DLR-IB-AT-KP-2017-9

Transient performance calculation of the Rolls-Royce/MAN Turbo RB 153 engine with special focus on heat soakage

Robin Martinelle

Cologne, January 2017

Deutsches Zentrum
DLR für Luft- und Raumfahrt

Deutsches Zentrum
für Luft- und Raumfahrt e.V

Transient performance calculation of the Rolls-Royce/MAN Turbo RB 153 engine with special focus on heat soakage

Robin Martinelle

Second year internship
September 2016 – January 2017
Cologne

Industrial supervisor: Maximilian Vieweg

ENSIAME supervisor: Prof. Céline Morin

ENSIAME

DLR

Contents

Nomenclature

Roman symbols

A	Area	[m^2]
Ah	Heat transfer area	[m^2]
Cp	Mass heat capacity	[J/kgK]
h	Heat transfer coefficient	[W/m^2K]
k	Material thermal conductivity	[W/mK]
m	Mass	[kg]
P	Pressure	[Pa]
Q	Heat flux	[W]
R	Gas constant	[J/kgK]
T	Temperature	[K]
t	Time	[sec]
W	Mass flow	[kg/sec]
Wc	Corrected mass flow	[kg/sec]

Greek symbols

θ	Referred temperature	[-]
δ	Referred pressure	[-]
η	Efficiency	[-]
γ	Heat capacity ratio	[-]
τ	Time constant	[-]

Abbreviations

Bld	Bleed
BldDct	Bleed duct
BPR	Bypass ratio
BrnPri	Burner
DctPri	Primary duct
DctSec	Secondary duct
FAR	Fuel-air ratio
FN	Net thrust
HPC	High pressure compressor
HPSHAFT	High pressure shaft
HPT	High pressure turbine
LPC	Low pressure compressor
LPSHAFT	Low pressure shaft
LPT	Low pressure turbine
Mix	Mixer
Nrel	Ratio current corrected speed to design corrected speed
nH	High pressure rotor speed
nHmax	High pressure rotor maximal speed
NozPri	Nozzle
PR	Pressure ratio
Splt	Splitter
S	Station

List of figures

List of tables

Acknowledgments

First of all, I would like to thank Dr.-Ing A. Döpelheuer, head of Engine department, for accepting me in his team.

I wish to thank my supervisor, Maximilian Vieweg, who followed my work, guided me throughout the study and helped me whenever I needed it. R. Becker und F. Wolters also helped me and have been implied in the project.

I would like to thank my ENSIAME supervisor Prof. Morin who also followed my work.

Many thanks to my German teacher Prof. Baginski; he helped me finding this internship and had the first contact with the DLR.

Finally, I would like to express my gratitude to the whole Engine department team; they have been very kind and thanks to them, these four months have been really great.

These months in Cologne have provided an overview of the engineer work in a research center. The internship has been full of valuable assets as learning to work with thermodynamic tools, improving communication and language skills and brought knowledge about fluid mechanics and heat transfers.

1. Introduction

1.1. Motivations

In a context of globalization, the aeronautic industry experiences one of the best economic growths these last years. Mainly due to the developing countries, the plane world fleet should double within the next twenty years. But this expected evolution will set a true challenge to the actors of the industry. From the conception to operation, the whole chain of production will face issues: gas emission, fuel consumption for example. The work of the aeronautic research field is to find solutions and offer means to improve plane performances.

From the first steam engine to the modern turbojet engine, the motorization is, without any doubt, the part of the plane that has been most improved. Indeed, the specific fuel consumption has been divided by 2.5 in just 50 years [9], which means plane engines are far more efficient than before. Improving engines is not simple though: They must deliver high level of performance in two very different operating regimes, stationary and transient. The first is about the regime where the engine parameters are not changing with time and the latter consists of operation where these parameters are changing very fast. Besides, other parameters like heat transfers are present in transient maoeuvres, which add more constraints to the studies. Both configurations have to be taken into account in the research work, where engineers are working on reducing exhaust, noise emission and more environmentally-friendly planes. This sets a large range of innovations in the next years.

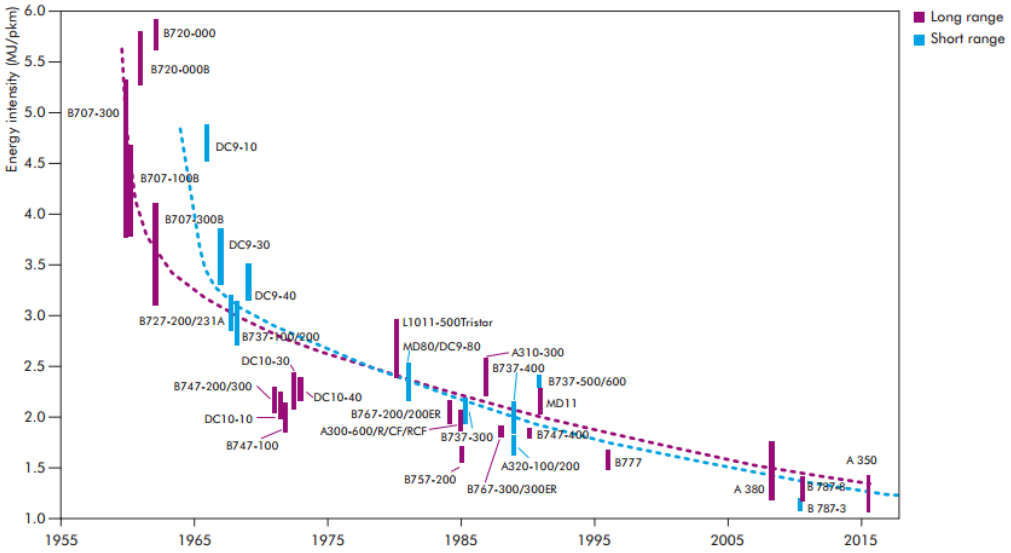

Figure 1: Evolution of fuel efficiency in the last 60 years [9]

1.2. Work place

The DLR (Deutsches Zentrum für Luft- und Raumfahrt) is the German Aerospace Center. It has its headquarters in Cologne, where 1,500 people are working. The main fields of research here are space flight, aviation research and energy technology.

