the map
 - The solver determines the required value for the Rline such that Wc returned from the map matches Wc entering the compressor, ensuring mass flow continuity through the compressor

4.3.3 Turbine

The following refer to the standard Turbine element and TurbinePRmap subelement.

- During Design mode
 - User inputs effDes and an initial guess for the pressure ratio in the Turbine element

- The solver determines the required turbine pressure ratio (PRbase) to balance the cycle
- The S_map socket uses the provided maps, PRdes, effDes and Wp to calculate map scalars
- During Off-Design mode
 - Np (corrected speed) and PRbase (pressure ratio) are used to read turbine performance from the map
 - The solver determines the required value for the map PR such that Wp returned from the map matches Wp entering the turbine, ensuring mass flow continuity through the turbine

4.3.4 Shaft

The shaft element provides a mechanical connection between the compressor and the turbine or any other element attached to it.

- During Design mode
 - User inputs shaft RPM (Nmech)
 - A solver dependent ensures that the torque on the shaft is zero (i.e. compressor torque equals turbine torque)
- During Off-Design mode
 - A solver independent varies shaft RPM (Nmech)
 - A solver dependent ensures that the net torque on the shaft is zero

4.3.5 Nozzle

The nozzle calculates engine exhaust conditions and gross thrust. Downstream static pressure is input, usually obtained from the Ambient element.

- During Design mode
 - If convergent nozzle, the exit area (same as throat area) is calculated such that the flow is choked (if nozzle pressure ratio is high enough) or exit static pressure equals the ambient one (if nozzle pressure ratio below that which would cause choked flow)

- If convergent-divergent nozzle, the nozzle throat area is calculated such that the flow is choked. If the nozzle pressure ratio is too low to produce choked flow, then an error is shown. The exit area is calculated such that the exit static pressure equals ambient static pressure
- During Off-Design mode
 - If convergent nozzle, the exit area is fixed (calculated from design point) and a solver dependent ensures that the exit condition is either choked flow (if nozzle pressure ratio is high enough) or exit static pressure equals ambient (if nozzle pressure ratio below that which would cause choked flow)
 - If convergent-divergent nozzle, the throat area is fixed (from design point) and the exit area is recalculated such that the exit static pressure equals ambient static pressure. A solver dependent ensures that the nozzle is choked

4.3.6 Turbojet

- During Design mode
 - User inputs the desired design point values
 - The solver varies turbine PR to drive shaft net torque to zero (automatic)
 - The solver varies fuel flow rate to achieve the desired turbine inlet temperature (user created)
 - The solver varies air flow to achieve the desired net thrust (user created)

Table 4.1 Turbojet independent variables at design point

Solver Independents		
	Solver Variable	Model Variable
<i>Automatic</i>	Trb.S_map.ind_PRbase	Trb.PRbase
<i>User created</i>	ind_Wfuel	FusEng.Wfuel
<i>User created</i>	ind_Wair	FsEng.W_in

Table 4.2 Turbojet dependent variables at Design point

Solver Dependents			
	Solver Variable	Left Hand Side	Right Hand Side
<i>Automatic</i>	Sh.integrate_Nmech	Sh.trqNet	0.0

<i>User created</i>	dep_T4	F04.Tt	T4_req
<i>User created</i>	dep_Fn	Perf.Fn	Fn_req

As mentioned before, during Design mode several values are set by the user, such as:

- Engine air flow rate (or an initial guess in case it is used as an independent variable)
- Altitude and Mach number
- Shaft RPM
- Compressor PR and eff
- Turbine eff
- Fuel flow rate (or an initial guess when added as an independent variable)
- Requested net thrust (for user created dependent)
- Requested turbine inlet temperature, Tt4 (for user created dependent)

On the contrary, during Design mode several values are calculated by the solver, such as:

- Fluid properties at each flow station (links between the components)
- Map scalars for the Compressor and Turbine elements
- Turbine PR (automatic solver variable)
- Engine air flow rate (when set as an independent variable)
- Fuel flow rate (when set as an independent variable)
- Nozzle throat/exit area (when using a convergent nozzle)
- Nozzle gross thrust
- Ram drag (inlet element)
- Engine net thrust and SFC

- During Off-Design mode
 - User inputs the desired off-design operating values
 - Define the dependent and independent variables and setup the Solver balances

Table 4.3 Turbojet independent variables at Off-Design

Solver Independents		
	Solver Variable	Model Variable
Automatic	Trb.S_map.ind_PRbase	Trb.PRbase
Automatic	Cmp.S_map.ind_RlineMap	Cmp.S_map.RlineMap
Automatic	Sh.ind_Nmech	Sh.Nmech
Automatic	FsEng.ind_W	FsEng.W
User created	ind_Wfuel	FusEng.Wfuel

Table 4.4 Turbojet dependent variables at Off-Design

Solver Dependents			
	Solver Variable	Left Hand Side	Right Hand Side
Automatic	Trb.S_map.dep_errWp	Trb.Wp	Trb.WpCalc
Automatic	Cmp.S_map.dep_errWc	Cmp.Wc	Cmp.WcCalc
Automatic	Sh.integrate_Nmech	Sh.trqNet	0.0
Automatic	NozPri.dep_Area	NozPri.WqAe	NozPri.WqAEdem
User created	dep_Fn	Perf.Fn	Fn_req

As mentioned above, during Off-Design mode, only a few values are set by the user, such as:

- Altitude and Mach number
- Requested net thrust (when set by the user)

On the other hand, during Off-Design mode, several values are calculated by the user, such as:

- Fluid properties at each flow station (links between the components)
- Engine air flow rate (automatic Solver variable)
- Shaft RPM (automatic solver variable)
- Fuel flow rate (automatic solver variable)
- Compressor and Turbine performance (PR and eff obtained from the scaled maps)
- Nozzle thrust
- Ram drag
- Engine net thrust and SFC

4.3.7 *Turbofan*

The following refer to a simple twin spool turbofan engine model [26]. Here, an additional splitter element is used which will be thoroughly discussed in the following chapters.

- During Design mode
 - User inputs the desired design point values
 - The solver varies the high and low pressure turbine PR to drive the high and low pressure shaft net torque to zero respectively (automatic)
 - The solver varies fuel flow rate to achieve the desired turbine inlet temperature (user created)
 - The solver varies air flow to achieve the desired net thrust (user created)

Table 4.5 Turbofan independent variables at Design point

Solver Independents		
	Solver Variable	Model Variable
Automatic	TrbL.S_map.ind_PRbase	TrbL.PRbase
Automatic	TrbH.S_map.ind_PRbase	TrbH.PRbase
User created	ind_Wfuel	FusEng.Wfuel
User created	ind_Wair	FsEng.W_in

Table 4.6 Turbofan dependent variables at Design point

Solver Dependents			
	Solver Variable	Left Hand Side	Right Hand Side
Automatic	ShL.integrate_Nmech	ShL.trqNet	0.0
Automatic	ShH.integrate_Nmech	ShH.trqNet	0.0
User created	dep_T4	F04.Tt	T4_req
User created	dep_Fn	Perf.Fn	Fn_req

- During Off-Design mode
 - User inputs the desired off-design operating values
 - Define the dependent and independent variables and setup the Solver balances

Table 4.7 Turbofan independent variables at Off-Design

Solver Independents		
	Solver Variable	Model Variable
Automatic	TrbL.S_map.ind_PRbase	TrbL.PRbase
Automatic	TrbH.S_map.ind_PRbase	TrbH.PRbase
Automatic	CmpL.S_map.ind_RlineMap	CmpL.S_map.RlineMap
Automatic	CmpH.S_map.ind_RlineMap	CmpH.S_map.RlineMap
Automatic	CmpFan.S_map.ind_RlineMap	CmpFan.S_map.RlineMap
Automatic	ShL.ind_Nmech	ShL.Nmech
Automatic	ShH.ind_Nmech	ShH.Nmech
Automatic	Slpit.ind_BPR	Split.BPR
Automatic	FsEng.ind_W	FsEng.W
User created	ind_Wfuel	FusEng.Wfuel

Table 4.8 Turbofan dependent variables at Off-Design

Solver Dependents			
	Solver Variable	Left Hand Side	Right Hand Side
Automatic	TrbL.S_map.dep_errWp	TrbL.Wp	TrbL.WpCalc
Automatic	TrbH.S_map.dep_errWp	TrbH.Wp	TrbH.WpCalc
Automatic	CmpL.S_map.dep_errWc	CmpL.Wc	CmpL.WcCalc
Automatic	CmpH.S_map.dep_errWc	CmpH.Wc	CmpH.WcCalc
Automatic	CmpFan.S_map.dep_errWc	CmpFan.Wc	CmpFan.WcCalc
Automatic	ShL.integrate_Nmech	ShL.trqNet	0.0
Automatic	ShH.integrate_Nmech	ShH.trqNet	0.0
Automatic	NozPri.dep_Area	NozPri.WqAe	NozPri.WqAEdem
Automatic	NozSec.dep_Area	NozSec.WqAe	NozSec.WqAEdem
User created	dep_Fn	Perf.Fn	Fn_req

A turbofan engine is essentially an evolution of a turbojet engine. Any new automatic dependent and independent variable depends on the additional components that the engine model consists of.

4.3.8 Turboshaft

As mentioned above, the additional dependent and independent variables depend on the various components of the engine and the user's will. Similarly, for the Turboshaft,

an evolution of the Turbojet engine as well, the dependent and independent variables are presented in the following tables:

Table 4.9 Turboshaft independent variables at Design point

Solver Independents		
	Solver Variable	Model Variable
Automatic	TrbL.S_map.ind_PRbase	TrbL.PRbase
Automatic	TrbH.S_map.ind_PRbase	TrbH.PRbase

Table 4.10 Turboshaft dependent variables at Design point

Solver Dependents			
	Solver Variable	Left Hand Side	Right Hand Side
Automatic	ShL.integrate_Nmech	ShL.trqNet	0.0
Automatic	ShH.integrate_Nmech	ShH.trqNet	0.0

Table 4.11 Turboshaft independent variables at Off-Design

Solver Independents		
	Solver Variable	Model Variable
Automatic	TrbFP.S_map.ind_PRbase	TrbFP.PRbase
Automatic	TrbH.S_map.ind_PRbase	TrbH.PRbase
Automatic	CmpH.S_map.ind_RlineMap	CmpH.S_map.RlineMap
Automatic	ShFP.ind_Nmech	ShFP.Nmech
Automatic	ShH.ind_Nmech	ShH.Nmech
User created	Brn.ind_FAR	Brn.FAR
User created	ShFP.ind_SHP	ShFP.HPX

Table 4.12 Turboshaft dependent variables at Off-Design

Solver Dependents			
	Solver Variable	Left Hand Side	Right Hand Side
Automatic	TrbFP.S_map.dep_errWp	TrbFP.Wp	TrbFP.WpCalc
Automatic	TrbH.S_map.dep_errWp	TrbH.Wp	TrbH.WpCalc
Automatic	CmpH.S_map.dep_errWc	CmpH.Wc	CmpH.WcCalc
Automatic	ShFP.integrate_Nmech	ShFP.trqNet	0.0
Automatic	ShH.integrate_Nmech	ShH.trqNet	0.0
Automatic	NozPri.dep_Area	NozPri.WqAe	NozPri.WqAEdem

<i>User created</i>	dep_Np	ShFP.Nmech	NP_dmd
<i>User created</i>	dep_SHP	Perf.SHP	SHP_dmd

4.4 Ports

Keeping in mind that the ultimate purpose of this report is to form the first step of a possible interface between PROOSIS and NPSS, we will present the differences between the two as far as the ports are concerned. Knowing the kind of information transferred through the ports is very important in order to interface the programs. Each program has different philosophy and demands different values to be transferred through the ports [32]. In the following tables we will present the variables that are transferred through each type of port in each program, discussing the differences between the programs as well.

Table 4.13 Fluid port variables comparison

FLUID PORT		
NPSS		PROOSIS
Tt		W
Pt		Tt
ht	Enthalpy based on total conditions	Pt
ut	Internal energy based on total conditions	FARB
Cpt	Specific heat (constant pressure)	FARU
Cvt	Specific heat (constant volume)	WAR
gamt	Ratio of specific heats	Ang
s	Entropy	
rhot	Density	
mut	Viscosity	
kt	Conductivity	
Rt	Gas constant	
W		
FAR		
WAR		
MN		
V		
A		
Ts		
Ps		

hs	
us	
Cps	
Cvs	
gams	
rhos	
mus	
ks	
Rs	
Cd	

Table 4.14 Fuel port variables comparison

FUEL PORT	
NPSS	PROOSIS
Wfuel	W
Tfuel	T
Pfuel	P
fuelType	
LHV	
hFuel	
Tref	
hRef	
Carbon	
Hydrogen	
Nitrogen2	
Oxygen	

Table 4.15 Mechanical port variables comparison

MECHANICAL PORT	
NPSS	PROOSIS
Nmech	Nmech
pwr	trq
trq	inertia
inertia	inertia_tot

As seen in the tables above, PROOSIS and NPSS transfer different information through their ports. This is of high importance, especially for the interfacing process, as special attention should be given in order to ensure that the same type of ports convey the same variables. Especially in the case of the fluid port, one can observe that NPSS transfers a large amount of information compared to PROOSIS.

An important difference between the programs, is that PROOSIS transfers only the main parameters and calculates anything else needed (i.e. enthalpy, entropy) inside of each component. On the contrary, NPSS passes all the information seen above through every element without the need to perform additional calculations in each element. This is a fundamental modeling difference between the programs, that someone must take into consideration when choosing to interface the software, either by creating the corresponding NPSS elements and integrating them into the PROOSIS library, or by establishing a direct communication channel between the two.

Moreover, NPSS supports some functions to be executed in the ports, such as the *setTotal* or *setStatic* which set the properties of a flow as a function of temperature and pressure or any other supported combination (i.e. h and P, S and P, h and S), *copyFlow* which copies the information of a flow and *burn* function which simulates a burner component. This is something that must also be taken into consideration when interfacing the programs.

5

NPSS MAP INTEGRATION IN PROOSIS

Prior to simulating and exporting results from the two programs it is important to ensure that the exact same parameters are set in both cases. Besides the input data, this includes the maps used from the compressor and turbine components. Maps define the performance of each element, thus it is vital to use the same map files in both simulations. Subsequently, we opted to integrate the NPSS compressor and turbine maps into PROOSIS in order to utilize the graphical environment.

Map files used in NPSS simulations are named `**RlineMap_EEE_*.map` (i.e. `CompressorRlineMap_EEE_fan.map` for the fan map). EEE refers to the Energy Efficient Engine program funded by NASA at 1978 to develop technologies suitable for more efficient high by-pass Turbofan engines. The program's goal was to improve thrust-specific fuel consumption by 12% compared to a GE CF6-50C engine, reduce operating costs by 5% of the same engine on an equivalent aircraft, reduce noise levels, comply with the EPA 1981 emission standards for new engines and finally achieve a minimum of 50% reduction in the rate of performance deterioration in service compared to the engine named above. Large aerospace manufacturers, such as Boeing, General Electric and Pratt & Whitney, contributed in the program and provided aircrafts and Turbofans required for testing respectively. P&W built a 9-stage high pressure compressor, whereas GE developed a 23:1 ratio, 10-stage high pressure compressor. All companies benefited from this project, especially GE who used their EEE HPC in the GE90 and GEnx engines later produced for the Boeing 777 and 777X aircrafts respectively [24].

5.1 Compressor R-line map

In the beginning of each compressor R-line map file the following variables, that are specific to the map, are defined: the stator vane angle (`alphaMapDes = 0.0`), the corrected speed (`NcMapDes = 1.0`), the R-line parameter (`RlineMapDes = 2.0`) at the

unscaled map Design Point, as well as the R-line parameter ($R_{lineStall} = 1.0$) at the stall line. Each map line is defined by the stator vane angle, the corrected speed, the R-line value and the corrected flow. For each stator vane angle (ALPHA) value (0, 1) the following are defined:

- **Fan:** for a number of corrected speed ($SPED$) values, several corrected flow ($FLOW$) values are given, each one corresponding to a specific R-line (R) value. R-line parameter varies from 1.0 to 2.5. The exact same philosophy is followed for the efficiency (EFF) and pressure ratio (PR) lines. However, in the case of the Fan map, the R-line parameter is given fewer values for the first three speed points (corresponding to the lower part of the map), meaning it varies from 1.0 to 2.0 for these three and from 1.0 to 2.5 for the rest speed values as it can be seen in the following picture. On the contrary, the HPC and LPC R-line parameters vary from 1.0 to 2.5 throughout every speed value.

```
// Fan OD corrected flow VS. stator angle, fraction of design speed, and R-line
// Note: this map has ALPHA = 0 and ALPHA = 1, but both submaps are identical
Table TB_Wc(real ALPHA, real SPED, real R) {
    ALPHA = 0.0 {
        SPED = 0.4 {
            R = { 1.0, 1.25, 1.5, 1.75, 2.0 }
            FLOW = { 386.838, 552.308, 670.769, 762.906, 830.598 }
        }
        SPED = 0.5 {
            R = { 1.0, 1.25, 1.5, 1.75, 2.0 }
            FLOW = { 544.786, 659.487, 766.667, 855.043, 918.974 }
        }
        SPED = 0.6 {
            R = { 1.0, 1.25, 1.5, 1.75, 2.0 }
            FLOW = { 693.333, 781.709, 881.368, 956.581, 1018.632 }
        }
        SPED = 0.7 {
            R = { 1.0, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5 }
            FLOW = { 830.598, 911.453, 996.068, 1060.0, 1114.53, 1152.137, 1182.222 }
        }
        SPED = 0.8 {
            R = { 1.0, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5 }
            FLOW = { 965.983, 1037.436, 1116.41, 1165.299, 1221.709, 1249.915, 1263.077 }
        }
        SPED = 0.85 {
            R = { 1.0, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5 }
            FLOW = { 1037.436, 1108.889, 1184.103, 1227.35, 1280.0, 1304.444, 1313.846 }
        }
        SPED = 0.9 {
            R = { 1.0, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5 }
            FLOW = { 1120.171, 1191.624, 1259.316, 1300.684, 1343.932, 1364.615, 1370.256 }
        }
    }
}
```

Figure 5.1 Excerpt of the Fan R-line map file

- **LPC and HPC:** as mentioned above the same apply for the Compressor maps, however the R-line parameter varies from 1.0 to 2.5 for all points and different speed variation is noticed as well.

For intermediate points of the map, interpolation and extrapolation is used. More specifically, for the corrected speed and R-line values a 3rd degree Lagrange polynomial

interpolation and a linear extrapolation are used, whereas for the stator vane angle a linear interpolation and extrapolation is used by default. As for the intermediate PR points no specific method is used.

5.2 Turbine PR map

Turbine PR map files are very similar to the ones used for the Compressor components. In the beginning, the map specific variables *PRmapDes* and *NpMapDes* are assigned the values 4.975 and 100.0 respectively. Then, each map line is defined by the corrected speed (*SPED*), pressure ratio (*PR*) and corrected flow (*FLOW*), as in the Compressor map file:

- **LPT and HPT:** in both map files for each corrected speed value a set of the pressure ratio and corrected flow values are defined. However, in Turbine maps the PR parameter is handled similarly to the R-line parameter found in Compressor map files, PR = 6.0 corresponds to surge, whereas PR = 2.0 corresponds to the lowest flow rate.

For the intermediate points of the map interpolation and extrapolation is used as in the case of Compressor maps. In this instance however, a 2nd degree Lagrange polynomial interpolation is used for the speed and a 3rd degree for the pressure ratio, whereas extrapolation is not set to be used in either of the two.

5.3 NPSS map integration process

The main difference between NPSS and PROOSIS map files is that the later uses BETA and ZETA maps, whereas the former uses R-line and PR maps for the Compressor and the Turbine respectively.

BETA and ZETA maps in PROOSIS are consisted of BETA and ZETA lines. These lines are defined by the BETA parameter, corrected speed, pressure ratio, corrected flow and isentropic efficiency in the Compressor and the ZETA parameter, corrected speed, corrected flow and efficiency in the Turbine.

In the beginning of each BETA map, values of the corrected flow and pressure ratio that define the surge line are given. The surge line is defined by the points that correspond to the lowest corrected flow for every corrected speed value. Then a table of values for each variable is defined, similarly to the NPSS maps. It is notable that the number of lines of the corrected flow, efficiency and pressure ratio is equal to the number of the corrected speed values in each case, whereas the number of columns of the table is equal to the number of BETA parameter values. Turbine ZETA maps are modeled similarly to the BETA maps.

A notable difference between the R-line and BETA parameter is not only the different range but the state of the engine at the parameter limits as well. The BETA parameter ranges from 0 (choke) to 1.0 (surge), whereas the R-line parameter ranges from 2.5 (choke) to 1.0 (surge).

In order to integrate NPSS maps in PROOSIS format we modified the BETA and ZETA maps using the corresponding tables found in the NPSS maps, following the process that will be described below.

Since each program uses different lines to define its maps, it was decided to handle the R-lines similarly to the BETA lines. This means we replaced the BETA values with the R-line values, defining a correlation between the two and adjusting the range limits accordingly. However, it is important to mention again that in the NPSS fan map file there are fewer points for the first three corrected speed values. In order to diminish the need to increase the number of points through interpolation, as the PROOSIS maps require the same point number allocation throughout each the map file, we integrated the PROOSIS map file in NPSS instead. The ultimate goal is to ensure that both software use the same maps in order to evaluate any differences in the simulations.

Finally, in both programs linear interpolations were set. Moreover, in order for the units to be compatible between the programs a modification to the units of corrected flow was made to convert lbm/sec to kg/s and the points were inserted reversely, as the R-line value that refers to choked flow is 2.5 (range 1÷2.5) while the BETA value is 0 (range 0÷1). The exact same process is followed for each Compressor map (i.e. LPC, HPC) and each Turbine map as well (i.e. LPT and HPT), whereas the reverse process is followed for the fan map file as mentioned above.

It is important at this point, however, to mention that differences in the range of the map parameters can cause differences as far as the positioning of design and off-design points on the maps is concerned. Small differences between the corresponding maps parameters can cause small deviation in the position of operating points on the maps. Consequently, this creates small differences in the performance of the related components, especially in the cases of the engine models, as it will be presented in the following chapters.

6

NPSS FLUID MODEL INTEGRATION IN PROOSIS

Having ensured that both programs will use the same performance map files, it is important to mention that another source of differences is the fluid model. Consequently, the next step is to make sure that in both cases the same fluid model is used. Since there is no immediate correlation between the fluid model options between the programs, we decided to insert the NPSS fluid model in PROOSIS. As mentioned before, the allFuel thermodynamic package was used in NPSS, which provides users with a variety of different fuel options. As mentioned in NPSS thermo guide [27], the different fuel types are “JP”, “OLJP”, “H2”, “GAS”, “CH4”, “WBH4” and “UNIV”, however only the JP and CH4 options are recommended. “OLJP” stands for OLD JP and has an H-C ratio of 2:1 but with Keenan and Keyes air composition, with no CO₂. “H2” stands for pure hydrogen gas with the 1962 atmosphere air composition. “GAS” is a methane model with the 1945 Keenan and Keyes air composition. “CH4” is the current methane model with the 1962 atmosphere air composition and is a GE standard. “WBH2” is a pure hydrogen fuel with pure oxygen rather than air as the working fluid. “UNIV” is a more complex option to address arbitrary fuel composition of H-C-O-N-A atoms with the 1962 atmosphere air composition and it can not be used without additional information and guidance. Finally, “JP” is designed to produce properties for the combustion products of fuel, water and air, where the fuel H-C ratio is 2:1. This model applies a 1962 atmosphere (with 0.03% CO₂) and is a GE current standard. It is also important to mention that allFuel is a simple (fast) “burned properties only” thermo package. It does not keep track of the Cp, SG, rho, etc. properties of unburned fuels. Therefore, it does not need to distinguish between JP4, JP7, JP8, AvGas, Kerosene, etc.. The 2:1 H-C ratio value is close for all of these JP fuels. An important note is that the zero enthalpy datum is at zero degrees Rankine/Kelvin.

It would be beneficial at this point to present the fluid models differences. Therefore, in order to do so, the JP option was selected from the NPSS allFuel thermo package, whereas the closest fuel option that PROOSIS has to offer is the JP4_noDiss.

Obtaining the fluid properties from PROOSIS was easy, since we had access to the corresponding fluid model file with containing the fluid properties tables. As far as NPSS is concerned, a simple program was created in order to obtain the fluid properties of the specified fuel. For a number of different temperature and FAR values we extracted the enthalpy, entropy and the isentropic coefficient values. The NPSS program is presented below:

```
setThermoPackage("allFuel");
FlowStation fs;
fs.switchFuelType = "JP";
#include "ThermoView.view"
```

Select the desired thermo package, specify the fuel and a viewer file to print the results in a customized format. Then, create new variables for pressure, temperature and temperature step.

```
real Pt; //pressure, psia
real Tt; // temperature, Rankine
real dT; // temperature, Rankine

fs.FAR = 0.000;
fs.WAR = 0.000;

do {
Pt = 101325*getUnitsFactor("Pa", "psia"); //psia
Tt = 200*getUnitsFactor("dK", "dR"); // Rankine
dT = 50*getUnitsFactor("dK", "dR");
cout << "FAR = " << fs.FAR << " " << endl;

    do {
        fs.setTotalTP(Tt, Pt);
        run();
        CASE++;
        rowSheet.update();
        Tt = Tt + dT;
    } while (Tt<=5400);

    fs.FAR = fs.FAR + 0.010;
    rowSheet.display();
} while (fs.FAR<=0.040);
```

Note that since NPSS uses the imperial unit system, pressure is given in psia and temperature in Rankine. The *getUnitsFactor* command returns a factor and serves in unit conversion. The simulations were executed for a temperature range of 200 to 3000 K by 50 K increments and for fuel-to-air ratio of 0.0 to 0.04 by 0.01 increments. The comparison between the fluid models of each program is presented in the following figures. Note that only data of FAR = 0.0 are plotted in the following figures, as similar curves are observed for the rest FAR values. Finally, it is also important to mention that a reference temperature of 200 K was assumed for the delta increase in both cases.

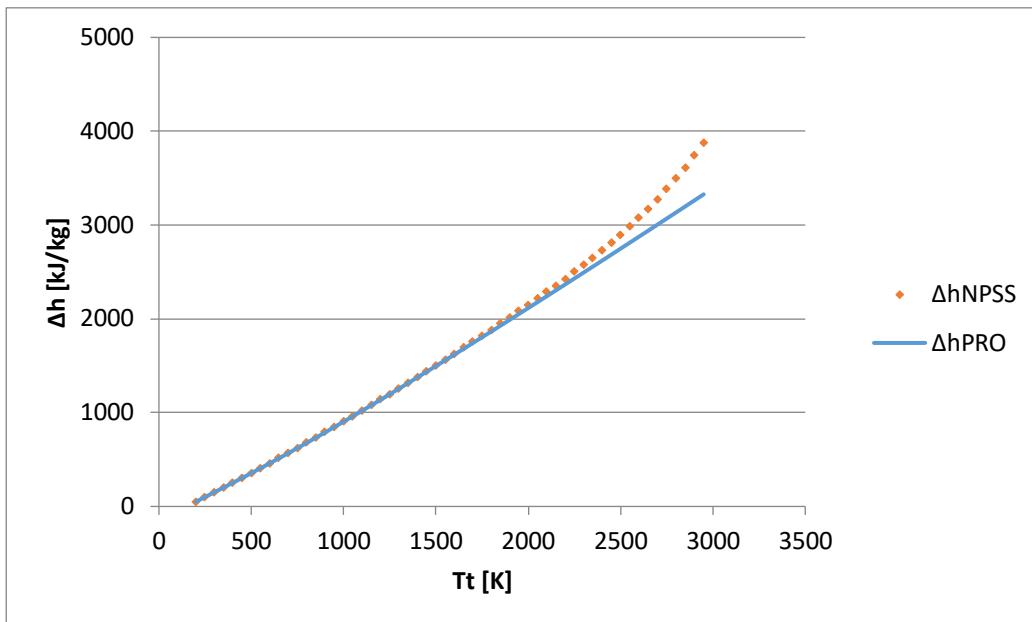


Figure 6.1 Comparison of NPSS and PROOSIS fluid model enthalpy

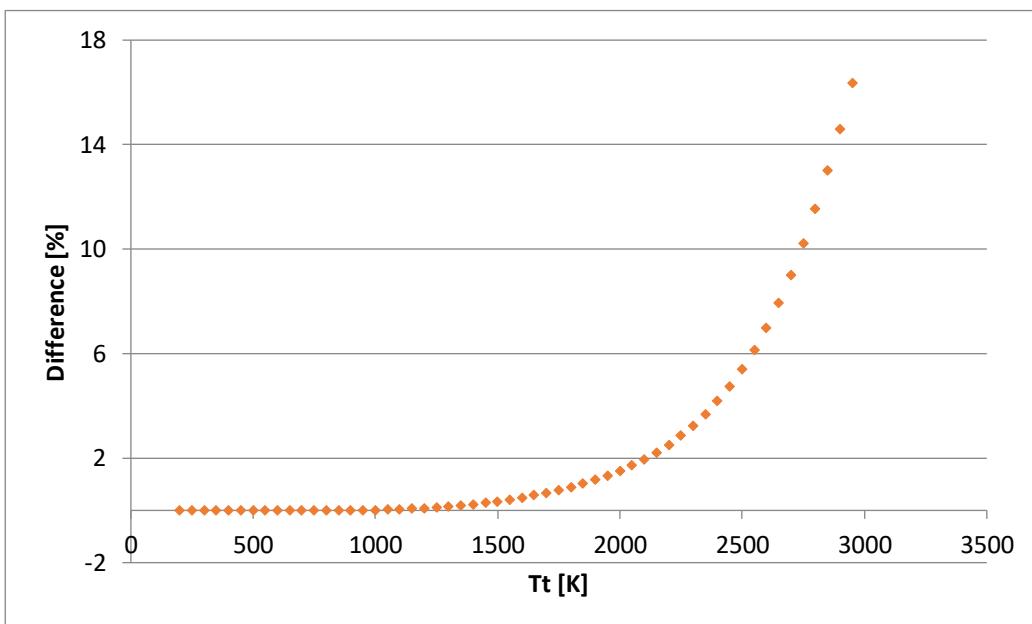


Figure 6.2 NPSS and PROOSIS fluid model enthalpy percentage differences

We can clearly see that as the temperature increases so do the enthalpy differences between the fluid models. For temperatures up to 1250 K differences are below 0.1%. For 1300-1900 K differences are lower than 1% and the maximum difference is at 3000 K equal to 16.3%. However, for typical jet engines temperatures of 1200-1600 K an average difference of 0.2% is observed. It is also important to mention that as the FAR increases, similarly differences increase. More specifically, the maximum difference for FAR = 0.01 is 20%, for FAR = 0.02 is 23.5% and for FAR = 0.04 is 28.4%. Similarly, the entropy function comparison is presented below:

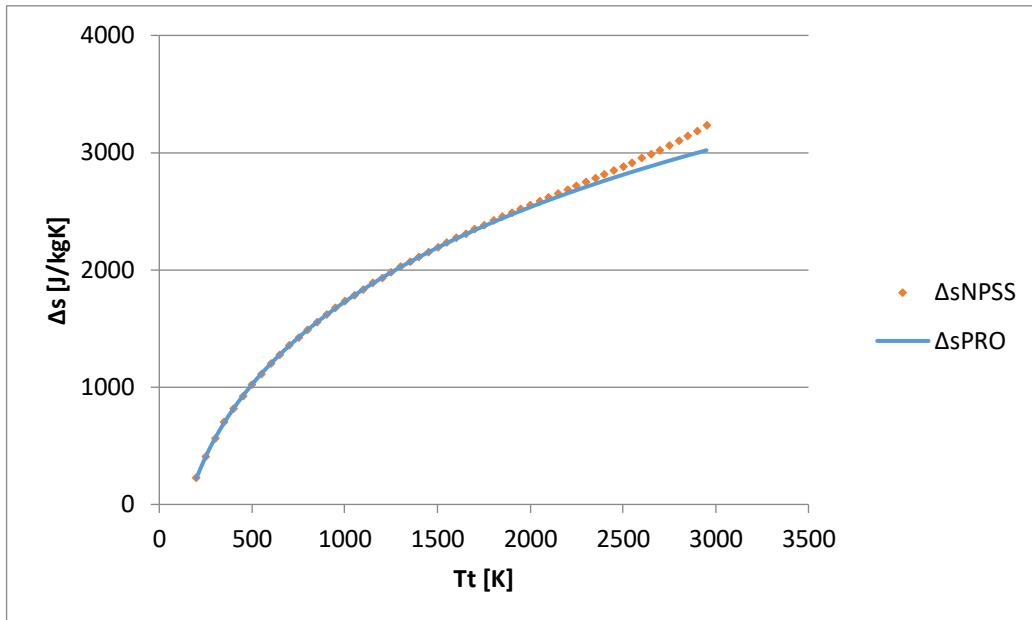


Figure 6.3 Comparison of NPSS and PROOSIS fluid model entropy

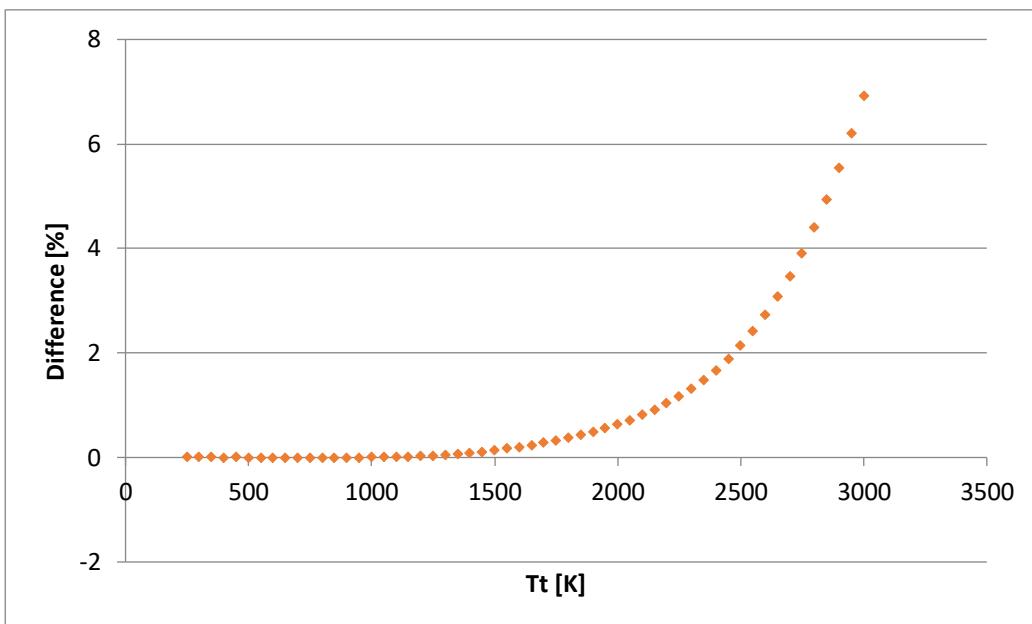


Figure 6.4 NPSS and PROOSIS fluid model entropy percentage differences

For temperatures up to 2150 K differences in entropy are below 0.1%, whereas the maximum difference is 6.9% at 3000 K. For FAR = 0.01 the maximum difference is 8%, for FAR = 0.02 is 8.8% and for FAR 0.04 is 10%. Finally, differences of the isentropic coefficient (γ) between the programs is presented in the following figures:

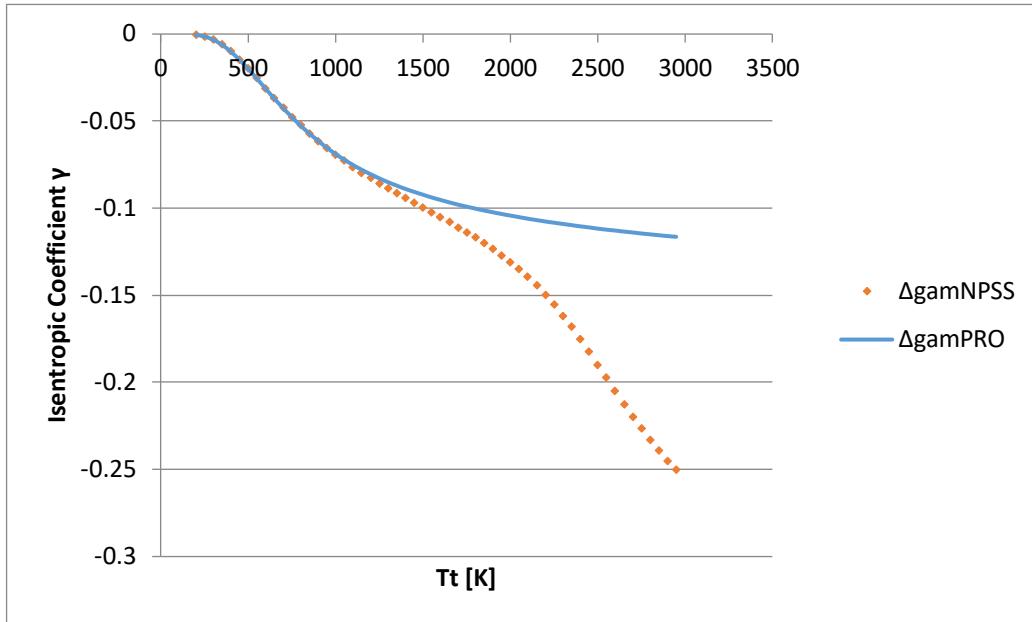


Figure 6.5 Comparison of NPSS and PROOSIS fluid model isentropic coefficient

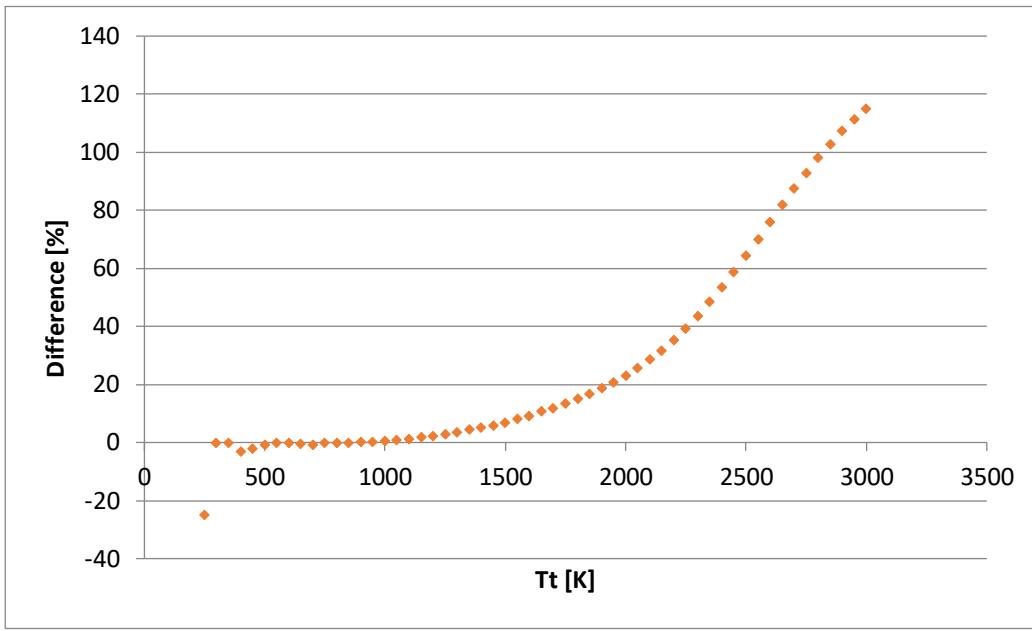


Figure 6.6 NPSS and PROOSIS fluid model isentropic coefficient percentage differences

The maximum difference observed for the isentropic coefficient across the considered temperature range is 115% for FAR = 0.0. For FAR = 0.01 the maximum

difference is 108.4%, for FAR = 0.02 it is 102% and finally for FAR = 0.04 it is 97.8%. The reference temperature, as in previous cases, is considered at 200 K.

Having seen the differences in the fluid models between the programs, the next step is to insert the NPSS allFuel JP fluid model in PROOSIS. Doing so will diminish any differences due to the fluid model differences and will reveal any possible differences in the thermodynamic calculations. In order to integrate the NPSS fluid model in PROOSIS, the default JP4_noDiss fuel file was modified and the tables of enthalpy, entropy and gas constant were inserted.

7

COMPONENT LEVEL COMPARISON

As mentioned in the introduction the goal of this project is to compare results from PROOSIS and NPSS. Consequently, component level as well as engine level comparison are necessary. In this chapter we will discuss the process of creating individual component models, in each of the two programs, and finally evaluate the results. The results will be presented in the form of tables and graphs. Additionally, it is important to mention that for every calculation in this chapter standard day assumptions are made. Finally, it is important to mention at this point that only minor differences are expected since the same fluid model and maps are used.

7.1 Inlet

7.1.1 PROOSIS

As stated in the previous chapter, creating a new library and schematic is a standard procedure. In this case we drag-and-drop the *InletAtm* component symbol from the TURBO library to the schematic window, which simulates the Inlet duct of an engine and receives the atmospheric conditions as an input. In addition, we insert the *General* and *Atmosphere* symbols from the TURBO library, required for the instantiation of the inlet duct component.

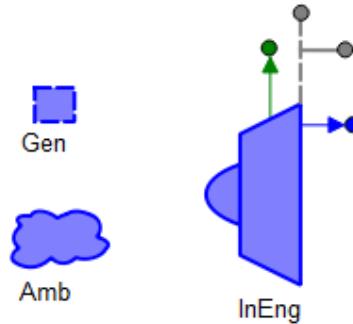


Figure 7.1 Inlet schematic diagram in PROOSIS environment

For the instantiation of the component only the pressure loss of the duct is required to be specified. This can be done by right clicking on the *Inlet* symbol on the schematic window and selecting edit attributes.