The firm is divided into institutes and then into departments. The internship takes place in the Institute of Propulsion Technology, department Engine.

The Institute of Propulsion Technology works on the improvement of gas turbines, responding to the needs of industry but also society. These consist mainly in efficiency, safety but also noise emission and exhaust.

The Engine department focuses its work on modelling the different components of gas turbines, their interactions, and their behavior in operation. They use innovative calculation methods to analyze engine performance and to test new technologies and concepts.

Figure 2: DLR place in Cologne

Furthermore, they develop an interactive software for gas turbine called GTlab. It incorporates performance and predesign calculators which request and update data via a common interface and thus simplifies the multidisciplinary design procedure.

1.3. Objectives

It is important to gather data about gas turbine performances and to analyze different engines. The study is focused on the Rolls-Royce/MAN Turbo RB 153 engine. The first objective is to build its model with the simulation tool NPSS. Then, using the model, a transient calculation (acceleration and deceleration) will be run. Identify and quantify the heat soakage between the gas and the engine structure will also be a great part of the work.

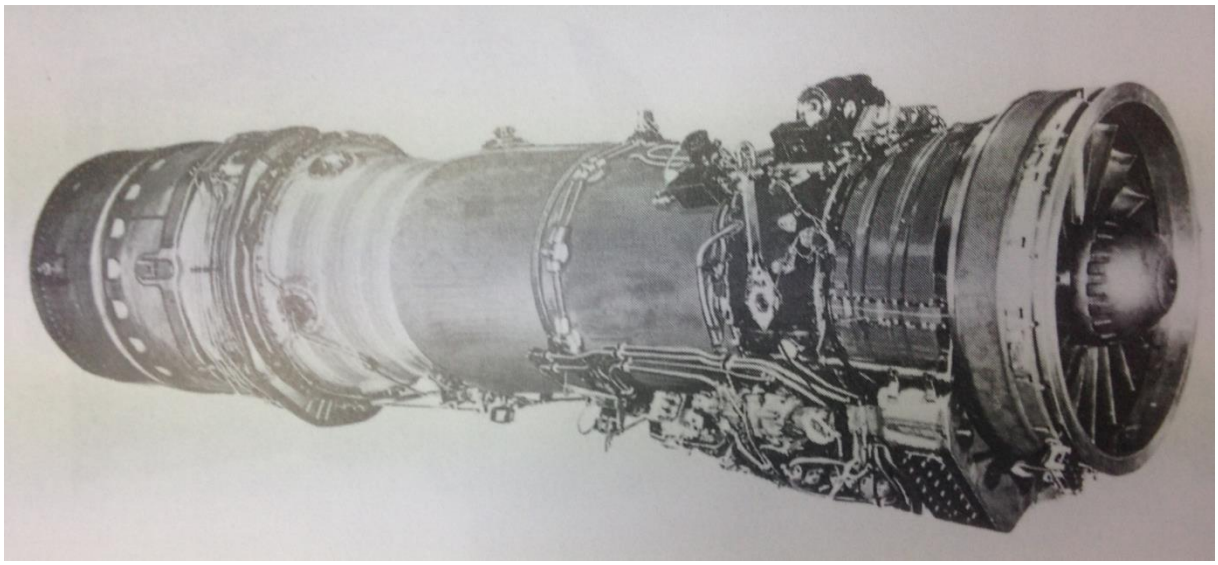

Figure 3: Picture of the RB153 engine, [2]

The work is based on a PhD dissertation by Dr.-Ing. Klaus Bauerfeind [1]. He had the opportunity to work with the RB153 engine and to deeply study it. The fact that a complete engine study is available is rare; this dissertation is valuable and with that the RB153 can be modelled numerically.

2. Background

2.1. The NPSS environment

NPSS (Numerical Propulsion System Simulation) is an object oriented, engineering design and simulation environment. It is used to study aerospace and thermodynamic systems, but also to analyze a large variety of fluid or thermal subjects and overall vehicle emissions.
In order to create a model in NPSS, the user has to define the different elements, the components of the system, and provide the data that describes their performance. Compressors performance calculation is performed with efficiency-based maps submitted by the user. In Design operation, scalars are calculated so that the unscaled map matches the desired Design point, and in Off-Design operation those scalars are held constant and are applied to calculate pressure ratio, efficiency and corrected mass flow. The NPSS environment incorporates a library of standard elements and thermodynamic properties for an engine cycle. The elements are defined in a user file, typically in a text editor and all the simulations are launched with a command window. Once the model is complete, the user must setup the problem: Solver parameters must be specified. The solver, which drives the model to a solution, has to adjust the independent conditions to match the dependent conditions.
There are different ways to view the output data. The user can choose to display the data directly on screen or to send it to an output file. This allows using the results and analyzing them with postprocessing tools. Figure 4 represents the structure of a model in NPSS (applied to the RB153 engine).

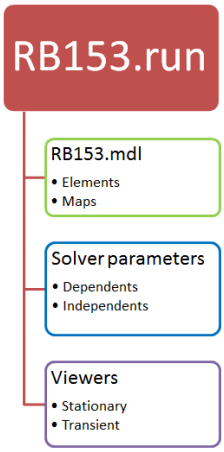

Figure 4: NPSS structure

2.2. The RB153 engine

In the early 1960s emerged a German project for a VTOL (Vertical Take-Off and Landing) fighter: The VJ101. A consortium of the companies Heinkel, Messerschmidt and Bölkow, EWR (Entwicklungsring Süd), designed the VJ101 C and two prototypes have been built. The VJ101C X-1 became the first VTOL plane to break the sound barrier.
As the German Air army needs evolved, the company had to work on a new project, which was basically an optimization of the previous design, the VJ101 D. Conceived to fly at Mach 2, with a higher climbing ceiling, the VJ101 had to get a new motorization which consisted in two Rolls-Royce/MAN Turbo RB 153 engines. The project has been cancelled in 1968 due to the German government's loss of interest in VTOL projects. Therefore, the jet actually never flew, and the engine has been kept as a test bed. [3]

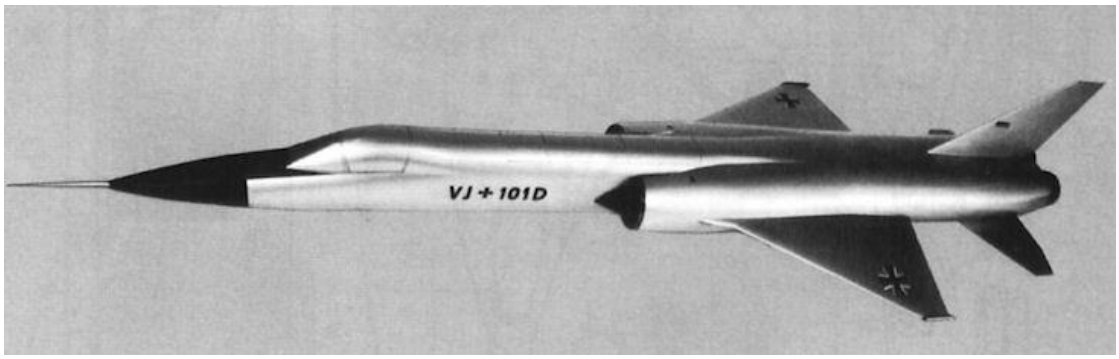

Figure 5: The VJ 101 D design

The engine itself is a bypass engine, which means the air is split into two flows: One through the core channel and one through the bypass channel.

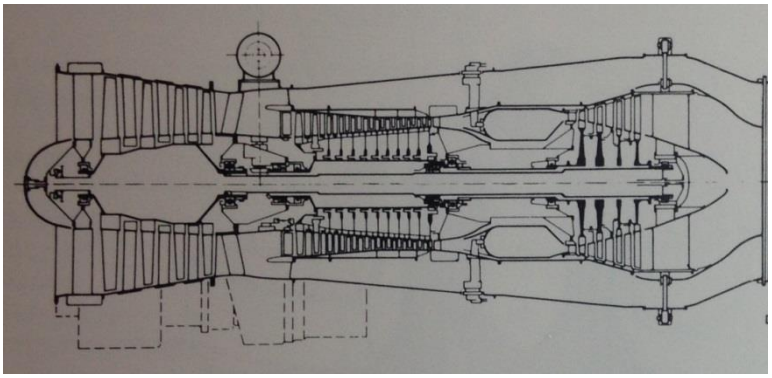

Figure 6: Sectional view of the RB153 engine, [2]

3. The model

3.1. Definition

The graphic model of the engine has been designed with the DLR in-house tool GTlab [4], [5], because NPSS does not include a graphic interface. It is represented in figure 7. This model includes a representation of each element and the links between them (called Stations). The elements are:

- Intake
- Low Pressure Compressor and High Pressure Compressor
- Flow Splitter and Flow Mixer
- Combustion Chamber or Burner
- Low Pressure Turbine and High Pressure Turbine
- Ducts
- Nozzle
- Shafts
- Secondary Air System