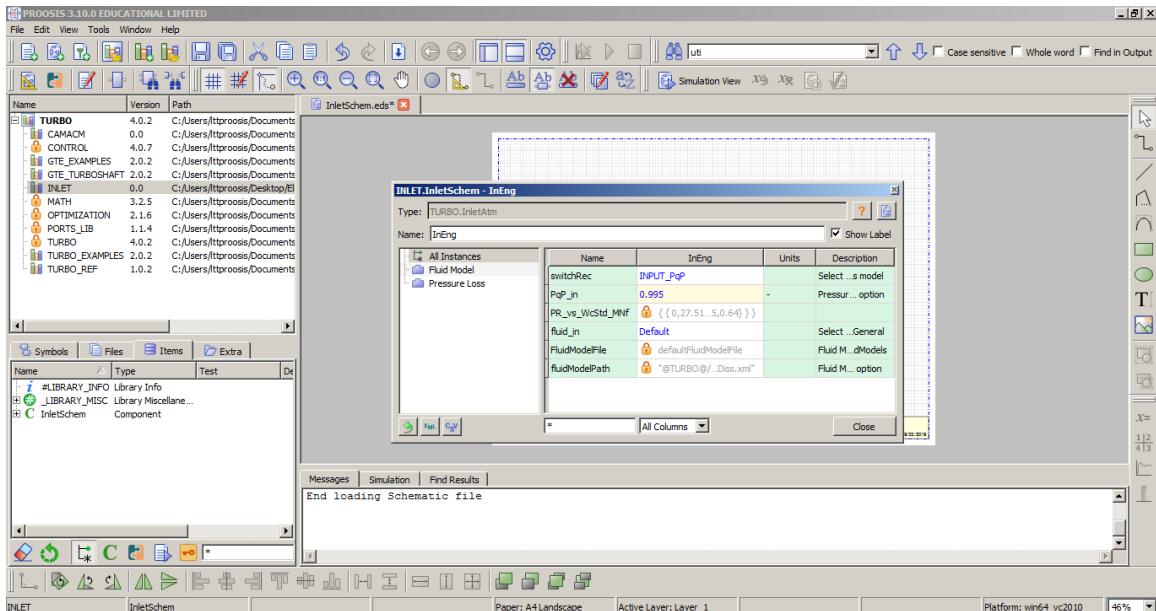


Figure 7.2 Editing attributes of the Inlet component in PROOSIS

Once the setup is complete we can compile the model by pressing F7 to ensure the schematic is correct and then generate a new *default partition* followed by a new *wizard experiment* on the left side of the workspace. Under the experiment we can add a *Standard Case* where we subsequently add a *parametric calculation* as pictured below:

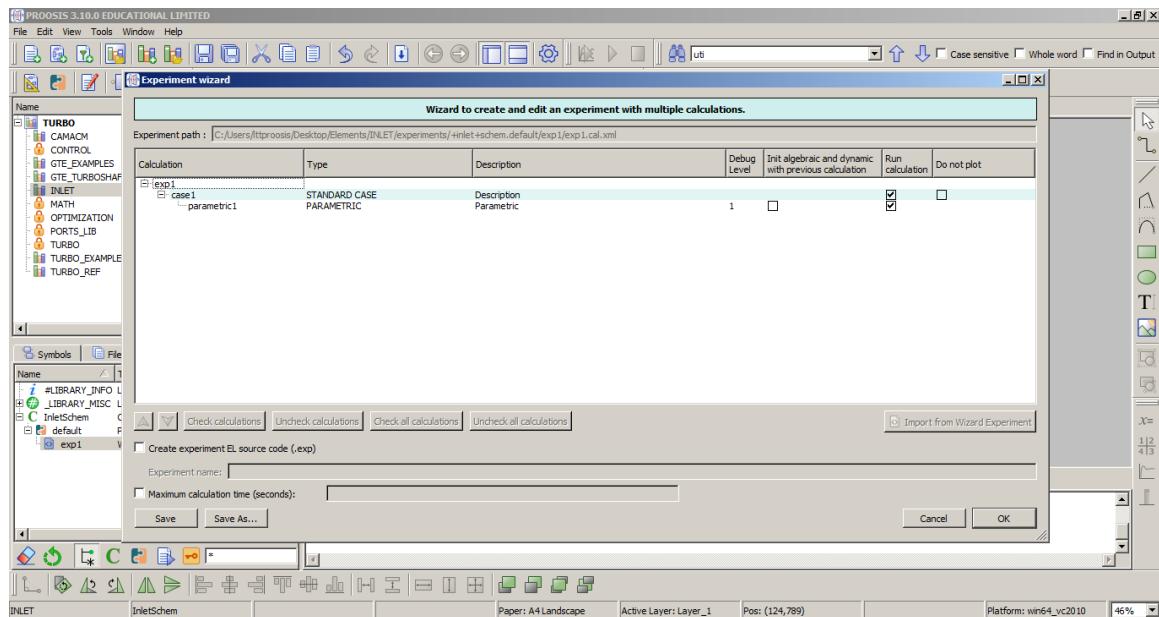


Figure 7.3 Creating a new experiment in PROOSIS

Furthermore, we edit the calculation accordingly. In our case we desire the Altitude and Inlet mass flow to be constant, whereas only the Mach number to vary, allowing the exit pressure of the inlet duct to be calculated. The parametric calculation is set by defining the initial and the final value of each variable as well as the number of steps we desire.

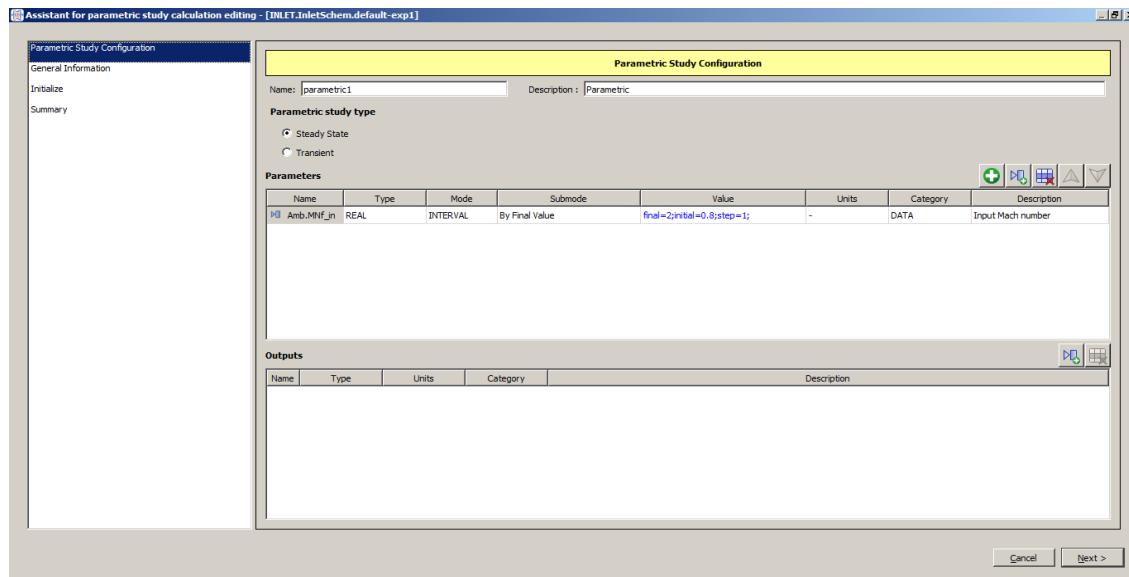


Figure 7.4 Setting-up the experiment in PROOSIS

Finally, to execute the solution and export the results of the simulation we can select the option *Simulate in Monitor* by right clicking on the experiment that we previously generated. In the monitor we can customize how the results, maps and plots are displayed:

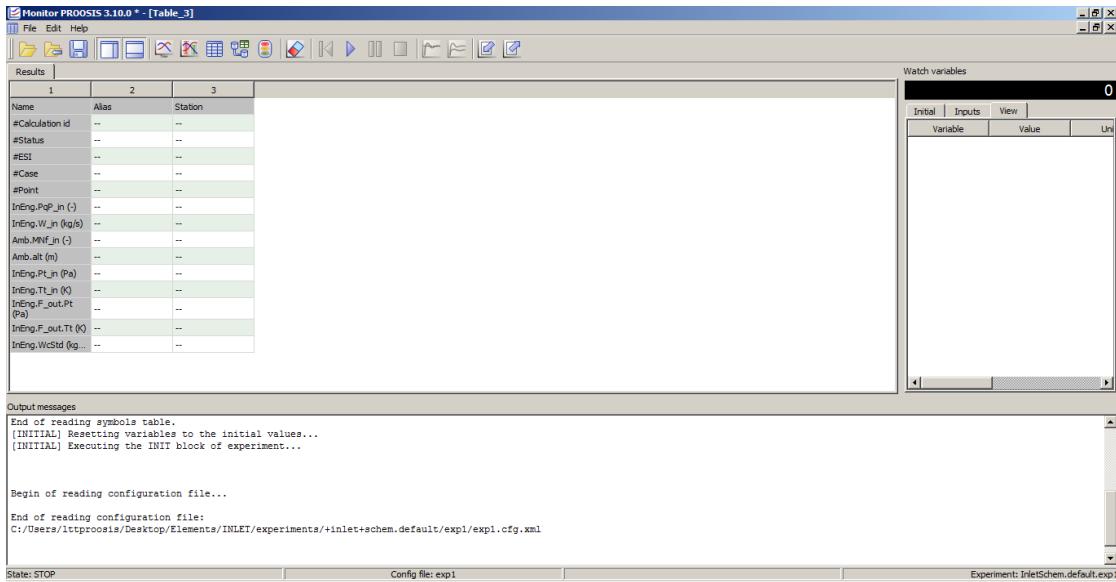


Figure 7.5 Monitor window in PROOSIS

In the end, we can run the simulation by clicking the play button located at the top toolbar and evaluate the results.

This element simulates a simple pressure drop, so results are not expected to have significant differences between the two programs.

7.1.2 NPSS

The generation of the Inlet element has been presented in a previous chapter. The only difference however, between the code presented in the previous chapter with the one used for the purpose of the comparison is that in this case metric system is used instead of the imperial, which is the default option in NPSS. For example, the altitude in the *Ambient* element and the inlet mass flow in the *InletStart* element are given in the metric system:

```
Element Ambient Amb {
    switchMode = "ALDTMN";
    alt_in = 11000. "m";
    dTs_in = 0.;
    MN_in = 0.8;
}
```

At *switchMode* option we select *ALDTMN*. This refers to altitude, dT and Mach number which will define the inlet conditions. The reason we choose the *ALDTMN* option is to ensure correlation between the two programs, as the exact option was selected in PROOSIS as well. The values selected for the three variables are identical in both programs.

```
Element InletStart InletStart {
    AmbientName = "Amb";
    W_in = 100. "kg/sec";
}
```

Finally, in order to print results to screen, one has to issue the *cout* command as previously stated. However, since we are using the metric system, we will need to print the results in this system instead of the imperial. NPSS provides us with commands that can easily convert results between the two systems. The command is the following:

```
getUnitsFactor("lbm/sec","kg/sec")
```

This command returns the factor required to convert *lbm/sec* to *kg/sec*. Thus, the printing commands will be transformed in the following:

```
cout << "W = " << F1.W *getUnitsFactor("lbm/sec","kg/sec") << " " << "kg/sec" << endl;
cout << "Ptin = " << F2.Pt* getUnitsFactor("psia","Pa") << " " << "Pa" << endl;
cout << "Ttin = " << F2.Tt*getUnitsFactor("dR","dK") << " " << "K" << endl;
```

As stated earlier, only the Mach number is varied during this comparison. The value of a variable can be changed outside of each element block, at any part of the model code, using proper strings (i.e. *Amb.MN_in* = new value to change the altitude).

Results from both programs have been exported to an Excel spreadsheet where comparison is greatly facilitated.

7.1.3 Results Comparison

Using the models described above, two cases were simulated in both programs. In each case an altitude of 11000m was set. The results are presented in the following table:

Table 7.1 PROOSIS-NPSS results comparison: Inlet component

	PROOSIS Pt2 [kPa]	NPSS Pt2 [kPa]	Difference [%]
MN=0.8	34.161	34.161	8.84E-05
MN=2.0	175.468	175.480	0.007121

As expected, only minor differences are observed.

7.2 Ambient

7.2.1 PROOSIS

To simulate ambient conditions only the symbols of *General* and *Ambient* were used from the TURBO library. Then a simulation for a set of different Altitude, Mach number and dT values was carried out.



Figure 7.6 Ambient schematic diagram in PROOSIS environment

7.2.2 NPSS

The model of the Ambient element in NPSS is generated similarly to the one described above and consists of the *Ambient*, *InletStart*, *Inlet* and *FlowEnd* elements. Note that the *Inlet* element is needed in order to establish the necessary port connections to run the simulation.

7.2.3 Results Comparison

In both programs the same cases were simulated and are presented in the following table:

Table 7.2 Ambient component simulation; considered flight conditions

A/A	Flight Altitude [m]	Mach Number [-]	Deviation from standard day [K]
1	300	0.2	0
2	9000	0.8	0
3	11500	0.85	0
4	300	0.2	10
5	9000	0.8	10
6	11500	0.85	10

The results from the two calculations are presented in the following tables. Only minor differences are observed (<0.02%).

Table 7.3 PROOSIS Ambient component simulation results

A/A	Ambient Pressure [kPa]	Ambient Temperature [K]	Free stream total pressure [kPa]	Free stream total temperature [K]	True Air Speed [m/s]
1	97.772	286.20	100.537	288.49	67.83
2	30.742	229.65	46.868	259.11	243.12
3	20.916	216.65	33.551	248.04	250.91
4	97.772	296.2	100.537	298.57	69.00
5	30.742	239.65	46.869	270.38	248.35
6	20.916	226.65	33.551	259.47	256.62

Table 7.4 NPSS Ambient element simulation results

A/A	Ambient Pressure [kPa]	Ambient Temperature [K]	Free stream total pressure [kPa]	Free stream total temperature [K]	True Air Speed [m/s]
1	97.772	286.20	100.538	288.49	67.82
2	30.742	229.65	46.872	259.11	243.12
3	20.916	216.65	33.554	248.04	250.90
4	97.772	296.20	100.537	298.57	68.99
5	30.742	239.65	46.873	270.38	248.35
6	20.916	226.65	33.554	259.47	256.62

7.3 Duct

7.3.1 PROOSIS

In order to create the component of a Duct, users may follow the instructions described previously. However, in this instance a complete model of the Duct component from the TURBO_REF library has been used. Here, a parametric study is conducted where the Mach number and dPqP (pressure drop) are varied. As previously, results are displayed at performance monitor.

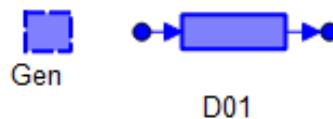


Figure 7.7 Duct schematic diagram in PROOSIS environment

7.3.2 NPSS

The generation of a model has been described in previous steps. However, it is important to remind that the declaration of the thermodynamic package should be the first statement in a code. Additionally, a view file will be included in order to print results in a desired format.

An important distinction for the *Duct* element compared to the *Inlet* model described above, is that in this case an *Ambient* element is not required as the flow of air will be set using the *FlowStart* element. The *FlowStart* element can define manually the properties of the incoming air flow.

```
#include "DuctView.view"

Element FlowStart FsAir {
    Pt = 100000 "Pa"; // Pa
    Tt = 300 "K"; // K
    W = 10 "kg/sec"; // kg/s
    WAR = 0; // Water to Air ratio
}
```

Moreover, we instantiate the *Duct* element defining the switch for the pressure drop as input:

```
Element Duct D025{
    switchDP="INPUT";
    dPqP_in = 0.0;
}
```

As previously stated, the termination of the flow must be defined, followed by the element linkages, the execution command and the display/print commands:

Element FlowEnd FeAir;

```
linkPorts( "FsAir.Fl_O" ,      "D025.Fl_I" ,      "F2" );
linkPorts( "D025.Fl_O" ,      "FeAir.Fl_I" ,      "F3" );
```

The next step is to program the parametric calculations and print results to a file. This is presented in the code below:

```
D025.Fl_IMN = 0.4;
D025.Fl_O.A = 0.500 "m2";
D025.dPqP_in = 0.0;
run();
rowSheet.update(); //prints results to the specified file
do {
    D025.Fl_O.A = D025.Fl_IA;
    run();
    CASE++;
    rowSheet.update();
    D025.dPqP_in = D025.dPqP_in + 0.01;
} while (D025.dPqP_in<=0.1);
```

```
D025.Fl_IMN = 0.5;
D025.Fl_O.A = 0.500 "m2";
D025.dPqP_in = 0.0;
run();
rowSheet.update();
do {
    D025.Fl_O.A = D025.Fl_IA;
    run();
    CASE++;
    rowSheet.update();
    D025.dPqP_in = D025.dPqP_in + 0.01;
} while (D025.dPqP_in<=0.1);
```

```
D025.Fl_IMN = 0.6;
D025.Fl_O.A = 0.500 "m2";
D025.dPqP_in = 0.0;
run();
```

```

rowSheet.update();
do {
    D025.Fl_O.A = D025.Fl_I.A;
    run();
    CASE++;
    rowSheet.update();
    D025.dPqP_in = D025.dPqP_in + 0.01;
} while (D025.dPqP_in<=0.1);
rowSheet.display();

```

7.3.3 Results Comparison

In both programs the inlet conditions were set the same: inlet total pressure 100 kPa, inlet total temperature 300 K, inlet mass flow rate 10 kg/s. Then, a parametric case was set varying the inlet Mach number (0.4÷0.6) and fractional pressure loss (0.0÷0.1). The duct area is derived to match the inlet Mach number for the considered mass flow rate, inlet total pressure and temperature.

The results of PROOSIS and NPSS calculations displayed no major differences (**<0.03%**) and are presented in [Figure 7.8](#) and [Figure 7.9](#), where respectively the exit Mach number and fractional pressure loss are plotted against the dynamic head loss coefficient dPL defined as:

$$dPL = \frac{Pt_1 - Pt_2}{Pt_1 - Ps_1}$$

Equation 7.1 Dynamic head loss coefficient

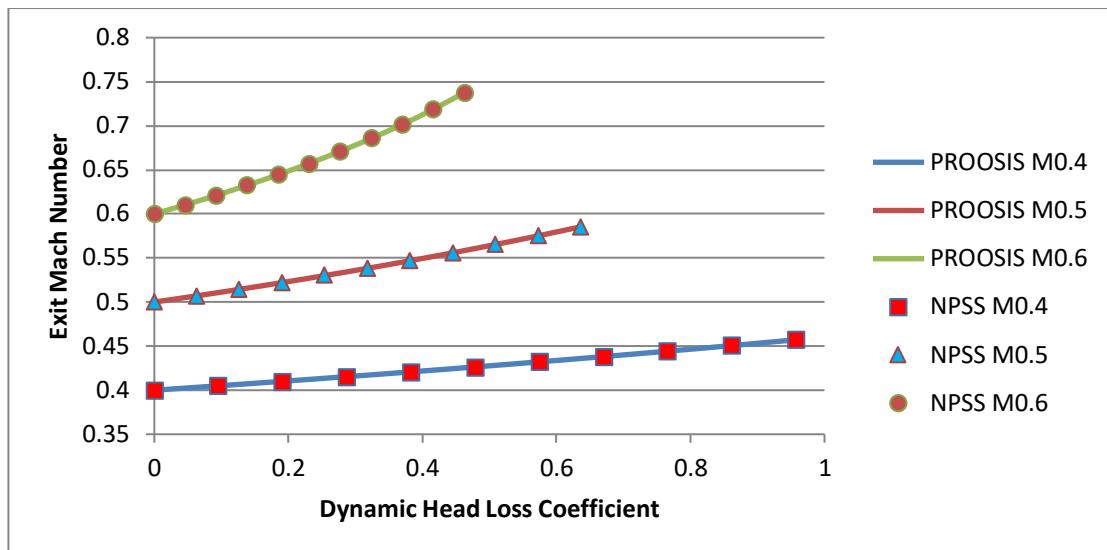


Figure 7.8 Duct outlet Mach number versus dynamic head loss factor

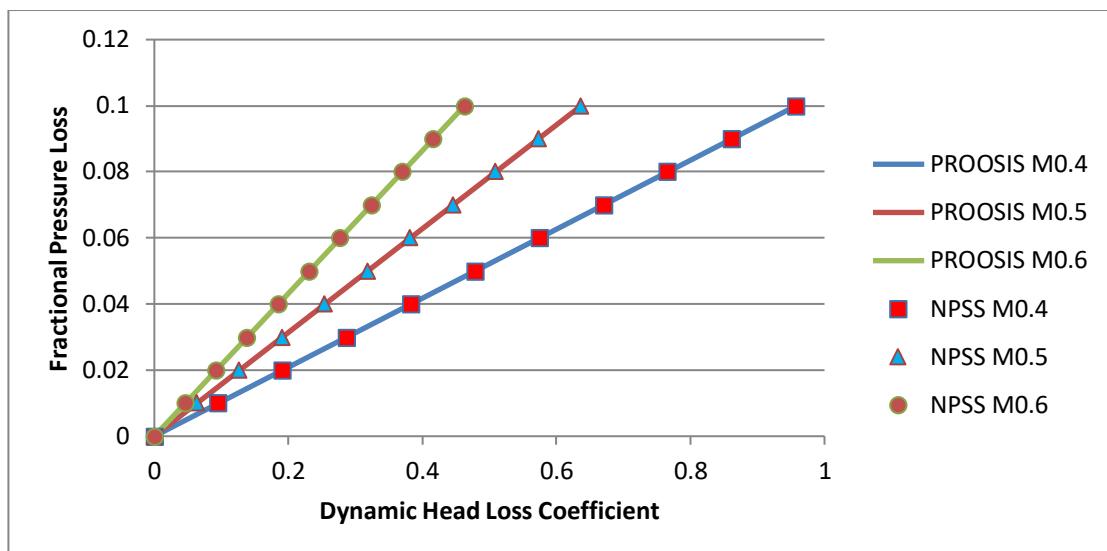


Figure 7.9 Duct fractional pressure loss versus dynamic head loss factor

7.4 Compressor

7.4.1 PROOSIS

For the Compressor element a model was generated using the *General* and the *Compressor* symbols from the TURBO library.

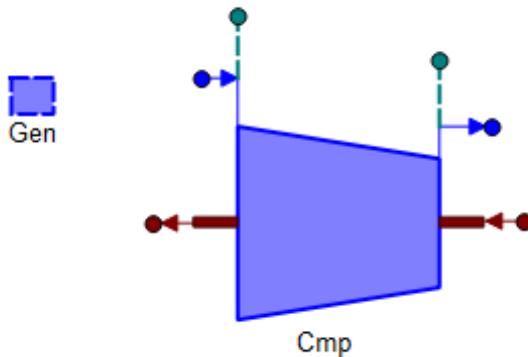


Figure 7.10 Compressor schematic diagram in PROOSIS environment

An off-design analysis has been set varying the compressor pressure ratio and mechanical speed.

7.4.2 NPSS

In order to simulate the compressor element in NPSS, one has to add custom dependent and independent variables and avoid using the *autoSolverSetup* command. The first set of custom variables is identical to the ones added by the default solver when the *autoSolverSetup* option is executed (i.e. RlineMap set as independent and Wc as dependent to achieve mass flow continuity through the component). Then a dependent variable is added to achieve a specific power request along with an independent that varies the incoming air flow. A parametric case is then set varying the power request and the mechanical speed, setting the same values that resulted from the PROOSIS simulation.

It is appropriate at this point to present the code of this model and the instantiation of the shaft and compressor elements used in the NPSS model as described below:

```
setThermoPackage("allFuel");
FlowStation fs;
fs.switchFuelType = "JP";
```

```
#include "CmpView.view";

// Start the flow of air
Element FlowStart FsAir {
    Pt = 101325 "Pa"; // Pa
    Tt = 288.15 "K"; // K
    W = 100. "kg/sec"; // kg/s
    WAR = 0; // Water to Air ratio
}

Element Compressor CmpH {
    #include "hpcE3.map";
    PRdes = 25;
    effDes= 0.9;
}
```

The first statement within the block ensures that the program will include and use a file named *hpcE3.map* containing the compressor map which will be plugged into the compressor element's *S_map* socket. The quotation marks around the file name causes NPSS to search for the file only in the current directory. The following statements of the block define the technical data that describe the specific component at design point.

The shaft element is instantiated as follows:

```
Element Shaft ShH {
    ShaftInputPort MeCmpH;
    Nmech = 10000;
}
```

The first statement creates a new input port for the *shaft* element and assigns a name to it. The rest of the statements describe the characteristics of this component. The termination of the flow, port linkages and execution of the code have already been described previously, however it would be beneficial to demonstrate the linking of the mechanical ports.

```
Element FlowEnd FeAir;

linkPorts( "FsAir.Fl_O" ,      "CmpH.Fl_I" ,      "F35" );
linkPorts( "CmpH.Fl_O" ,      "FeAir.Fl_I" ,      "F40" );
linkPorts( "CmpH.Sh_O" ,      "ShH.MeCmpH" ,      "MeCmpH" );
```

```
setOption("switchDes", "DESIGN");
run();
```

The command *setOption* specifies whether the model will run in design or in off-design mode. At first, a design point is simulated to calculate map scalars, followed by an off-design analysis. At this point, custom dependent and independent variables will be created as follows:

```
setOption("switchDes", "OFFDESIGN");
```

```
Independent ind_Rline{
    varName = "CmpH.S_map.RlineMap";
}
```

```
Dependent dep_Wc {
    eq_lhs = "CmpH.Wc"; //model value
    eq_rhs = "CmpH.WcCalc"; //target value
}
```

```
solver.addIndependent("ind_Rline");
solver.addDependent("dep_Wc");
```

```
real P = -48200.0774*getUnitsFactor("kW", "hp");
```

A new variable is created, that will specify the power demand in the corresponding dependent variable. The *getUnitsFactor* command, returns a value that serves for unit conversion (here we convert kW to hp, as the power is given in horsepower in NPSS).

```
Dependent dep_pwr {
    eq_lhs = "MeCmpH.pwr"; //model value
    eq_rhs = "P"; //target value
}
```

```
Independent ind_Wair{
    varName = "FsAir.W";
}

solver.addDependent("dep_pwr");
solver.addIndependent("ind_Wair");

run(); //run the first off-design point
```

```

rowSheet.update(); //print results to the specified file

P = -45458.1367*getUnitsFactor("kW","hp");
ShH.Nmech = 9900;
run(); //run another off-design point
rowSheet.update();

```

Users, can add as many off-design points as they desire, setting a new value for the power command and mechanical speed.

7.4.3 Results Comparison

The same inlet and design point conditions were set in both programs as well as map files and can be seen in the following table. Then a parametric study was in each program as mentioned above. The design point results, as expected, present minor differences, lower than **0.03%**.

Table 7.5 Input data for Compressor component comparison

Total inlet pressure [Pa]	101325
Total inlet temperature [K]	288.15
Inlet mass flow [kg/s]	100
Pressure ratio	25
Efficiency	0.9
Mechanical speed at design point [rpm]	10000
BETA parameter at design point	0.33
R-line parameter at design point	2.0

Note that the default R-line at design point is set equal to 2.0. Consequently, the corresponding BETA parameter value is equal to 0.33 as presented in the table above.

Comparisson between efficiency and pressure ratio during off-design analysis is presented in [Figure 7.11](#), where differences lower than **0.03%** are observed as expected.

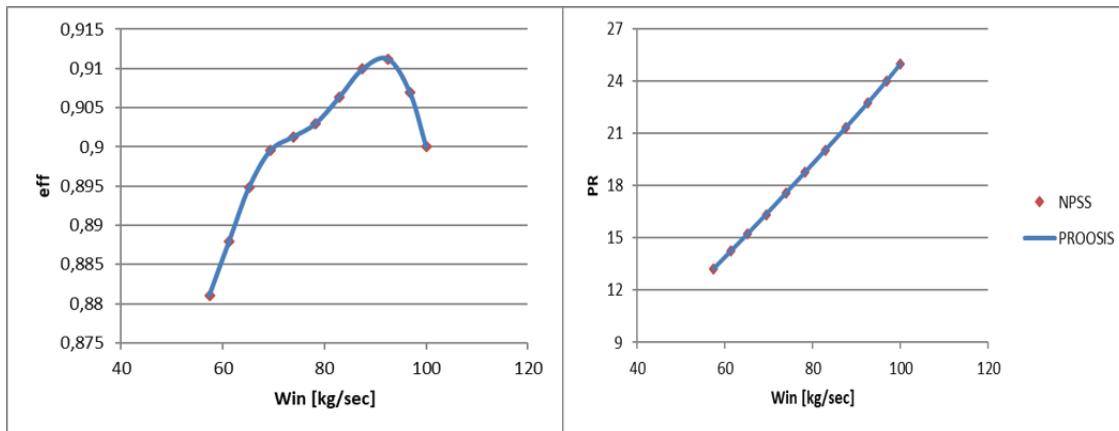


Figure 7.11 Compressor off-design performance in PROOSIS and NPSS

7.4.4 Fan

A comparison between the fan components between the programs will not be presented, as the NPSS uses the same compressor element to simulate the fan as well. As mentioned earlier, the differentiation is possible with different map files. Since we have ensured that the map files used in each program are identical, similar differences to the compressor element are also expected in the case of the fan.

It is important however to mention an important difference between the components of the two programs. PROOSIS fan component, is able to divide the incoming flow into two streams, apply different pressure ratio on each stream and use different map files as well. NPSS fan element on the other hand, cannot divide the flow and uses a single map file as well, not allowing for different pressure ratio on each stream (this can be achieved with different modeling solutions that will be presented in the following chapters 8.2). Consequently, in order for someone to compare the fan components between the programs, a compressor component with the fan map would need to be used in PROOSIS.

7.5 Burner

7.5.1 PROOSIS

To simulate the Burner component in PROOSIS only the *General* and the *Burner* symbols were used from the TUBRO library. Additionally, ambient conditions, the pressure drop through the burner, the burner efficiency and the respective switches, the lower heating value of the fuel and the fuel type were defined. Finally, a parametric study has been conducted varying the fuel flow rate.

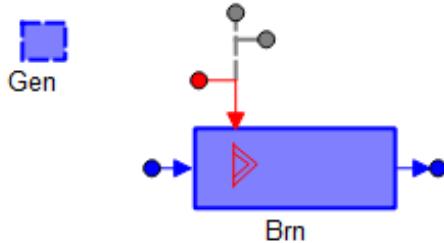


Figure 7.12 Burner schematic diagram in PROOSIS environment

7.5.2 NPSS

For the Burner component after the definition of the thermodynamic package, it is required to define the desired fuel type. This action is completed through the creation of a flow station in which we can select the type of fuel through a range of different fuels that are included in the program.

```
setThermoPackage("allFuel");
FlowStation fs;
fs.switchFuelType = "JP";
```

Flow stations can be greatly useful, especially in large models, as they can instantiate easily a flow of air or fuel.

The definition of the desired fuel is completed through the fuel switch of the flow station. Note that NPSS *allFuel* station does not keep track of the properties of the unburnt fuel; therefore, it does not distinguish JP4, JP7 and JP8. The only differentiation between these fuels comes through their lower heating value.

As shown in previous models instantiation of the air flow is coded as follows:

```
Element FlowStart FsAir {
    Pt = 101325 "Pa"; // Pa
    Tt = 600 "K"; // K
    W = 100 "kg/sec"; // kg/s
    WAR = 0; // Water to Air ratio
}
```

Moreover, we need to instantiate the flow of fuel and define the lower heating value of the fuel with the *FuelStart* element:

```
Element FuelStart FusEng {
    LHV = (4.3124*10**7)*getUnitsFactor("J/kg", "Btu/lbm");
}
```

Recall that the command *getUnitsFactor* returns a value that is used for unit conversion and the LHV value is set equal to the one used in PROOSIS.

The Burner element is coded as follows:

```
Element Burner BrnPri {
    dPqP_dmd = 0.05; // user input friction relative pressure drop (Pin - Pout)/Pin
    effBase = 0.98; // user input burner adiabatic efficiency
    switchBurn = "FUEL";
    Wfuel = 0.2 "kg/sec";
}
```

Inside the *Burner* block we assign the attributes needed to perform calculations. The *switchBurn* option determines if burner is running to fuel flow, FAR or Tt4. For the fuel flow there are two possible options, *FUEL* and *WFUEL*. The former expects the fuel flow to be specified inside the *burner* block, whereas the later acquires the incoming fuel flow that is specified in the *FuelStart* block. Finally, the last statement determines the fuel flow rate. The rest of the code for the Burner model is similar to the models described previously; therefore, it will not be presented.

7.5.3 Results Comparison

Identical conditions were set in both cases and are presented in the following table. Then an experiment is simulated in which the fuel flow rate is set to vary from 1 to 4 kg/s (the resulting fuel-to-air ratio varies from 0.01 to 0.04).

Table 7.6 Input data for Burner component comparison

Total inlet pressure [Pa]	101325
Total inlet temperature [K]	600
Inlet mass flow [kg/s]	100
Relative pressure drop	0.05

Efficiency	0.9995
Fuel flow [kg/s]	1

The calculated burner exhaust temperature from PROOSIS and NPSS for the examined operating conditions is shown in [Table 7.7](#). The maximum absolute difference is **0.012%**.

Table 7.7 PROOSIS-NPSS result comparison: Burner component

A/A	Fuel to air ratio	PROOSIS Burner Exhaust Temperature [K]	NPSS Burner Exhaust Temperature [K]	Difference [%]
1	0.01	977.44	977.49	-0.024
2	0.02	1312.80	1312.81	-0.096
3	0.03	1615.94	1615.73	-0.322
4	0.04	1889.69	1889.60	-0.754

In the following figure readers can observe graphically the results of the simulation considered above.

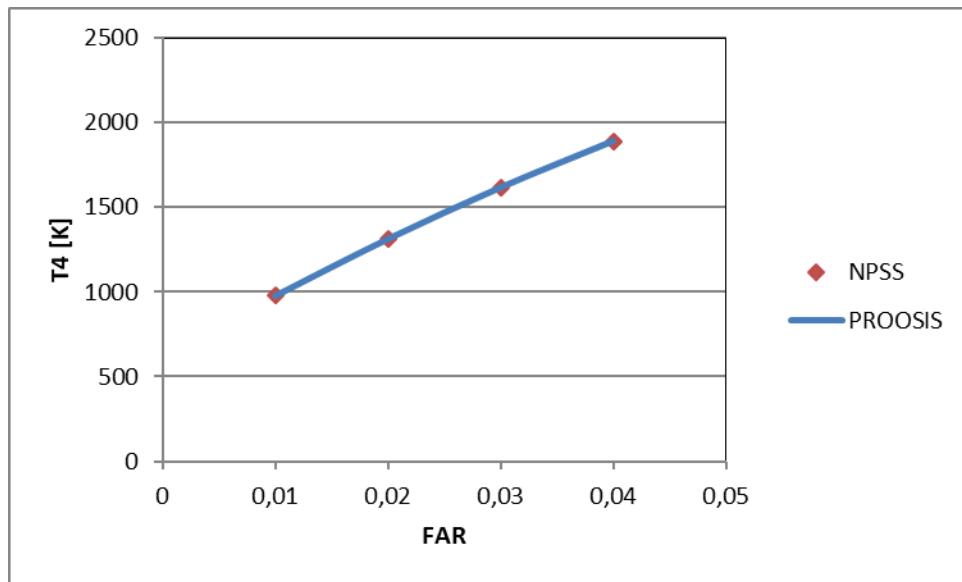


Figure 7.13 Burner exhaust temperature against fuel-to-air ratio

It would be beneficial at this point to present results at the burner component between NPSS and one of PROOSIS included fluid models. Taking a closer look on the results, it is easy to notice that for increasing temperature rise, the difference between the

results is also increasing. The maximum difference is 1.8% at FAR = 0.05, which is attributed to differences in the fluid models at high temperatures, as indicated in the previous chapter. However, 0.05 is an extreme high value, used here only to emphasize the difference between the default fluid models, whereas for the most common jet engine operating conditions (FAR 0.02), differences drop to around 0.2%. In PROOSIS the JP4 fuel was selected whereas in NPSS the JP option. As mentioned before, in NPSS the JP option is designed to produce properties for the combustion products of fuel, water and air, where the fuel H-C atom ratio is 2:1. It does not keep track of the properties of unburned fuels; therefore, it does not need to distinguish between JP4, JP7, JP8, where the H-C ratio value is close for all these fuels.

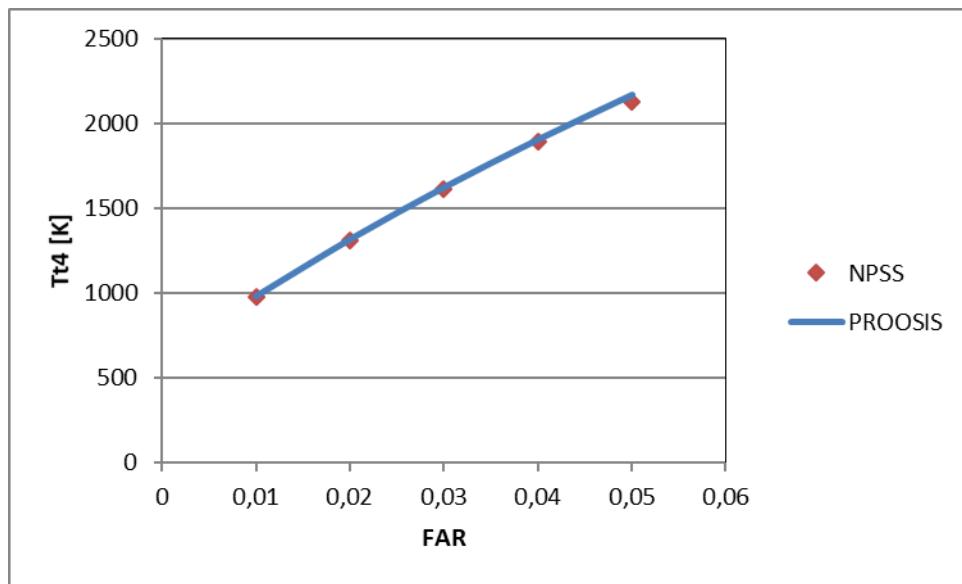


Figure 7.14 Burner exhaust temperature against fuel-to-air-ratio between the allFuel NPSS thermodynamic package and PROOSIS JP4 fluid model

7.6 Turbine

7.6.1 PROOSIS

The schematic of the PROOSIS consists of the *General* and the *Turbine* components. In the parametric case the pressure ratio and mechanical speed were varied as it will be described in the following paragraph.

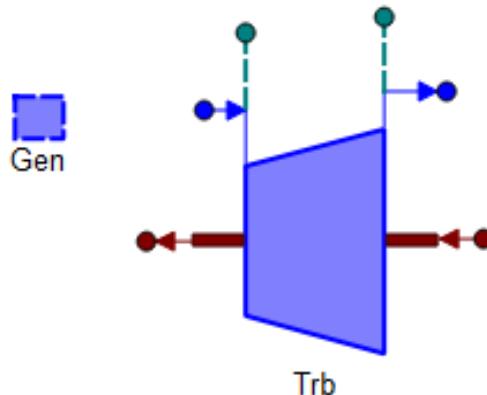


Figure 7.15 Turbine schematic diagram in PROOSIS environment

7.6.2 NPSS

For the case of the Turbine element, we opted to use results from a simple turbojet model in NPSS. This means that results of the turbine part were used as inputs for the PROOSIS turbine model, ensuring the same component operating conditions as well. More specifically inlet conditions, pressure ratio and mechanical speed were used as inputs in the parametric study in PROOSIS.

The Turbojet model will not be presented here, as it will be described separately in the following chapter.

7.6.3 Results Comparison

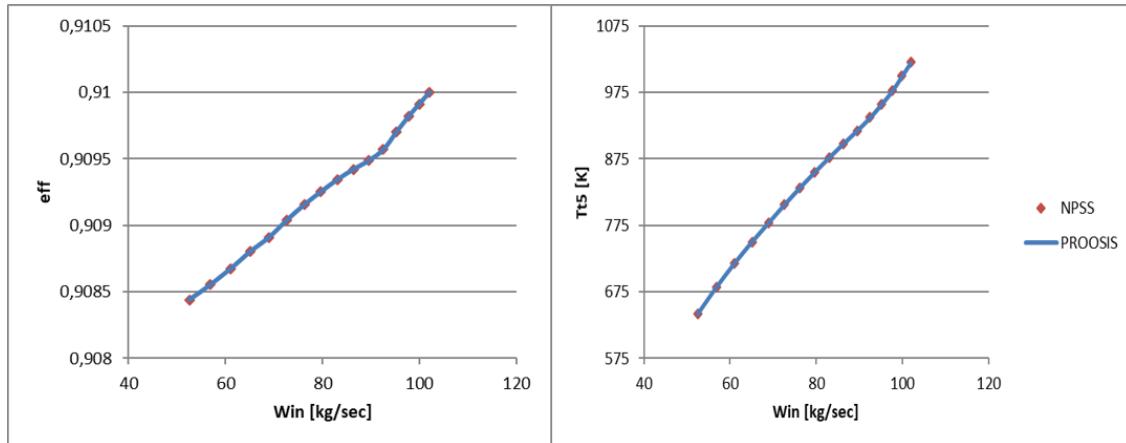
As mentioned above, PROOSIS *Turbine* component is compared against the NPSS Turbine element results from the Turbojet model described in Chapter 8.1.2. The first step is to scale the turbine map at design point conditions and then to simulate different off-design cases varying the fuel flow rate.

Additionally, a turbine model is generated in PROOSIS, a new experiment is set and the design point conditions from the NPSS simulation are set as boundary values. Then, the off-design points are defined by setting the mechanical speed and pressure ratio resulted from the turbine component in the Turbine model.

In the following table the design point values used in the PROOSIS turbine model are presented.

Table 7.8 PROOSIS Turbine component experiment boundaries

Total inlet Pressure [kPa]	742.387
Total inlet temperature [K]	1350.785
Inlet mass flow [kg/s]	102
Inlet FAR	0.02
Efficiency	0.91
Mechanical speed [rpm]	10000
Pressure ratio	3.766
ZETA parameter at design point	0.74
PR-line parameter at design point	4.96

**Figure 7.16 Turbine off-design performance in PROOSIS and NPSS**

Minor differences are observed, as expected, and the maximum absolute difference in the off-design analysis is **0.013%** for the total exit temperature (T_{t5}).

7.7 Nozzle

7.7.1 PROOSIS

A model of a converging Nozzle from the TURBO_REF library was used in this case. The only symbols needed for the schematic are the *General* and the *Nozzle*.

In the parametric case the total inlet pressure is set to vary, whereas the ambient conditions and the inlet conditions are assigned specific values.