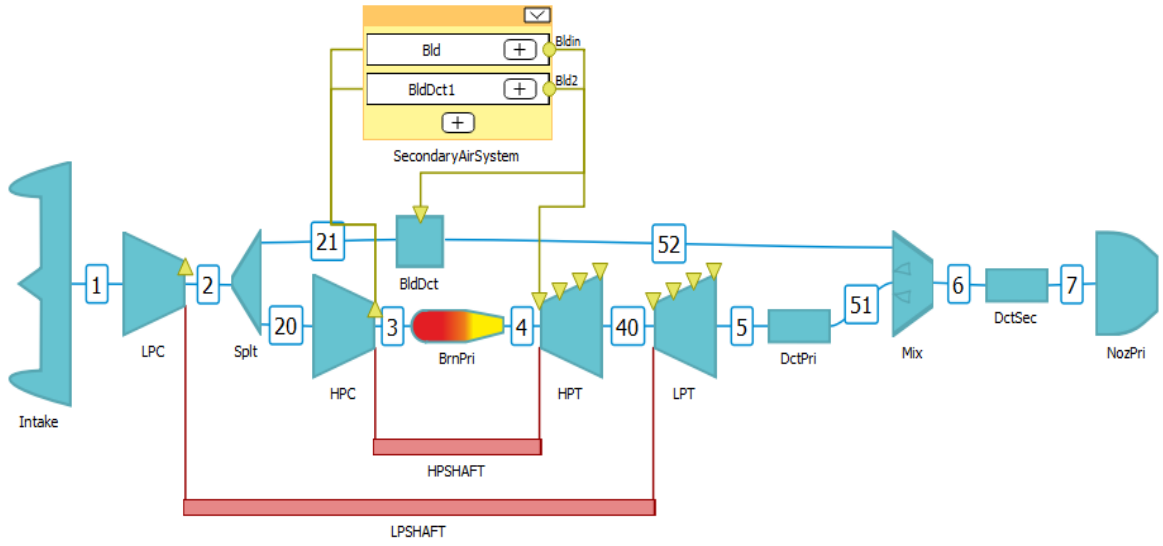

Figure 7: RB153 model in GTlab graphic interface

All these elements and the links between them have been defined in a model file in the NPSS environment using the text editor Notepad++. The following figure is an example of elements definition. Here is defined the Ambient and InletStart elements. They set the ambient conditions and the incoming air flow in the engine, and they are the two first elements of the simulation, InletStart corresponding to the Intake element. As NPSS is an American software, imperial units and different naming conventions are used. It is then necessary to convert the different values in the correct unit.

```
//Ambient Amb
Element Ambient Amb {
     alt_in=0.0;
     Ps_in {value=14.7; units="psia";};
     Ts_in {value=288.15; units="K";};

}

//InletStart InletStart
real W_in_SI {value =55; units="kg/sec";}
real W_in_US {value = convertUnits("W_in_SI", "lbm/sec");
units = "lbm/sec";}

Element InletStart InletStart{
     AmbientName = "Amb";
     W_in = W_in_US;
}
```

Figure 8: Elements in NPSS

The next figure is an example of the different links between the elements: each element has an out port (O) and an in port (I) which can be Fluid (Fl) or Fuel (Fu), depending of the element's purpose.
An instruction like "*linkPorts("CmpH.Fl_O", "BrnPri.Fl_I", "S3")*" will create a link between the fluid out port of the high pressure compressor and the fluid in port of the burner with the station name S3.

```
//Ambient to Inlet
linkPorts("InletStart.Fl_O", "Inl.Fl_I", "S0");

//Primary Cold Section
linkPorts("Inl.Fl_O", "CmpL.Fl_I", "S1");
linkPorts("CmpL.Fl_O", "Splt.Fl_I", "S2");
linkPorts("Splt.Fl_O1", "CmpH.Fl_I", "S20");
linkPorts("FusEng.Fu_O", "BrnPri.Fu_I", "S3f");
linkPorts("CmpH.Fl_O", "BrnPri.Fl_I", "S3");
```

```
//Primary Hot Section
linkPorts("BrnPri.Fl_O", "TrbH.Fl_I", "S4");
linkPorts("TrbH.Fl_O", "TrbL.Fl_I", "S40");
```

Figure 9: Links in NPSS

3.2. Parameters investigation

The solver needs a Design Point to build the stationary model and then to run further calculation. This point has been chosen to be at 97% of the maximal rotational speed of the high pressure rotor. It is a choice, though: In fact, most of the data in [1] is at this value, it is then easier to use that one.
Many variables are needed to study the engine. Bauerfeind gives some in [1], and some others have been calculated or estimated. The assumptions have to take into account the 1960's technologies: The compressors and turbines efficiencies should be smaller than today's standards. The model fidelity has to consider physics reality, so some variables have been added, even if they were not mentioned in [1] as pressure losses in different elements.

The following table is a summary of the different variables used to set the model. Calculation details are in the following pages.

Parameter	Unit	Value	Source
Air mass flow at intake	kg/sec	55	[3]
Altitude	m	0	[1]
Ambient pressure	bar	1.01325	[1]
Ambient temperature	K	288.15	[1]
Burner efficiency	-	0.98	Assumption
Bypass ratio	-	0.7	Assumption
Cooling air flow	%	4	Assumption
Fuel flow	kg/sec	0.588	[1]
High pressure rotor inertia	kg*m²	1.5	Assumption
High pressure rotor maximal speed	rpm	16800	[1]
HPC polytropic efficiency	-	0.883	Calculation
HPC pressure ratio	-	7.5	Assumption
HPT polytropic efficicency	-	0.864	Design output
HPT pressure ratio	-	3.34	Design output
Low pressure rotor inertia	kg*m²	2	Assumption
LPC polytropic efficiency	-	0.846	Calculation
LPC pressure ratio	-	2.4	Assumption
LPT polytropic efficiency	-	0.86	Design output
LPT pressure ratio	-	2.01	Design output
Mach number at mixer's primary entrance	-	0.6	Assumption
Net thrust	kN	31.4	[1]
nH	% of nHmax	97	[1]
Pressure loss in the high pressure rotor	%	2	Assumption
Pressure loss in the low pressure rotor	%	1	Assumption
Pressure loss in the primary duct	%	1	Assumption
Pressure loss in the secondary duct	%	1.5	Assumption

Table 1: Parameters of the model

First, the ambient conditions correspond to ISA (International Standard Atmosphere). The air mass flow at the intake, the fuel flow and the net thrust have been found in [1].
Compressor pressure ratios have been estimated based on the data.

The figures 10 and 11 are the working lines of the compressors and they are reproductions of the ones found in [1]. Working line is a part of the compressor map which is a diagram used to predict and calculate the performance of the turbomachine, and more widely the engine performance.

Here, the lowest point is at 90% nHmax and the highest is at 95% nHmax. As the Design Point is at 97% nHmax, assumptions have to be done to find appropriate pressure ratios.