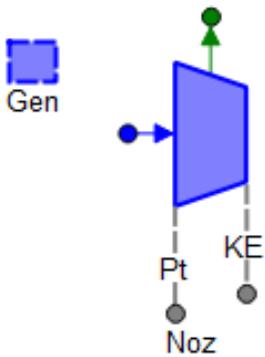


Figure 7.17 Nozzle schematic diagram in PROOSIS environment

7.7.2 NPSS

In the case of the Nozzle there is a big distinction not only in the instantiation process between PROOSIS and NPSS, but in the nozzle coefficients as well. NPSS requires different variables to be instantiated compared to PROOSIS. More specifically the variables that are required to instantiate the Nozzle element in PROOSIS are the total inlet pressure and temperature, the exit area, the mass flow and the ambient pressure. On the contrary, the default NPSS nozzle solver requires the total inlet pressure and temperature, mass flow and exit pressure as inputs in order to calculate the exit area. The exit pressure can be either user-defined or set equal to the ambient pressure. Subsequently, we created a new dependent (*Aexit*) and independent (*Wair*) variable in NPSS in order to produce the same simulation in both software.

The code of the Nozzle model is presented below:

```
setThermoPackage("allFuel");
Element FlowStart FsAir {
    Pt = 30000 "Pa"; // Pa
    Tt = 1000. "K"; // K
    W = 14.15 "kg/sec"; // kg/s initial guess
    WAR = 0; // Water to Air ratio
}
```

The Nozzle element is coded as shown:

```
Element Nozzle NozPri {
```

```

switchType = "CONIC";
switchCfg = "CALCULATE";
dPqP = 0.00;
PsExh = 22630 "Pa";
}

```

The *switchType* option specifies the type of nozzle geometry (i.e. convergent, convergent-divergent, fixed). *SwitchCfg* option determines that correction to ideal gross thrust is calculated and not defined. There is also a *switchCoef* option that determines the type of coefficient that will be used, however it will be discussed in the following paragraph. Every coefficient in each program was assigned a value of 1. The third statement specifies that there is no pressure loss across the Nozzle. Finally, the fourth statement defines the static outlet pressure of the Nozzle. Instead of the *PsExh* command, one may use the *PsExhName* command, where the user must declare the element from which the program will obtain the exit pressure for the Nozzle. It is common for users to select the Ambient element so that the exit pressure of the Nozzle is set equal to the atmospheric conditions (*PsExhName = "Amb.Ps";*).

```

// End the flow of air
Element FlowEnd FeAir;

// SET ELEMENT LINKAGES
linkPorts( "FsAir.Fl_O" ,           "NozPri.Fl_I" ,      "F060" );
linkPorts( "NozPri.Fl_O" ,          "FeAir.Fl_I" ,       "F090" );

```

Define a new variable and set the custom dependent and independent variables in order to achieve a specific exit area by varying the inlet mass flow. Firstly, run a design point to calculate the nozzle properties and then run the desired off-design points.

```

real A {
    value = 0.; IOstatus = INPUT; units = NONE;
    description = "Aexit";
}

```

```

Independent W_in {
    varName = "FsAir.W"; // <-- the variable that this Indep. can vary
}

```

```

Dependent Target_A {
    eq_rhs = "NozPri.Aexit";
}

```

```

eq_lhs = "A";
}

A = 655.806312; //0.4231 m2

setOption( "switchDes", "DESIGN" );
autoSolverSetup();
solver.addDependent("Target_A");
solver.addIndependent("W_in");
run();

Independent Win { // new independent for the off-design
    varName = "FsAir.W";
}

setOption( "switchDes", "OFFDESIGN" );
solver.clear();
autoSolverSetup();
solver.addIndependent("Win");

// run multiple off-design points varying the total inlet pressure
FsAir.Pt = 40000 "Pa";
run();

```

7.7.3 **Nozzle coefficient comparison**

Besides the default solver setup difference between the programs mentioned above, the nozzle coefficients that each nozzle component is equipped with also differ. In the following table the various coefficients that each program can use are presented:

Table 7.9 Nozzle coefficient comparison

PROOSIS	NPSS
C_d	$C_{d\text{Th}}$
C_x	C_v
C_{fg}	C_{fg}
	C_{ang}
	C_{qua}
	$C_{mixcorr}$

PROOSIS Nozzle

- C_d = Discharge coefficient
 - Used in the calculation of effective exit area, $A_{exit\ eff}$
- C_x = Thrust coefficient
 - Used in the calculation of gross thrust, F_g
- C_{fg} = Ideal thrust coefficient
 - Is calculated by the program

NPSS Nozzle

- C_{dTh} = Throat discharge coefficient
 - Used in the calculation of throat effective area
 - Can be obtained from a subelement
 - Can be calculated by the program
- C_v = Exit velocity coefficient
 - Used in the calculation of gross thrust
 - Can be obtained from a subelement
 - Can be calculated by the program
- C_{fg} = Gross thrust coefficient
 - Used in the calculation of gross thrust, F_g
 - Can be obtained from a subelement
 - Can be calculated by the program
- C_{ang} = Exit flow angle coefficient
 - Used in the calculation of actual gross thrust and V_{actual}
 - Used only when C_v is used
- C_{qua} = Throat thermal expansion coefficient
 - Used in the calculation of throat effective area
- $C_{mixcorr}$ = Thrust correction due to partial mixing upstream or in the nozzle
 - Used in the calculation of actual gross thrust and V_{actual}
 - Used only when C_v is used

From the above, we can see that the first three coefficients of each program at [Table 7.9](#), are identical. However, NPSS provides users with additional options giving us the

ability to consider additional effects in the nozzle area, thus allowing for different fidelity simulations.

7.7.4 Results Comparison

As mentioned previously, the nozzle component requires different instantiation in each program. A big distinction between the two is that the default solver in NPSS demands an exit pressure to be given in order for the code to calculate the exit area, whereas in PROOSIS users can define a specific exit area. An additional difference is that each program uses different coefficients. Subsequently, aiming for an analogous comparison, only the Cfg (nozzle exit gross thrust coefficient) variable was used for comparison, whereas the rest of the coefficients were set equal to 1. Finally, a parametric study was set varying the nozzle inlet total pressure.

Differences are below **0.02%** in all cases, as expected since the same fluid model is used in both programs. In [Table 7.10](#) input data used for instantiation are shown and in [Figure 7.18](#) a comparison between few performance parameters is presented.

Table 7.10 Input data for Nozzle component comparison

Inlet temperature [K]	1000
Inlet pressure [kPa]	30
Ambient pressure [kPa]	22.63
Exit Area [m²]	0.4231

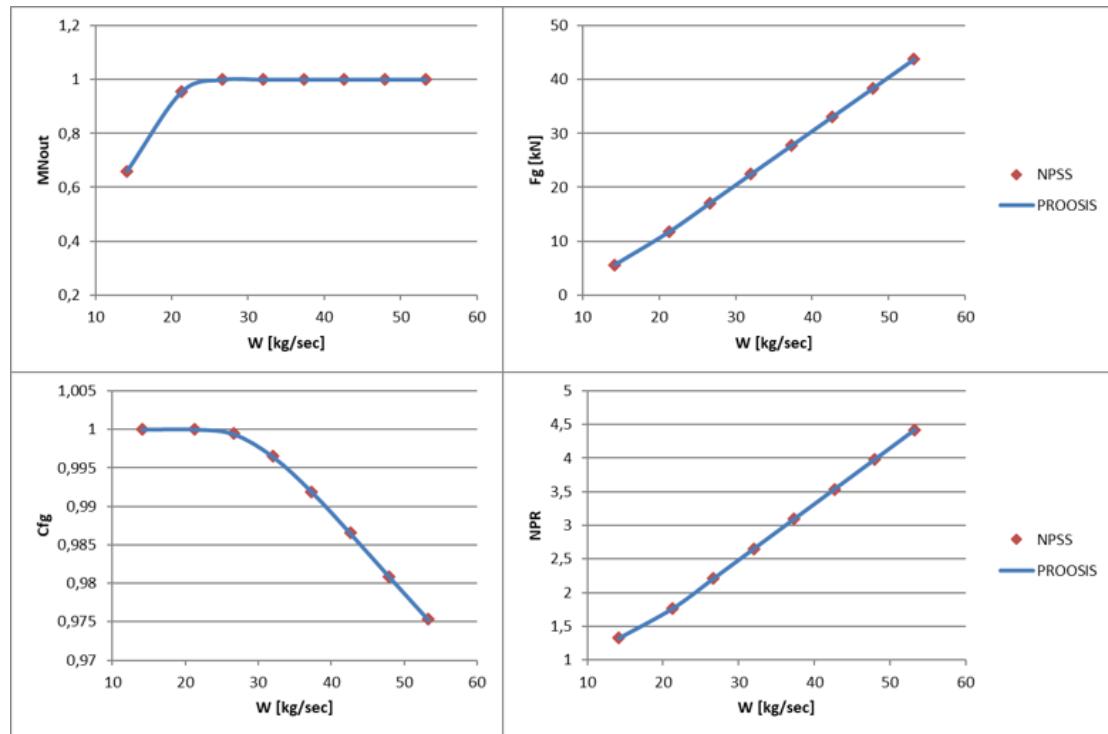


Figure 7.18 Nozzle performance comparison in PROOSIS and NPSS

8

ENGINE LEVEL COMPARISON

Following individual component generation, in the present chapter complete engine models and simulations will be presented. At first a simple Turbojet engine model will be compared, followed by a Turbofan engine and finally a Turboshaft engine. As in the case of individual component models, similarly in the case of a complete engine model, the generation process of each program is different and will be presented in the following pages. Finally, as mentioned in the previous chapter, standard day assumptions are made and the same fluid model and performance maps are used for every calculation presented in this chapter.

8.1 Turbojet

8.1.1 PROOSIS

Component model generation has been described thoroughly in the previous chapter. Therefore, it is now simple to create a complete model of a simple Turbojet engine by linking all the different components of the engine mentioned before.

Following the same procedure as we did in each individual component we can now start building a Turbojet model by dragging-and-dropping the included components from the TURBO library in a new schematic window.

Firstly, the *General* and *Atmosphere* symbols are inserted which will provide the input variables of the atmospheric conditions for the *Inlet* of the engine. Moreover, we continue by adding the *Inlet* of the engine, followed by the high pressure compressor, a duct to simulate a pressure loss, the burner, the high pressure turbine and finally a convergent nozzle. In the next step the fluid ports of each component are linked accordingly so as the outlet of one component is connected to the inlet of the next component in order to transfer the necessary information through the engine. Then a shaft

to connect the HPC and HPT is added, as well as the *shaft_start* and *shaft_end* components to dictate the power flow through the engine. *Shaft_start* is connected to the HPT whereas the *shaft_end* is connected to the HPC as power is generated in the Turbine and is transferred through the shaft to the Compressor and rotates it. Finally, a *Performance Monitor* is added which is connected to all necessary ports of each component and is used to extract the results of the thermodynamic calculations.

The complete model of the Turbojet engine can be seen in the image below:

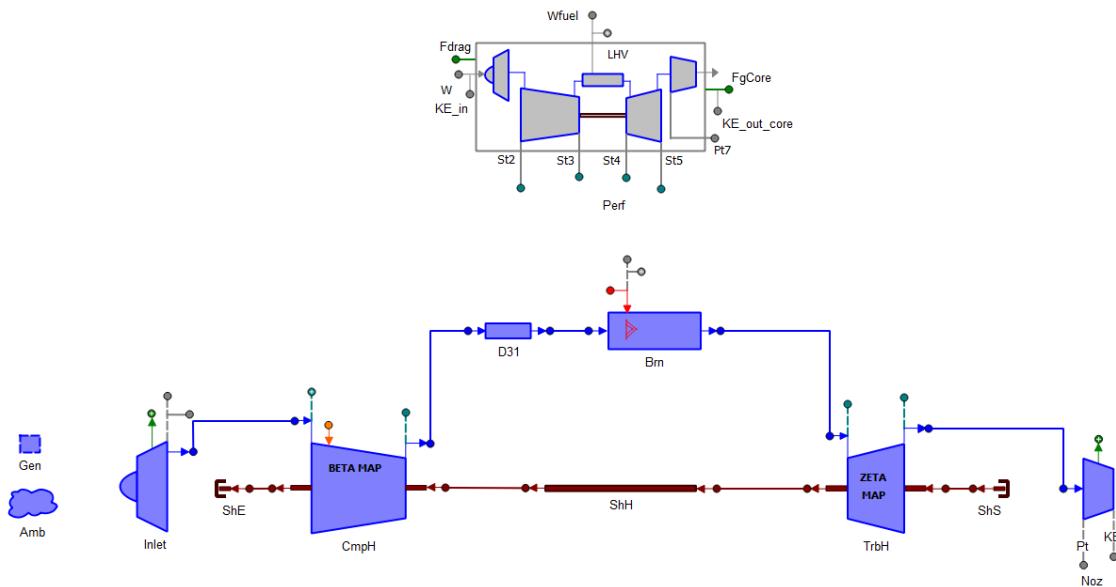


Figure 8.1 Turbojet schematic in PROOSIS environment

Having now finished building our model we can Compile it by pressing F7 and then continue setting-up the experiment. The first step is to define all necessary data and run a Design point (*Extended Steady calculation*) to set the maps scalars and later run the Off-Design points through a *Parametric* calculation. Finally, the simulation can be done in the monitor where the results and maps are displayed in the desired format.

8.1.2 NPSS

As mentioned in the previous chapter, creating a component and thus a complete engine model in NPSS differs from PROOSIS. Aside from the absence of the graphical interface in NPSS, the coding and instantiation process of a model is different between the software. In the following image we can see the corresponding block diagram of a simple Turbojet engine. Note that since NPSS does not have a graphical interface, this diagram is only for visualization of the layout and facilitation of the programming process.

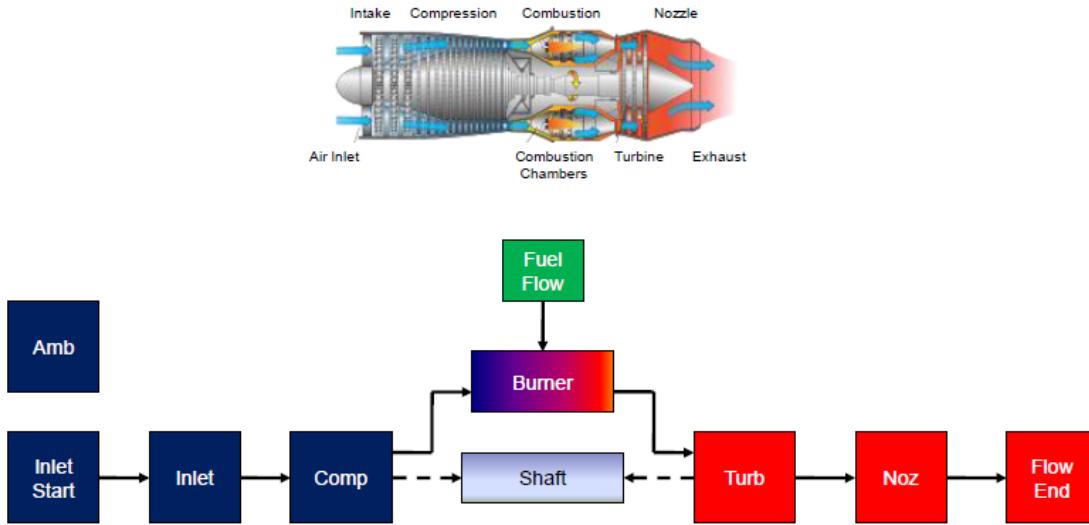


Figure 8.2 Turbojet schematic on NPSS, [26]

Model generation in NPSS is similar to the steps followed for each component, only that in this case linking of the elements altogether is needed, keeping in mind the layout of our model. Thus, the first step of the process is to define the thermodynamic package as well as the type of fuel to be used in the simulation and the view file to be used to export the results of the simulation:

```

setThermoPackage("allFuel");
FlowStation fs;
fs.switchFuelType = "JP";
#include "TurbojetView.view"
  
```

Then, the atmospheric conditions are set and the flow of air is initiated:

```

Element Ambient Amb {
    switchMode = "ALDTMN";
    alt_in     = 38000.;
    dTs_in    = 0.;
    MN_in     = 0.8;
}
  
```

```

Element InletStart InletStart {
    AmbientName = "Amb";
    W_in = 100. "kg/sec";
}
  
```

The Inlet duct, HPC and the duct connecting the compressor exit to the inlet of the Burner follows:

```
Element Inlet InEng {
    PqP_in = 0.995;
}

Element Compressor CmpH {
    #include "hpcE3.map";
    PRdes = 25;
    effDes = 0.9;
}

Element Duct D035{
    switchDP="INPUT";
    dPqP_in=0.002;
}
```

Additionally, the initialization of the fuel is added as well as the fuel properties (i.e. lower heating value), followed by the Burner element. It is important to note that the LHV must be set equal in both NPSS and PROOSIS:

```
Element FuelStart FusEng {
    LHV = (4.3124*10**7)*getUnitsFactor("J/kg", "Btu/lbm");
}

Element Burner BrnPri {
    dPqP_dmd = 0.05;
    effBase = 0.98;
    switchBurn = "FUEL";
    Wfuel = 2 "kg/sec"; //initial guess
}
```

Burner is followed by the HPT and then the converging exit Nozzle:

```
Element Turbine TrbH {
    #include "hptE3.map";
    PRbase = 4; // pressure ratio initial guess
    effDes = 0.91;
}
```

```
Element Nozzle NozPri {
    switchType = "CONIC";
    switchCfg = "CALCULATE";
    dPqP = 0.00;
    PsExhName = "Amb.Ps";
}
```

It is important to mention that during model generation in PROOSIS no pressure ratio of the Turbine was needed to be given at any point; instead it was a variable that was calculated from the program. On the contrary, on NPSS an initial guess of the pressure ratio of the Turbine is needed for it to be calculated.

Next follows the shaft to connect the power flow between the HPC and HPT and then the termination of the flow:

```
Element Shaft ShH {
    ShaftInputPort MeCmpH, MeTrbH;
    Nmech = 10000;
}
```

Element FlowEnd FeAir;

Afterwards, the *EngPerf* element must be added in order to extract results of the overall engine performance such as OPR (overall pressure ratio):

```
Element EngPerf Perf {
}
```

Finally, linkages of the fluid, fuel and shaft ports of the elements is presented below:

<i>linkPorts ("InletStart.Fl_O",</i>	<i>"InEng.Fl_I",</i>	<i>"F010");</i>
<i>linkPorts ("InEng.Fl_O",</i>	<i>"CmpH.Fl_I",</i>	<i>"F020");</i>
<i>linkPorts ("CmpH.Fl_O",</i>	<i>"D035.Fl_I",</i>	<i>"F030");</i>
<i>linkPorts ("D035.Fl_O",</i>	<i>"BrnPri.Fl_I",</i>	<i>"F031");</i>
<i>linkPorts ("FusEng.Fu_O",</i>	<i>"BrnPri.Fu_I",</i>	<i>"FU035");</i>
<i>linkPorts ("BrnPri.Fl_O",</i>	<i>"TrbH.Fl_I",</i>	<i>"F040");</i>
<i>linkPorts ("TrbH.Fl_O",</i>	<i>"NozPri.Fl_I",</i>	<i>"F050");</i>
<i>linkPorts ("NozPri.Fl_O",</i>	<i>"FeAir.Fl_I",</i>	<i>"F060");</i>
<i>linkPorts ("CmpH.Sh_O",</i>	<i>"ShH.MeCmpH",</i>	<i>"MeCmpH");</i>
<i>linkPorts ("TrbH.Sh_O",</i>	<i>"ShH.MeTrbH",</i>	<i>"MeTrbH");</i>

It is important to mention that in both programs the Design point was set so that the fuel flow is calculated such that a specific thrust is achieved. In order to set this calculation in NPSS, custom dependent and independent variables have been added in the solver. The additional code can be seen below:

```
real Fn_req = 15819.426727751; // lbf, requested net thrust
```

```
Independent ind_Wfuel{
    varName = "BrnPri.Wfuel";
}

Dependent dep_Fn {
    eq_lhs = "Perf.Fn"; // model value
    eq_rhs = "Fn_req"; // target value
}
```

At this point the building process of the Turbojet model has been completed. Next step is to run a simulation and evaluate the results. To run the Off-Design points, one has first to run the Design point. NPSS does not need a file to store the results of Design point simulation in order to use it later for the Off-Design, as PROOSIS does. Simply one has to run the code in Design mode and thereupon run the code again but in Off-Design mode. The program uses Design data automatically to calculate the Off-Design points, provided a Design run has first been completed as previously mentioned.

Therefore, the commands required to simulate the Design and Off-Design points can be seen below:

```
setOption( "switchDes", "DESIGN" );
autoSolverSetup();
solver.addIndependent("ind_Wfuel");
solver.addDependent("dep_Fn");
run();
cout << "Solver converged (1 = yes, 0 = no) ? = " << solver.converged << endl;
cout << "Iterations = " << solver.iterationCounter << endl;
rowSheet.update();

setOption( "switchDes", "OFFDESIGN" );
solver.debugLevel = "ITERATION_DETAILS";
solver.diagnosticFile = "solverOffDes.out";
solver.clear();
autoSolverSetup();
```

```

solver.addIndependent("ind_Wfuel");
solver.addDependent("dep_Fn");

Fn_req = 0.95*15819.426727751; // run the first off-design point
run();
rowSheet.update();
Fn_req = 0.90*15819.426727751; //run another off design point
run();
rowSheet.update();
rowSheet.display();

```

Commands such as `solver.debugLine`, `solver.diagnosticFile`, `solver.converged` and `solver.iterationCounter`, simply provide users with more information about the iterations and convergence of the code, whereas `autoSolverSetup` ensures that the default setup of the selected simulation mode is set. As previously mentioned, this command sets the default dependent and independent variables of the objects that have the attribute `autoSetup` set to `TRUE`, ensures power equilibrium on the shaft and is often used when different simulation modes take place in the same model, such as in our case where we have a Design mode followed by an Off-Design mode.

Changing the requested thrust on Off-Design mode will generate a new Off-Design operation point. Note that users can run multiple Off Design points simply by setting different values to the desired variables. There are other ways to run multiple Off-Design points as well. The simplest one would be to change manually the value of `BrnPri.Wfuel` (or any desired variable) each time and then execute the code. However, when a large number of Off-Design points are simulated this method is not suggested as it is very time consuming. Usually, a do-while loop is used in cases such as this one and it is the fastest and simplest way of exporting a large amount of data from different simulations. A possible example is presented below:

```

BrnPri.Wfuel = 0.2 "kg/sec";
do {
    run();
    rowSheet.update();
    BrnPri.Wfuel = BrnPri.Wfuel + 0.2;
} while (BrnPri.Wfuel<=2);
rowSheet.display();

```

Finally, the command `rowSheet.update()`; serves to export the results of the simulation using the view file specified in the beginning of the model, whereas `rowSheet.display()`; prints user information, the version of the program and date and time of the simulation on the output file.

8.1.3 Results Comparison

As already mentioned, firstly a design point is simulated to calculate map scalars. The boundary values selected for this simulation are presented in the following table.

Table 8.1 Turbojet design point input data

Flight Altitude [ft]	38000
Mach Number	0.8
Inlet mass flow [kg/s]	100
Inlet PqP	0.995
Compressor pressure ratio	25
Compressor efficiency	0.9
Duct dPqP	0.002
Burner efficiency	0.98
Burner dPqP	0.05
Turbine efficiency	0.91
Mechanical speed [rpm]	10000
BETA parameter	0.33
ZETA parameter	0.74
R-line parameter	2.0
PR-line parameter	4.96
Requestet Thrust [kN]	70.368

In the following figure, the design point percentage differences for various cycle parameters are presented.

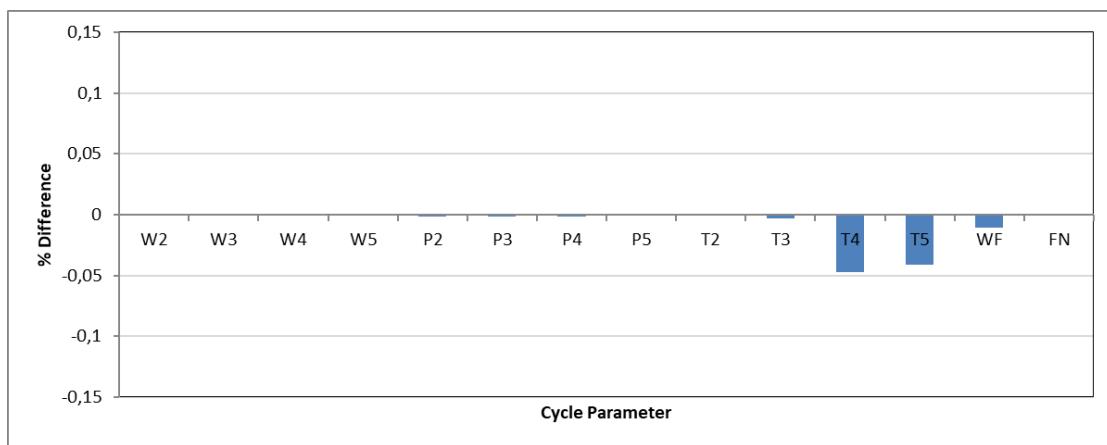


Figure 8.3 PROOSIS-NPSS turbojet cycle percentage differences at design point

Notice that differences are well within **0.05%**, whereas the maximum difference in design point is located in the burner total exit temperature, equal to 0.47%, as seen in [Figure 8.3](#). Small differences in the burner component can in turn affect slightly the performance of the downstream turbine.

Following, an off-design analysis was performed in each program, where an operating line of 18 points was simulated by varying the requested thrust (100÷15 % of the design thrust by 5% increments). [Figure 8.4](#) shows the maximum, minimum and average percentage differences between PROOSIS and NPSS. For all cycle parameters, the differences are within **0.05%**, as well. It is also important to mention that the maximum difference observed is 0.047% for the total exit temperature of the burner component.

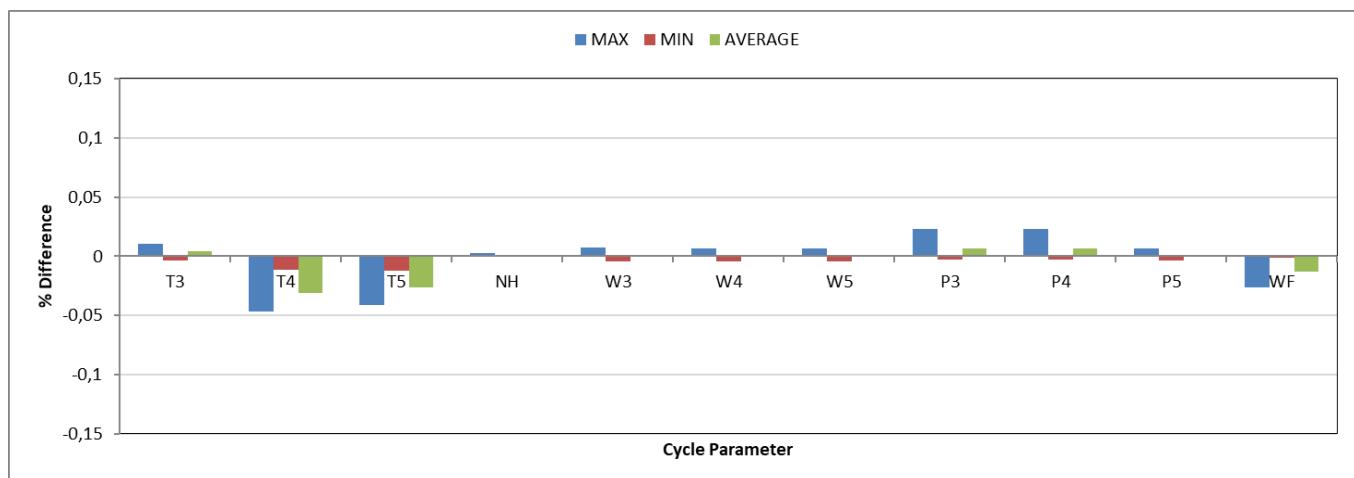


Figure 8.4 PROOSIS-NPSS turbojet cycle percentage differences at off-design

It is important to mention that the larger differences observed in the Turbojet model, compared to the ones found in component level comparisons, can be attributed to differences in the map parameters as mentioned in paragraph [5.3](#). Due to differences in the range of map parameters of each program, the resulting operating points that each program produces when the engine balance is completed differ, which in turn creates differences in the performance of each engine component. However, these differences are very small in all cases, lower than 0.05%.

[Figure 8.5](#) compares the performance of PROOSIS and NPSS models in terms of specific fuel consumption and mechanical speed against net thrust FN. The maximum difference is that of SFC at **0.02%**.

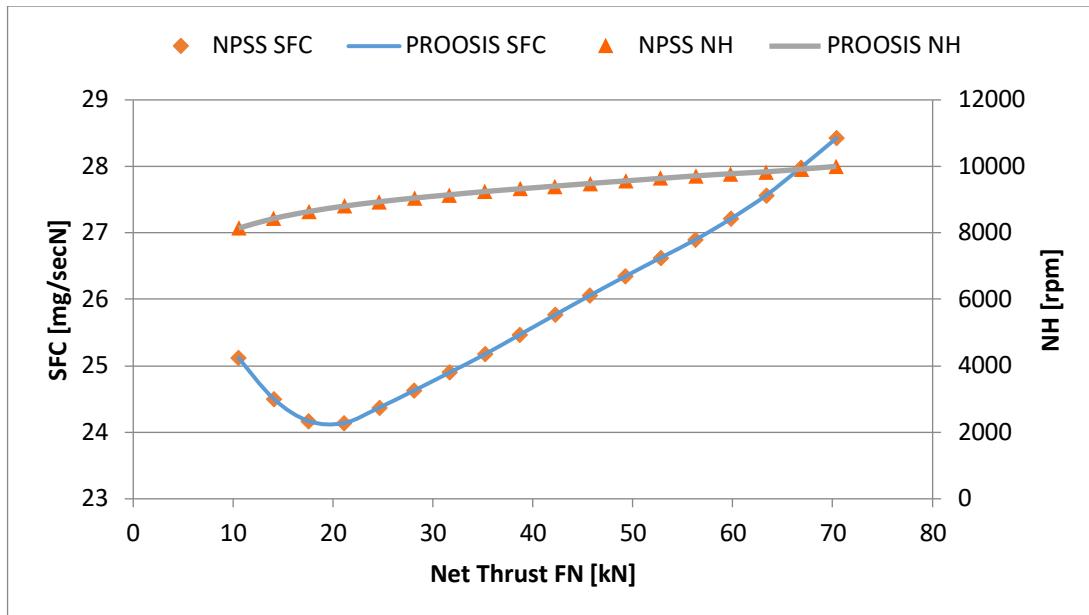


Figure 8.5 PROOSIS-NPSS turbojet main performance parameter comparison

8.2 Turbofan

The process to create a Turbofan model in both programs is similar to the one used in the Turbojet case, only with a few additions; however, it is fairly simple to do so.

8.2.1 PROOSIS

Having already completed models for individual components and for a Turbojet engine as well, users can easily create a new Turbofan engine model just by adding a Fan component and another Turbine. Note that there are a number of different modelling solutions and assemblies for a Turbofan engine, however in this case we will use a model based on the modeling philosophy of NPSS. We should always keep in mind that the purpose of this report is to compare not only the modeling philosophy of each program, but the simulation results as well, meaning that identical engine models must be used in each program.

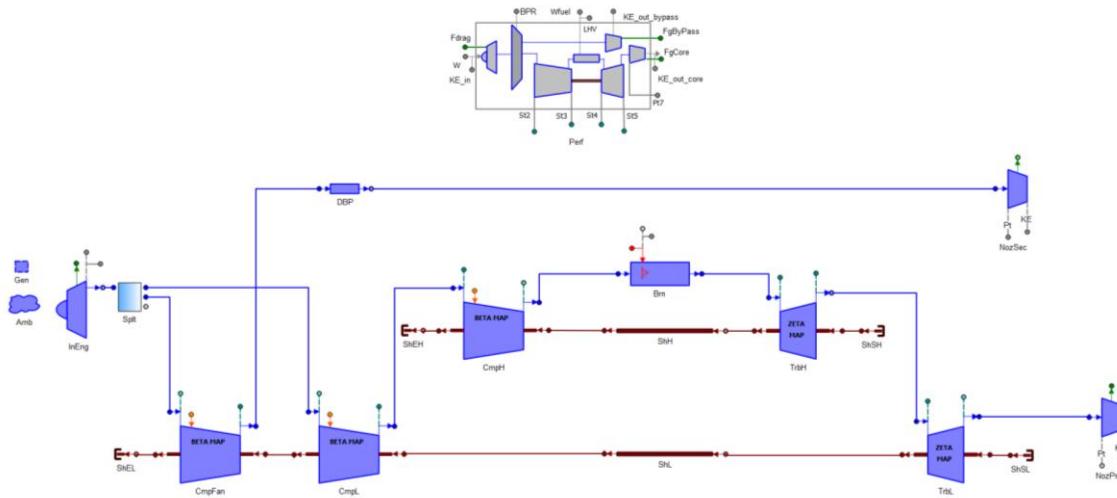


Figure 8.6 Turbofan schematic in PROOSIS environment

PROOSIS provides users with a number of different Fan components, in which the flow is divided into two streams and different pressure ratios on each stream can be applied. However, as previously mentioned there is no specific fan element in NPSS, only a compressor one where the map differentiates its performance. This imposes great differences in the modeling of a Turbofan engine between the programs, as the fan element in NPSS does not split the flow nor can apply different pressure ratios on each stream. The NPSS modeling philosophy requires a splitter element that divides the inlet stream, a fan element to apply the pressure ratio on the by-pass stream and finally a low pressure compressor/booster where the corresponding fan core pressure ratio is incorporated into. More about the NPSS model will be discussed in the following paragraph.

To create an identical model layout in PROOSIS at first a simple splitter component had to be created, as it was not included in the TURBO library. Additionally, a compressor component was used to model the fan section for the by-pass stream, similarly to NPSS, followed by the booster.

At first an experiment is generated and then a Design point is simulated to calculate the maps scalars, followed by the Off-Design points. Finally, results are exported on the monitor window of the program.

8.2.2 NPSS

The NPSS corresponding model for the Turbofan model presented above can be seen in the image below:

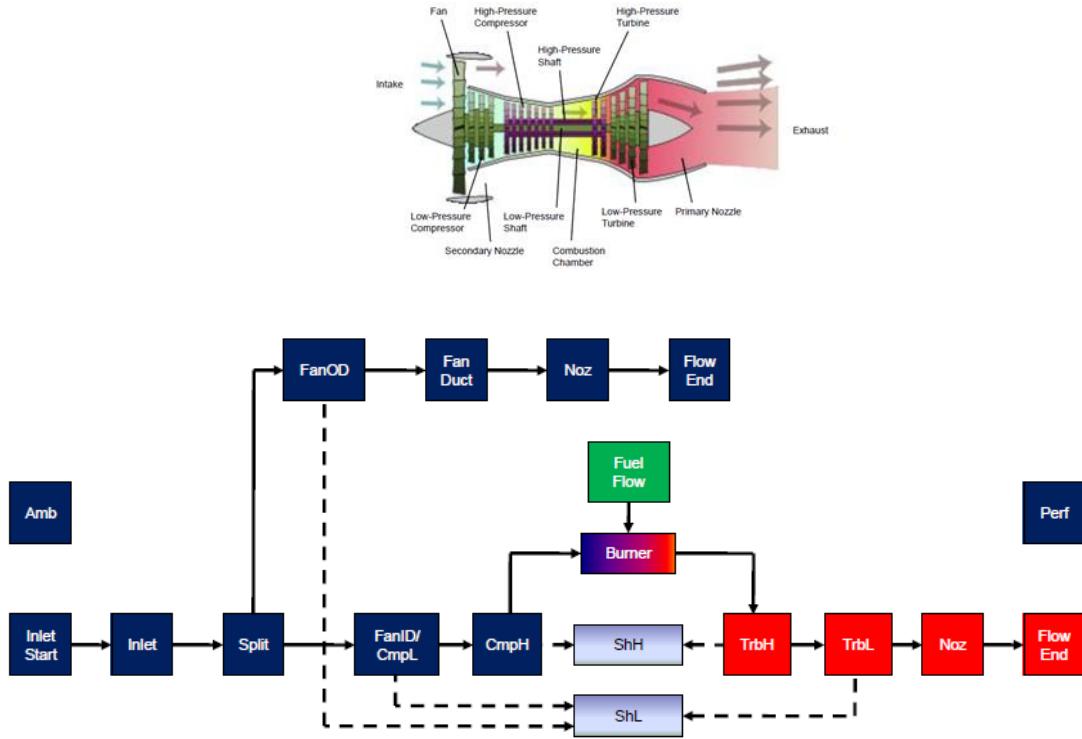


Figure 8.7 Turbofan schematic in NPSS

Having already presented the instantiation of each individual element and the building process of the Turbojet engine model, it is now very simple to create a Turbofan model. The only addition in this case is the implementation of the splitter element. However, it is considered important to present the Turbofan code, as a few different programming options are shown. The code of the Turbofan model can be seen below. Note that this model is based on the one created by P. Johnson for the NPSS Training Module [26].

```
setThermoPackage("allFuel");
FlowStation fs;
fs.switchFuelType = "JP";
#include "TurbofanViewDP.view";
```

```
Element Ambient Amb {
    switchMode = "ALDTMN";
    alt_in = 0.;
    MN_in = 0.;
    dTs_in = 0.;
} // END Amb
```

```
Element InletStart FsEng {
    AmbientName = "Amb";
    W_in = 317.465658;
} // End InletStart
```

```
Element Inlet InEng {
    PqP_in = 0.995;
} // END InEng
```

```
Element Splitter SpltFan {
    BPRdes = 5.;
} // END SpltFan
```

As mentioned in the beginning of the chapter, the splitter element splits the incoming flow into two streams, the core and the by-pass stream. Only the by-pass ratio is required for the instantiation of this element.

```
Element Compressor CmpFSec {
    #include "fanPRO.map";
    PRdes = 1.5;
    effDes = 0.85;
} // End CmpFSec
```

```
Element Duct DBP {
    switchDP = "INPUT";
    dPqP_in = 0.015;
} // END DBP
```

```
Element Compressor CmpL {
    #include "lpcE3.map";
    PRdes = 3.0;
    effDes = 0.85;
} // End CmpL
```

```
Element Compressor CmpH {
    #include "hpcE3.map";
    PRdes = 10.0;
    effDes = 0.85;
} // End CmpH
```

```
Element FuelStart FusEng {
    Wfuel = 2.0;
    LHV = (4.3124*10**7)*getUnitsFactor("J/kg", "Btu/lbm");
} // End FusEng
```

```
Element Burner BrnPri {
    dPqP_dmd      = 0.05;
    effBase       = 0.98;
    switchBurn   = "WFUEL";
} // End BrnPri
```

```
Element Turbine TrbH {
    #include "hptE3.map";
    effDes = 0.87;
    PRbase = 4.5;
} // End TrbH
```

```
Element Turbine TrbL {
    #include "lptE3.map";
    effDes = 0.87;
    PRbase = 4.5;
} // End TrbL
```

```
Element Nozzle NozPri {
    PsExhName = "Amb.Ps";
    switchType = "CONIC";
} //END NozPri
```

```
Element Nozzle NozSec {
    PsExhName = "Amb.Ps";
    switchType = "CONIC";
} //END NozSec
```

```
Element Shaft ShL {
    ShaftInputPort Sh_ICmpFSec;
    ShaftInputPort Sh_ICmp;
    ShaftInputPort Sh_ITrb;
    Nmech = 5000.;
} // End ShL
```

```

Element Shaft ShH {
    ShaftInputPort Sh_ICmp;
    ShaftInputPort Sh_ITrb;
    Nmech = 10000.;
} // End ShH

```

```

Element FlowEnd FePri { } // End FrPri

```

```

Element FlowEnd FeSec { } // End FeSec

```

```

Element EngPerf Perf { } // End Perf

```

```

Element Cycle Cycle {
    EPR_numName = "CmpH.Fl_O";
    EPR_denName = "InEng.Fl_O";
    FPR_numName = "CmpFSec.Fl_O";
    FPR_denName = "InEng.Fl_O";
} //END Cycle

```

A cycle element is used in order to easily obtain overall engine performance values, such as engine pressure ratio and fan pressure ratio.

```

// Primary Hot Section
linkPorts( "FsEng.Fl_O"      , "InEng.Fl_I"      , "F0" );
linkPorts( "InEng.Fl_O"      , "SpltFan.Fl_I"   , "F020" );
linkPorts( "SpltFan.Fl_O1"   , "CmpL.Fl_I"      , "F020A" );
linkPorts( "CmpL.Fl_O"       , "CmpH.Fl_I"      , "F025" );
linkPorts( "CmpH.Fl_O"       , "BrnPri.Fl_I"    , "F030" );
linkPorts( "BrnPri.Fl_O"     , "TrbH.Fl_I"      , "F040" );
linkPorts( "TrbH.Fl_O"       , "TrbL.Fl_I"      , "F045" );
linkPorts( "TrbL.Fl_O"       , "NozPri.Fl_I"    , "F050" );
linkPorts( "NozPri.Fl_O"     , "FePri.Fl_I"     , "F090" );

// Fan duct section
linkPorts( "SpltFan.Fl_O2"   , "CmpFSec.Fl_I"  , "F120" );
linkPorts( "CmpFSec.Fl_O"    , "DBP.Fl_I"       , "F130" );
linkPorts( "DBP.Fl_O"         , "NozSec.Fl_I"   , "F170" );
linkPorts( "NozSec.Fl_O"     , "FeSec.Fl_I"     , "F190" );

// Link Fuel Ports
linkPorts( "FusEng.Fu_O"     , "BrnPri.Fu_I"    , "Fu_In" );

```

```
// Link Shaft Ports
linkPorts( "CmpFSec.Sh_O" , "ShL.Sh_ICmpFSec" , "MeCmpFSec" );
linkPorts( "CmpL.Sh_O" , "ShL.Sh_ICmp" , "MeCmpL" );
linkPorts( "TrbL.Sh_O" , "ShL.Sh_ITrb" , "MeTrbL" );

linkPorts( "CmpH.Sh_O" , "ShH.Sh_ICmp" , "MeCmpH" );
linkPorts( "TrbH.Sh_O" , "ShH.Sh_ITrb" , "MeTrbH" );
```

Since the splitter produces two separate flows, it has two output fluid ports to be linked accordingly. One port will be connected to the inlet of the oncoming core element whereas the other port will be connected to the by-pass stream. The output ports can be seen above under the name *FL_O1* and *FL_O2*.