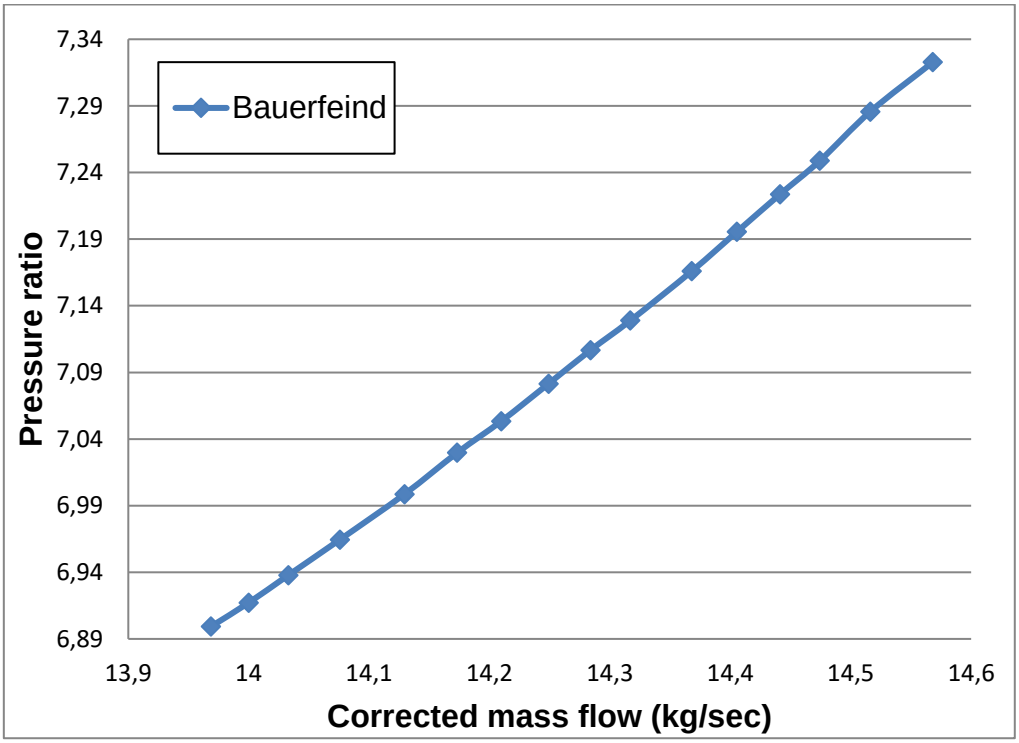

Figure 10: HPC working line from [1]

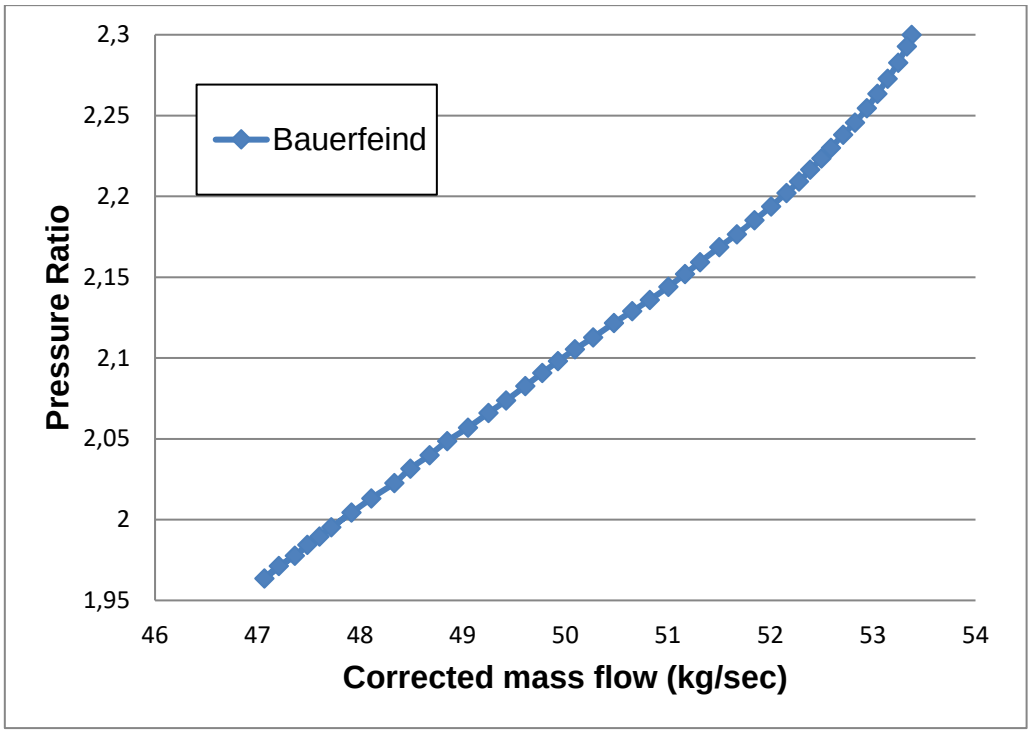

Figure 11: LPC working line from [1]

These stationary lines are built with two parameters: Pressure ratio and corrected flow. The pressure ratio of a compressor is given as follows:

$$PR = \frac{P_{out}}{P_{in}} \tag{1}$$

Corrected flow represents the mass flow that would pass through a compressor if the inlet conditions were corresponding to ISA. Calculation is in formula 2:

$$Wc = \frac{W\sqrt{\frac{T}{288.15}}}{\frac{P}{101.325}} = \frac{W\sqrt{\theta}}{\delta} \tag{2}$$

With these formulae and the figures, an assumption is possible. It is though important to know that this is estimation and not a true value.

In NPSS, the solver calculates directly the corrected flow, and it basically uses the same formula as it can be seen in figure 12.

```
// calculate misc. fluid condition related variables
real delta = PtIn / C_PSTD ; // ratio of inlet pressure to
standard day pressure
theta = TtIn / C_TSTD; // ratio of inlet temperature to
standard day temperature
sqrtTheta = sqrt(theta);

// calculate corrected flow related variables
Wc =  Fl_I.W * sqrtTheta / delta ;
```

Figure 12: Calculation of the corrected flow in NPSS

The bypass ratio can also be estimated with figures 10 and 11, the pressure ratios and the corrected mass flows: As it can be seen in formula 2, the mass flow can be extracted from corrected mass flow equation. The bypass ratio represents the ratio of air going through core channel and bypass channel. It is defined in formula 3:

$$BPR = \frac{W_2}{W_1} \tag{3}$$

At first, it has been estimated at 0.86, with W_2= 24.3 and W_1= 28.2. The model fidelity imposed then to decrease it to 0.7.
The polytropic efficiencies of the compressor have been calculated using formula 4. It is based on the isentropic relation but the polytropic efficiency is added in the formula.

$$\frac{T_{out}}{T_{in}} = \left(\frac{P_{out}}{P_{in}}\right)^{\frac{\gamma-1}{\gamma*\eta_{poly}}} \tag{4}$$

With γ:

$$\gamma = \frac{1}{1 - \frac{R}{Cp}} \tag{5}$$

γ is assumed to be constant in this calculation, which is a simplification because it is not strictly true. Then, as the values for Cp (1011 J/kgK) and R (287 J/kgK) are known, the polytropic efficiency can be calculated:

$$\eta_{poly} = \frac{\gamma - 1}{\gamma * \frac{\log\left(\frac{T_{out}}{T_{in}}\right)}{\log\left(\frac{P_{out}}{P_{in}}\right)}} \quad (6)$$

The other parameters are not mentioned in [1]. However, they are very important for the results to be reliable. Pressure losses and inertias have been estimated according to the geometry of the different elements; burner efficiency has been set to an expected value considering the 1960's technologies for example.

Finally, there is a secondary air system between the high pressure compressor and the high pressure turbine and the bypass channel. This system exists but there is no information about it, it is only mentioned.
The percentage of air going through the cooling air flow channel has been estimated with data from other Rolls-Royce engines built in this time period.
The amount of air going through the system between the high pressure compressor and the bypass channel (handling bleed) has been scheduled. Thus, the handling bleed flow depends on the rotational speed of the high pressure rotor.

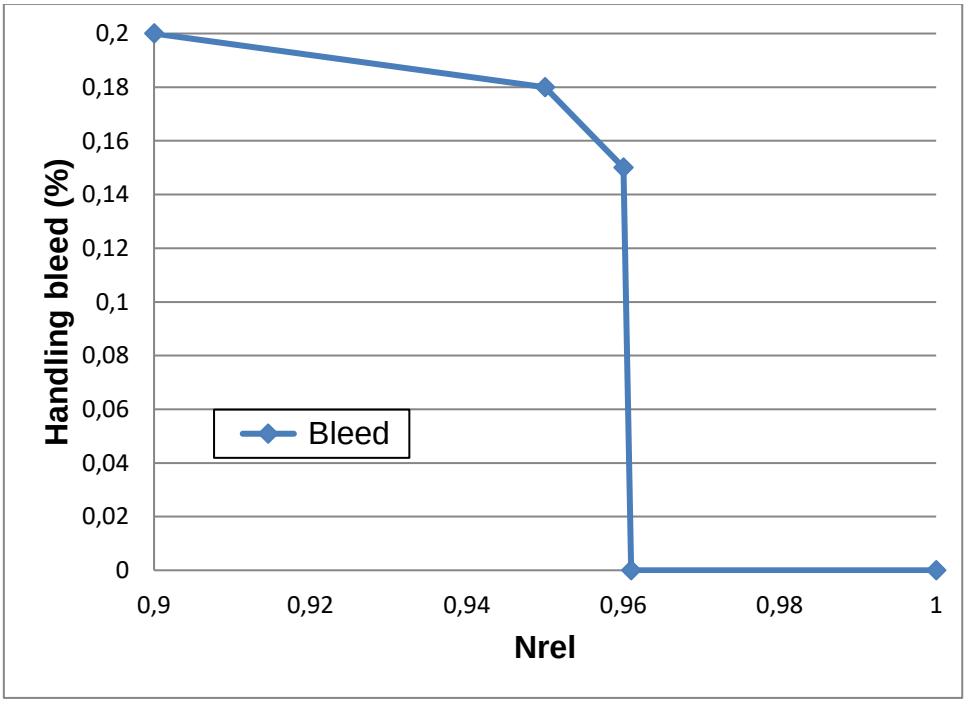

Figure 13: Percentage of handling bleed

Once the elements' parameters are specified, constraints and goals have to be explained.

The dependent and independent parameters are defined in a specific file that contains the solver parameters. In the simulation, the net thrust is the dependent parameter; the solver has to match this variable. To reach that solution, it can modify the independent parameter which receives only a start value. In NPSS, the independent is the Fuel-Air Ratio (FAR). It represents the mass ratio of fuel to air in the combustion process that takes place in the burner and it is shown in formula 7. Figure 14 shows the implementation of the independents and dependents in NPSS.

$$FAR = \frac{W_{fuel}}{W_{air}} \tag{7}$$

```
//FAR ind_FAR
Independent ind_FAR {
     varName="BrnPri.FAR";
     indepRef = "0.019";
     dxLimit = 0.2;
     dxLimitType = "FRACTIONAL";
     description = "vary the FAR to achieve the desired FN";
}
//FN dep_FN
real FN_in_SI {value= 31400; units="N";}
real FN_in {value=convertUnits("FN_in_SI", "lbf");
units="lbf";}

Dependent dep_FN {
     eq_rhs = "FN_in";
     eq_lhs = "Perf.Fn";
     eq_Ref = "FN_in";
     toleranceType = "FRACTIONAL";
     description = "desired FN";
}
```

Figure 14: Independent and dependent in NPSS

The viewer has to be created next. As it has been said in the NPSS introduction, there are two ways to look at the results: directly on the command window or in a specific file.