At this point, new solver variables will be created, as well as custom dependents, independents and constraints.

```
real Fn_req = 10000;
```

```
real NmechL_max = 5200;
real NmechH_max = 11000;
```

```
Independent ind_Wfuel{
    varName = "FusEng.Wfuel";
}
```

```
Dependent dep_Fn {
    eq_lhs = "Perf.Fn";
    eq_rhs = "Fn_req";
}
```

```
// Solver Constraints
Dependent dep_NmechL_max {
    eq_lhs = "ShL.Nmech";
    eq_rhs = "NmechL_max";
}
```

```
Dependent dep_NmechH_max {
    eq_lhs = "ShH.Nmech";
    eq_rhs = "NmechH_max";
}
```

```
dep_Fn.addConstraint( "dep_NmechL_max", "MAX");
dep_Fn.addConstraint( "dep_NmechH_max", "MAX");
```

Note that the solver constraints are created as dependents, as mentioned in previous chapters.

The next step is to set-up the design run. We can follow the same process described in the Turbojet model; however, it would be beneficial to present another programming option which is highly preferable when using multiple files.

```
// Setup the Design Run
void setupDesign(){

    setOption("switchDes", "DESIGN");
    setOption( "solutionMode", "STEADY_STATE" );

    solver.debugLevel = "ITERATION_DETAILS";
    solver.diagnosticFile = "solverDes.out";

    autoSolverSetup();

    solver.addIndependent("ind_Wfuel");
    solver.addDependent("dep_Fn");

    dep_Fn.useConstraints = FALSE; // Turn off the constraints during design mode
}
```

The `void` command creates a function that can be called at any part of the code the user desires. It is common to set the design and off-design run using a void command in a different file than the model, which can be later used when running the code. This serves greatly for organization purposes and to simplify the code. For example, one can use a file to store the engine model code, another file to store custom functions and inputs and finally another file where the execution of the code is done. Note that in the last file a command to include all other files must be issued.

```
setupDesign(); //execute the setupDesign function
run();
cout << "Solver converged (1 = yes, 0 = no) ? = " << solver.converged << endl;
cout << "Iterations = " << solver.iterationCounter << endl;
rowSheet.update();
rowSheet.display();
```

```

void setupOffDesign(){
    setOption("switchDes", "OFFDESIGN");

    solver.debugLevel = "ITERATION_DETAILS";
    solver.diagnosticFile = "solverDes.out";

    autoSolverSetup();

    solver.addIndependent("ind_Wfuel");
    solver.addDependent("dep_Fn");

    dep_Fn.useConstraints = TRUE;
}

Fn_req = 9000;
run();
rowSheet.update();
rowSheet.display();

```

At this point multiple off-design points can be simulated by varying the requested thrust.

Other Turbofan modeling options

As previously mentioned there are many different engine architectures depending on the application and the manufacturer. The architecture most commonly used can be seen in the image below, which uses two compressors and two turbines. The splitter in this case is placed after the fan element which ensures that the fan pressure ratio will be applied to both the core and by-pass stream.

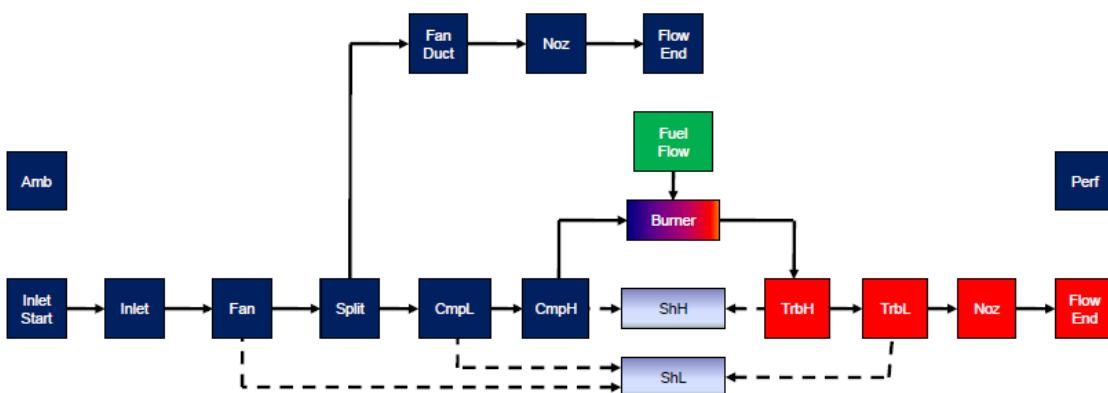


Figure 8.8 NPSS Turbofan modeling option 1, [26]

Another modeling option can be seen in the following image. In this case the splitter element is also placed after the fan element and is the simplest engine model, compared to others presented here.

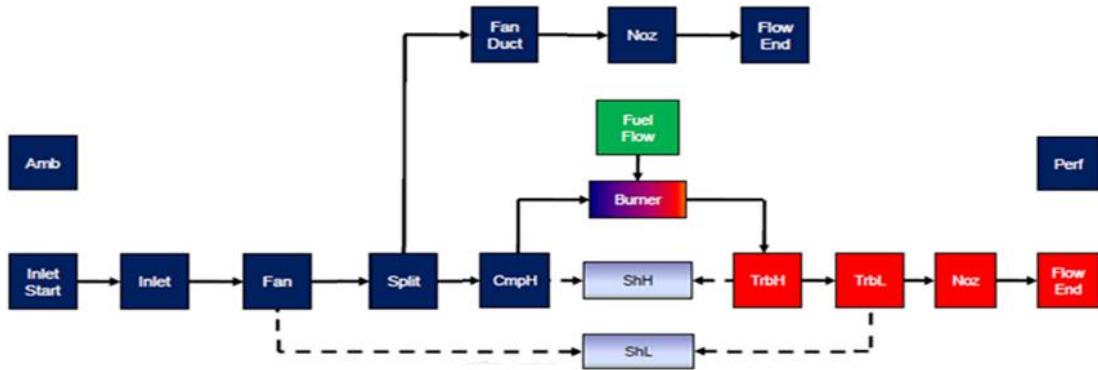


Figure 8.9 NPSS Turbofan modeling option 2, [26]

Finally, the last modeling option in NPSS can be seen in the figure below. In this case different pressure ratios in the core and by-pass stream can be applied using two fan elements, one for the core and one for the by-pass.

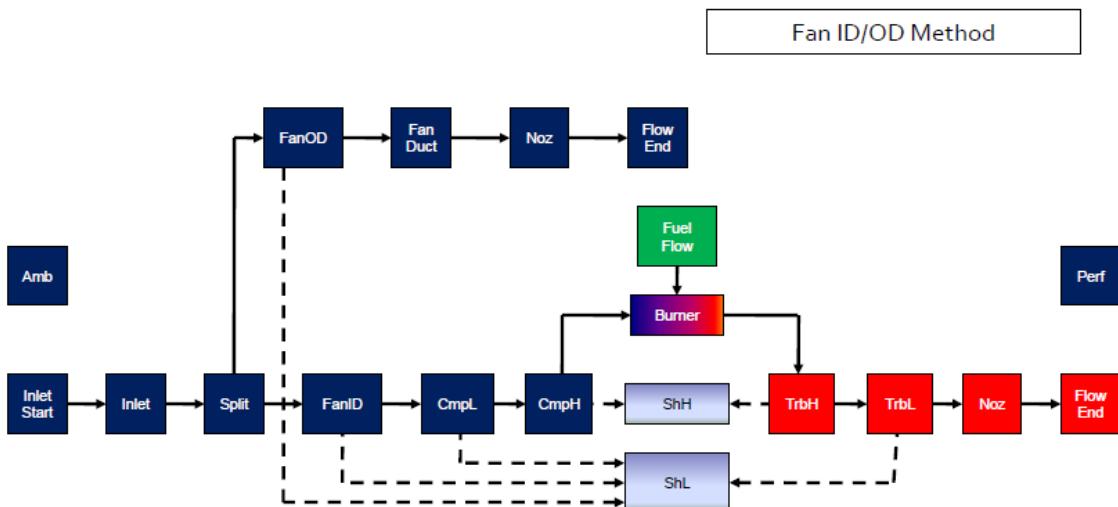


Figure 8.10 NPSS Turbofan modeling option 3, [26]

8.2.3 Results Comparison

As in the turbojet case, similarly here a design point calculation was set to calculate map scalars. The same input data were set in both programs and are presented in the following table:

Table 8.2 Turbofan design point input data

Flight Altitude [m]	0
Mach Number	0
Inlet mass flow rate [kg/s]	144
Inlet PqP	0.995
Fan pressure ratio	1.5
Fan efficiency	0.85
By pass ratio	5
By pass duct dPqP	0.015
LP Compressor pressure ratio	3
LP Compressor efficiency	0.85
HP Compressor pressure ratio	10
HP Compressor efficiency	0.85
Burner efficiency	0.98
Burner dPqP	0.05
HP Turbine efficiency	0.87
LP Turbine efficiency	0.87
HP shaft mechanical speed [rpm]	10000
LP shaft mechanical speed [rpm]	5000
Requested Thrust [kN]	44.482
Fan BETA parameter	0.473684
HPC BETA parameter	0.33
LPC BETA parameter	0.33
HPT ZETA parameter	0.74
LPT ZETA parameter	0.59
Fan R-line parameter	1.789474
HPC R-line parameter	2.0
LPC R-line parameter	2.0
HPT PR-line parameter	4.96
LPT PR-line parameter	4.36

The percentage differences for various cycle parameters of the simulation are presented in the following figure.

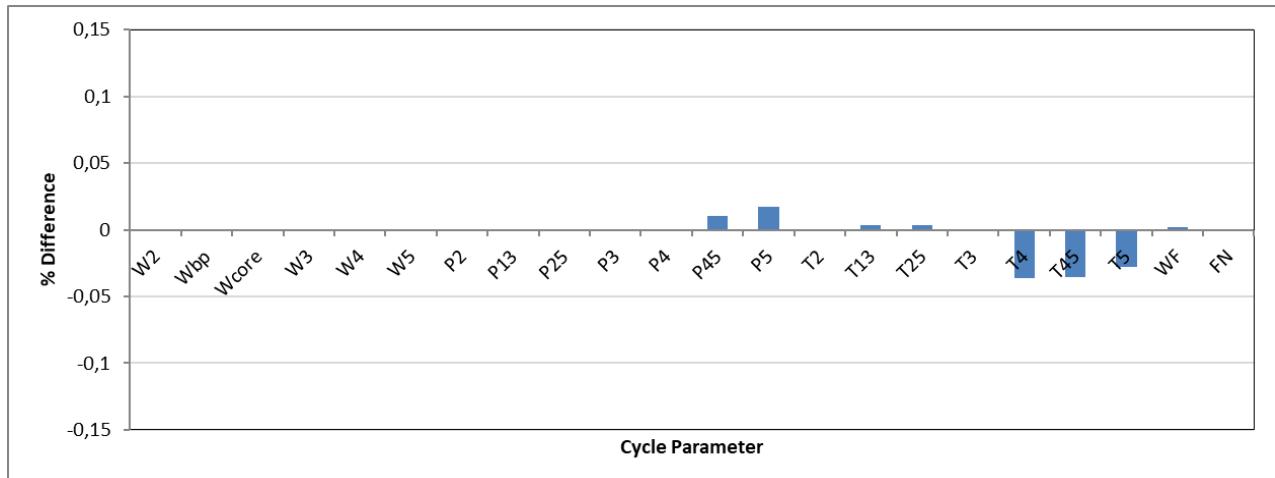


Figure 8.11 PROOSIS-NPSS turbofan cycle percentage differences at design point

Notice that at design point minor differences are similarly observed as in the case of the Turbojet engine, whereas the maximum difference is **0.035%** for the burner total exit temperature.

A parametric case is then set varying the desired net thrust (100÷21 % of design point desired thrust), producing an operating line of 21 points. [Figure 8.12](#) shows the maximum, minimum and average percentage differences between PROOSIS and NPSS. For all cycle parameters, the differences are within **0.05%**, whereas the maximum difference is similarly located in the burner total exit temperature, equal to 0.041%.

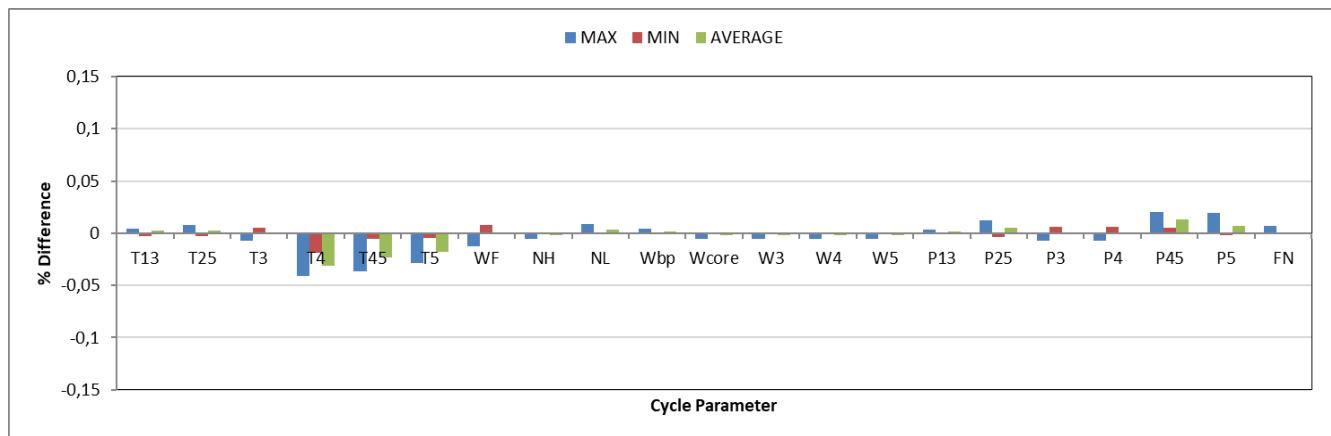


Figure 8.12 PROOSIS-NPSS turbofan cycle percentage differences at off-design

As in the Turbojet case, similar differences are observed here, which can be attributed to differences in the map parameters as previously mentioned.

Figure 8.13 compares the performance of PROOSIS and NPSS models in terms of specific fuel consumption against net thrust FN. The maximum SFC percentage difference is **0.01%** at the last off-design point.

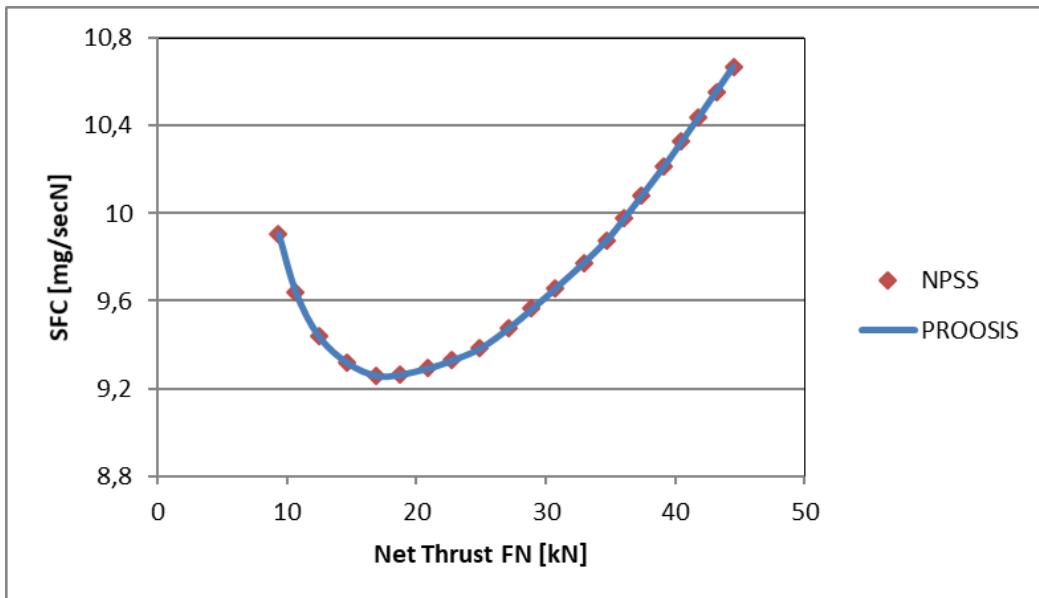


Figure 8.13 PROOSIS-NPSS turbofan main performance parameter comparison

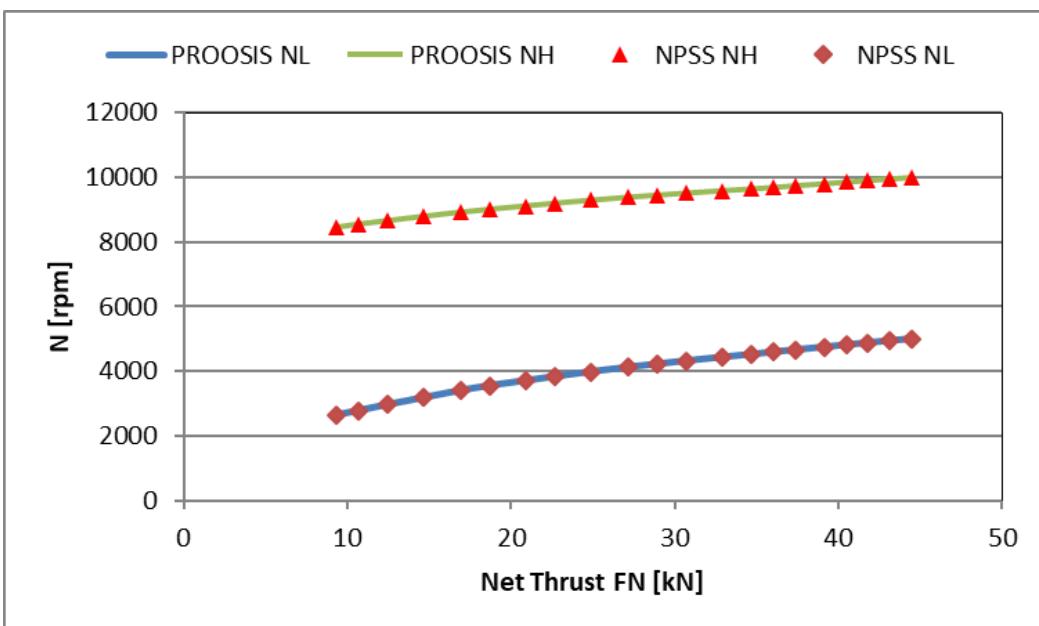


Figure 8.14 PROOSIS-NPSS turbofan mechanical speed comparison

Figure 8.14 compares the performance of PROOSIS and NPSS models in terms of mechanical speed against net thrust FN. In both occasions, differences are below 0.01%.

8.3 Turboshaft

8.3.1 PROOSIS

In this case, a complete Turboshaft engine model was used from the TURBO_VALIDATION library. An experiment is generated and then a Design point is simulated to scale the maps, followed by the Off-Design points through a *Parametric* calculation. Finally, results are exported on the monitor window of the program as in the previous models.

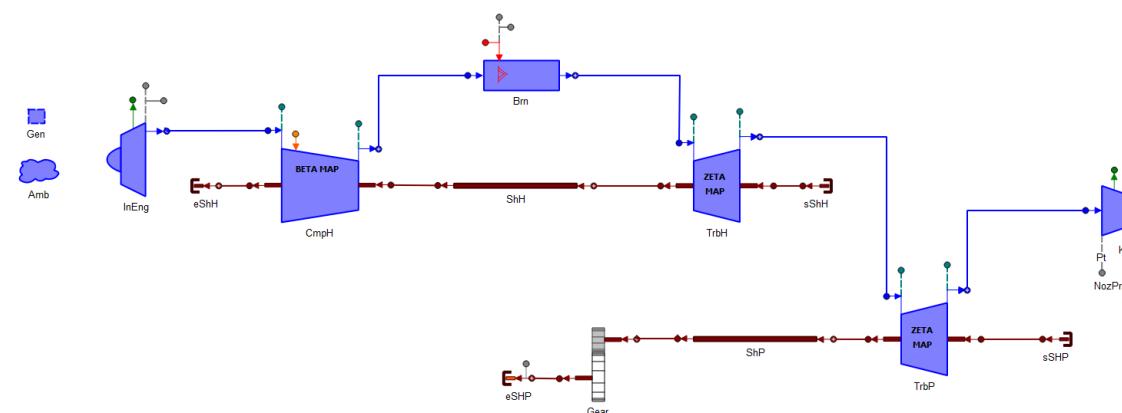


Figure 8.15 Turboshaft schematic in PROOSIS environment

8.3.2 NPSS

An identical model to the one described above was also created in NPSS. The building philosophy is very similar to the other engine models; however, in this case it is required to add a few calculations in the code, as there is no element in the included library to simulate the performance of this type of engine.

The first difference to the engine models previously presented is that in this case there is a free power turbine and shaft. The instantiation of the shaft element is different, as it requires the power extraction from the shaft to be given, as seen below:

```
Element Shaft ShFP {
    ShaftInputPort MeTrbFP;
    HPX = 2000; //value in horsepower
    Nmech = 20000.;

}
```

The major difference in the code of the turboshaft model is the modified *EngPerf* element. As mentioned earlier, users can modify the source code or even create their own elements. However, it is also possible to modify an element inside a model file as presented below. Users can easily create their own variables and add the necessary equations. Note that the command *void postexecute()* is a member function that allows modification of an element in the specific model, diminishing the need to access the source code. This member function command will allow users to override variable values, add custom calculations and compute extra attributes not calculated by the standard library element, after the default calculations have been completed. This modification is used only in the specific model and is not applied to all similar elements found in other models.

```
Element EngPerf Perf {

    // Additional inputs
    real gearRatio {
        value = 0.3; IOstatus = INPUT; units = RPM;
        description = "Gearbox ratio";
    }

    // Additional outputs
    real N1 {
        value = 0.; IOstatus = OUTPUT; units = RPM;
        description = "High pressure shaft speed";
    }
    real N2 {
        value = 0.; IOstatus = OUTPUT; units = RPM;
        description = "Free power turbine shaft speed";
    }
    real ND {
        value = 0.; IOstatus = OUTPUT; units = RPM;
        description = "Output Speed";
    }
    real Q {
        value = 0.; IOstatus = OUTPUT; units = FT_LBF;
        description = "Free power turbine actual shaft torque";
    }
    real SHP {
        value = 0.; IOstatus = OUTPUT; units = HORSEPOWER;
        description = "Free power turbine actual shaft power";
    }
    real BSFC {
```

```

value = 0.; IOstatus = OUTPUT; units = LBM_PER_HR_HP;
description = "Brake specific fuel consumption";
}

real effth {
    value = 0.; IOstatus = OUTPUT; units = NONE;
    description = "Thermal efficiency";
}

// Additional calculations
void postexecute() {

    effth = ShFP.pwrIn*getUnitsFactor("hp", "W")/
(BrnPri.Wfuel*getUnitsFactor("lbm/sec", "kg/sec")*4.3124*10**7);

    // Machine speeds
    N1 = ShH.Nmech;
    N2 = ShFP.Nmech;
    ND = N2 * gearRatio;

    // Torque
    Q = ShFP.trqIn;

    // Power
    SHP = ShFP.pwrIn;

    // Specific fuel consumption
    BSFC = (BrnPri.Wfuel / SHP) * C_HOURtoSEC;
}

} //END Perf

```

The port linking commands are similar to the ones from previous models and have already been presented. Hence, this part of the code does not need to be presented. Note that since there is not a gear element in the library and is user created within another element in the model code, it does not have any ports to be connected.

8.3.3 Results Comparison

First a design point followed by several off-design points was calculated. Design point input data can be seen in the following table:

Table 8.3 Turbofan design point input data

Altitude [m]	0
Mach Number	0
Inlet mass flow [kg/s]	5
Inlet PqP	0.995
Compressor pressure ratio	15
Compressor efficiency	0.858
Burner FAR	0.0192
Burner efficiency	0.995
Burner dPqP	0.025
HP Turbine efficiency	0.913
Free power Turbine efficiency	0.915
HP shaft mechanical speed [rpm]	40000
FP shaft mechanical speed [rpm]	20000
Power derived from the FP shaft [kW]	1619.386
Gear ratio	0.3
Gear efficiency	1
HPC BETA parameter	0.33
HPT ZETA parameter	0.74
FPT ZETA parameter	0.59
HPC R-line parameter	2.0
HPT PR-line parameter	4.96
FPT PR-line parameter	4.36

The percentage differences for various cycle parameters of the design point simulation are presented below, where differences are well within **0.01%**.

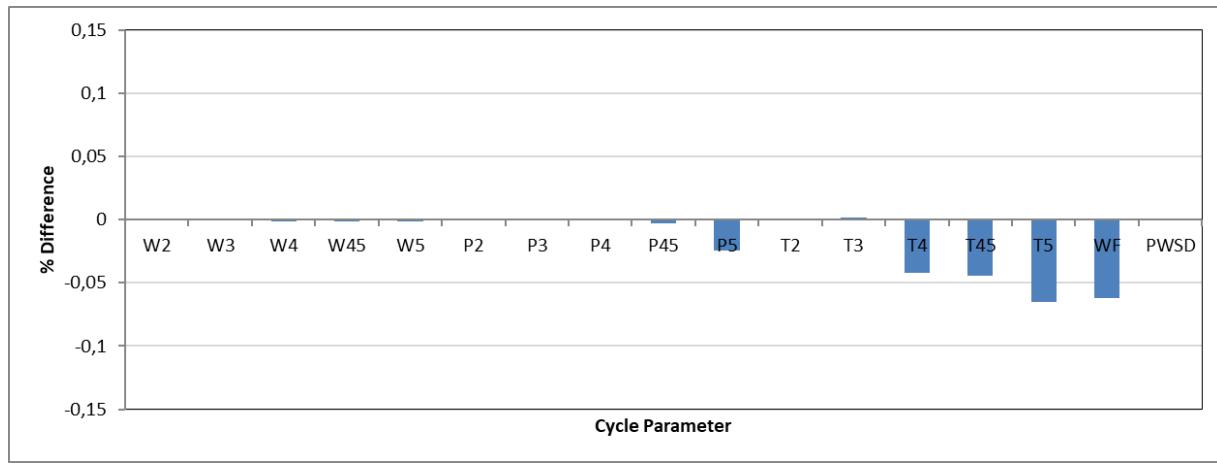


Figure 8.16 PROOSIS-NPSS turboshaft cycle percentage differences at design point

A parametric case is then simulated varying the power extracted from the free power shaft, producing an operating line of 31 points ($1619 \div 450$ kW). Figure 8.17 shows the maximum, minimum and average percentage differences between PROOSIS and NPSS. For all cycle parameters, the differences are within **0.01%**. It is also important to mention that the maximum difference observed is that of the free power turbine total exit temperature at **0.065%**.

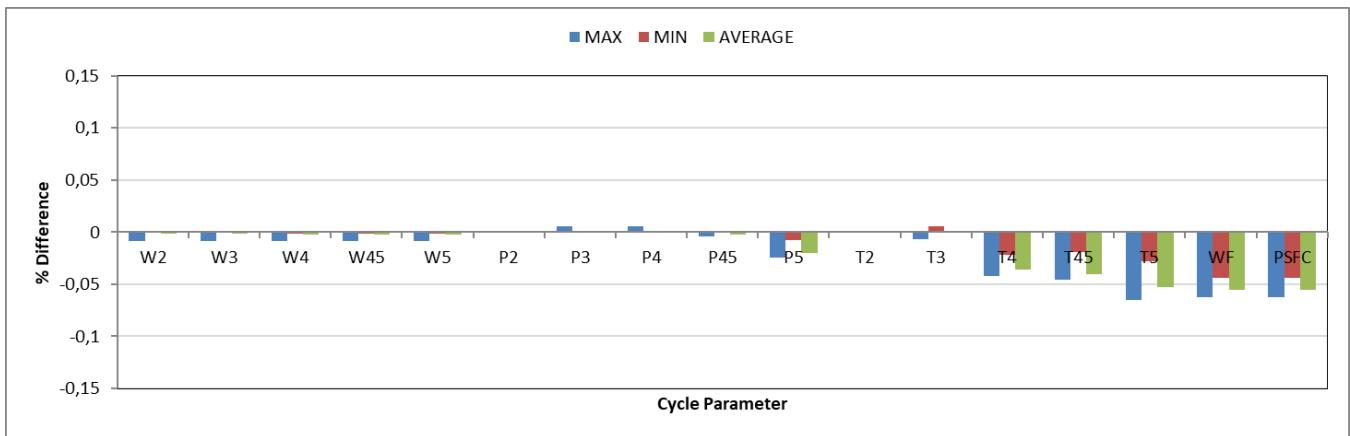


Figure 8.17 PROOSIS-NPSS turboshaft engine cycle percentage differences at off-design

Figure 8.18 compares the performance of PROOSIS and NPSS models in terms of specific fuel consumption SFC and thermal efficiency Eth against shaft power delivered PWSD. The maximum SFC percentage difference is **0.06%** and the maximum thermal efficiency difference is **0.06%** as well.

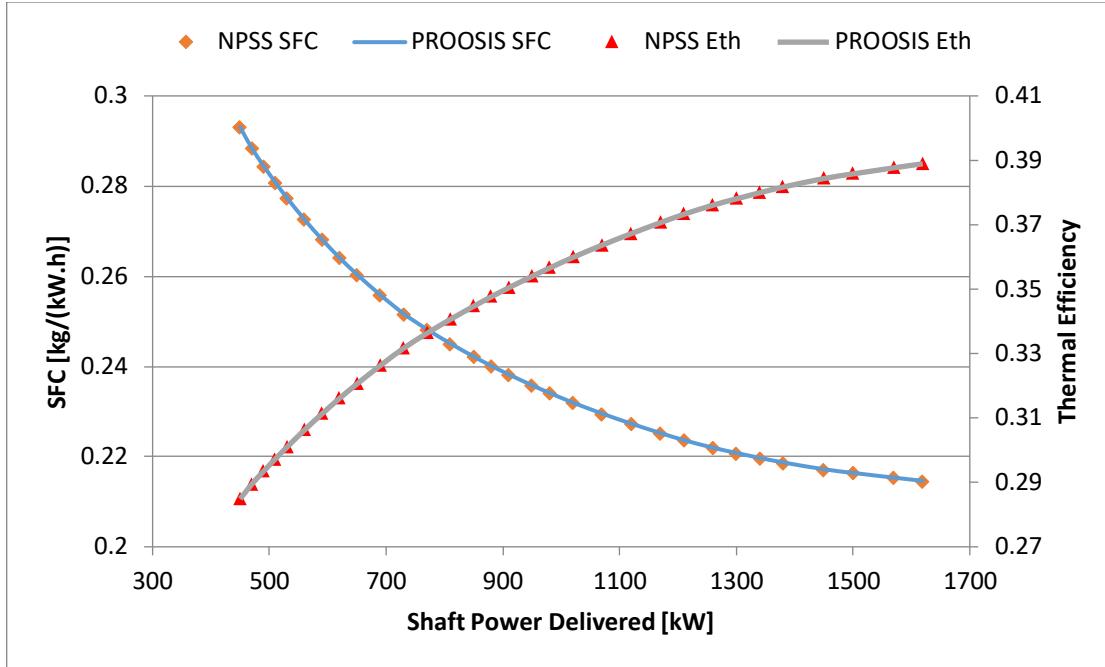


Figure 8.18 PROOSIS-NPSS turboshaft main performance parameters comparison

As in previous cases, similarly here, only small differences are observed and can be attributed to differences in the map parameters as previously mentioned. Differences in the range of the map parameters can produce different parameter values during calculations, which in turn place the operating points on slightly different positions, thus producing the above differences, however they are very small (<0.1%).

9

SUMMARY AND FUTURE WORK

PROOSIS and NPSS are very powerful commercial software, offering a wide area of applications and are continuously expanding their abilities to meet the industrial and customer needs. Through the work conducted for the purpose of this diploma thesis, we learned how to operate NPSS. Then, we created not only component, but complete engine models as well and compared the simulation results against the corresponding PROOSIS models. From this comparison we can see that the modeling philosophy of the two software is similar and that differences are below 0.1% in all cases, which leads us to the conclusion that as far as thermodynamic calculations are concerned, the two programs are identical. Minor differences produced mainly at engine level calculations can be attributed to differences in the map parameters. As previously mentioned, due to the different range of the map parameters between the programs, the resulting parameter values during engine simulations also differ slightly. This means that when each program balances its engine cycle, the corresponding map parameters differ which in turn places the operating points on a slightly different position on the map. This in turn creates small differences in the performance of the engine, however they are very small. Another possible source of differences are the computational methods. More specifically, different interpolation methods for the fluid model file, can in turn create small differences as far as the burner performance is concerned, adding up to the differences produced from the map parameters. Finally, minor differences could also be attributed to different solver tolerance between the programs.

As mentioned in the introduction, the purpose of this report is to serve as an initial step to a possible future interface between NPSS and PROOSIS. Connecting these two programs is a high industry demand, as it will facilitate the communication between engine manufacturers, especially in cases where different companies, that operate different software, develop different parts of the same engine. Having familiarized with the NPSS modeling philosophy and its programming environment and having learned how to program NPSS, the next step would be to develop an interface between the programs. One possible idea, would be to interface the two software at source code level. This means that the code of one program would be converted in such a way that it can be processed by the

other. For example, PROOSIS is capable of exporting the source code not only for a component, but for a complete engine model as well in C++ programming language, the one that NPSS is based on. Another possible idea, would be to interface the programs through deck generation. Decks are compiled and standalone executable programs that can be used by external programs (such as Excel) as a black box, only by providing specific inputs. Both PROOSIS and NPSS are capable of deck generation. This means that for example, an NPSS deck for an element or even a complete engine model could be created, which would be later called and executed as a black box in PROOSIS and vice versa.

Finally, since both software are thermodynamically equal and the only major difference between them are the performance maps, it would also be possible to simply modify the current PROOSIS components such that they can read and operate with NPSS maps (i.e. R-line and PR-line maps).

10

**ΕΚΤΕΤΑΜΕΝΗ
(ΕΛΛΗΝΙΚΑ)**

ΠΕΡΙΛΗΨΗ

Η συνεχής εξέλιξη της αεροδιαστηματικής βιομηχανίας αναγκάζει τους επιστήμονες να εξοπλίζονται με τα κατάλληλα εργαλεία ώστε να συνεχίσουν να πρωτοπορούν με καινοτόμες λύσεις στον τομέα τους. Το PROOSIS και το NPSS είναι δύο λογισμικά προσομοίωσης με μεγάλο εύρος δυνατοτήτων και εφαρμογών, ιδιαίτερως στον τομέα των θερμικών στροβιλομηχανών, και είναι τα κυρίαρχα λογισμικά που χρησιμοποιούνται σήμερα από μεγάλες εταιρείες παραγωγής αεροπορικών κινητήρων. Το NPSS είναι ένα λογισμικό που δημιουργήθηκε από την NASA και χρησιμοποιείται κυρίως στις Ηνωμένες Πολιτείες. Αντιθέτως, το PROOSIS δημιουργήθηκε και χρησιμοποιείται κατά κύριο λόγο στην Ευρώπη. Πολλές κατασκευάστριες εταιρείες χρησιμοποιούν ένα από τα δύο προγράμματα, ωστόσο υπάρχουν περιπτώσεις όπου οι εταιρείες μέλη και των δύο κοινοπραξιών χρησιμοποιούν και τα δύο λογισμικά, κάτι το οποίο δημιουργεί δυσκολίες ιδιαίτερα σε συνεργατικά έργα όπου ο κάθε κατασκευαστής συμβάλλει σε διαφορετικό μέρος του κινητήρα. Επομένως, χρήζει ιδιαίτερου ενδιαφέροντος μία πιθανή διασύνδεση αυτών των δύο προγραμμάτων, βελτιώνοντας συγχρόνως την εξέλιξη των αεροπορικών κινητήρων. Η παρούσα διπλωματική εργασία αποτελεί το πρώτο βήμα στη φιλοσοφία διασύνδεσης των λογισμικών, παρουσιάζοντας μια αναλυτική σύγκριση μεταξύ των δύο αυτών προγραμμάτων, δίνοντας ωστόσο μεγαλύτερη έμφαση στο NPSS, καθώς το PROOSIS χρησιμοποιείται ήδη εδώ και χρόνια ως το βασικό λογισμικό στο Εργαστήριο Θερμικών Στροβιλομηχανών του Εθνικού Μετσόβιου Πολυτεχνείου. Αρχικά θα γίνει μία παρουσίαση των δύο λογισμικών. Έπειτα, θα παρουσιαστεί αναλυτικά ο τρόπος δημιουργίας μοντέλων τόσο για τις διάφορες συνιστώσες μιας στροβιλομηχανής, όσο και για ολόκληρους κινητήρες, έτσι ώστε να γίνει περισσότερο κατανοητή η φιλοσοφία μοντελοποίησης αλλά και να εντοπιστούν τυχόν διαφορές σε επίπεδο προγραμματισμού μεταξύ των δύο. Στο τέλος, θα παρουσιαστούν αναλυτικά τα αποτελέσματα των προσομοιώσεων και θα γίνει σύγκριση μεταξύ αυτών. Ακόμα, αξίζει να αναφερθεί ότι για την σύγκριση των λογισμικών οι εκδόσεις που χρησιμοποιήθηκαν είναι PROOSIS 3.10.0 και NPSS 2.8 student version.

10.1 Εισαγωγή

Η εφαρμογή νέων τεχνολογιών στην αεροδιαστημική βιομηχανία γίνεται ολοένα και πιο δαπανηρή, καθώς απαιτεί δοκιμή, αξιολόγηση και πλήρη πιστοποίηση κινητήρων πλήρους μεγέθους, έτσι ώστε μελετώνται πλήρως οι αλληλεπιδράσεις μεταξύ των διάφορων συνιστωσών.

Τα λογισμικά προσομοίωσης προσφέρουν στους μηχανικούς την δυνατότητα να δημιουργούν ποικίλα μοντέλα συστημάτων και συνιστωσών, ιδιαίτερα των αεριοστροβίλων, και να αξιολογούν νέες ιδέες στις αρχικές φάσεις του σχεδιασμού.

Έχουν επίσης επιτρέψει την αξιολόγηση λειτουργικών προβλημάτων μέσω υπολογιστικών προσομοιώσεων, ειδικά σε περιπτώσεις προβλημάτων που αντιμετωπίζονται σε συνθήκες οι οποίες θα ήταν δύσκολο να προσομοιωθούν πειραματικά. Τα προγράμματα αυτά επέτρεψαν σημαντικά την μείωση του κόστους ανάπτυξης και αξιολόγησης που κάποτε απαιτούνταν, λόγω των επαναλαμβανόμενων δοκιμών και επανασχεδιασμών προϊόντων μεγάλης κλίμακας.

Η φρενήρης πρόοδος που σημειώνεται τα τελευταία χρόνια στην υπολογιστική μηχανική, σε συνδυασμό με την ταχεία αύξηση της υπολογιστικής ισχύος και της παράλληλης επεξεργασίας, έχει καταστήσει τα μοντέλα αυτά ιδιαιτέρως ισχυρά και έχει εμπλουτίσει τις δυνατότητές τους και την πιστότητά τους στη συσχέτιση προσομοιωμένων και πραγματικών συνθηκών λειτουργίας.

Τα κυρίαρχα εμπορικά λογισμικά προσομοίωσης κινητήρων αεριοστροβίλων που χρησιμοποιούνται επί του παρόντος από μεγάλους κατασκευαστές κινητήρων είναι το PROOSIS και το NPSS.

Το PROOSIS χρησιμοποιείται κυρίως στην Ευρώπη είναι το βασικό εργαλείο του Εργαστηρίου Θερμικών Στροβιλομηχανών, ενώ το NPSS, που αναπτύχθηκε από τη NASA, χρησιμοποιείται στις Η.Π.Α. καθώς και σε άλλες χώρες. Λαμβάνοντας ωστόσο υπόψη ότι το NPSS αποτελεί ένα νέο λογισμικό για το εργαστήριο, η έκθεση αυτή θα δώσει έμφαση στις δυνατότητες και τη φιλοσοφία μοντελοποίησης του NPSS.

Το NPSS έχει χρησιμοποιηθεί σε μεγάλο αριθμό έργων είτε ως το κύριο εργαλείο είτε όχι. για παράδειγμα έχει χρησιμοποιηθεί για τις προσομοιώσεις του επονομαζόμενου Nuclear Thermal Rocket (NTR), η οποία είναι μία τεχνολογία που αφορά επανδρωμένες αποστολές στον Άρη. Επίσης, έχει χρησιμοποιηθεί για διάφορα μοντέλα διαστημικής πρόωσης, για προσομοίωση αναρρόφησης του οριακού στρώματος και βελτιστοποίηση του σχήματος εισόδου σε κινητήρα ενός υβριδικού αεροσκάφους N2B. Επιπλέον, έχει βρει εφαρμογή ως επιλύτης σε θάλαμο καύσης κινητήρα scramjet, όπου η ροή στο θάλαμο καύσης είναι υπερηχητική και τέλος για την ανάπτυξη μοντέλων κινητήρων ανοικτού ρότορα.

Τα τελευταία χρόνια διεξήχθησαν εκτενείς μελέτες για την ανάπτυξη υβριδικών αλλά και πλήρως ηλεκτρικών συστημάτων πρόωσης με την χρήση του NPSS. Νέες τεχνολογίες, όπως τα συστήματα πρόωσης για αεροσκάφη κάθετης ανύψωσης έχουν

αναπτυχθεί ομοίως με το NPSS. Τέλος, ακολουθώντας την τάση για μετάβαση σε πλήρως ηλεκτρικά συστήματα, έχουν αναπτυχθεί πολλές ηλεκτρικές συνιστώσες, ενώ αντίστοιχα μοντέλα μπαταριών θα αναπτυχθούν συντόμως.

Ωστόσο, πρέπει να αναφερθεί ότι η πλειονότητα των εταιρειών χρησιμοποιούν μόνο το ένα ή ακόμα και τα δύο προγράμματα σε περιπτώσεις που είναι μέλη και στις δύο κοινοπραξίες. Αυτό είναι ιδιαιτέρως σημαντικό, ειδικά σε έργα όπου διαφορετικοί κατασκευαστές συνεισφέρουν σε διαφορετικά μέρη του ίδιου κινητήρα, καθώς αυτή τη στιγμή δεν είναι δυνατόν το NPSS να επεξεργάζεται αρχεία του PROOSIS και αντίστροφα.

Αυτή η εργασία επιχειρεί να εντοπίσει και να αξιολογήσει τις διαφορές μεταξύ των δύο προγραμμάτων ως προς τους υπολογισμούς και τα θερμοδυναμικά πακέτα, καθώς επίσης και τις διαφορές στη δημιουργία και την προσομοίωση συγκεκριμένων μοντέλων αεροπορικών κινητήρων.