The model is now complete and the simulation is almost ready to be run. The different components of the model are gathered in one file and the solver is set, then the simulation can be launched. Figure 15 is the Design calculation in the *run* file: The solver is set to “DESIGN”, independents and dependents are added and the program itself is run.

```
#include "RB153.mdl"
#include "printRB153.view"
#include "RB153solverparams.inc"

//Switch to DESIGN calculation
setOption( "switchDes", "DESIGN" );
autoSolverSetup();

//Add dependent and independent to the solver
solver.addDependent("dep_FN");
solver.addIndependent("ind_FAR");

//Run the program
CASE++;
run();
printRB153.update();
```

Figure 15: Run file in NPSS

3.3. Design and Off-Design Calculation

The model has been built with parameters values at 97% nHmax. It is the Design point and the basis of the stationary model. Bauerfeind has been able to measure the temperatures at different stations. This data will be used to check the model's accuracy: The assumptions and calculations might be correct if the temperatures are matching.

Station	Temperatures at 97% nHmax (K)
S1	288
S2	387
S20	387
S52	387
S3	725
S4	1377
S40	1048
S5	900
S51	900
S6	698

Table 2: Temperatures at Design

Once the Design Point is run, the solver can be changed to Off-Design calculation. Basically, it consists in changing the throttle, and in this case the rotational speed of the rotors. Design calculation allows the solver in NPSS to calculate performance data that will be used in Off-Design calculation for instance design map scalars, mixer and nozzle entrance area, etc. Employing a loop, several points are calculated and create working lines for the high pressure compressor and the low pressure compressor. The lowest steady-state point (Idle point) is chosen at 63% nHmax. Bauerfeind gives temperatures at this point and the same checking as in Design calculation can be done with the results:

Station	Temperatures at 63% nHmax (K)
S1	288
S2	301
S20	301
S51	301
S3	430
S4	880
S40	700
S5	670
S51	670
S6	480

Table 3: Temperatures at Idle

Both working lines have been implemented in the maps used for the two compressors. They are represented in red in the following figures, and the speed lines of the maps are in blue. Data from [1] has been added in green but it is only from 90 to 95% nHmax.