Εικόνα 10.1 Εφαρμογές του NPSS

10.2 Περιγραφή Λογισμικών

10.2.1 PROOSIS

Το PROOSIS (Propulsion Object Oriented Simulation Software) είναι ένα αυτόνομο και ευέλικτο αντικειμενοστραφές περιβάλλον προσομοίωσης για την μοντελοποίηση αεροπορικών κινητήρων και άλλων συστημάτων, όπως για παράδειγμα ελέγχου, ηλεκτρικά, θερμικά, υδραυλικά και μηχανικά συστήματα. Αρχικά αναπτύχθηκε στο πλαίσιο του ευρωπαϊκού προγράμματος VIVACE (Value Improvement through a Virtual Aeronautical Collaborative Enterprise) από μία κοινοπραξία ευρωπαϊκών πανεπιστημάτων, ερευνητικών ιδρυμάτων και ιδιωτικών εταιρειών. Είναι βασισμένο στο EcosimPro, ένα εργαλείο προσομοίωσης που αναπτύχθηκε από την Empresarios Agrupados Internacional S.A., το οποίο είναι και το βασικό πρόγραμμα που χρησιμοποιεί η Ευρωπαϊκή Διαστημική Υπηρεσία για την ανάπτυξη πυραυλικών συστημάτων, περιβαλλοντικό έλεγχο και μελέτη συστημάτων υποστήριξης ζωής. Το PROOSIS διαθέτει όλες τις δυνατότητες του EcosimPro, καθώς επίσης και ορισμένες πρόσθετες δυνατότητες, οι οποίες είναι απαραίτητες για την προσομοίωση αεροπορικών κινητήρων, όπως για παράδειγμα ο χειρισμός χαρτών και ο σχεδιασμός με περιορισμούς. Επιπλέον, διαθέτει τη βιβλιοθήκη TURBO, η οποία περιλαμβάνει όλες τις τυπικές συνιστώσες για μοντελοποίηση οποιουδήποτε τύπου αεριοστροβίλου.

Το εν λόγω πρόγραμμα είναι ικανό για steady state και transient προσομοιώσεις και μπορούν να πραγματοποιηθούν ποικίλοι τύποι υπολογισμών, όπως design, off-design, single ή multi-point, αναλύσεις δοκιμών, ευαισθησίας, παραμετρικές, καθώς επίσης και μελέτες βελτιστοποίησης, ανάλυση αποστολών, διάγνωση και σχεδιασμός συστημάτων ελέγχου.

Όλα τα παραπάνω χαρακτηριστικά, καθιστούν το PROOSIS ένα χρήσιμο εργαλείο με εφαρμογές σε όλες τις φάσεις του κύκλου ζωής ενός κινητήρα, από τον προκαταρκτικό σχεδιασμό έως και την υποστήριξη κατά τη διάρκεια χρήσης και λειτουργίας.

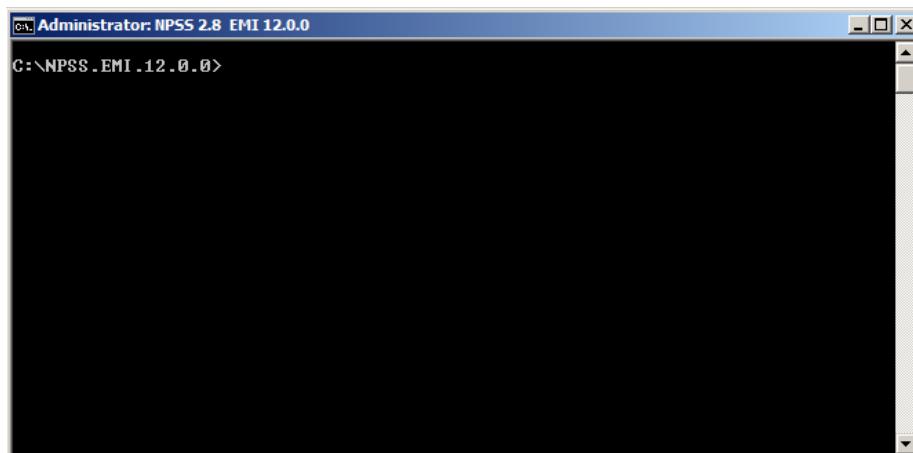
Επιπροσθέτως, το PROOSIS διαθέτει ένα προηγμένο γραφικό περιβάλλον και χρησιμοποιεί μία γλώσσα υψηλού επιπέδου (EL) για μοντελοποίηση μεγάλου εύρους συστημάτων, αλλά δίνει επίσης τη δυνατότητα στους χρήστες να δημιουργήσουν νέες συνιστώσες και βιβλιοθήκες ή ακόμα και να επεκτείνουν τις υπάρχουσες.

10.2.2 NPSS

To NPSS (Numerical Propulsion System Simulation) είναι ένα αντικειμενοστραφές σχεδιαστικό περιβάλλον προσομοίωσης που επιτρέπει την ανάπτυξη και την ενσωμάτωση μοντέλων ποικίλων συστημάτων. Οι κύριες εφαρμογές του λογισμικού περιλαμβάνουν αεροδιαστηματικά συστήματα (μοντέλα επιδόσεων κινητήρων για πρόωση αεροσκαφών), ανάλυση θερμοδυναμικών συστημάτων, όπως για παράδειγμα κύκλοι Rankine και Brayton, ανάπτυξη πυραυλικών συστημάτων και βιομηχανική τυποποίηση για την δυνατότητα κοινής χρήσης και ενσωμάτωσης μοντέλων. Αρχικά αναπτύχθηκε από το Κέντρο Έρευνας Glenn της NASA σε συνεργασία με την κυβέρνηση των Η.Π.Α., την αεροναυπηγική βιομηχανία και με πανεπιστήμια, με στόχο την δημιουργία ενός εξελιγμένου επιλύτη που να μπορεί να χρησιμοποιηθεί σε μεγάλο εύρος διαφορετικών έργων, ο οποίος να είναι συγχρόνως όσο το δυνατόν πιο γενικός. Αργότερα, η NASA μετέφερε τον έλεγχο, την συντήρηση και την ανάπτυξη του κώδικα σε μία βιομηχανική κοινοπραξία. Εν τέλει, το 2013, ο έλεγχος του κώδικα μεταφέρθηκε αποκλειστικά στο Southwest Research Institute (SwRI).

Δεδομένου ότι το πρόγραμμα είναι επιλύτης ροών, έχει επίσης εφαρμοστεί και σε έναν μεγάλο αριθμό ρευστών/θερμικών έργων, όπως σε συστήματα μεταφοράς θερμότητας, κύκλους ψύξης και ανάλυση εκπομπών μηχανών. Επιπροσθέτως, παρέχει στους χρήστες ένα μεγάλο εύρος τρόπων επίλυσης, όπως για παράδειγμα design, off-design, transient, παραμετρική ανάλυση και ανάλυση αποστολής.

Η διασύνδεση του προγράμματος γίνεται μέσω ενός παραθύρου εντολών (command window), όπου καθορίζονται τα αρχεία εισόδου για την εκτέλεση των προσομοιώσεων.



Εικόνα 10.2 Περιβάλλον του NPSS

Η δημιουργία ενός μοντέλου μπορεί να ολοκληρωθεί χρησιμοποιώντας οποιονδήποτε επεξεργαστή κειμένου. Άλλωστε, το NPSS είναι βασισμένο σε γλώσσα

προγραμματισμού C++, κάτι το οποίο διευκολύνει την αναγνωσιμότητα ενός μοντέλου μέσω της πλειονότητας των επεξεργαστών κειμένου.

Οποιοδήποτε μοντέλο μπορεί να δημιουργηθεί χρησιμοποιώντας έναν επεξεργαστή κειμένου, τις ενσωματωμένες θερμοδυναμικές βάσεις δεδομένων και βιβλιοθήκες συνιστωσών του προγράμματος.

Ο χρήστης μπορεί να δημιουργήσει ένα μοντέλο οποιουδήποτε αεροπορικού κινητήρα χρησιμοποιώντας στοιχεία, που ονομάζονται elements, και ρυθμίζοντας τα τεχνικά χαρακτηριστικά που καθορίζουν την απόδοσή τους. Τέλος, πρέπει να συνδέσει όλες τις συνιστώσες μεταξύ τους, μέσω των εντολών linkPorts. Σημαντικό επίσης, είναι να αναφερθεί ότι το NPSS είναι ευέλικτο, παρέχοντας στους χρήστες πρόσβαση στον κώδικα της κάθε συνιστώσας, επιτρέποντας συγχρόνως την δημιουργία νέων στοιχείων, αλλά και την τροποποίηση των ήδη υπαρχόντων.

Έχοντας ολοκληρώσει ένα μοντέλο, ο χρήστης μπορεί πλέον να καθορίσει το πρόβλημα προς επίλυση, να επιλέξει θερμοδυναμικό πακέτο, να καθορίσει τους στόχους και περιορισμούς και τέλος να χρησιμοποιήσει τις αυτόματες ή ακόμα και προσαρμοσμένες ρυθμίσεις του επιλύτη (solver).

Σε απλά μοντέλα, οι χρήστες μπορούν να καθορίσουν το μοντέλο σε ένα ενιαίο αρχείο. Σε μεγαλύτερα και πολυπλοκότερα συστήματα, ωστόσο, είναι χρήσιμο να γίνεται χρήση πολλαπλών ξεχωριστών αρχείων τα οποία καταμερίζουν τον κώδικα και συμβάλλουν στην οργάνωση. Οι τύποι αρχείων που χρησιμοποιούνται συνήθως είναι οι ακόλουθοι:

.run: το κυρίως αρχείο του μοντέλου, μέσα στο οποίο καθορίζονται όλα τα επιπλέον αρχεία που τυχόν θα χρησιμοποιηθούν, το θερμοδυναμικό πακέτο, οι ιδιότητες των ρευστών και τέλος τα αρχεία εξόδου.

.mdl: περιέχει όλες τις συνιστώσες και τι διασυνδέσεις που απαιτούνται και αντιπροσωπεύουν μοντέλο ενός ρευστού/θερμικού συστήματος.

.int or .dll: περιέχει όλες τις μηχανολογικές μεθόδους όλων των στοιχείων του μοντέλου (π.χ. συμπιεστής, στρόβιλος).

.inc: περιέχει όλα τα στοιχεία που απαιτούνται για την οργάνωση της προσομοίωσης, όπως για παράδειγμα ένα αρχείο με τις ρυθμίσεις του επιλύτη.

.fnc: περιέχει συναρτήσεις καθορισμένες από τον χρήστη για την υλοποίηση προσαρμοσμένων υπολογισμών.

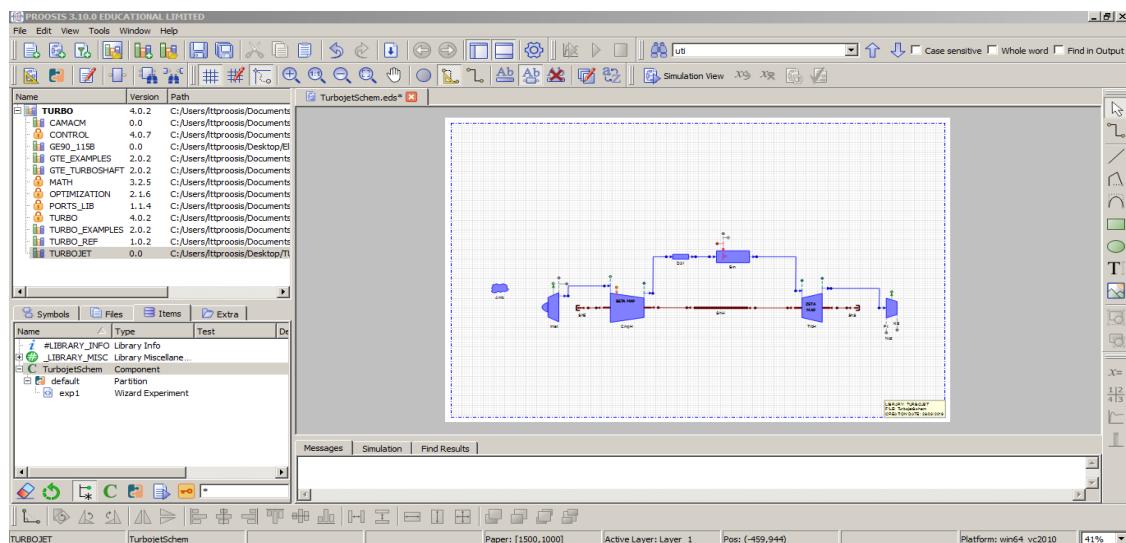
.view: καθορίζει τα αρχεία στα οποία θα γίνει η εξαγωγή των δεδομένων της προσομοίωσης με την επιθυμητή διάταξη.

Τέλος, αξίζει να αναφερθεί ότι τα δεδομένα εξόδου μπορούν να εμφανίζονται είτε στο παράθυρο εντολών, είτε να αποθηκεύονται σε ένα προκαθορισμένο αρχείο. Τα δεδομένα μπορούν να παρουσιάζονται με τη μορφή στηλών ή γραμμών, χρησιμοποιώντας προκαθορισμένες εντολές ή, όπως γίνεται στην πλειονότητα των περιπτώσεων, να χρησιμοποιηθεί ένα αρχείο *view* για την επίτευξη της επιθυμητής διάταξης των δεδομένων.

10.3 Δημιουργία Μοντέλων

10.3.1 Μοντελοποίηση στο PROOSIS

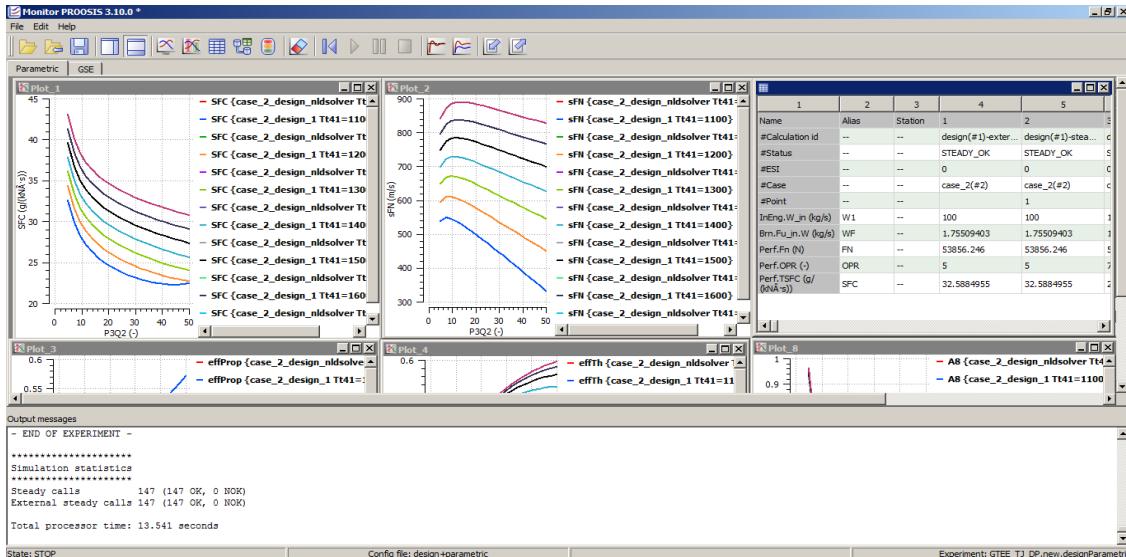
Για να πραγματοποιηθεί οποιαδήποτε προσομοίωση στο περιβάλλον του PROOSIS, πρέπει αρχικά ο χρήστης να δημιουργήσει μία νέα βιβλιοθήκη και ένα νέο σχηματικό. Στο σχηματικό αυτό, μπορεί να χτιστεί το μοντέλο οποιουδήποτε κινητήρα, εισάγοντας τις απαραίτητες συνιστώσες από την ενσωματωμένη βιβλιοθήκη TURBO. Στην κάτωθι εικόνα φαίνεται το γραφικό περιβάλλον του PROOSIS και η γραφική απεικόνιση ενός κινητήρα Turbojet:



Εικόνα 10.3 Γραφικό περιβάλλον στο PROOSIS

Έχοντας πλέον δημιουργήσει το σχηματικό του προς μελέτη κινητήρα, δημιουργούμε ένα νέο *partition* και ένα νέο *experiment*, όπου και κάνουμε όλες τις απαραίτητες ρυθμίσεις για την προσομοίωση. Τέλος για να εκτελέσουμε τον κώδικα επιλέγουμε το στοιχείο *Simulate*. Η εξαγωγή των αποτελεσμάτων μπορεί να γίνει πολύ εύκολα στο Monitor, όπου ο κάθε χρήστης μπορεί να εκτυπώνει τα αποτελέσματα των

υπολογισμών με την διάταξη που επιθυμεί, καθώς επίσης και να εμφανίζει τα αποτελέσματα αυτά με την μορφή διαγραμμάτων. Εξίσου σημαντικό είναι να αναφερθεί ότι τα αποτελέσματα των υπολογισμών μπορούν επίσης να εξαχθούν σε αρχεία της μορφής csv.



Εικόνα 10.4 Παράθυρο εξαγωγής αποτελεσμάτων των προσομοιώσεων

10.3.2 Μοντελοποίηση στο NPSS

Η δημιουργία μοντέλων συνιστωσών, καθώς επίσης και μοντέλων ολοκληρωμένων συστημάτων πρόωσης στο NPSS διαφέρει από το PROOSIS. Εκτός από την έλλειψη γραφικού περιβάλλοντος, το NPSS ομοίως διαφέρει και στον τρόπο διαμόρφωσης ενός μοντέλου. Στη συγκεκριμένη περίπτωση ένα μοντέλο δημιουργείται δίνοντας κατάλληλες εντολές προγραμματισμού σε ένα πρόγραμμα επεξεργασίας κειμένου και η εκτέλεση του κώδικα γίνεται μέσω του παραθύρου εντολών (command window). Η διαδικασία είναι απλή και παρουσιάζεται στα ακόλουθα βήματα:

1. Άνοιγμα ενός νέου αρχείου text σε ένα πρόγραμμα επεξεργασίας κειμένου που υποστηρίζει γλώσσα προγραμματισμού C++.
2. Καθορισμός του επιθυμητού θερμοδυναμικού πακέτου, μέσω του οποίου καθορίζονται οι ιδιότητες των ρευστών/αντιδρώντων:

```
setThermoPackage("allFuel");
```

Άλλα θερμοδυναμικά πακέτα που συμπεριλαμβάνονται στις βιβλιοθήκες του προγράμματος είναι: GasTbl, Janaf, CEA, FPT and REFPROP.

3. Δημιουργία και αρχικοποίηση των απαραίτητων συνιστωσών.

Χρησιμοποιώντας ως παράδειγμα το μοντέλο ενός απλού αγωγού εισόδου, θα πρέπει στην αρχή να ορίσουμε τις ατμοσφαιρικές συνθήκες/συνθήκες εισόδου.

```
Element Ambient Amb {
    switchMode = "ALDTMN";
    alt_in = 0.;
    dTs_in = 0.;
    MN_in = 0.8;
}
```

Ο κώδικας ακολουθεί μία απλή σύνταξη σε μπλοκ όπως φαίνεται παραπάνω. Οι εντολές κάθε μπλοκ πρέπει να περικλείονται μέσα σε αγκύλες. Το κάθε μπλοκ αντιπροσωπεύει μία συνιστώσα (Element).

Η πρώτη γραμμή αρχικοποιεί τη συνιστώσα *Ambient*, που βρίσκεται μέσα στην βιβλιοθήκη του προγράμματος, θέτοντας σε αυτή το όνομα *Amb*. Η ονομασία των συνιστωσών καθορίζεται αποκλειστικά από τον χρήστη. Επομένως χρήσιμο είναι να επιλέγονται ξεχωριστά ονόματα για κάθε συνιστώσα, ιδιαίτερα σε περιπτώσεις μεγαλύτερων μοντέλων (turbofan) όπου για παράδειγμα η συνιστώσα του συμπιεστή μπορεί να υπάρχει περισσότερες από μία φορά μέσα στη διάταξη (χρήση της συνιστώσας *Compressor* για μοντελοποίηση του συμπιεστή χαμηλής, του συμπιεστή υψηλής, καθώς επίσης και του ανεμιστήρα).

Η πρώτη εντολή εντός του μπλοκ καθορίζει ποιες μεταβλητές θα χρησιμοποιηθούν για την αρχικοποίηση της συνιστώσας. Η συγκεκριμένη επιλογή κάνει χρήση του υψόμετρου, της δέλτα-θερμοκρασίας και του αριθμού Mach. Αξίζει να σημειωθεί ότι κάθε εντολή πρέπει να τερματίζεται με το σύμβολο (:).

Ακολούθως, δημιουργούμε το στοιχείο *InletStart*, μέσω του οποίου εκκινείται και καθορίζεται η παροχή εισόδου, καθώς επίσης και τη συνιστώσα από την οποία θα λάβει τις συνθήκες εισόδου, ώστε να οριστούν πλήρως τα χαρακτηριστικά της ροής.

```
Element InletStart InletStart {
    AmbientName = "Amb";
    W_in = 100.;
}
```

Έπειτα, δημιουργούμε τη συνιστώσα του αγωγού εισόδου ως ακολούθως. Η μόνη εντολή που απαιτείται είναι ο προσδιορισμός της πτώσης πίεσης κατά μήκος του αγωγού:

```
Element Inlet InEng {
    PqP_in = 0.995;
}
```

Έχοντας πλέον ολοκληρώσει την δημιουργία των απαραίτητων συνιστωσών του μοντέλου, πρέπει τώρα να ορίσουμε το πέρας της ροής, όπως φαίνεται κάτωθι:

Element FlowEnd FeAir;

4. Καθορισμός των απαραίτητων συνδέσεων μεταξύ των συνιστωσών.

```
linkPorts( "InletStart.Fl_O",      "InEng.Fl_I",      "F1" );
linkPorts( "InEng.Fl_O",          "FeAir.Fl_I",     "F2" );
```

Κάθε μία εντολή δημιουργεί έναν «σταθμό» με μία θύρα εισόδου και μία θύρα εξόδου. Ο χρήστης μπορεί να επιλέξει το επιθυμητό όνομα για κάθε σταθμό, ωστόσο συνηθίζεται να ακολουθείται η αρίθμηση του κινητήρα, δηλαδή η έξοδος από τον συμπιεστή θα χαρακτηρίζεται από τον αριθμό 3.

5. Τέλος, για την εκτέλεση του κώδικα απαιτείται η κάτωθι εντολή:

run();

6. Τα αποτελέσματα των υπολογισμών μπορούν να εκτυπώνονται στο παράθυρο εντολών ή να εξάγονται σε αρχεία. Για την εμφάνιση στο παράθυρο εντολών απαιτούνται οι ακόλουθες εντολές, χρησιμοποιώντας το όνομα του κάθε σταθμού που επιθυμούμε:

```
cout << "W = " << F1.W << " " << F1.W.units << endl;
cout << "Ptin = " << F2.Pt << " " << F2.Pt.units << endl;
cout << "Ttin = " << F2.Tt << " " << F2.Tt.units << endl;
```

Για διευκόλυνση στην ανάγνωση και αξιολόγηση των αποτελεσμάτων, χρήσιμη είναι η εξαγωγή των σε ένα αρχείο με την ακόλουθη εντολή.

```
OutFileStream results {
    filename = "res.txt";
}
```

```
results << "W = " << F1.W << " " << F1.W.units << endl;
```

Επιπροσθέτως, οι χρήστες μπορούν να αξιοποιήσουν κάποια αρχεία που παρέχει το πρόγραμμα για τυποποιημένη εξαγωγή των δεδομένων, τα επονομαζόμενα Viewers. Μερικά από αυτά είναι:

- VarDumpViewer (αλφαριθμητική λίστα των επιθυμητών μεταβλητών)

- CaseColumnViewer (κάθε γραμμή αποτελεί μία μεταβλητή, ενώ κάθε στήλη μία διαφορετική προσομοίωση)
- CaseRowViewer (κάθε γραμμή αποτελεί μία ξεχωριστή προσομοίωση, ενώ κάθε στήλη μία μεταβλητή)
- PageViewer (μία σελίδα με προκαθορισμένη από τον χρήστη μορφή)

Για την εξαγωγή των δεδομένων σε αρχείο της μορφής *.view*, απαιτούνται οι ακόλουθες εντολές:

```
#include "Results.view"
```

Η εντολή αυτή βρίσκεται συνήθως στην αρχή του κώδικα, έπειτα από τον καθορισμό του θερμοδυναμικού πακέτου.

Μετά την εκτέλεση του κώδικα, οι επόμενες εντολές θα τυπώσουν τα επιθυμητά δεδομένα στο προκαθορισμένο αρχείο.

```
rowSheet.update();
rowSheet.display();
```

7. Εκτέλεση του κώδικα.

Οπως έχει προαναφερθεί η εκτέλεση του κώδικα γίνεται στο παράθυρο εντολών, μέσω της εντολής *runnpss* όνομα_αρχείου.run.



Εικόνα 10.5 Εκτέλεση ενός μοντέλου στο NPSS

Αξίζει να σημειωθεί ότι το αρχείο για εκτέλεση πρέπει να έχει την επέκταση *.run* και να βρίσκεται στον φάκελο εγκατάστασης του λογισμικού. Ακόμα, σε περίπτωση που εντοπιστούν σφάλματα κατά την εκτέλεση, αυτά θα εμφανιστούν στο παράθυρο εντολών. Συνήθως, διορθώνοντας το πρώτο σφάλμα είναι επαρκές για την επιτυχή εκτέλεση του κώδικα.

10.4 O solver tou NPSS, τα subelements και τα ports

O solver tou NPSS χρησιμοποιεί μεθόδους Newton-Raphson με ενημερώσεις Broyden, μέχρι όλες οι ανεξάρτητες και εξαρτημένες μεταβλητές να ικανοποιηθούν ώστε εν τέλει να επιτευχθεί σύγκλιση.

10.4.1 Εξαρτημένες και ανεξάρτητες μεταβλητές

Κατά τον προγραμματισμό ενός μοντέλου στο NPSS, οι χρήστες μπορούν να επιλέξουν τις προεπιλεγμένες ανεξάρτητες και εξαρτημένες μεταβλητές, ή ακόμα να ορίσουν εκείνες που οι ίδιοι επιθυμούν. Το NPSS διευκολύνει περαιτέρω την ρύθμιση του solver, παρέχοντας την επιλογή autoSolverSetup. Όταν εκτελείται αυτή η εντολή, αναζητά στον κώδικα όλες τις μεταβλητές οι οποίες έχουν την επιλογή autoSetup ρυθμισμένη σε TRUE και τις προσθέτει, μαζί με τις προεπιλεγμένες μεταβλητές, στον solver. Παρακάτω φαίνεται η ρύθμιση προσαρμοσμένων ανεξάρτητων και εξαρτημένων μεταβλητών:

```
Independent BrnWFuel {
    varName = "BrnPri.Wfuel"; //μεταβλητή που θα μεταβάλλεται από τον solver
    autoSetup = TRUE
}
```

```
Dependent NetThrust {
    eq_lhs = "Perf.Fn"; //αντίστοιχη μεταβλητή στο μοντέλο
    eq_rhs = "22500"; //επιθυμητή τιμή
}
```

```
solver.addIndependent("BrnWfuel");
solver.addDependent("NetThrust");
```

Να σημειωθεί ότι είναι απαραίτητο να υπάρχει ίσος αριθμός ανεξάρτητων και εξαρτημένων μεταβλητών προκειμένου να γίνει επίλυση του προβλήματος. Επίσης, αυτές οι μεταβλητές μπορούν να είναι ενεργές για όσο ο χρήστης επιθυμεί, ενώ μπορεί να τις απενεργοποιήσει με την εντολή removeIndependent/Dependent.

10.4.2 Περιορισμοί

Ενισχύοντας την λειτουργικότητα των παραπάνω, το NPSS έχει την δυνατότητα να επιλύει προβλήματα υπό περιορισμούς, χρησιμοποιώντας ειδικές εξαρτημένες μεταβλητές που σε αυτή την περίπτωση ονομάζονται περιορισμοί.

Οι χρήστες μπορούν ορίσουν την ελάχιστη και μέγιστη τιμή που μπορούν να λάβουν ορισμένες μεταβλητές. Όταν αυτά τα όρια έχουν ξεπεραστεί, τότε η τιμή της εν λόγω εξαρτημένης μεταβλητής γίνεται ίση με το όριο του περιορισμού που έχει τεθεί. Τέλος, αξίζει να σημειωθεί ότι υπάρχει η δυνατότητα πολλαπλών περιορισμών σε μία συγκεκριμένη εξαρτημένη μεταβλητή (περιορισμός της συνολικής ώσης με βάση την θερμοκρασία εξόδου από τον θάλαμο καύσης και τις μηχανικές στροφές της ατράκτου).

10.4.3 Elements, Subelements και Sockets

Τα Elements αποτελούν δομικά στοιχεία του μοντέλου και παρέχουν έναν τρόπο μοντελοποίησης όλων των απαραίτητων υπολογιστικών διαδικασιών. Οι μεταβλητές που αναφέρθηκαν παραπάνω, αποτελούν μέρος των Elements και αντιπροσωπεύουν ποσότητες, όπως για παράδειγμα συντελεστές, φυσικές ιδιότητες και ιδιότητες αερίων. Μερικές από αυτές τις μεταβλητές, ωστόσο, χρησιμοποιούνται για την επιλογή της κατάστασης λειτουργίας (Design, Off-Design) και μαζί με κάποιους διακόπτες μπορούν να ελέγχουν πλήρως τα χαρακτηριστικά και την συμπεριφορά ενός δομικού στοιχείου. Όπως έχει αναφερθεί ήδη στην εισαγωγή, τα στοιχεία αυτά έχουν σχεδιαστεί ούτως ώστε να είναι όσο το δυνατόν πιο γενικά. Αυτό, διευκολύνεται με την χρήση των subelements (υπο-στοιχεία).

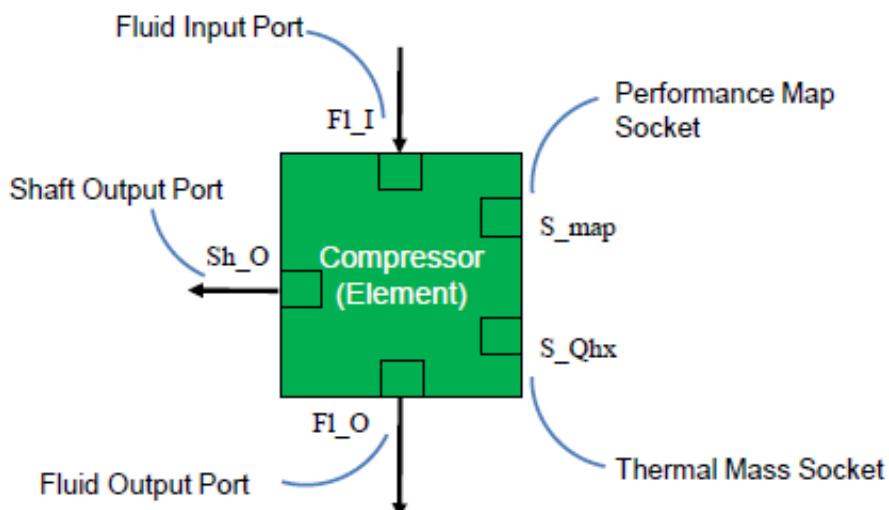
Τα subelements ενισχύουν τα βασικά elements με τρόπο τέτοιο ώστε να είναι πιο λεπτομερή στην ανάλυσή τους (π.χ. χάρτες επιδόσεων συμπιεστή). Έτσι, μπορεί το ίδιο στοιχείο (compressor, turbine) να μοντελοποιήσει όλους τους διαφορετικούς τύπους συμπιεστών, ανεμιστήρων και στροβίλων. Η διάκριση για παράδειγμα, μεταξύ συμπιεστή χαμηλής και υψηλής πίεσης, γίνεται μέσω των subelements, δηλαδή των χαρτών που προσδιορίζουν εν τέλει την απόδοσή τους. Τα subelements περιέχουν υπολογισμούς, πίνακες και συναρτήσεις που αντιστοιχούν σε συγκεκριμένο συμπιεστή ή στρόβιλο. Ακόμα, να αναφερθεί ότι ένα υπο-στοιχείο μπορεί να είναι ένας χάρτης ή ακόμα και ένα αρχείο με τα αποτελέσματα μια ανάλυσης CFD.

Αυτή η αρχιτεκτονική με τα subelements, παρέχει μεγάλη ευελιξία δίνοντας την δυνατότητα στους χρήστες να μπορούν να επέμβουν σε ένα στοιχείο με τις επιθυμητές τροποποιήσεις, καθώς επίσης ελαχιστοποιεί τον πηγαίο κώδικα, καταμερίζοντάς τον σε διαφορετικά αρχεία. Για παράδειγμα, οι επιδόσεις ενός συμπιεστή υπολογίζονται με βάση τις συνθήκες εισόδου, τον βαθμό απόδοσης και τον λόγο πίεσης, κάτι το οποίο επιτρέπει σε κάθε συμπιεστή να χρησιμοποιεί τον ίδιο πηγαίο κώδικα. Ωστόσο, ο βαθμός απόδοσης

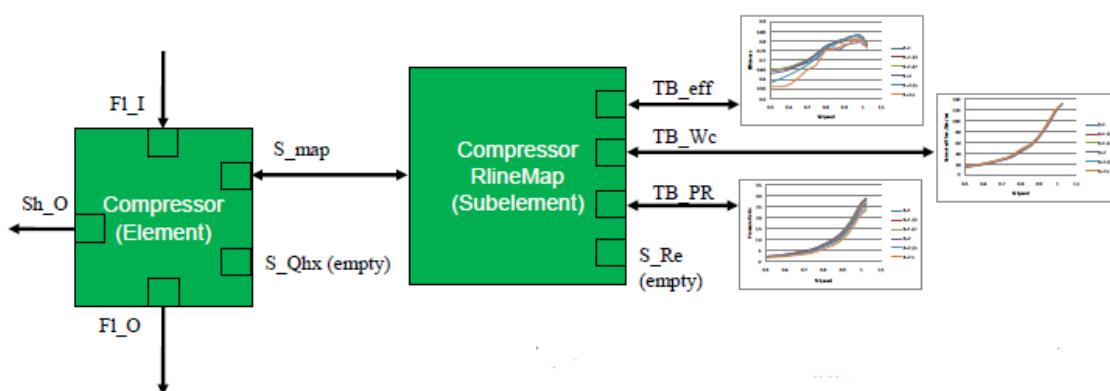
και ο λόγος πίεσης σε κάποιο σημείο εκτός του σημείου λειτουργίας, προκύπτει από το αντίστοιχο subelement. Τέλος, αυτή η αρχιτεκτονική διευκολύνει ιδιαίτερα σε περιπτώσεις όπου απαιτούνται εξατομικευμένοι υπολογισμοί, καθώς έτσι απαιτείται μόνο αναπροσαρμογή του subelement χωρίς περαιτέρω πρόσβαση στον κώδικα του element. Σε περίπτωση δε που ο κώδικας ήταν ενιαίος, θα απαιτούταν να εντοπισθεί το συγκεκριμένο σημείο προς τροποποίηση.

Τα sockets ουσιαστικά, είναι κανάλια που διευκολύνουν την επικοινωνία μεταξύ των βασικών δομικών στοιχείων (elements) και των υπο-στοιχείων (subelements).

Στις ακόλουθες εικόνες παρουσιάζεται σχηματικά το στοιχείο του συμπιεστή με όλες τις σχετικές υποδοχές, καθώς επίσης και η σύνδεσή του με το αντίστοιχο subelement/χάρτη επιδόσεων.



Εικόνα 10.6 Σχηματικό ενός τυπικού συμπιεστή



Εικόνα 10.7 Σχηματικό ενός τυπικού συμπιεστή με προσάρτηση χάρτη επιδόσεων στην αντίστοιχη υποδοχή

10.4.4 Μεταβλητές σε Design και Off-Design

1. InletStart

Το στοιχείο *InletStart* εκκινεί τη ροή και θέτει τις συνθήκες εισόδου στον κινητήρα. Το στοιχείο αυτό λαμβάνει τις συνθήκες πτήσης από το στοιχείο *Ambient* που καθορίζεται από τον χρήστη.

- Στο Design: ο χρήστης εισάγει το *W_in* (παροχή αέρα)
- Στο Off-Design : o solver μεταβάλλει την παροχή αέρα

2. Συμπιεστής

Τα ακόλουθα ισχύουν για το τυπικό στοιχείο συμπιεστή και το αντίστοιχο CompressorRlineMap subelement.

- Στο Design
 - Ο χρήστης εισάγει τα PRdes και effDes
 - Το S_map socket χρησιμοποιεί τους χάρτες, τα PRdes, effDes και Wc για να υπολογίσει τα scalars των χαρτών
- Στο Off-Design
 - Τα Nc (ανηγμένες στροφές), R-line και alpha (γωνία εισόδου της ροής) χρησιμοποιούνται για να διαβάσουν τις επιδόσεις από τον χάρτη
 - Ο solver καθορίζει την απαιτούμενη τιμή για την παράμετρο R-line έτσι ώστε η ανηγμένη παροχή που προκύπτει από τον χάρτη να ισούται με αυτήν που εισέρχεται στον συμπιεστή, εξασφαλίζοντας την αρχή της συνέχειας

3. Στροβίλος

Τα ακόλουθα ισχύουν για το τυπικό στοιχείο στροβίλου και το αντίστοιχο TurbinePRmap subelement.

- Στο Design
 - Ο χρήστης εισάγει το effDes και μία αρχική εκτίμηση για τον λόγο πίεσης
 - Ο solver καθορίζει τον απαιτούμενο λόγο πίεσης του στροβίλου (PRbase) για να ισορροπήσει τον κύκλο
 - Το S_map socket χρησιμοποιεί τους χάρτες, τα PRdes, effDes και Wp για να υπολογίσει τα scalars των χαρτών
- Στο Off-Design

- Τα Np (ανηγμένες στροφές) και PRbase χρησιμοποιούνται για να διαβάσουν τις επιδόσεις από τον χάρτη
- Ο solver καθορίζει την απαιτούμενη τιμή για τον λόγο πίεσης έτσι ώστε η ανηγμένη παροχή που προκύπτει από τον χάρτη να ισούται με αυτήν που εισέρχεται στον στροβίλο, εξασφαλίζοντας την αρχή της συνέχειας

4. Ατρακτος

Το στοιχείο της ατράκτου παρέχει την μηχανική σύνδεση μεταξύ των συμπιεστών και των στροβίλων που υπάρχουν σε ένα μοντέλο.

- Στο Design
 - Ο χρήστης εισάγει τις μηχανικές στροφές της ατράκτου (Nmech)
 - Μία εξαρτημένη μεταβλητή διασφαλίζει το ισοζύγιο ισχύος στην άτρακτο (για παράδειγμα η ροπή του συμπιεστή να ισούται με την ροπή του στροβίλου)
- Στο Off-Design
 - Μία ανεξάρτητη μεταβλητή μεταβάλλει τις στροφές της ατράκτου
 - Μία εξαρτημένη μεταβλητή εξασφαλίζει ότι η ροπή στην άτρακτο ισούται με μηδέν

5. Ακροφύσιο

Το στοιχείο του ακροφυσίου υπολογίζει τις συνθήκες εξόδου του κινητήρα και την συνολική ώση. Η πίεση εξόδου αποτελεί είσοδο και συνήθως λαμβάνεται από το στοιχείο *Ambient*.

- Στο Design
 - Αν πρόκειται για συγκλίνον ακροφύσιο, η διατομή εξόδου υπολογίζεται έτσι ώστε η ροή να είναι στραγγαλισμένη (αν ο λόγος πίεσης του ακροφυσίου είναι αρκετά μεγάλος) ή ώστε η στατική πίεση εξόδου να ισούται με την πίεση περιβάλλοντος (αν ο λόγος πίεσης του ακροφυσίου είναι μικρότερος από αυτόν που θα προκαλούσε στραγγαλισμό της ροής).
 - Αν πρόκειται για συγκλίνον-αποκλίνον ακροφύσιο, η διατομή του λαιμού υπολογίζεται έτσι ώστε η ροή να είναι στραγγαλισμένη. Αν ο λόγος πίεσης του ακροφυσίου είναι αρκετά χαμηλός για να οδηγήσει σε στραγγαλισμό, τότε ένα μήνυμα σφάλματος εμφανίζεται. Η διατομή εξόδου υπολογίζεται έτσι ώστε η στατική πίεση εξόδου να ισούται με την πίεση περιβάλλοντος.

- Στο Off-Design
 - Αν πρόκειται για συγκλίνον ακροφύσιο, η διατομή εξόδου είναι σταθερή (ίση με αυτή που υπολογίστηκε στο Design) και μία εξαρτημένη μεταβλητή εξασφαλίζει είτε η ροή να είναι στραγγαλισμένη, είτε η στατική πίεση εξόδου να ισούται με την πίεση περιβάλλοντος.
 - Αν πρόκειται για συγκλίνον-αποκλίνον ακροφύσιο, η διατομή του λαιμού είναι σταθερή (ίση με αυτή που υπολογίστηκε στο Design) και η διατομή εξόδου επανυπολογίζεται έτσι ώστε η στατική πίεση εξόδου να είναι ίση με την πίεση περιβάλλοντος. Μία εξαρτημένη μεταβλητή διασφαλίζει ότι η ροή είναι στραγγαλισμένη.

6. Turbojet

- Στο Design
 - Ο χρήστης εισάγει τις επιθυμητές τιμές σε όλα τα στοιχεία
 - Ο solver μεταβάλλει τον λόγο πίεσης του στροβίλου ώστε να διασφαλίσει τη διατήρηση ροπής στην άτρακτο (αυτόματα)
 - Ο solver μεταβάλλει την παροχή καυσίμου έτσι ώστε να πετύχει την ζητούμενη θερμοκρασία εισόδου στον στρόβιλο (καθορίζεται από τον χρήστη)
 - Ο solver μεταβάλλει την παροχή αέρα έτσι ώστε να πετύχει την επιθυμητή ώση (από τον χρήστη)

Πίνακας 10.1 Ανεξάρτητες μεταβλητές στο Design point σε κινητήρα Turbojet

	Ανεξάρτητες μεταβλητές	
	Μεταβλητή στον solver	Μεταβλητή στο μοντέλο
Αυτόματο	Trb.S_map.ind_PRbase	Trb.PRbase
Από χρήστη	ind_Wfuel	FusEng.Wfuel
Από χρήστη	ind_Wair	FsEng.W_in

Πίνακας 10.2 Εξαρτημένες μεταβλητές στο Design point σε κινητήρα Turbojet

	Εξαρτημένες μεταβλητές		
	Μεταβλητή στον solver	Αριστερό μέλος	Δεξί μέλος
Αυτόματο	Sh.integrate_Nmech	Sh.trqNet	0.0
Από χρήστη	dep_T4	F04.Tt	T4_req

<i>Από χρήστη</i>	dep_Fn	Perf.Fn	Fn_req
-------------------	--------	---------	--------

- Στο Off-Design
 - Ο χρήστης εισάγει τις επιθυμητές τιμές στην κατάσταση λειτουργίας εκτός του σημείου σχεδίασης
 - Ο χρήστης καθορίζει τις εξαρτημένες και ανεξάρτητες μεταβλητές και κάνει τις επιθυμητές ρυθμίσεις στον solver

Πίνακας 10.3 Ανεξάρτητες μεταβλητές στο Off-Design σε κινητήρα Turbojet

	Ανεξάρτητες μεταβλητές	
	Μεταβλητή στον solver	Μεταβλητή στο μοντέλο
<i>Αντόματο</i>	Trb.S_map.ind_PRbase	Trb.PRbase
<i>Αντόματο</i>	Cmp.S_map.ind_RlineMap	Cmp.S_map.RlineMap
<i>Αντόματο</i>	Sh.ind_Nmech	Sh.Nmech
<i>Αντόματο</i>	FsEng.ind_W	FsEng.W
<i>Από χρήστη</i>	ind_Wfuel	FusEng.Wfuel

Πίνακας 10.4 Εξαρτημένες μεταβλητές στο Off-Design σε κινητήρα Turbojet

	Εξαρτημένες μεταβλητές		
	Μεταβλητή στον solver	Αριστερό μέλος	Δεξί μέλος
<i>Αντόματο</i>	Trb.S_map.dep_errWp	Trb.Wp	Trb.WpCalc
<i>Αντόματο</i>	Cmp.S_map.dep_errWc	Cmp.Wc	Cmp.WcCalc
<i>Αντόματο</i>	Sh.integrate_Nmech	Sh.trqNet	0.0
<i>Αντόματο</i>	NozPri.dep_Area	NozPri.WqAe	NozPri.WqAEdem
<i>Από χρήστη</i>	dep_Fn	Perf.Fn	Fn_req

7. Turbofan

- Στο Design
 - Ο χρήστης εισάγει τις επιθυμητές τιμές για όλα τα στοιχεία
 - Ο solver μεταβάλλει το λόγο πίεσης των στροβίλων υψηλής και χαμηλής ώστε να εξασφαλίζει την διατήρηση ισχύος στην άτρακτο υψηλής και χαμηλής αντίστοιχα (αυτόματα)
 - Ο solver μεταβάλλει την παροχή καυσίμου ώστε να πετύχει την επιθυμητή θερμοκρασία εισόδου στον στρόβιλο (από τον χρήστη)
 - Ο solver μεταβάλλει την παροχή αέρα ώστε να επιτύχει την απαιτούμενη ώση (από τον χρήστη)

Πίνακας 10.5 Ανεξάρτητες μεταβλητές στο Design point σε κινητήρα Turbofan

	Ανεξάρτητες μεταβλητές	
	Μεταβλητή στον solver	Μεταβλητή στο μοντέλο
Αυτόματο	TrbL.S_map.ind_PRbase	TrbL.PRbase
Αυτόματο	TrbH.S_map.ind_PRbase	TrbH.PRbase
Από χρήστη	ind_Wfuel	FusEng.Wfuel
Από χρήστη	ind_Wair	FsEng.W_in

Πίνακας 10.6 Εξαρτημένες μεταβλητές στο Design point σε κινητήρα Turbofan

	Εξαρτημένες μεταβλητές		
	Μεταβλητή στον solver	Αριστερό μέλος	Δεξιό μέλος
Αυτόματο	ShL.integrate_Nmech	ShL.trqNet	0.0
Αυτόματο	ShH.integrate_Nmech	ShH.trqNet	0.0
Από χρήστη	dep_T4	F04.Tt	T4_req
Από χρήστη	dep_Fn	Perf.Fn	Fn_req

- Στο Off-Design
 - Ο χρήστης εισάγει τις επιθυμητές τιμές για την λειτουργία σε εκτός του σημείου σχεδίασης
 - Ο χρήστης καθορίζει τις ανεξάρτητες και εξαρτημένες μεταβλητές και κάνει τις επιθυμητές ρυθμίσεις στον solver

Πίνακας 10.7 Ανεξάρτητες μεταβλητές στο Off-Design σε κινητήρα Turbofan

	Ανεξάρτητες μεταβλητές	
	Μεταβλητή στον solver	Μεταβλητή στο μοντέλο
Αντόματο	TrbL.S_map.ind_PRbase	TrbL.PRbase
Αντόματο	TrbH.S_map.ind_PRbase	TrbH.PRbase
Αντόματο	CmpL.S_map.ind_RlineMap	CmpL.S_map.RlineMap
Αντόματο	CmpH.S_map.ind_RlineMap	CmpH.S_map.RlineMap
Αντόματο	CmpFan.S_map.ind_RlineMap	CmpFan.S_map.RlineMap
Αντόματο	ShL.ind_Nmech	ShL.Nmech
Αντόματο	ShH.ind_Nmech	ShH.Nmech
Αντόματο	Slpit.ind_BPR	Split.BPR
Αντόματο	FsEng.ind_W	FsEng.W
Από χρήστη	ind_Wfuel	FusEng.Wfuel

Πίνακας 10.8 Εξαρτημένες μεταβλητές στο Off-Design σε κινητήρα Turbofan

	Εξαρτημένες μεταβλητές		
	Μεταβλητή στον solver	Αριστερό μέλος	Δεξιό μέλος
Αντόματο	TrbL.S_map.dep_errWp	TrbL.Wp	TrbL.WpCalc
Αντόματο	TrbH.S_map.dep_errWp	TrbH.Wp	TrbH.WpCalc
Αντόματο	CmpL.S_map.dep_errWc	CmpL.Wc	CmpL.WcCalc
Αντόματο	CmpH.S_map.dep_errWc	CmpH.Wc	CmpH.WcCalc
Αντόματο	CmpFan.S_map.dep_errWc	CmpFan.Wc	CmpFan.WcCalc
Αντόματο	ShL.integrate_Nmech	ShL.trqNet	0.0
Αντόματο	ShH.integrate_Nmech	ShH.trqNet	0.0
Αντόματο	NozPri.dep_Area	NozPri.WqAe	NozPri.WqAEdem
Αντόματο	NozSec.dep_Area	NozSec.WqAe	NozSec.WqAEdem
Από χρήστη	dep_Fn	Perf.Fn	Fn_req

Ο κινητήρας Turbofan ουσιαστικά αποτελεί μία εξέλιξη του κινητήρα Turbojet. Οποιεσδήποτε τυχόν νέες εξαρτημένες και ανεξάρτητες μεταβλητές οφείλονται στις επιπλέον συνιστώσες από τις οποίες αποτελείται ένας κινητήρας.

8. Turboshaft

Όπως αναφέρθηκε παραπάνω, τυχόν επιπλέον ανεξάρτητες και εξαρτημένες μεταβλητές, εξαρτώνται αποκλειστικά από τις συνιστώσες από τις οποίες αποτελείται ένας

κινητήρας αλλά και την κρίση του χρήστη. Ο κινητήρας Turboshaft ομοίως είναι μία εξέλιξη του κινητήρα Turbojet και οι εξαρτημένες και ανεξάρτητες μεταβλητές παρουσιάζονται στους ακόλουθους πίνακες:

Πίνακας 10.9 Ανεξάρτητες μεταβλητές στο Design point σε κινητήρα Turboshaft

	Ανεξάρτητες μεταβλητές	
	Μεταβλητή στον solver	Μεταβλητή στον solver
Αυτόματο	TrbL.S_map.ind_PRbase	TrbL.PRbase
Αυτόματο	TrbH.S_map.ind_PRbase	TrbH.PRbase

Πίνακας 10.10 Εξαρτημένες μεταβλητές στο Design point σε κινητήρα Turboshaft

	Εξαρτημένες μεταβλητές		
	Μεταβλητή στον solver	Αριστερό μέλος	Δεξιό μέλος
Αυτόματο	ShL.integrate_Nmech	ShL.trqNet	0.0
Αυτόματο	ShH.integrate_Nmech	ShH.trqNet	0.0

Πίνακας 10.11 Ανεξάρτητες μεταβλητές στο Off-Design σε κινητήρα Turboshaft

	Ανεξάρτητες μεταβλητές	
	Μεταβλητή στον solver	Μεταβλητή στον solver
Αυτόματο	TrbFP.S_map.ind_PRbase	TrbFP.PRbase
Αυτόματο	TrbH.S_map.ind_PRbase	TrbH.PRbase
Αυτόματο	CmpH.S_map.ind_RlineMap	CmpH.S_map.RlineMap
Αυτόματο	ShFP.ind_Nmech	ShFP.Nmech
Αυτόματο	ShH.ind_Nmech	ShH.Nmech
Από χρήστη	Brn.ind_FAR	Brn.FAR
Από χρήστη	ShFP.ind_SHP	ShFP.HPX

Πίνακας 10.12 Εξαρτημένες μεταβλητές στο Off-Design σε κινητήρα Turboshaft

	Εξαρτημένες μεταβλητές		
	Μεταβλητή στον solver	Αριστερό μέλος	Δεξιό μέλος
Αυτόματο	TrbFP.S_map.dep_errWp	TrbFP.Wp	TrbFP.WpCalc
Αυτόματο	TrbH.S_map.dep_errWp	TrbH.Wp	TrbH.WpCalc
Αυτόματο	CmpH.S_map.dep_errWc	CmpH.Wc	CmpH.WcCalc
Αυτόματο	ShFP.integrate_Nmech	ShFP.trqNet	0.0

<i>Αντόματο</i>	ShH.integrate_Nmech	ShH.trqNet	0.0
<i>Αντόματο</i>	NozPri.dep_Area	NozPri.WqAe	NozPri.WqAEdem
<i>Από χρήστη</i>	dep_Np	ShFP.Nmech	NP_dmd
<i>Από χρήστη</i>	dep_SHP	Perf.SHP	SHP_dmd

10.4.5 *Ports*

Λαμβάνοντας υπόψη ότι ο απότερος σκοπός αυτής της εργασίας είναι να αποτελέσει τον θεμέλιο λίθο για μία μελλοντική διασύνδεση του PROOSIS και του NPSS, θα παρουσιάσουμε σε αυτό το σημείο τις διαφορές που παρουσιάζουν τα δύο λογισμικά όσον αφορά τα ports και την πληροφορία που αντά μεταφέρουν. Στους παρακάτω πίνακες παρουσιάζονται οι μεταβλητές που μεταφέρονται μέσω των αντίστοιχων port σε κάθε πρόγραμμα:

Πίνακας 10.13 Σύγκριση των μεταβλητών που μεταφέρονται μέσω του fluid port

FLUID PORT		
NPSS		PROOSIS
Tt		W
Pt		Tt
ht	Ενθαλπία βασισμένη στις ολικές συνθήκες	Pt
ut	Εσωτερική ενέργεια	FARB
Cpt	Ειδική θερμοχωρητικότητα (σταθερή πίεση)	FARU
Cvt	Ειδική θερμοχωρητικότητα (σταθερός όγκος)	WAR
gamt	Λόγος ειδικών θερμοχωρητικοτήτων	Ang
s	Εντροπία	
rho	Πυκνότητα	
mut	Ιξώδες	
kt	Αγωγιμότητα	
Rt	Σταθερά αερίων	
W		
FAR		
WAR		
MN		
V		
A		
Ts		
Ps		

hs	
us	
Cps	
Cvs	
gams	
rhos	
mus	
ks	
Rs	
Cd	

Πίνακας 10.14 Σύγκριση των μεταβλητών που μεταφέρονται μέσω του fuel port

FUEL PORT	
NPSS	PROOSIS
Wfuel	W
Tfuel	T
Pfuel	P
fuelType	
LHV	
hFuel	
Tref	
hRef	
Carbon	
Hydrogen	
Nitrogen2	
Oxygen	

Πίνακας 10.15 Σύγκριση των μεταβλητών που μεταφέρονται μέσω του mechanical port

MECHANICAL PORT	
NPSS	PROOSIS
Nmech	Nmech
pwr	trq
trq	inertia
inertia	inertia_tot

Παρατηρούμε ότι σε κάθε πρόγραμμα μεταφέρεται διαφορετική ποσότητα πληροφορίας μέσω των ports. Το NPSS μεταφέρει περισσότερη πληροφορία σε σχέση με το PROOSIS, καθώς το δεύτερο υπολογίζει οτιδήποτε χρειασθεί (ενθαλπία, εντροπία) μέσα σε κάθε συνιστώσα αντιθέτως. Αυτή η διαφορά είναι σημαντική και θα πρέπει να δοθεί ιδιαίτερη προσοχή κατά την διασύνδεση των δύο λογισμικών (είτε κατασκευαστούν τα elements του NPSS και προσαρμοστούν στις βιβλιοθήκες του PROOSIS, είτε επικοινωνούν απευθείας τα προγράμματα), ώστε να εξασφαλιστεί ότι η ίδια πληροφορία μεταφέρεται μέσω των αντίστοιχων ports και στα δύο προγράμματα.

Επιπλέον, ένα ακόμη σημείο που χρήζει προσοχής είναι ότι το NPSS έχει την δυνατότητα να εκτελεί ορισμένες συναρτήσεις μέσα στα ports. Για παράδειγμα οι *setTotal* και *setStatic* οι οποίες καθορίζουν τα χαρακτηριστικά της ροής με βάση την πίεση και την θερμοκρασία ή οποιονδήποτε άλλο υποστηριζόμενο συνδυασμό, η *copyFlow* η οποία αντιγράφει τα χαρακτηριστικά μίας ροής και η *burn* η οποία πραγματοποιεί καύση του εργαζόμενου μέσου, όπως ένα στοιχείο θαλάμου καύσης. Αυτή η επιπλέον δυνατότητα του NPSS πρέπει να ληφθεί υπόψη κατά την διασύνδεση των λογισμικών.

10.5 Ενσωμάτωση των χαρτών επιδόσεων του NPSS στο PROOSIS

Όπως είναι γνωστό, οι χάρτες καθορίζουν τις επιδόσεις των συνιστωσών. Έτσι, εκτός από τα δεδομένα, έτσι και οι χάρτες πρέπει να είναι ίδιοι ώστε η σύγκριση να είναι ακριβής. Επομένως, επιλέχθηκε να αξιοποιηθεί το γραφικό περιβάλλον που παρέχει το PROOSIS και να εισαχθούν τα αρχεία των χαρτών του NPSS σε αυτό.

Τα αρχεία χαρτών στο NPSS έχουν το όνομα ***RlineMap_EEE_**.map* (π.χ. *CompressorRlineMap_EEE_fan.map* για τον χάρτη του ανεμιστήρα). Το EEE αναφέρεται σε ένα πρόγραμμα (Energy Efficient Engine) που χρηματοδοτήθηκε από την NASA το 1978 και είχε ως σκοπό την ανάπτυξη πιο αποδοτικών κινητήρων Turbofan υψηλού λόγου παράκαμψης. Μεγάλοι κατασκευαστές της αεροδιαστημικής βιομηχανίας, όπως η Boeing, η General Electric και η Pratt & Whitney συμμετείχαν στο πρόγραμμα και παρείχαν τα απαιτούμενα αεροσκάφη και κινητήρες για τις δοκιμές. Η P&W κατασκεύασε έναν συμπιεστή υψηλής πίεσης 9 βαθμίδων, ενώ η GE έναν συμπιεστή υψηλής πίεσης 10 βαθμίδων με λόγο πίεσης 23. Όλοι όσοι συνέβαλαν στο εγχείρημα επωφελήθηκαν, ιδιαίτερα η GE, η οποία χρησιμοποίησαν τον συμπιεστή τους στους κινητήρες GE90 και GEnx που χρησιμοποιήθηκαν αργότερα στα Boeing 777 και 777X αντίστοιχα.

10.5.1 Χάρτης συμπιεστή (R-line map)

Στην αρχή του κάθε αρχείου ορίζονται κάποιες μεταβλητές, οι οποίες είναι συγκεκριμένες για τον χάρτη. Οι μεταβλητές αυτές είναι η γωνία του στάτορα (α MapDes = 0.0), οι διορθωμένες στροφές (Nc MapDes = 1.0), η παράμετρος R (R lineMapDes = 2.0) και η παράμετρος R στο σημείο πάλμωσης (R lineStall = 1.0). Κάθε καμπύλη του χάρτη ορίζεται από τη γωνία του στάτορα, την τιμή της παραμέτρου R και την ανηγμένη παροχή. Για κάθε τιμή της γωνίας ορίζονται τα ακόλουθα:

- **Ανεμιστήρας:** για διάφορες τιμές των ανηγμένων στροφών, δίνονται ζεύγη τιμών για την ανηγμένη παροχή και την παράμετρο R, η οποία λαμβάνει τιμές από 1 έως 2.5. με τον ίδιο τρόπο κατασκευάζονται και οι καμπύλες για τον βαθμό απόδοσης και τον λόγο πίεσης. Ωστόσο, στην συγκεκριμένη περίπτωση του ανεμιστήρα, για τις 3 πρώτες καμπύλες ανηγμένων στροφών η παράμετρος R λαμβάνει λιγότερες τιμές σχέση με τα υπόλοιπα σημεία. Δηλαδή, οι τιμές κυμαίνονται από 1.0 έως 2.0 για αυτά τα 3 πρώτα σημεία, ενώ από 1.0 έως 2.5 για τα υπόλοιπα.
- **Συμπιεστής χαμηλής και υψηλής πίεσης:** όλα τα παραπάνω ισχύουν και στην περίπτωση του συμπιεστή, ωστόσο η παράμετρος R έχει το ίδιο εύρος τιμών για όλες τις τιμές ανηγμένων στροφών.

Για ενδιάμεσα σημεία, χρησιμοποιείται παρεμβολή και προεκβολή. Πιο συγκεκριμένα, για τις ανηγμένες στροφές και την παράμετρο R, χρησιμοποιείται πολυωνυμική παρεμβολή Lagrange 3rd βαθμού, ενώ για την γωνία του στάτορα χρησιμοποιείται γραμμική παρεμβολή και προεκβολή. Για τα ενδιάμεσα σημεία του λόγου πίεσης δεν ορίζεται κάποια συγκεκριμένη μέθοδος παρεμβολής.

10.5.2 Χάρτης στροβίλου (Turbine PR map)

Οι χάρτες των στροβίλων ορίζονται όμοια με εκείνους των συμπιεστών. Αρχικά ορίζονται τιμές για τον λόγο πίεσης στο σημείο σχεδίασης (PR mapDes = 4.975) και τις ανηγμένες στροφές (Np MapDes = 100). Η κάθε καμπύλη ορίζεται από τις ανηγμένες στροφές, τον λόγο πίεσης και την ανηγμένη παροχή.

- **Στρόβιλος υψηλής και χαμηλής πίεσης:** και στις δύο περιπτώσεις για κάθε τιμή των ανηγμένων στροφών, δίνεται ένα ζεύγος τιμών για τον λόγο πίεσης και την ανηγμένη παροχή. Η παράμετρος PR στους χάρτες των στροβίλων χρησιμοποιείται με παρόμοιο τρόπο όπως η παράμετρος R στους συμπιεστές, $PR = 6.0$ αντιστοιχεί σε στραγγαλισμό, ενώ $PR = 2.0$ αντιστοιχεί στη μικρότερη παροχή.

Αντίστοιχα, για τα ενδιάμεσα σημεία των στροφών χρησιμοποιείται πολυωνυμική παρεμβολή Lagrange 2nd βαθμού και 3rd βαθμού για τον λόγο πίεσης.

10.5.3 Διαδικασία ενσωμάτωσης χαρτών

Μία βασική διαφορά μεταξύ των χαρτών συμπιεστών του PROOSIS και του NPSS είναι ότι το πρώτο χρησιμοποιεί γραμμές BETA και ZETA, ενώ το δεύτερο γραμμές R και PR για τους συμπιεστές και τους στροβίλους αντίστοιχα. Επιπλέον διαφορές παρατηρούνται όχι μόνο στο εύρος τιμών των παραμέτρων των δύο προγραμμάτων αλλά και στην κατάσταση που βρίσκεται ο κινητήρας στα όρια των παραμέτρων αυτών. Η παράμετρος BETA έχει εύρος από 0 (στραγγαλισμός) έως 1.0 (πάλμωση), ενώ η παράμετρος R έχει εύρος από 2.5 (στραγγαλισμός) έως 1.0 (πάλμωση).

Για να ενσωματωθούν οι χάρτες του NPSS στη μορφή του PROOSIS, τροποποιήθηκαν οι χάρτες BETA και ZETA χρησιμοποιώντας τους αντίστοιχους πίνακες του NPSS. Αφού σε κάθε πρόγραμμα χρησιμοποιούνται διαφορετικές παράμετροι, ορίστηκε μία σχέση αντιστοίχησης μεταξύ των δύο, ρυθμίζοντας κατάλληλα τα εύρη. Ωστόσο, όσον αφορά την περίπτωση του χάρτη του ανεμιστήρα του NPSS, όπως αναφέρθηκε και προηγουμένως δεν υπάρχει ένας ενιαίος αριθμός σημείων (λιγότερα σημεία για τις τρεις πρώτες τιμές των ανηγμένων στροφών). Προκειμένου να μην χρειασθεί να κάνουμε παρεμβολές ώστε να παράξουμε ίσο αριθμό σημείων για όλες τις τιμές των ανηγμένων στροφών, καθώς αυτό απαιτείται στους χάρτες του PROOSIS, επιλέξαμε να εισάγουμε τον χάρτη του ανεμιστήρα του PROOSIS στο NPSS αντίστροφα. Εν τέλει, όλοι οι προκύπτοντες χάρτες που χρησιμοποιήθηκαν είναι πανομοιότυποι, ενώ επιπλέον ορίστηκε και στα δύο προγράμματα η χρήση γραμμικής παρεμβολής για τα ενδιάμεσα σημεία. Τέλος, αξίζει να αναφερθεί ότι έγινε επιπλέον και κατάλληλη μετατροπή μονάδων, αφού στο NPSS η παροχή μετράται σε lbm/sec, ενώ στο PROOSIS σε kg/sec.

Είναι σημαντικό σε αυτό το σημείο να αναφερθεί ότι οι διαφορές που παρατηρούνται όσον αφορά τα όρια των παραμέτρων των χαρτών μεταξύ των λογισμικών, μπορούν να δημιουργήσουν με τη σειρά τους διαφορές τόσο σε επίπεδο συνιστωσών όσο και σε επίπεδο μηχανής. Μικρές διαφορές των παραμέτρων των δύο λογισμικών, τοποθετούν τα αντίστοιχα σημεία λειτουργίας επάνω στον χάρτη με μία μικρή απόκλιση μεταξύ τους, κάτι που με την σειρά του δημιουργεί διαφορές, έστω και αμελητέες, στις επιδόσεις των συνιστωσών, όπως θα παρουσιαστεί και στις επόμενες παραγράφους.

10.6 Ενσωμάτωση του θερμοδυναμικού μοντέλου του NPSS στο PROOSIS

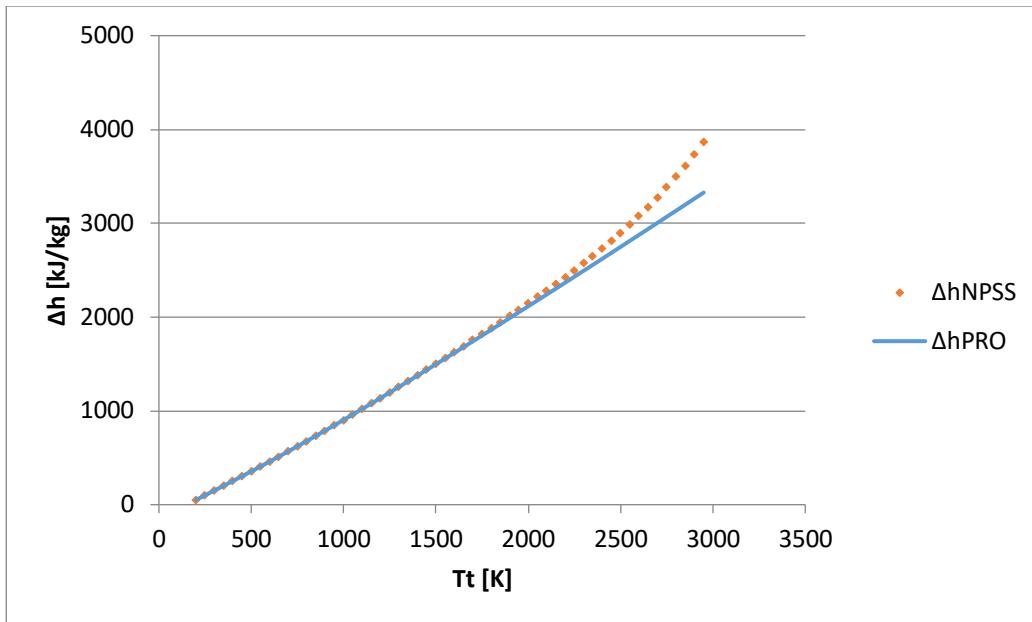
Έχοντας εξασφαλίσει στην προηγούμενη παράγραφο ότι οι χάρτες επιδόσεων που θα χρησιμοποιήσουν τα προγράμματα στις προσομοιώσεις είναι ίδιοι, γνωρίζουμε ότι ένα ακόμη στοιχείο που μπορεί να δημιουργήσει διαφορές είναι το εργαζόμενο μέσο. Επομένως, το επόμενο βήμα είναι να εξασφαλίσουμε ομοίως ότι θα χρησιμοποιηθεί το

Ενσωμάτωση του θερμοδυναμικού μοντέλου του NPSS στο PROOSIS 10.27

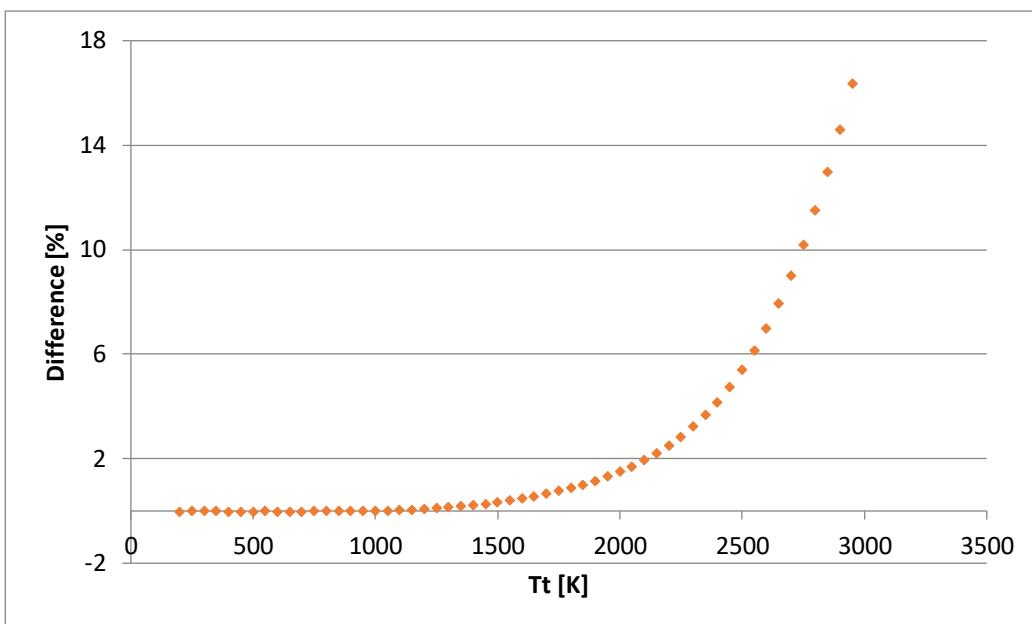
ίδιο θερμοδυναμικό μοντέλο και στις δύο περιπτώσεις. Εφόσον όμως δεν υπάρχει άμεση συσχέτιση μεταξύ των μοντέλων στα δύο προγράμματα, αποφασίστηκε να εισαχθεί το θερμοδυναμικό μοντέλο του NPSS στο PROOSIS. Οπως έχει αναφερθεί προηγουμένως, στο NPSS χρησιμοποιήσαμε το θερμοδυναμικό πακέτο allFuel, το οποίο παρέχει μία πληθώρα επιλογών για καύσιμα. Όπως αναφέρεται στο NPSS thermo guide [27], τα διαθέσιμα καύσιμα είναι τα “JP”, “OLJP”, “H2”, “GAS”, “CH4”, “WBH4” και “UNIV”, ωστόσο μόνο τα JP και CH4 συνιστώνται για χρήση. Το “OLJP” αντιπροσωπεύει το OLD JP και έχει αναλογία H-C 2:1 αλλά με την σύσταση αέρα Keenan και Keyes χωρίς CO2. Το “H2” αντιπροσωπεύει το καθαρό υδρογόνο με την σύσταση αέρα Keenan and Keyes του 1945. Το “CH4” είναι το μοντέλο μεθανίου με ατμοσφαιρικό μοντέλο 1962 και είναι ένα πρότυπο της GE. Το “WBH2” είναι καθαρό υδρογόνο με καθαρό οξυγόνο ως το εργαζόμενο μέσο. Το “UNIV” είναι μία σύνθετη επιλογή για αυθαίρετη σύνθεση ατόμων H-C-O-N-A με το μοντέλο ατμόσφαιρας 1962 και δεν μπορεί να χρησιμοποιηθεί χωρίς επιπλέον πληροφορίες και καθοδήγηση. Τέλος, το “JP” είναι σχεδιασμένο να παράγει τις ιδιότητες των προϊόντων καύσης καυσίμου, νερού και αέρα, όπου ο λόγος H-C του καυσίμου είναι 2:1. Το συγκεκριμένο μοντέλο χρησιμοποιεί το ατμοσφαιρικό μοντέλο 1962 (με 0.03% CO2) και είναι το πρότυπο της GE. Το πακέτο allFuel είναι ένα απλό (γρήγορο) θερμοδυναμικό πακέτο και δεν λαμβάνει υπόψη τις ιδιότητες των άκαυστων μιγμάτων. Ως εκ τούτου δεν κάνει διάκριση μεταξύ των JP4, JP7, JP8, AvGas, κηροζίνη κ.α.. Η αναλογία H-C σε όλα αυτά τα καύσιμα είναι 2:1. Τέλος, αξίζει να αναφερθεί ότι η μηδενική ενθαλπία αναφοράς ορίζεται στους μηδέν Rankine/Kelvin.

Σε αυτό το σημείο θα ήταν χρήσιμο να παρουσιάσουμε τις διαφορές μεταξύ των θερμοδυναμικών πακέτων των δύο προγραμμάτων. Έτσι, για να πραγματοποιήσουμε αυτή τη σύγκριση, επιλέχθηκε το καύσιμο JP από το πακέτο allFuel του NPSS και παράλληλα το καύσιμο JP4_noDiss του PROOSIS.

Απόκτηση των ιδιοτήτων από το πακέτο του PROOSIS ήταν εύκολο, καθώς υπήρχε πρόσβαση στο αντίστοιχο αρχείο με τους ανάλογους πίνακες. Όσον αφορά το NPSS, ένα απλό πρόγραμμα δημιουργήθηκε για εξαγωγή των ιδιοτήτων του εργαζόμενου μέσου. Για διαφορετικό αριθμό FAR και για ένα εύρος διαφορετικών θερμοκρασιών τυπώθηκαν οι αντίστοιχες τιμές για την ενθαλπία, την εντροπία και την ισεντροπική σταθερά γ. Πιο συγκεκριμένα για FAR από 0.0 έως 0.04 με βήμα 0.01 δόθηκαν τιμές για την θερμοκρασία σε εύρος 200 έως 3000 K με βήμα 50 K. Η σύγκριση μεταξύ των ιδιοτήτων των εργαζόμενων μέσων παρουσιάζεται στα κάτωθι διαγράμματα. Να σημειωθεί ότι παρουσιάζονται δεδομένα μόνο για FAR = 0.0, καθώς και στις υπόλοιπες περιπτώσεις οι καμπύλες είναι παρόμοιες. Τέλος, για τις διαφορές ενθαλπίας και εντροπίας ορίστηκε ως θερμοκρασία αναφοράς οι 200 K.



Διάγραμμα 10.1 Σύγκριση ενθαλπικής αύξησης μεταξύ των θερμοδυναμικών μοντέλων του NPSS και του PROOSIS

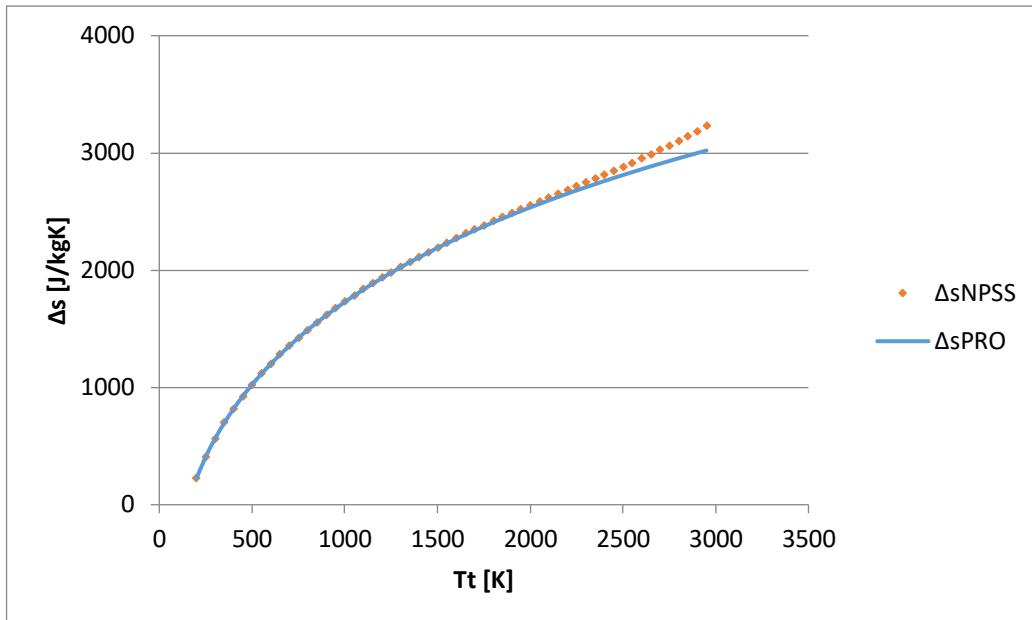


Διάγραμμα 10.2 Ποσοστιαίες διαφορές ενθαλπικής αύξησης μεταξύ των θερμοδυναμικών μοντέλων του NPSS και του PROOSIS

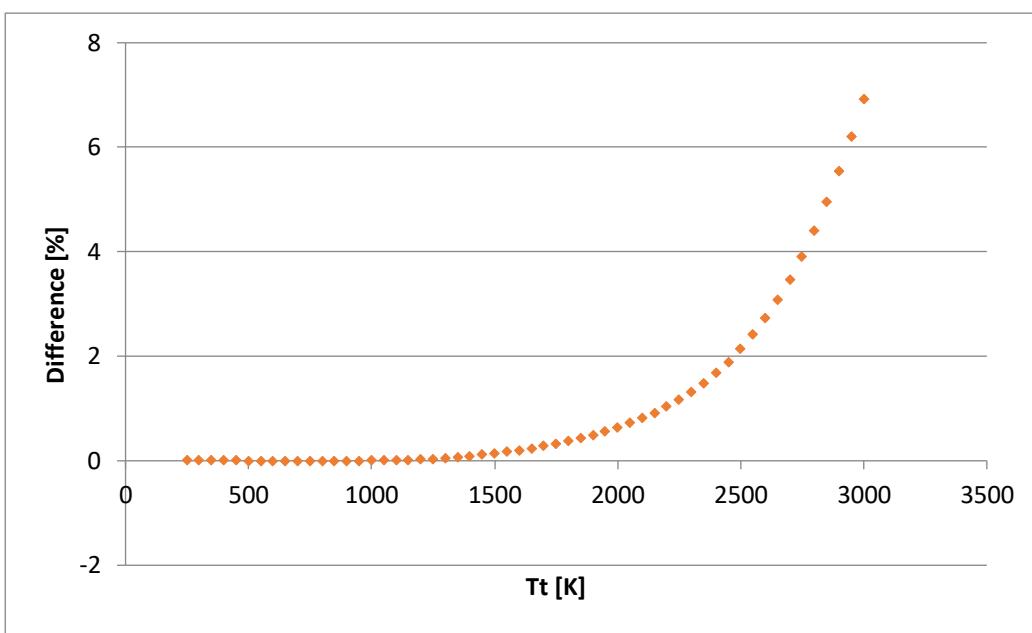
Εύκολα μπορούμε να παρατηρήσουμε ότι καθώς αυξάνεται η θερμοκρασία, ομοίως αυξάνονται οι διαφορές στις ενθαλπίες των δύο μοντέλων. Για θερμοκρασίες μέχρι 1250 K οι διαφορές είναι μικρότερες από 0.1%. Για θερμοκρασίες από 1300-1900 K οι διαφορές είναι μικρότερες από 1% ενώ η μέγιστη διαφορά αγγίζει το 16.3% στους 3000 K. Ωστόσο, για τυπικές θερμοκρασίες λειτουργίας των αεροπορικών κινητήρων 1200-

Ενσωμάτωση του θερμοδυναμικού μοντέλου του NPSS στο PROOSIS 10.29

1600 K, παρατηρείται μία μέση απόκλιση 0.2%. Πιο συγκεκριμένα η μέγιστη διαφορά για FAR = 0.01 είναι 20%, για FAR = 0.02 είναι 23.5% και για FAR = 0.04 είναι 28.4%. Ομοίως και για την συνάρτηση εντροπίας η σύγκριση φαίνεται παρακάτω.

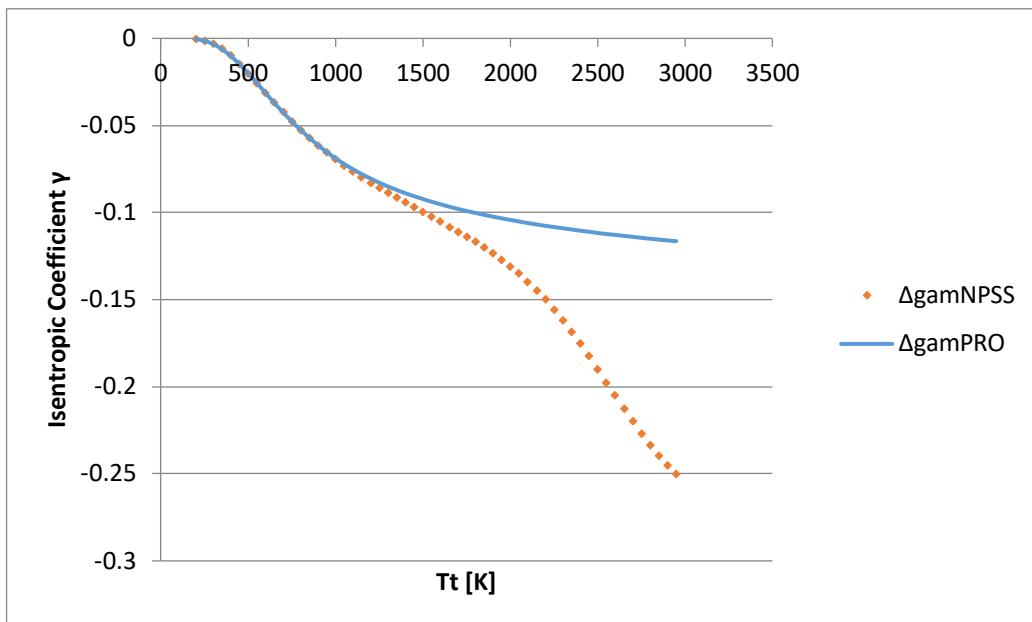


Διάγραμμα 10.3 Σύγκριση εντροπικής αύξησης μεταξύ των θερμοδυναμικών μοντέλων του NPSS και του PROOSIS

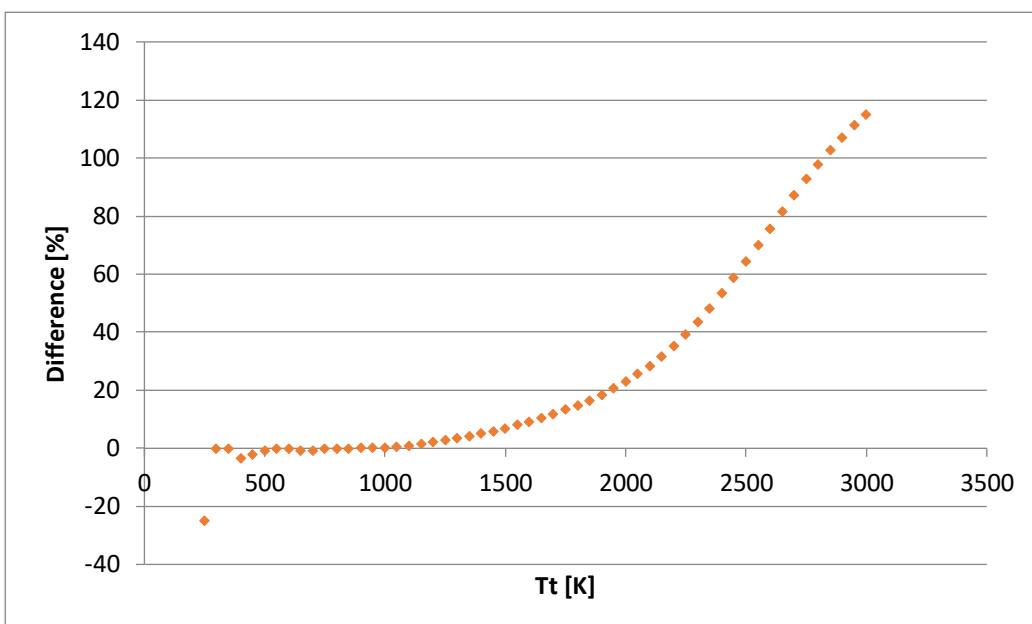


Διάγραμμα 10.4 Ποσοστιαίες διαφορές εντροπικής αύξησης μεταξύ των θερμοδυναμικών μοντέλων του NPSS και του PROOSIS

Για θερμοκρασίες μέχρι 2150 K οι διαφορές στην εντροπία είναι μικρότερες από 0.1%, ενώ η μέγιστη διαφορά είναι 6.9% στους 3000 K. Για $FAR = 0.01$ η μέγιστη διαφορά είναι 8%, για $FAR = 0.02$ είναι 8.8% ενώ για $FAR = 0.04$ είναι 10%. Τέλος, στα παρακάτω διαγράμματα παρουσιάζονται οι διαφορές στην ισεντροπική σταθερά γ μεταξύ των προγραμμάτων:



Διάγραμμα 10.5 Σύγκριση ισεντροπικής σταθεράς γ μεταξύ των θερμοδυναμικών μοντέλων του NPSS και του PROOSIS



Διάγραμμα 10.6 Ποσοστιαίς διαφορές ισεντροπικής σταθεράς γ μεταξύ των θερμοδυναμικών μοντέλων του NPSS και του PROOSIS

Να υπενθυμίσουμε ότι όπως και στις προηγούμενες περιπτώσεις, η θερμοκρασία αναφοράς ορίστηκε στους 200 K. Η μέγιστη διαφορά που παρατηρείται για FAR = 0.0 ισούται με 115%, για FAR = 0.01 είναι 108.4%, για FAR = 0.02 είναι 102% και τέλος για FAR = 0.04 ισούται με 97.8%.

Έχοντας δει τις διαφορές μεταξύ των θερμοδυναμικών πακέτων των δύο προγραμμάτων, το επόμενο βήμα είναι να εισάγουμε τα δεδομένα του πακέτου allFuel με καύσιμο το JP του NPSS στο PROOSIS. Με αυτόν τον τρόπο θα εξαλειφθούν τυχόν διαφορές που δημιουργούνται λόγω της διαφοράς στις ιδιότητες του εργαζόμενου μέσου και θα φανερώσει τυχόν άλλες όσον αφορά τους θερμοδυναμικούς υπολογισμούς. Επομένως, για να εισαχθούν οι ιδιότητες του NPSS στο PROOSIS, τροποποιήθηκε το αρχείο καυσίμου JP4_noDiss και εισήχθησαν οι αντίστοιχοι πίνακες ενθαλπίας, εντροπίας και σταθεράς γ.

10.7 Σύγκριση σε επίπεδο συνιστωσών

Στο σημείο αυτό θα παρουσιαστούν οι συγκρίσεις που έγιναν σε επίπεδο συνιστωσών, έτσι ώστε να γίνει μία αναλυτικότερη σύγκριση μεταξύ των λογισμικών. Αξίζει να αναφερθεί, ότι για όλους του υπολογισμούς που έγιναν στο πλαίσιο της σύγκρισης αυτής, έγινε η ίδια αρχικοποίηση και χρησιμοποιήθηκαν ακριβώς τα ίδια δεδομένα, το ίδιο θερμοδυναμικό μοντέλο και οι ίδιοι χάρτες επιδόσεων. Τέλος, συνθήκες τυπικής ημέρας τέθηκαν για όλες τις προσομοιώσεις. Όπως ήταν αναμενόμενο, οι διαφορές που παρατηρήθηκαν είναι ελάχιστες.

10.7.1 Αγωγός εισόδου

Η σύγκριση έγινε σε υψόμετρο 11000m και τα αποτελέσματα της προσομοίωσης παρουσιάζονται στον κάτωθι πίνακα, όπου παρατηρούνται αμελητέες διαφορές:

Πίνακας 10.16 Σύγκριση αποτελεσμάτων αγωγού εισόδου

	PROOSIS Pt2 [kPa]	NPSS Pt2 [kPa]	Διαφορά [%]
MN=0.8	34.161	34.161	8.84E-05
MN=2.0	175.468	175.480	0.007121

10.7.2 Ατμοσφαιρικές συνθήκες

Οι συνθήκες που επιλέχθηκαν για σύγκριση φαίνονται στον παρακάτω πίνακα:

Πίνακας 10.17 Συνθήκες προσομοίωσης

A/A	Υψόμετρο [m]	Αριθμός Mach [-]	Δέλτα θερμοκρασία[K]
1	300	0.2	0
2	9000	0.8	0
3	11500	0.85	0
4	300	0.2	10
5	9000	0.8	10
6	11500	0.85	10

Τα αποτελέσματα των προσομοιώσεων παρουσιάζονται στους παρακάτω πίνακες και παρατηρούνται μικρές διαφορές (<0.02%).

Πίνακας 10.18 Αποτελέσματα του PROOSIS

A/A	Πίεση [kPa]	Θερμοκρασία [K]	Ολική πίεση [kPa]	Ολική θερμοκρασία [K]	Ταχύτητα αέρα [m/s]
1	97.772	286.20	100.537	288.49	67.83
2	30.742	229.65	46.868	259.11	243.12
3	20.916	216.65	33.551	248.04	250.91
4	97.772	296.2	100.537	298.57	69.00
5	30.742	239.65	46.869	270.38	248.35
6	20.916	226.65	33.551	259.47	256.62

Πίνακας 10.19 Αποτελέσματα του NPSS

A/A	Πίεση [kPa]	Θερμοκρασία [K]	Ολική πίεση [kPa]	Ολική θερμοκρασία [K]	Ταχύτητα αέρα [m/s]
1	97.772	286.20	100.538	288.49	67.82
2	30.742	229.65	46.872	259.11	243.12
3	20.916	216.65	33.554	248.04	250.90
4	97.772	296.20	100.537	298.57	68.99
5	30.742	239.65	46.873	270.38	248.35
6	20.916	226.65	33.554	259.47	256.62

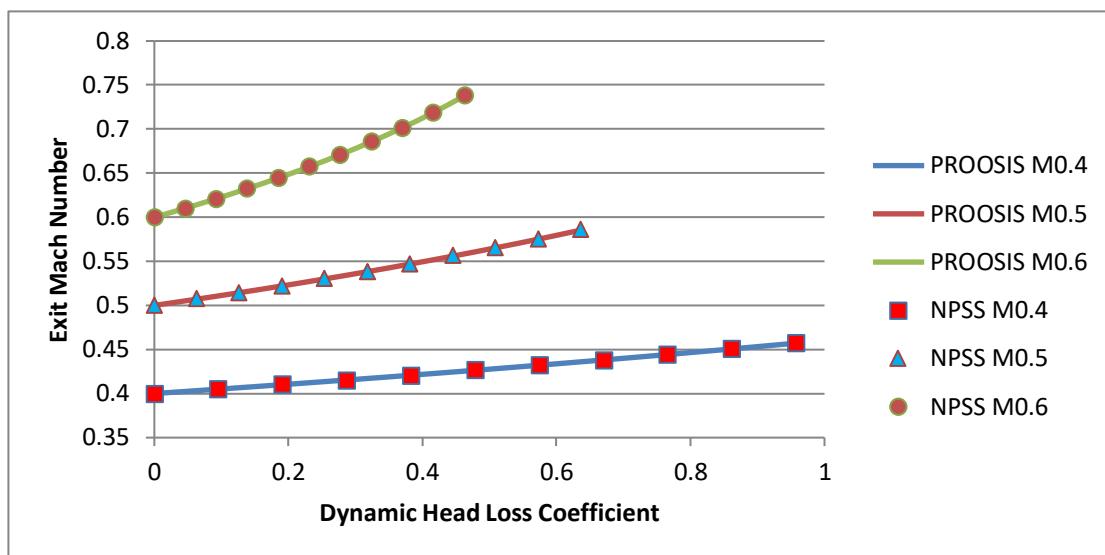
10.7.3 Αγωγός

Οι συνθήκες εισόδου τέθηκαν ίδιες και στα δύο προγράμματα: ολική πίεση εισόδου 100 kPa, ολική θερμοκρασία εισόδου 300 K, παροχή εισόδου 10 kg/s. Έπειτα, έγινε μία παραμετρική μελέτη μεταβάλλοντας τον αριθμό Mach (0.4÷0.6) και την πτώση πίεσης δια του αγωγού (0.0÷0.1).

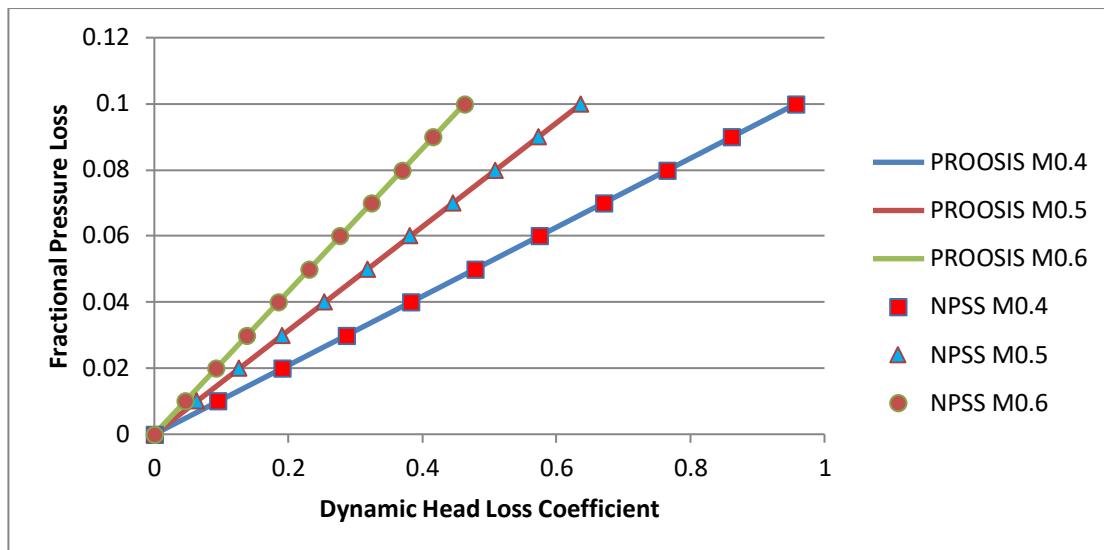
Παρατηρώνται μικρές διαφορές (<0.03%), ενώ τα αποτελέσματα παρουσιάζονται στα παρακάτω γραφήματα, όπου ο αριθμός Mach και η πτώση πίεσης σχεδιάζονται σε συνάρτηση με τον δυναμικό συντελεστή απωλειών dPL, που ορίζεται ως:

$$dPL = \frac{Pt_1 - Pt_2}{Pt_1 - Ps_1}$$

Εξίσωση 10.1 Δυναμικός συντελεστής απωλειών



Διάγραμμα 10.7 Αριθμός Mach εξόδου σε συνάρτηση με τον δυναμικό συντελεστή απωλειών



Διάγραμμα 10.8 Πτώση πίεσης σε συνάρτηση με τον δυναμικό συντελεστή απωλειών

10.7.4 Συμπιεστής

Το ίδιο σημείο σχεδίασης ορίστηκε και στα δύο προγράμματα, ενώ η γραμμή λειτουργίας δημιουργήθηκε με μεταβολή του λόγου πίεσης και των μηχανικών στροφών.

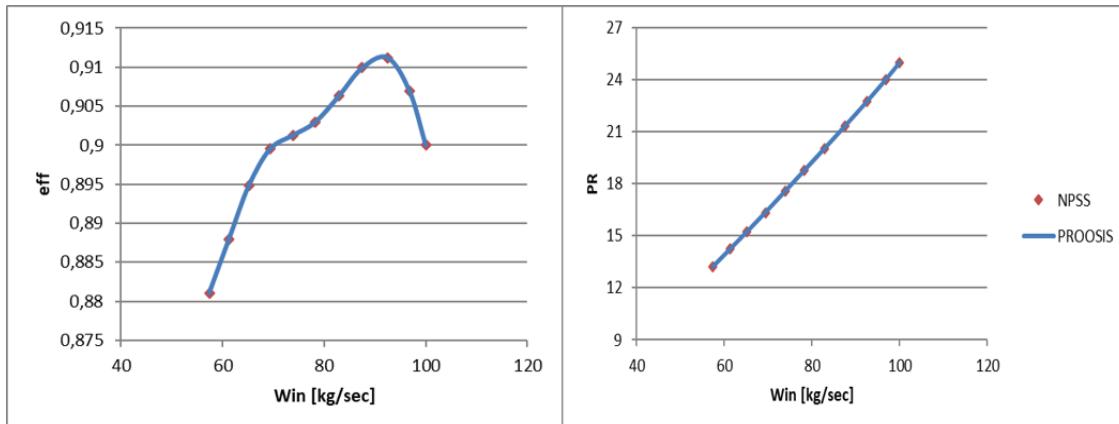
Σε αυτό το σημείο πρέπει να αναφερθεί ότι για να προσομοιωθεί το στοιχείο του συμπιεστή στο NPSS χρειάζεται να προσθέσουμε δικές μας ανεξάρτητες και εξαρτημένες μεταβλητές, αποφεύγοντας την χρήση της εντολής *autoSolverSetup*. Το πρώτο ζεύγος μεταβλητών που προσθέτουμε solver είναι το ίδιο με το προεπιλεγμένο, όταν εκτελείται η εντολή *autoSolverSetup* (δηλαδή το *RlineMap* τίθεται ως ανεξάρτητη μεταβλητή ενώ η ανηγμένη παροχή *Wc* ως εξαρτημένη ώστε να τηρείται η αρχή της συνέχειας). Έπειτα, ορίζεται μία εξαρτημένη μεταβλητή ώστε να επιτυγχάνεται συγκεκριμένη απαίτηση ισχύος και ανεξάρτητη που θα μεταβάλλει την παροχή αέρα. Έπειτα, ορίζεται μία παραμετρική ανάλυση σε λειτουργία off-design με μεταβολή της απαίτησης ισχύος και των μηχανικών στροφών, τα οποία λαμβάνουν τις αντίστοιχες τιμές με το PROOSIS.

Πίνακας 10.20 Δεδομένα εισόδου στον συμπιεστή

Ολική πίεση εισόδου [Pa]	101325
Ολική θερμοκρασία εισόδου [K]	288.15
Παροχή εισόδου [kg/s]	100
Λόγος πίεσης	25
Βαθμός απόδοσης	0.9
Μηχανικές στροφές στο σημείο σχεδίασης [grpm]	10000
Παράμετρος ΒΕΤΑ στο design point	0.33

Παράμετρος R-line στο design point	2.0
------------------------------------	-----

Όπως ήταν αναμενόμενο, διαφορές μικρότερες του **0.03%** παρατηρούνται, ενώ τα αποτελέσματα παρουσιάζονται στα παρακάτω διαγράμματα:



Διάγραμμα 10.9 Επιδόσεις συμπιεστή στο PROOSIS και στο NPSS

Ανεμιστήρας

Δεν πραγματοποιήθηκε κάποια σύγκριση μεταξύ των συνιστωσών του ανεμιστήρα μεταξύ των προγραμμάτων, καθώς το στοιχείο που χρησιμοποιεί το NPSS για να προσομοιώσει τον ανεμιστήρα είναι το ίδιο με αυτό που προσομοιώνει τον συμπιεστή (compressor element). Όπως αναφέρθηκε προηγουμένως, η διαφοροποίηση γίνεται με την χρήση διαφορετικού χάρτη. Έχοντας ωστόσο διασφαλίσει ότι οι χάρτες που χρησιμοποιούν τα προγράμματα είναι πανομοιότυποι, παρόμοιες διαφορές με το στοιχείο του συμπιεστή αναμένονται και για τον ανεμιστήρα.

Μία βασική διαφορά που πρέπει να τονισθεί, είναι ότι η συνιστώσα του ανεμιστήρα στο PROOSIS μπορεί να χωρίσει την ροή σε δύο ρεύματα, να εφαρμόσει διαφορετικό λόγο πίεσης σε κάθε ρεύμα και συγχρόνως να χρησιμοποιήσει διαφορετικό χάρτη σε κάθε περίπτωση. Αντιθέτως, το στοιχείο του ανεμιστήρα στο NPSS δεν μπορεί να διαιρέσει την ροή, ούτε να εφαρμόσει διαφορετικό λόγο πίεσης (αυτό μπορεί να επιτευχθεί με διαφορετικούς τρόπους μοντελοποίησης που θα αναφερθούν στην συνέχεια). Προκειμένου ωστόσο, να συγκρίνει κάποιος τα αποτελέσματα των εν λόγω συνιστωσών, θα χρειαστεί να χρησιμοποιήσει ένα στοιχείο συμπιεστή με τον αντίστοιχο χάρτη του ανεμιστήρα στο PROOSIS ώστε να υπάρχει πλήρης αντιστοιχία μεταξύ των λογισμικών.

10.7.5 Θάλαμος καύσης

Ίδιες συνθήκες εισόδου τέθηκαν και στα δύο προγράμματα και παρουσιάζονται στον ακόλουθο πίνακα. Έπειτα ορίστηκε μία παραμετρική μελέτη με μεταβολή της παροχής καυσίμου έτσι ώστε να μεταβάλλεται ο λόγος καυσίμου-αέρα (FAR) από 0.01 έως 0.04.

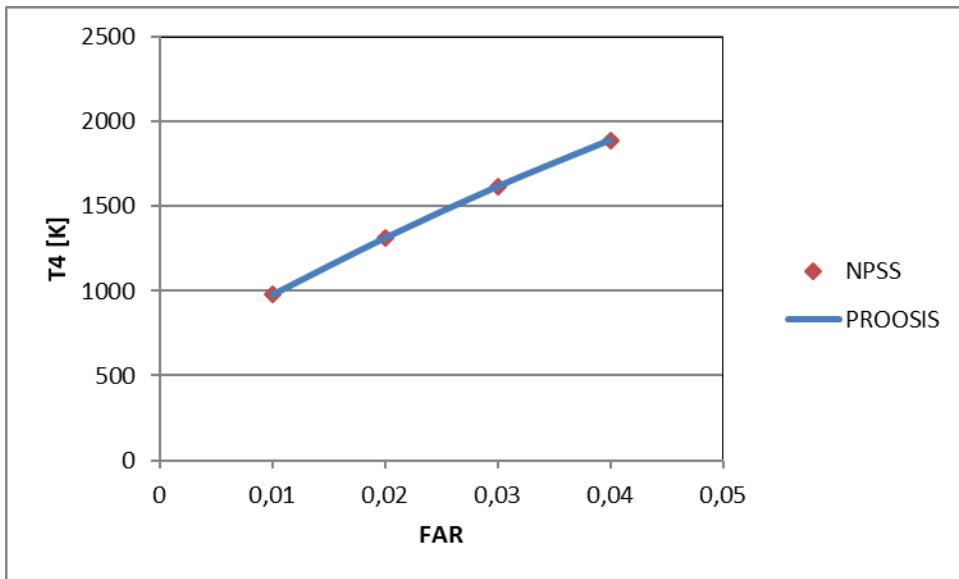
Πίνακας 10.21 Δεδομένα εισόδου στον θάλαμο καύσης

Ολική πίεση εισόδου [Pa]	101325
Ολική θερμοκρασία εισόδου [K]	600
Παροχή εισόδου [kg/s]	100
Σχετική πτώση πίεσης	0.05
Βαθμός απόδοσης	0.9995
Παροχή καυσίμου [kg/s]	1

Πίνακας 10.22 Σύγκριση αποτελεσμάτων του θαλάμου καύσης

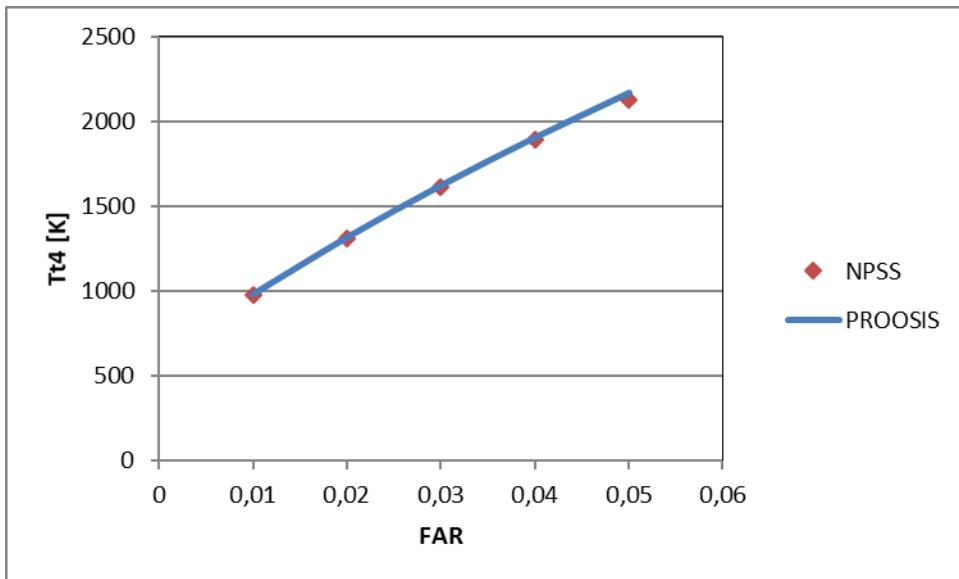
A/A	Λόγος καυσίμου-αέρα	Θερμοκρασία εξόδου στο PROOSIS [K]	Θερμοκρασία εξόδου στο NPSS [K]	Διαφορά [%]
1	0.01	977.44	977.49	-0.024
2	0.02	1312.80	1312.81	-0.096
3	0.03	1615.94	1615.73	-0.322
4	0.04	1889.69	1889.60	-0.754

Οι παραπάνω διαφορές παρουσιάζονται ακολούθως με την μορφή γραφήματος, ενώ η μέγιστη διαφορά που παρατηρείται είναι **0.012%**:



Διάγραμμα 10.10 Θερμοκρασία εξόδου του θαλάμου καύσης σε συνάρτηση με τον λόγο καυσίμου-αέρα

Σε αυτό το σημείο ωστόσο, χρήσιμο θα ήταν να παρουσιάσουμε τις διαφορές που υπάρχουν μεταξύ του θερμοδυναμικού πακέτου του NPSS και της επιλογής JP4 του PROOSIS στο στοιχείο του θαλάμου καύσης. Όπως μπορεί να παρατηρηθεί στο παρακάτω διάγραμμα, οι διαφορές αυξάνονται καθώς κινούμαστε σε υψηλότερες θερμοκρασίες. Όπως φάνηκε από την σύγκριση των πακέτων σε προηγούμενη παράγραφο, η μέγιστη διαφορά που παρατηρείται εδώ είναι για $FAR = 0.05$ και ισούται με 1.8%. Όμως, λόγος καυσίμου-αέρα 0.05 είναι μία εξεζητημένη τιμή που χρησιμοποιήθηκε για να τονίσουμε τις διαφορές των πακέτων, ενώ για τις συνήθεις συνθήκες λειτουργίας των αεροπορικών κινητήρων, δηλαδή για FAR στην περιοχή του 0.02, οι διαφορές είναι της τάξης του 0.2%.



Διάγραμμα 10.11 Θερμοκρασία εξόδου του θαλάμου καύσης σε συνάρτηση με τον λόγο καυσίμου-αέρα μεταξύ του allFuel θερμοδυναμικού πακέτου στο NPSS και του JP4 στο PROOSIS

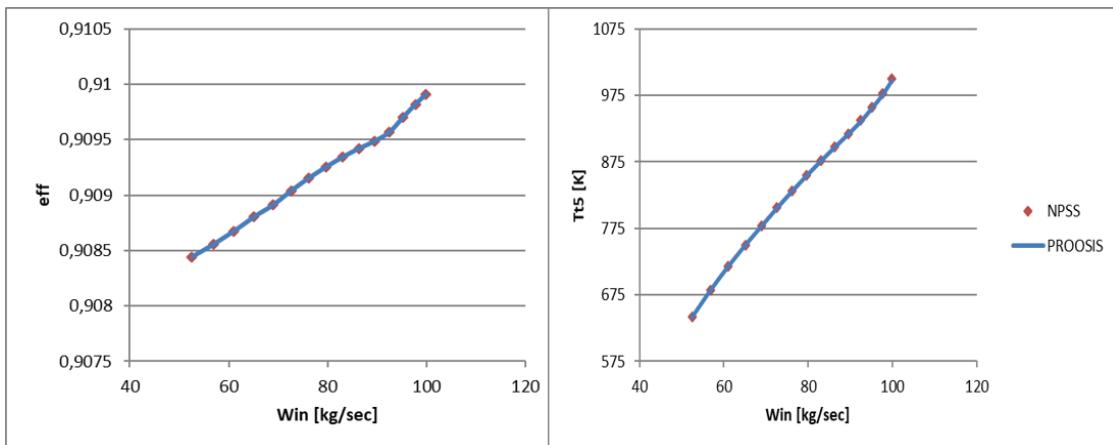
10.7.6 Στρόβιλος

Στην περίπτωση του στροβίλου, επιλέξαμε να συγκρίνουμε τα αποτελέσματα του στροβίλου από έναν κινητήρα Turbojet στο NPSS με ένα μοντέλο στροβίλου στο PROOSIS. Όπως σε όλες τις περιπτώσεις, έτσι και εδώ ορίστηκε το ίδιο σημείο σχεδίασης και οι ίδιες συνθήκες εισόδου μεταξύ των προγραμμάτων, όπως παρουσιάζεται στον ακόλουθο πίνακα.

Πίνακας 10.23 Δεδομένα εισόδου στον στρόβιλο

Ολική πίεση εισόδου [Pa]	742.387
Ολική θερμοκρασία εισόδου [K]	1350.785
Παροχή εισόδου [kg/s]	102
FAR εισόδου	0.02
Βαθμός απόδοσης	0.91
Μηχανικές στροφές [rpm]	10000
Λόγος πίεσης	3.766
Παράμετρος ZETA στο design point	0.74
Παράμετρος PR-line στο design point	4.96

Όπως ήταν αναμενόμενο, οι διαφορές είναι πολύ μικρές και μάλιστα η μέγιστη διαφορά που παρατηρείται είναι **0.013%** για την ολική θερμοκρασία εξόδου ($Tt5$).



Διάγραμμα 10.12 Επιδόσεις στροβίλου στο PROOSIS και στο NPSS

10.7.7 Ακροφύσιο

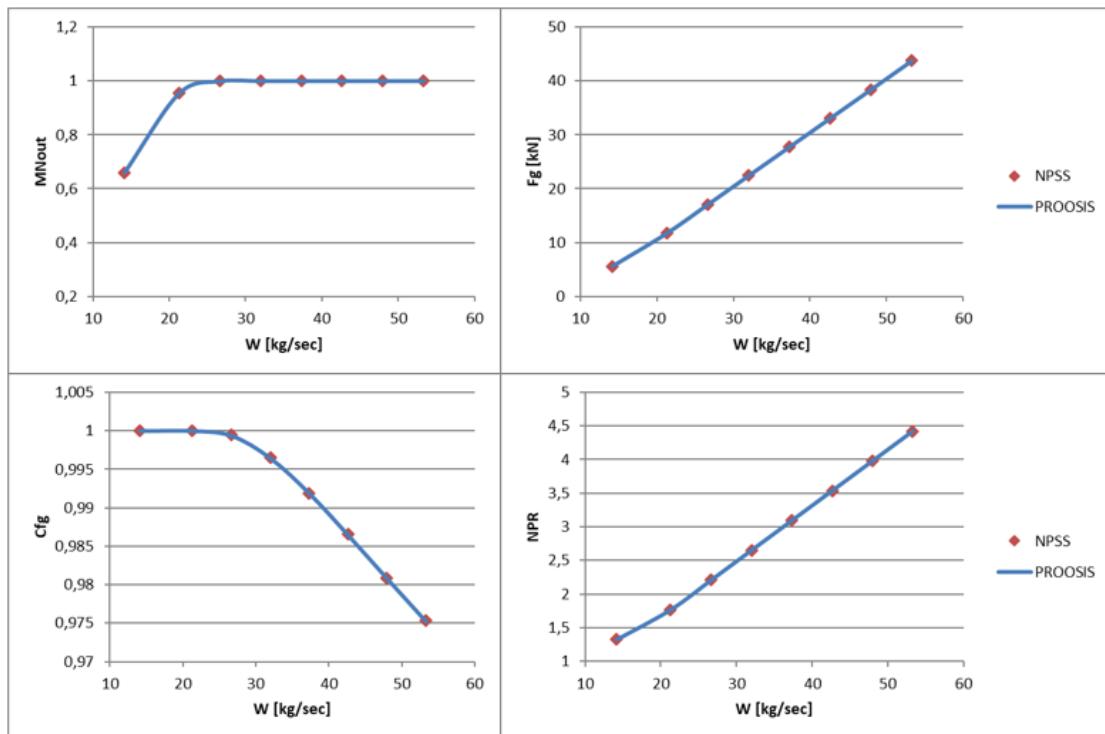
Στη συνιστώσα του ακροφυσίου παρατηρούνται αρκετές διαφορές μεταξύ των δύο λογισμικών. Στο κάθε πρόγραμμα υπάρχουν διαφορετικές μεταβλητές και συντελεστές που αρχικοποιούν και ορίζουν το ακροφύσιο.

Πιο συγκεκριμένα, οι μεταβλητές που απαιτούνται για την αρχικοποίηση του ακροφυσίου στο PROOSIS είναι η ολική πίεση και θερμοκρασία εισόδου, η διατομή εξόδου, η παροχή εισόδου και η ατμοσφαιρική πίεση. Αντιθέτως, το NPSS απαιτεί την ολική πίεση και θερμοκρασία εισόδου, την παροχή εισόδου και την πίεση εξόδου, ώστε να υπολογιστεί η διατομή εξόδου. Η πίεση εξόδου μπορεί να καθορίζεται απευθείας από τον χρήστη ή να ορίζεται ίση με την περιβαλλοντική. Επομένως, για να γίνει σύγκριση των προγραμμάτων μεταξύ ίδιων συνθηκών, προστέθηκαν στον NPSS επιπλέον μία ανεξάρτητη (παροχή εισόδου) και μία εξαρτημένη μεταβλητή (διατομή εξόδου) στον solver, ώστε να υπολογίζεται η παροχή εισόδου έτσι ώστε να επιτευχθεί μία δεδομένη διατομή εξόδου.

Μία παραμετρική ανάλυση πραγματοποιήθηκε, μεταβάλλοντας την ολική πίεση εισόδου. Τα δεδομένα εισόδου φαίνονται στον κάτωθι πίνακα, ενώ οι διαφορές είναι μικρότερες από 0.02%.

Πίνακας 10.24 Δεδομένα εισόδου στο ακροφύσιο

Θερμοκρασία εισόδου [K]	1000
Πίεση εισόδου [kPa]	30
Περιβαλλοντική πίεση [kPa]	22.63
Διατομή εξόδου [m^2]	0.4231



Διάγραμμα 10.13 Επιδόσεις ακροφυσίου στο PROOSIS και στο NPSS

Σύγκριση συντελεστών ακροφυσίων

Εκτός από την διαφορά στον τρόπο υπολογισμού των ακροφυσίων μεταξύ των δύο προγραμμάτων, υπάρχουν επιπλέον διαφορές και στους συντελεστές που χρησιμοποιεί η κάθε συνιστώσα ακροφυσίου. Στον παρακάτω πίνακα παραθέτονται οι συντελεστές που χρησιμοποιούνται σε κάθε πρόγραμμα:

Πίνακας 10.25 Συντελεστές ακροφυσίων

PROOSIS	NPSS
C_d	C_{dTh}
C_x	C_v
C_{fg}	C_{fg}
	C_{ang}
	C_{qua}
	$C_{mixcorr}$

Ακροφύσιο του PROOSIS

- $C_d =$ Συντελεστής εκβολής
 - Χρησιμοποιείται στον υπολογισμό της ενεργούς διατομής εξόδου, $A_{exiteff}$

- $C_x = \Sigma$ Συντελεστής ώσης
 - Χρησιμοποιείται στον υπολογισμό της μικτής ώσης, F_g
- $C_{fg} = \Sigma$ Συντελεστής ιδανικής ώσης
 - Υπολογίζεται από το πρόγραμμα

Ακροφύσιο του NPSS

- $C_{dTh} = \Sigma$ Συντελεστής εκβολής λαιμού
 - Χρησιμοποιείται στον υπολογισμό της ενεργούς διατομής στον λαιμό
 - Μπορεί να προέρχεται από subelement
 - Μπορεί να υπολογιστεί από το πρόγραμμα
- $C_v = \Sigma$ Συντελεστής ταχύτητας εξόδου
 - Χρησιμοποιείται στον υπολογισμό της μικτής ώσης
 - Μπορεί να προέρχεται από subelement
 - Μπορεί να υπολογιστεί από το πρόγραμμα
- $C_{fg} = \Sigma$ Συντελεστής μικτής ώσης
 - Χρησιμοποιείται στον υπολογισμό της μικτής ώσης, F_g
 - Μπορεί να προέρχεται από subelement
 - Μπορεί να υπολογιστεί από το πρόγραμμα
- $C_{ang} = \Sigma$ Συντελεστής γωνίας εξόδου
 - Χρησιμοποιείται στον υπολογισμό της μικτής ώσης και της V_{actual}
 - Χρησιμοποιείται μόνο όταν χρησιμοποιείται το C_v
- $C_{qua} = \Theta$ Θερμικός συντελεστής εκτόνωσης στον λαιμό
 - Χρησιμοποιείται στον υπολογισμό της ενεργούς διατομής στον λαιμό
- $C_{mixcorr} = \Delta$ Διόρθωση στην ώση λόγω μερικής ανάμιξης της ροής
 - Χρησιμοποιείται στον υπολογισμό της μικτής ώσης και της V_{actual}
 - Χρησιμοποιείται μόνο όταν χρησιμοποιείται το C_v

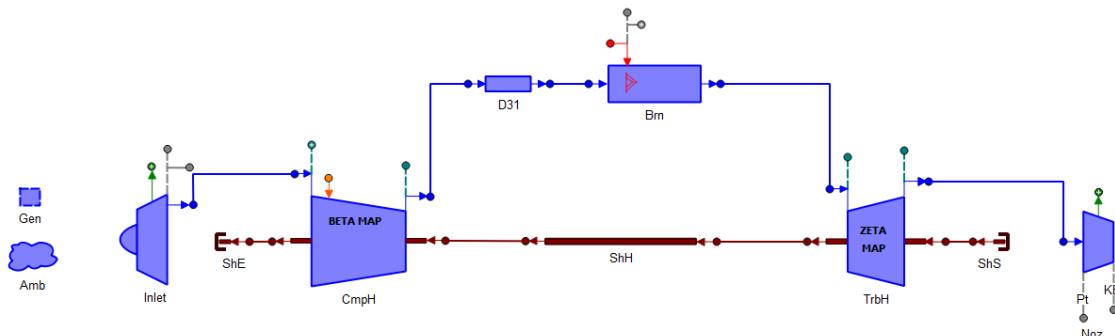
Από τα παραπάνω μπορούμε να πούμε ότι οι τρεις πρώτοι συντελεστές του κάθε λογισμικού που παρουσιάζονται στον [Πίνακας 10.25](#) είναι ταυτόσημοι. Ωστόσο, το NPSS δίνει την δυνατότητα στους χρήστες να εφαρμόσουν επιπλέον συντελεστές, επιτρέποντας την μελέτη δευτερευόντων φαινομένων στο ακροφύσιο, καθώς επίσης και διενέργεια διαφορετικής ακρίβειας υπολογισμών..

10.8 Σύγκριση σε επίπεδο κινητήρα

Όπως αναφέρθηκε και στην προηγούμενη παράγραφο, σε όλες τις προσομοιώσεις ιδιαίτερο ενδιαφέρον δόθηκε έτσι ώστε τα δεδομένα εισόδου να είναι πανομοιότυπα μεταξύ των δύο λογισμικών.

10.8.1 Turbojet

Το Turbojet είναι ένας απλός τύπος κινητήρα και μπορεί εύκολα να μοντελοποιηθεί συνδέοντας κατάλληλα όλες τις συνιστώσες που έχουν αναφερθεί στο παραπάνω κεφάλαιο.



Εικόνα 10.8 Σχηματικό κινητήρα Turbojet στο PROOSIS

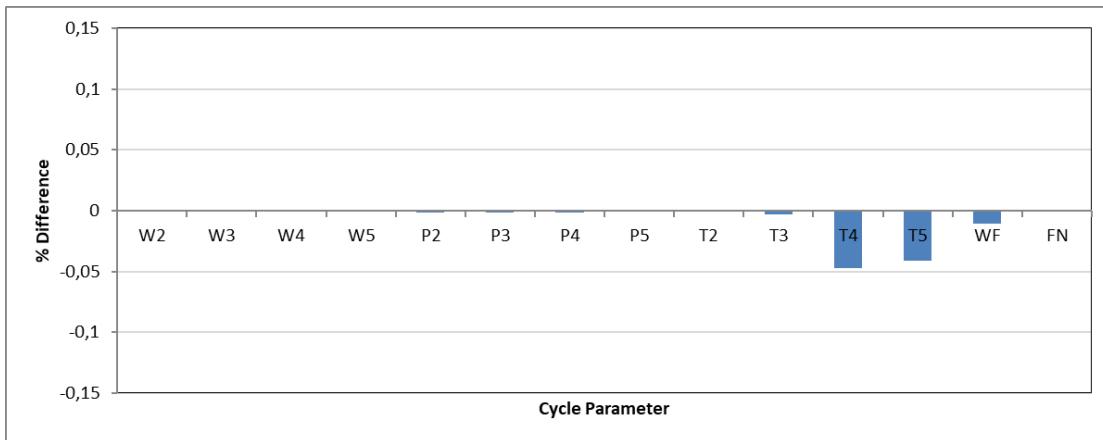
Ίδιες συνθήκες λειτουργίας τέθηκαν και στα δύο προγράμματα, ενώ η καμπύλη λειτουργίας κατασκευάστηκε μεταβάλλοντας την παροχή καυσίμου.

Πίνακας 10.26 Δεδομένα σημείου σχεδίασης για κινητήρα Turbojet

Έψος πτήσης [ft]	38000
Αριθμός Mach	0.8
Παροχή εισόδου [kg/s]	100
Πτώση πίεσης αγωγού εισόδου PqP	0.995
Λόγος πίεσης συμπιεστή	25
Βαθμός απόδοσης συμπιεστή	0.9
Πτώση πίεσης αγωγού $dPqP$	0.002
Παροχή καυσίμου [kg/s]	2
Βαθμός απόδοσης θαλάμου καύσης	0.98
Πτώση πίεσης θαλάμου καύσης $dPqP$	0.05

Βαθμός απόδοσης στροβίλου	0.91
Μηχανικές στροφές ατράκτου [rpm]	10000
Παράμετρος ΖΕΤΑ	0.74
Παράμετρος PR-line	4.96
Απαιτούμενη ώση [kN]	70.368

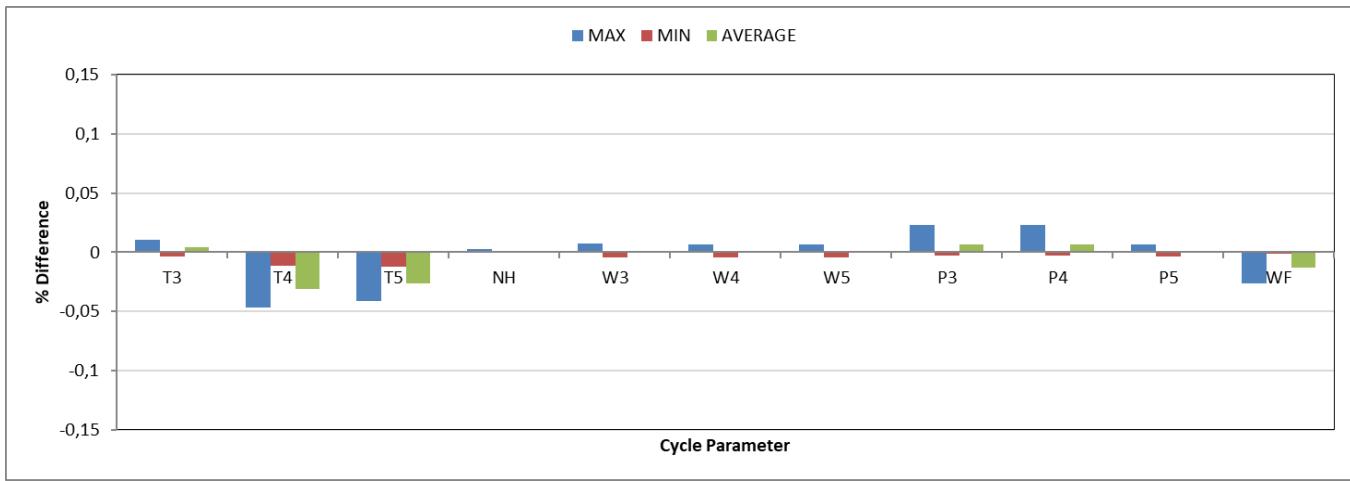
Στο παρακάτω γράφημα παρουσιάζονται οι διαφορές που παρατηρήθηκαν στο σημείο σχεδίασης:



Διάγραμμα 10.14 Ποσοστιαίες διαφορές κινητήρα Turbojet στο σημείο σχεδίασης

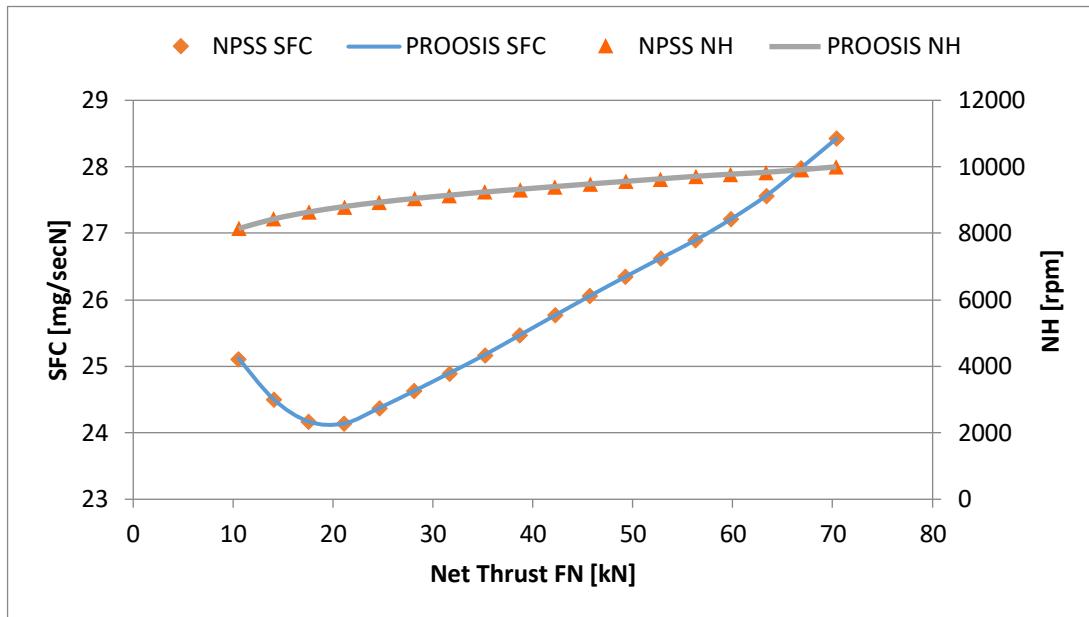
Οι διαφορές στο σημείο σχεδίασης είναι μικρότερες από **0.05%** σε όλες τις συνιστώσες. Η μέγιστη διαφορά εντοπίζεται στην θερμοκρασία εξόδου από τον θάλαμο καύσης, ίση με 0.47% και επηρεάζει σε ελάχιστο βαθμό τις επιδόσεις του στροβίλου.

Ακολούθως, η γραμμή λειτουργίας σε κατάσταση εκτός του σημείου λειτουργίας, αποτελείται από 18 σημεία με μεταβολή της απαιτούμενης ώσης ($100 \div 15\% \text{ της ώσης στο σημείο σχεδίασης}$ κατά διαστήματα 5%). Οι διαφορές είναι και εδώ μικρότερες από **0.05%** σε όλο το εύρος λειτουργίας ενώ η μέγιστη διαφορά που παρατηρείται είναι ομοίως **0.047%** για την ολική θερμοκρασία εξόδου από τον θάλαμο καύσης.



Διάγραμμα 10.15 Ποσοστιαίες διαφορές κινητήρα Turbojet εκτός του σημείου σχεδίασης

Στο ακόλουθο διάγραμμα παρουσιάζονται η ειδική κατανάλωση καυσίμου και οι μηχανικές στροφές σε συνάρτηση με την ώση, ενώ μέγιστη διαφορά εντοπίζεται στην ειδική κατανάλωση, ίση με **0.02%**.



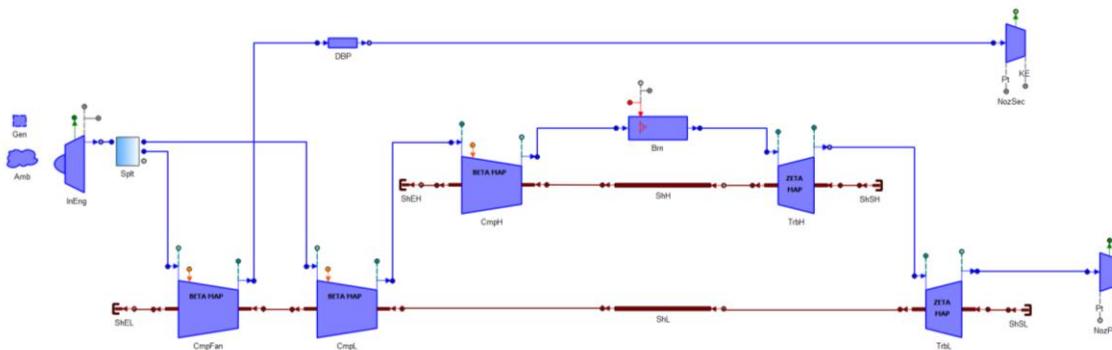
Διάγραμμα 10.16 Διαφορές ειδικής κατανάλωσης και μηχανικών στροφών σε κινητήρα Turbojet μεταξύ του NPSS και του PROOSIS

Είναι σημαντικό να αναφερθεί ότι οι μεγαλύτερες διαφορές που παρατηρούνται στον κινητήρα Turbojet, σε σχέση με τις διαφορές που παρατηρήθηκαν σε επίπεδο συνιστωσών, οφείλονται στις παραμέτρους των χαρτών όπως αναφέρθηκε στην παράγραφο 10.5.3. Οι διαφορές στα εύρη των εν λόγω παραμέτρων δημιουργούν διαφορές κατά την τοποθέτηση των σημείων λειτουργίας επάνω στους χάρτες, με αποτέλεσμα να δημιουργούνται διαφορές σε επίπεδο μηχανής. Ωστόσο οι διαφορές αυτές είναι πολύ μικρές σε όλες τις περιπτώσεις, μικρότερες από 0.05%.

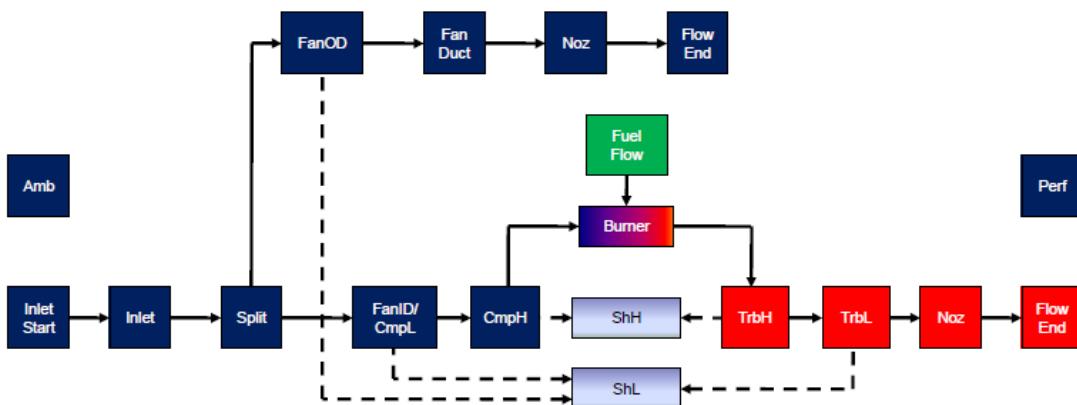
10.8.2 Turbofan

Όπως έχει αναφερθεί προηγουμένως, η συνιστώσα του ανεμιστήρα στο NPSS δεν μπορεί να χωρίσει την ροή σε δύο ρεύματα, ούτε να εφαρμόσει διαφορετικό λόγο πίεσης στο κάθε ρεύμα όπως γίνεται στην αντίστοιχη συνιστώσα του PROOSIS. Αντιθέτως χρησιμοποίει ένα στοιχείο Splitter για να χωρίσει τη ροή σε δύο ρεύματα. Η εφαρμογή διαφορετικού λόγου πίεσης σε κάθε ρεύμα μπορεί να υλοποιηθεί με διαφορετικούς τρόπους μοντελοποίησης. Ένας τρόπος είναι να χρησιμοποιηθεί μία επιπλέον συνιστώσα ανεμιστήρα και ο δεύτερος τρόπος είναι να ενσωματωθεί ο λόγος πίεσης του ανεμιστήρα του κυρίως ρεύματος στον συμπιεστή χαμηλής (booster).

Στο πλαίσιο αυτής της διπλωματικής, ακολουθήθηκε η δεύτερη σχεδιαστική φιλοσοφία. Ωστόσο, καθώς στο PROOSIS δεν υπάρχει στοιχείο διαχωριστή της ροής στις βιβλιοθήκες του, απαιτήθηκε η δημιουργία ενός. Το σχηματικό του εν λόγω κινητήρα παρουσιάζεται στην κάτωθι εικόνα:



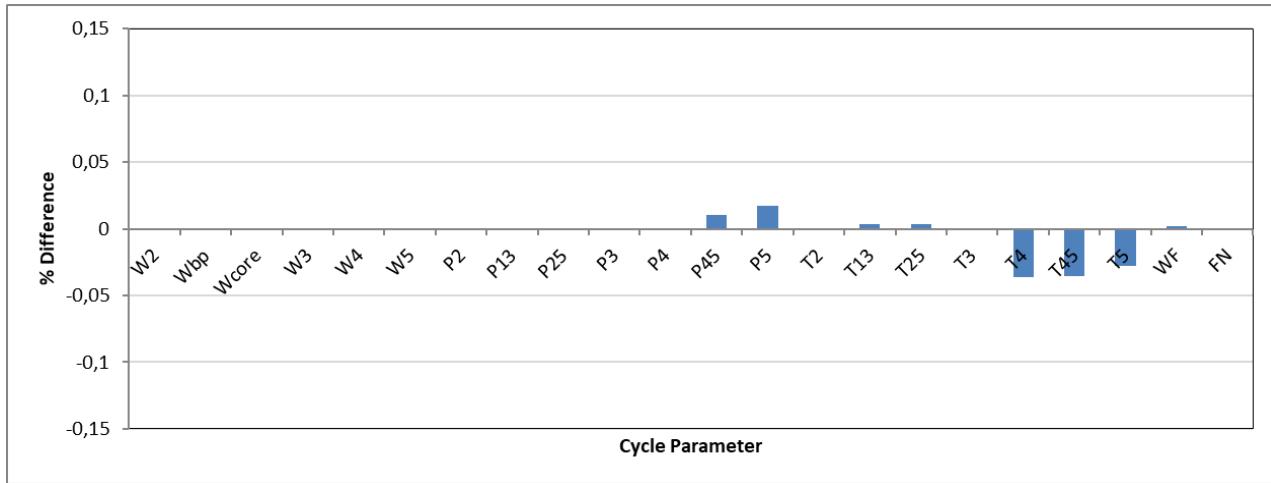
Εικόνα 10.9 Σχηματικό κινητήρα Turbofan στο PROOSIS



Εικόνα 10.10 Σχηματικό κινητήρα Turbofan στο NPSS

Πίνακας 10.27 Δεδομένα σημείου σχεδίασης για κινητήρα Turbofan

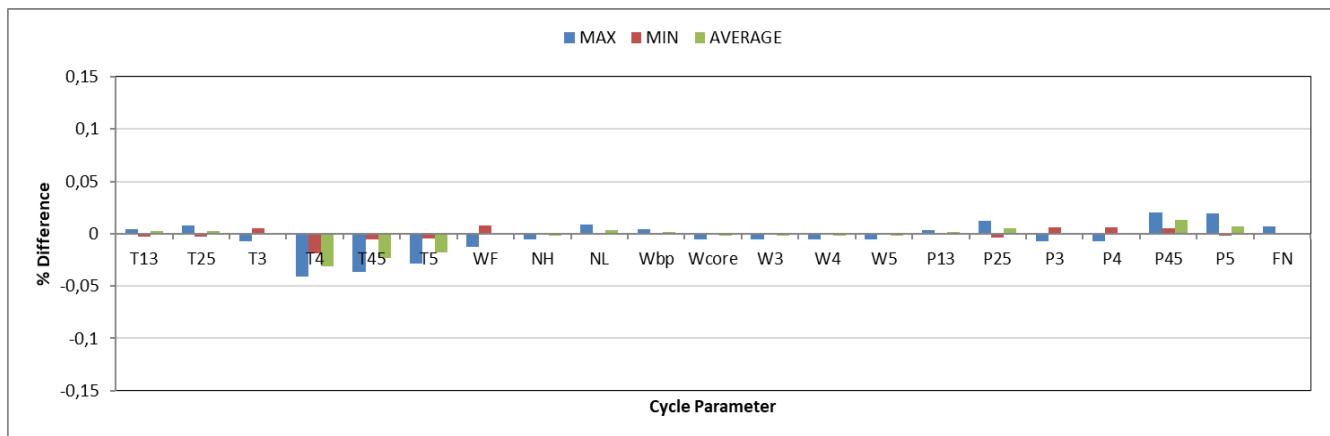
Υψος πτήσης [m]	0
Αριθμός Mach	0
Πτώση πίεσης αγωγού εισόδου ΔP_{qP}	0.995
Λόγος πίεσης ανεμιστήρα	1.5
Βαθμός απόδοσης ανεμιστήρα	0.85
Λόγος παράκαμψης	5
Πτώση πίεσης αγωγού παράκαμψης ΔP_{qP}	0.015
Λόγος πίεσης συμπιεστή χαμηλής	3
Βαθμός απόδοσης συμπιεστή χαμηλής	0.85
Λόγος πίεσης συμπιεστή υψηλής	10
Βαθμός απόδοσης συμπιεστή υψηλής	0.85
Βαθμός απόδοσης θαλάμου καύσης	0.98
Πτώση πίεσης θαλάμου καύσης ΔP_{qP}	0.05
Βαθμός απόδοσης στροβίλου υψηλής	0.87
Βαθμός απόδοσης στροβίλου χαμηλής	0.87
Μηχανικές στροφές ατράκτου υψηλής [rpm]	10000
Μηχανικές στροφές ατράκτου χαμηλής [rpm]	5000
Απαιτούμενη ώση [kN]	44.482
Παράμετρος BETA ανεμιστήρα	0.473684
Παράμετρος BETA συμπιεστή υψηλής	0.33
Παράμετρος BETA συμπιεστή χαμηλής	0.33
Παράμετρος ZETA στροβίλου υψηλής	0.74
Παράμετρος ZETA στροβίλου χαμηλής	0.59
Παράμετρος R-line ανεμιστήρα	1.789474
Παράμετρος R-line συμπιεστή υψηλής	2.0
Παράμετρος R-line συμπιεστή χαμηλής	2.0
Παράμετρος PR-line στροβίλου υψηλής	4.96
Παράμετρος PR-line στροβίλου χαμηλής	4.36



Διάγραμμα 10.17 Ποσοστιαίες διαφορές κινητήρα Turbofan στο σημείο σχεδίασης

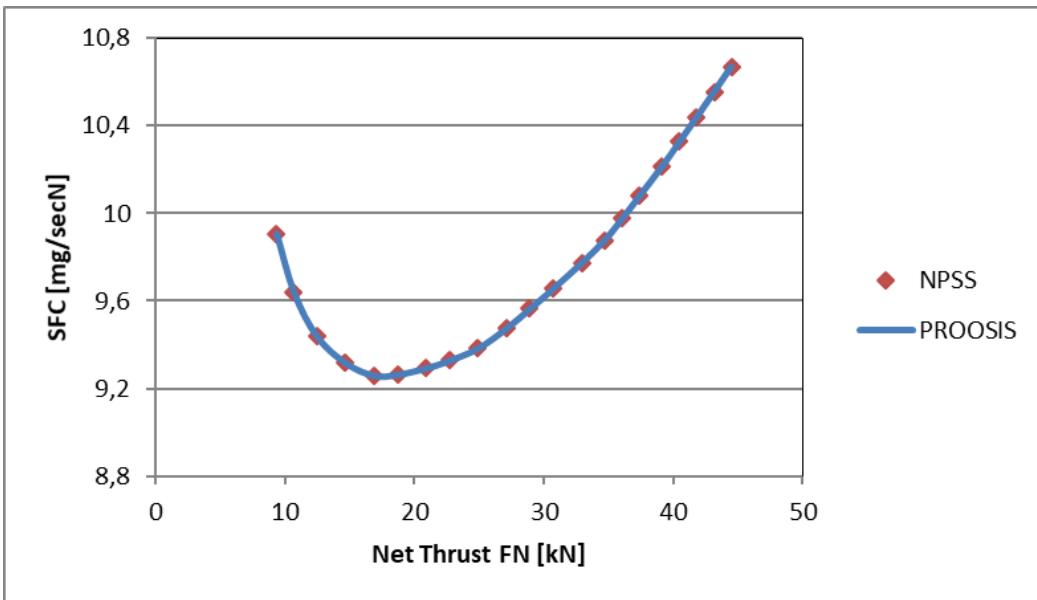
Όπως στην περίπτωση του κινητήρα Turbojet, έτσι και εδώ οι διαφορές είναι μικρότερες από **0.05%** και η μέγιστη διαφορά εντοπίζεται πάλι στην θερμοκρασία εξόδου από τον θάλαμο καύσης, ίση με 0.035%.

Η γραμμή λειτουργίας 21 σημείων προέκυψε με μεταβολή της απαιτούμενης ώσης ($100 \div 21\%$ της ώσης στο σημείο σχεδίασης), ενώ η μέγιστη διαφορά ισούται με 0.041% και παρατηρήθηκε στην ολική θερμοκρασία εξόδου από τον θάλαμο καύσης.

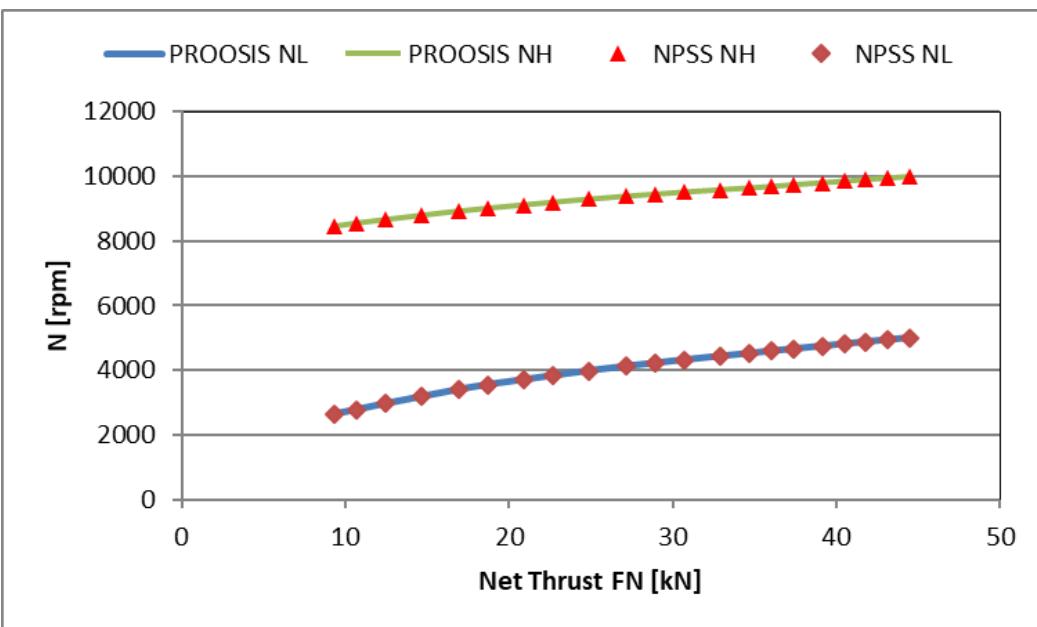


Διάγραμμα 10.18 Ποσοστιαίες διαφορές κινητήρα Turbofan εκτός του σημείου σχεδίασης

Στα ακόλουθα διαγράμματα παρουσιάζονται η ειδική κατανάλωση και οι μηχανικές στροφές σε συνάρτηση με την ώση για τις προσομοιώσεις του NPSS και του PROOSIS. Η μέγιστη διαφορά που παρατηρείται για την ειδική κατανάλωση ισούται με 0.01%, ενώ οι διαφορές στις μηχανικές στροφές είναι μικρότερες από 0.01%.



Διάγραμμα 10.19 Διαφορά ειδικής κατανάλωσης σε κινητήρα Turbofan μεταξύ του NPSS και του PROOSIS

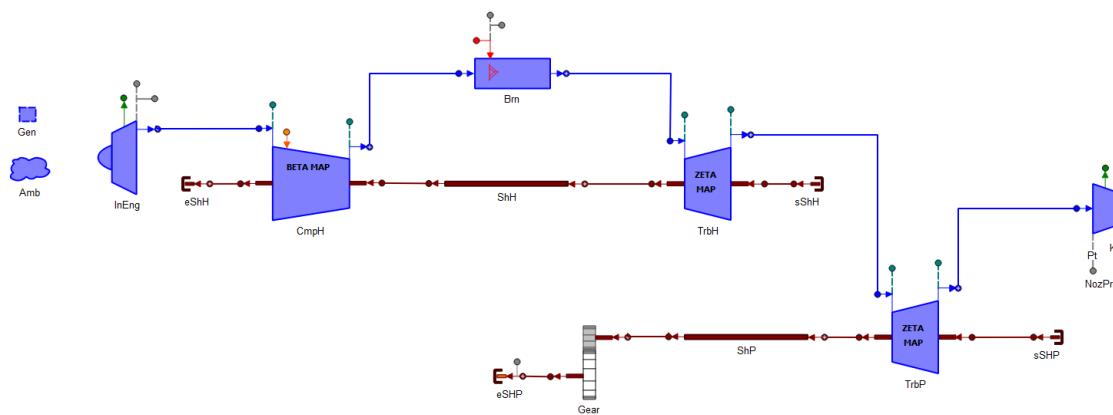


Διάγραμμα 10.20 Διαφορά μηχανικών στροφών σε κινητήρα Turbofan μεταξύ του NPSS και του PROOSIS

Όπως και στη περίπτωση του κινητήρα Turbojet, έτσι και εδώ, παρατηρούνται διαφορές μεταξύ των λογισμικών, έστω και ελάχιστες, οι οποίες μπορούν να αποδοθούν κυρίως σε διαφορές των παραμέτρων των χαρτών, όπως αναφέρθηκε προηγουμένως.

10.8.3 Turboshaft

Είναι αρκετά εύκολο να δημιουργηθεί ένα μοντέλο κινητήρα Turboshaft στο PROOSIS με μόνη προσθήκη ενός κιβωτίου ταχυτήτων και ενός στροβίλου ισχύος. Αντιθέτως, στο NPSS δεν υπάρχει κάποια συνιστώσα στις βιβλιοθήκες που να μπορεί να προσομοιώσει αυτόν τον τύπο κινητήρα και να κάνει τους σχετικούς υπολογισμούς. Για αυτόν τον λόγο γίνεται παρέμβαση σε μία συνιστώσα του NPSS (αφού δίνεται αυτή η δυνατότητα), όπου προστίθενται κατάλληλα οι απαιτούμενοι υπολογισμοί.



Εικόνα 10.11 Σχηματικό κινητήρα Turboshaft στο PROOSIS

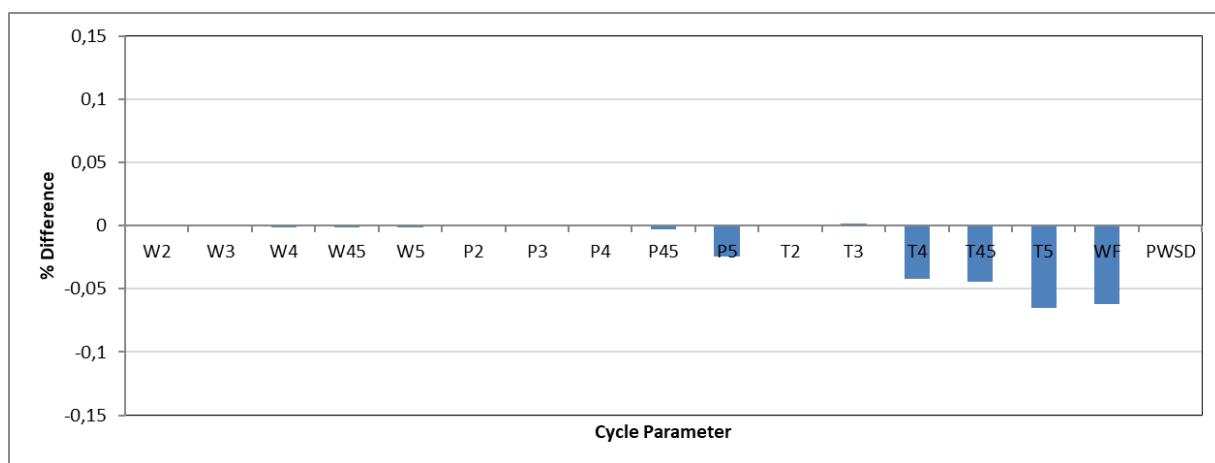
Όπως στις προηγούμενες περιπτώσεις, έτσι και εδώ χρησιμοποιήθηκαν ίδια δεδομένα ώστε να υπάρχει πλήρης αντιστοίχηση μεταξύ των δύο προγραμμάτων.

Πίνακας 10.28 Δεδομένα σημείου σχεδίασης για κινητήρα Turboshaft

Υψόμετρο [m]	0
Αριθμός Mach	0
Παροχή εισόδου [kg/s]	5
Πτώση πίεσης αγωγού εισόδου PqP	0.995
Λόγος πίεσης συμπιεστή	15
Βαθμός απόδοσης συμπιεστή	0.858
Λόγος καυσίμου-αέρα θαλάμου καύσης	0.0192
Βαθμός απόδοσης θαλάμου καύσης	0.995
Πτώση πίεσης θαλάμου καύσης dPqP	0.025
Βαθμός απόδοσης στροβίλου υψηλής	0.913
Βαθμός απόδοσης στροβίλου ισχύος	0.915
Μηχανικές στροφές ατράκτου υψηλής [rpm]	40000

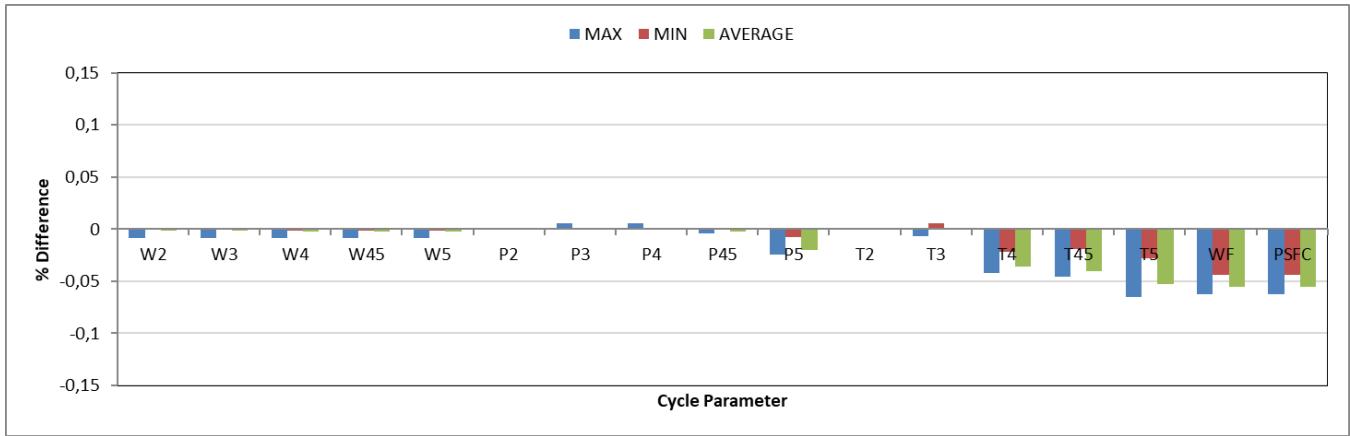
Μηχανικές στροφές ατράκτου ισχύος [rpm]	20000
Ισχύς που εξάγεται από την άτρακτο ισχύος [kW]	1619.386
Λόγος σχέσης του κιβωτίου	0.3
Βαθμός απόδοσης του κιβωτίου	1
Παράμετρος ΒΕΤΑ συμπιεστή υψηλής	0.33
Παράμετρος ΖΕΤΑ στροβίλου υψηλής	0.74
Παράμετρος ΖΕΤΑ στροβίλου ισχύος	0.59
Παράμετρος R-line συμπιεστή υψηλής	2.0
Παράμετρος PR-line στροβίλου υψηλής	4.96
Παράμετρος PR-line στροβίλου ισχύος	4.36

Οι διαφορές που προκύπτουν στο σημείο σχεδίασης παρουσιάζονται στο κάτωθι διάγραμμα και είναι μικρότερες από **0.1%**:



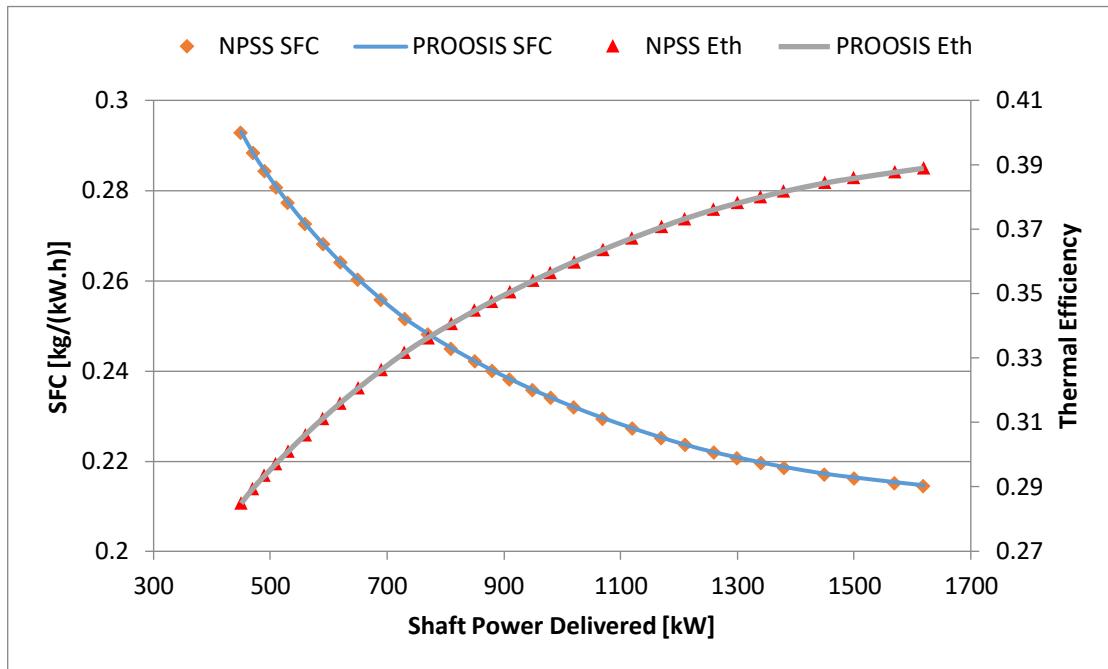
Διάγραμμα 10.21 Ποσοστιαίες διαφορές κινητήρα Turboshaft στο σημείο σχεδίασης

Ακολούθως, παράγθηκε μία γραμμή λειτουργίας 31 σημείων μεταβάλλοντας κάθε φορά την εξαγόμενη ισχύ από την άτρακτο ισχύος ($1619 \div 450$ kW). Η μεγαλύτερη διαφορά παρατηρείται στην θερμοκρασία εξόδου από τον στρόβιλο ισχύος και ισούται με **0.065%**.



Διάγραμμα 10.22 Ποσοστιαίες διαφορές κινητήρα Turboshaft εκτός του σημείου σχεδίασης

Στο επόμενο διάγραμμα παρουσιάζονται η ειδική κατανάλωση και ο θερμικός βαθμός απόδοσης σε συνάρτηση με την εξαγόμενη ισχύ. Η μεγαλύτερη διαφορά για την ειδική κατανάλωση ισούται με **0.06%**, ενώ η μέγιστη διαφορά για τον θερμικό βαθμό απόδοσης είναι επίσης **0.06%**.



Διάγραμμα 10.23 Διαφορές ειδικής κατανάλωσης και θερμικού βαθμού απόδοσης λειτουργίας κινητήρα Turboshaft μεταξύ του NPSS και του PROOSIS

Οπως και στις προηγούμενες περιπτώσεις, έτσι και εδώ οι διαφορές που παρατηρούνται είναι αμελητέες και αποδίδονται κυρίως σε διαφορές στις παραμέτρους των χαρτών.

10.9 Ανακεφαλαίωση και μελλοντική εργασία

Το PROOSIS και το NPSS είναι δύο ισχυρά εμπορικά λογισμικά, που έχουν ένα ευρύ φάσμα εφαρμογών και εξελίσσουν συνεχώς τις δυνατότητές τους. Μέσα από την δουλειά που έγινε στα πλαίσια αυτής της διπλωματικής εργασίας, γνωρίσαμε πώς λειτουργεί το NPSS, καθώς επίσης δημιουργήσαμε μοντέλα, τόσο σε επίπεδο συνιστωσών, όσο και σε επίπεδο μηχανής και συγκρίναμε τα αποτελέσματα προσομοιώσεων με τα αντίστοιχα μοντέλα στο PROOSIS. Από την εν λόγω σύγκριση συμπεραίνουμε ότι η φιλοσοφία μοντελοποίησης μεταξύ των λογισμικών είναι ίδια και οι διαφορές που παρατηρούνται είναι μικρότερες από 0.1% σε όλες τις περιπτώσεις, κάτι που μας οδηγεί βέβαια στο συμπέρασμα ότι όσον αφορά τους θερμοδυναμικούς υπολογισμούς, τα δύο λογισμικά ταυτίζονται. Οι ελάχιστες διαφορές που παρατηρούνται, κυρίως σε επίπεδο μηχανής, μπορούν να αποδοθούν σε διαφορές των παραμέτρων των χαρτών. Όπως έχει αναφερθεί προηγουμένως, λόγω της διαφοράς στο εύρος τιμών των παραμέτρων των χαρτών μεταξύ των προγραμμάτων, οι προκύπτουσες τιμές των παραμέτρων κατά τις προσομοιώσεις αποκλίνουν. Αυτό σημαίνει ότι κατά τον υπολογισμό του κύκλου σε έναν κινητήρα μεταξύ των προγραμμάτων, παράγονται διαφορετικές τιμές των αντίστοιχων παραμέτρων, οι οποίες τοποθετούν τα σημεία λειτουργίας σε ελάχιστα διαφορετικές θέσεις επάνω στους χάρτες. Αυτό με την σειρά του δημιουργεί διαφορές στις επιδόσεις των κινητήρων όπως παρουσιάστηκε ωστόσο οι διαφορές αυτές είναι πολύ μικρές. Ένα ακόμη στοιχείο που δημιουργεί μικρές διαφορές είναι πιθανώς οι διαφορετικοί τρόποι παρεμβολής όσον αφορά τα αρχεία του εργαζόμενου μέσου μεταξύ των προγραμμάτων. Τέλος διαφορές, έστω και αμελητέες, παράγονται λόγω της διαφορετικής ακρίβειας υπολογισμών μεταξύ των δύο, κάτι που προστίθεται στα προηγούμενα, δημιουργώντας τις διαφορές που είδαμε σε επίπεδο μηχανής.

Όπως αναφέρθηκε και στην εισαγωγή, ο σκοπός αυτής της διπλωματικής είναι να αποτελέσει το πρώτο βήμα σε μία πιθανή διασύνδεση μεταξύ του NPSS και του PROOSIS. Η διασύνδεση των δύο λογισμικών είναι μία επιθυμία της βιομηχανίας, καθώς θα διευκολύνει την επικοινωνίας μεταξύ των κατασκευαστών, ιδιαίτερα σε περιπτώσεις όπου διαφορετικές εταιρείες, που χρησιμοποιούν διαφορετικά λογισμικά, κατασκευάζουν διαφορετικά μέρη του ίδιου κινητήρα. Έχοντας πλέον εξοικειωθεί με τη φιλοσοφία μοντελοποίησης του NPSS και έχοντας μάθει πώς λειτουργεί το πρόγραμμα, το επόμενο βήμα είναι να αναπτύξουμε ένα περιβάλλον διασύνδεσης μεταξύ του NPSS και του PROOSIS. Μία πιθανή ιδέα είναι να γίνει η διασύνδεση αυτή σε επίπεδο πηγαίου κώδικα. Αυτό σημαίνει ότι ο κώδικας του ενός προγράμματος θα μετατραπεί με τρόπο τέτοιον ώστε να μπορεί να εκτελεστεί από το άλλο πρόγραμμα. Για παράδειγμα, γνωρίζουμε ότι το PROOSIS έχει την δυνατότητα να παράγει και να εξάγει τον κώδικα μίας συνιστώσας ή ακόμη και ολόκληρης μηχανής, σε γλώσσα προγραμματισμού C++, την ίδια δηλαδή στην οποία έχει βασιστεί και το NPSS. Μία ακόμα πιθανή ιδέα είναι να γίνει η διασύνδεση μέσω δημιουργίας κατάλληλων decks. Τα decks είναι εκτελέσιμα αρχεία τα οποία

μπορούν να χρησιμοποιηθούν ως μαύρο κουτί από ένα εξωτερικό πρόγραμμα (π.χ. Excel), δίνοντας παράλληλα τις κατάλληλες εισόδους. Τόσο το PROOSIS όσο και το NPSS έχουν την δυνατότητα παραγωγής deck. Αυτό σημαίνει ότι παραδείγματος χάριν, το NPSS μπορεί να δημιουργήσει ένα deck μίας συνιστώσας ή ακόμα και ενός μοντέλου μηχανής, το οποίο με τη σειρά του θα μπορεί να εκτελεστεί ως μαύρο κουτί από το PROOSIS και αντίστροφα.

Τέλος, αφού τα δύο λογισμικά είναι θερμοδυναμικά ισοδύναμα και η μοναδική ουσιαστική διαφορά μεταξύ αυτών είναι οι χάρτες επιδόσεων, θα ήταν δυνατό να τροποποιηθούν τα στοιχεία του PROOSIS με τέτοιο τρόπο ώστε να μπορούν να διαβάζουν και να λειτουργούν με χάρτες της φιλοσοφίας του NPSS (δηλαδή R-line και PR-line χάρτες).

11 REFERENCES

- [1] R. W. Claus, A. L. Evans, J. K. Lylte and L. D. Nichols, "Numerical Propulsion System Simulation", NASA Lewis Research Center, Interdisciplinary Technology Office, Cleveland, OH 44135, U.S.A., May 1991
- [2] Michael L. Belair, Charles J. Sarmiento and Thomas M. Lavelle, "Nuclear Thermal Rocket Simulation in NPSS", NASA/TM-2013-216553, July 2013
- [3] Dean Olson, "Pratt & Whitney Space Propulsion NPSS Usage", 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference, Session 42-ABP-7 Room Palm B, Fort Lauderdale, July 2004
- [4] Mingxuan Shi, Manish Pokhrel, Jonathan Gladin, Elena Garcia and Dimitri N. Mavris, "Modeling Fidelity Requirements of Mission-Level Analysis on Boundary Layer Ingestion Propulsion System", AIAA SciTech Forum, San Diego, January 2019
- [5] Hyoungjin Kim and Meng-Sing Liou, "Optimal Inlet Shape Design of N2B Hybrid Wing Body Configuration", 48th AIAA/ASME/SAE/ASEE Joint Propulsion Conference, Atlanta, July 2012
- [6] Long N. Vu and Donald R. Wilson, "Quasi-one-dimensional Scramjet Combustor Flow Solver Using the Numerical Propulsion System Simulation", AIAA Propulsion and Energy Forum, Cincinnati, July 2018
- [7] Eric S. Hendricks, "Development of an Open Rotor Cycle Model in NPSS Using a Multi-Design Point Approach", NASA/TM-2011-217225, GT2011-46694, October 2011
- [8] Christopher A. Perullo, David Trawick, William Clifton, Jimmy C.M. Tai and Dimitri N. Mavris, "Development of a Suite of Hybrid Electric Propulsion Modeling Elements Using NPSS", Proceedings of ASME Turbo Expo 2014: Turbine Technical Conference and Exposition, GT2014, Düsseldorf, June 2014

- [9] Francisco M. Capristan and Nathaniel J. Blaesser, "Analysis of the Parallel Electric-Gas Architecture with Synergistic Utilization Scheme (PEGASUS) Concept", NASA/TM-2019-220396, August 2019
- [10] Jonathan Kratz and George Thomas, "Dynamic Analysis of the STARC-ABL Propulsion System", AIAA Propulsion and Energy Forum 2019, Indianapolis, August 2019
- [11] Carl Russell and Wayne Johnson, "Conceptual Design of Vertical Lift Aircraft with Future Propulsion System Architectures", AHS International and AIAA Transformative Vertical Flight Concepts Joint Workshop on Enabling New Flight Concepts Through Novel Propulsion and Energy Architectures, Arlington, August 2014
- [12] Christopher A. Snyder and Michael T. Tong, "Modeling Turboshaft Engines for the Revolutionary Vertical Lift Technology Project", 75th Annual Vertical Flight Society (VFS 2019) Forum and Technology Display, Philadelphia, May 2019
- [13] Eric S. Hendricks, Robert D. Falck, Justin S. Gray, Eliot D. Aretskin-Hariton, Daniel J. Ingraham, Jeffryes W. Chapman, Sydney L. Schnulo, Jeffrey C. Chin, John P. Jasa and Jennifer D. Bergeson, "Multidisciplinary Optimization of a Turboelectric Tiltwing Urban Air Mobility Aircraft", AIAA Aviation 2019, Dallas, June 2019
- [14] Jeffrey T. Csank, David J. Sadey, George L. Thomas, Thomas M. Lavelle, Jennifer D. Bergeson and Jesús García Calderón, "Electrical Power System Sizing within the Numerical Propulsion System Simulation", AIAA Propulsion and Energy Forum and Exposition, Indianapolis, August 2019
- [15] David Sadey, Jeff Csank and Tom Lavelle, "NPSS Electrical Port Development", NASA Glenn Research Center, February 2019
- [16] EcosimPro website: https://www.ecosimpro.com/products/proosis/#PROOSIS_description
- [17] EcosimPro website: https://www.ecosimpro.com/products/ecosimpro/#ECOSIMPRO_description
- [18] EcosimPro website, PROOSIS brochure: https://www.ecosimpro.com/wp-content/uploads/2015/02/proosis_brochure.pdf
- [19] PROOSIS User Manual

-
- [20] “What is Numerical Propulsion System Simulation (NPSS)?”, Southwest Research Institute (SwRI), San Antonio, March 2016
 - [21] Scott M. Jones, “Steady-State Modeling of Gas Turbine Engines Using the Numerical Propulsion System Simulation Code”, Proceedings of ASME Turbo Expo 2010: Power for Land, Sea and Air, GT2010, Glasgow, June 2014
 - [22] G. Follen and M. auBuchon, “Numerical Zooming Between a NPSS Engine System Simulation and a One-Dimensional High Compressor Analysis Code”, NASA/TM-2000-209913, April 2000
 - [23] NPSS User’s Guide
 - [24] D. Y. Davis and E. M. Stearns, “Energy Efficient Engine-Flight Propulsion System Final Design and Analysis”, NASA CR-168219, R83AEB488, August 1985
 - [25] NPSS Training Module 2-1, “Configuring and Running a Turbojet Model”, Wolverine Ventures, 2012
 - [26] NPSS Training Module 2-2, “Creating a Turbofan Model”, Wolverine Ventures, 2012
 - [27] NPSS Thermo Guide, February 2016
 - [28] NPSS Reference Sheets, March 2008
 - [29] NPSS Fluid Network Thermo Guide, February 2016
 - [30] “Introduction to Propulsion Simulation Using NPSS”, Southwest Research Institute, Presentation at Cranfield University, UK, September 2017
 - [31] “Introduction to System Modeling”, Southwest Research Institute, 2012
 - [32] NPSS Training Module 1-1, 1-2, 2-1, 2-2, 3-1, 3-2, Wolverine Ventures, 2012