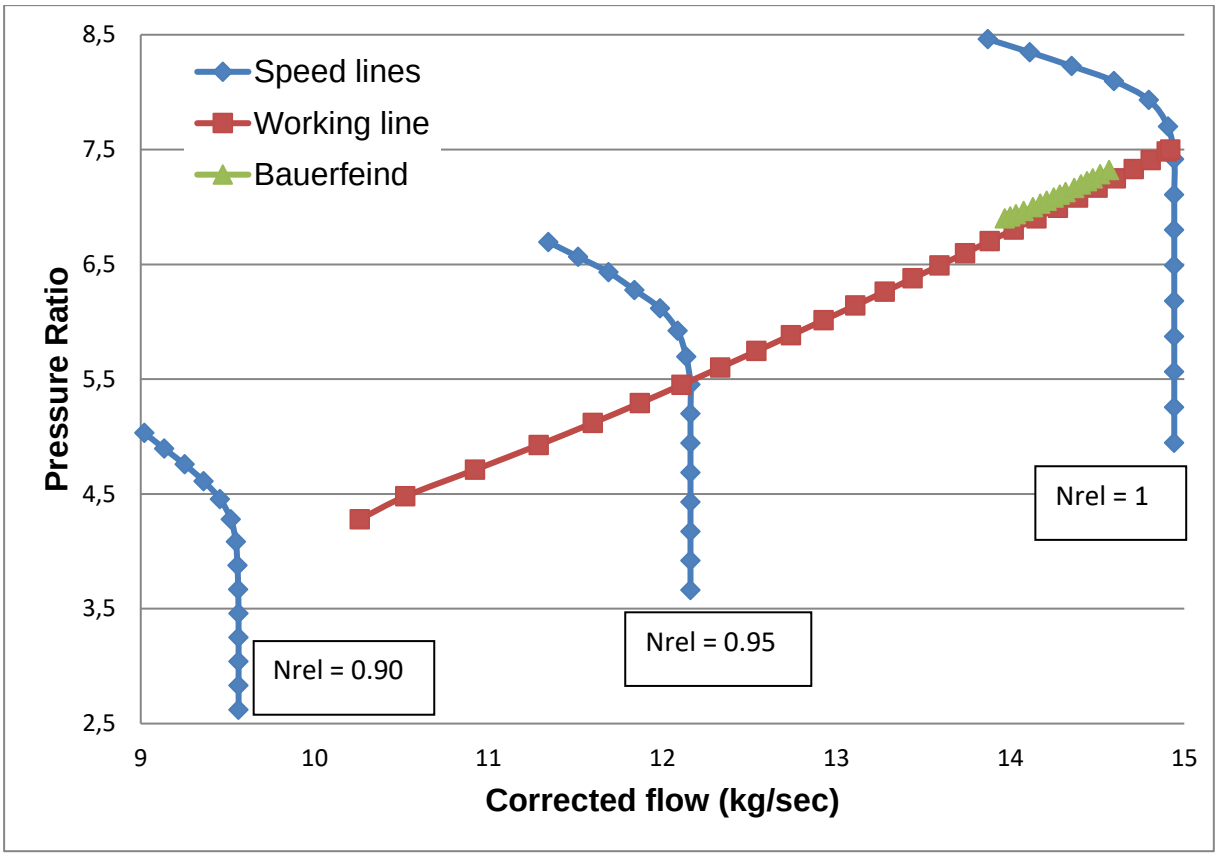

Figure 16: HPC map

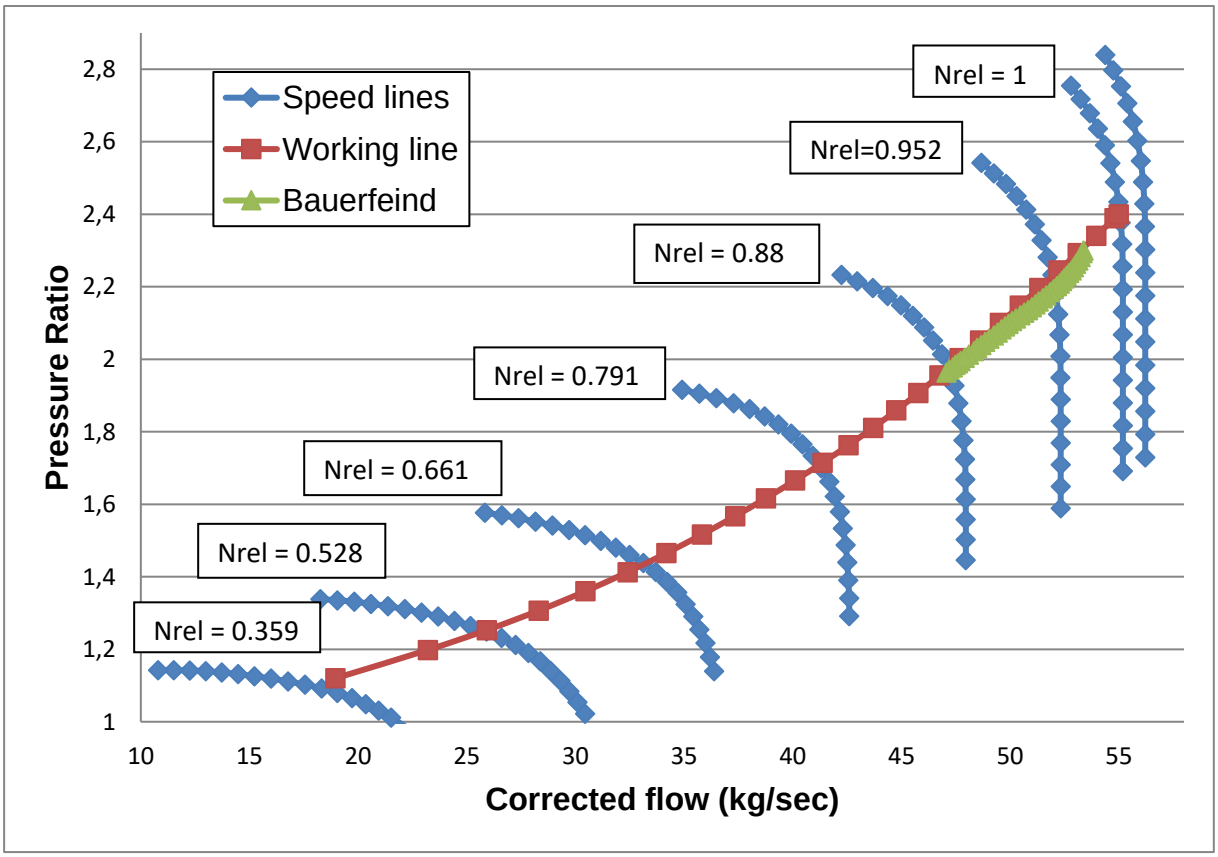

Figure 17: LPC map

4. Transient calculation

4.1. Rotor dynamics

Now that Steady-State calculation has been run properly, the work will focus on transient operation. This calculation mode is about the regime where parameters are changing with time. Transient operation differs from steady-state.
Fuel flow increase causes an acceleration, the turbines power increases along with the temperature and exceeds the power that the compressors need. This unbalanced power accelerates the spool and increases the air flow, pressures, temperatures, etc., until the new steady-state condition corresponding to the fuel flow is matched. For a deceleration, the unbalanced power would be negative. [6]

If there is no cooling air flow between the compressor and the turbine, the unbalanced power can be obtained as follows [6]:

$$UP = W_{turbine} * Cp * \left(T_{turb,out} - T_{turb,in}\right) - W_{compressor} * Cp * \left(T_{comp,out} - T_{comp,in}\right) \quad (8)$$

The unbalanced power is used to calculate the spool acceleration rate NDOT with J_{spool} the spool polar moment of inertia (kg*m²), N the rotational speed of the spool (rpm):

$$NDOT = \frac{UP}{J_{spool} * N * \left(\frac{\pi}{30}\right)^2} \quad (9)$$

The new spool speed is then as follows, with t_{step} being the time step:

$$N_{new} = N + NDOT * t_{step} \quad (10)$$

Therefore, a steady-state operating point must be run before the transient maneuver can be started: At Idle for acceleration and at Design for a deceleration. This is essential to reach a solution and perform the simulation properly.
The fuel flow is scheduled from Idle to Design operating point in the case of acceleration and from Design to Idle operating point in the case of deceleration. The engine responds to this flow. In NPSS, a table has been created for the schedule and is set as dependent parameter. The solver needs also to be set to transient mode and the time must be specified. Figure 18 shows the way to implement it and figure 19 represents the fuel flows in acceleration and deceleration. [1]

```
//Switch to TRANSIENT calculation
setOption("solutionMode", "TRANSIENT");
autoSolverSetup();

//Set the dependent variable RunCondition
RunCondition.eq_rhs = "TB_FuelSchedule(time)";

//Add independent and dependent to the solver
solver.addIndependent("ind_FAR_TR");
solver.addDependent("RunCondition");

transient.stopTime = 5;
transient.baseTimeStep = 0.1;
time = -0.1;

CASE++;
run();
```

Figure 18: Transient run in NPSS

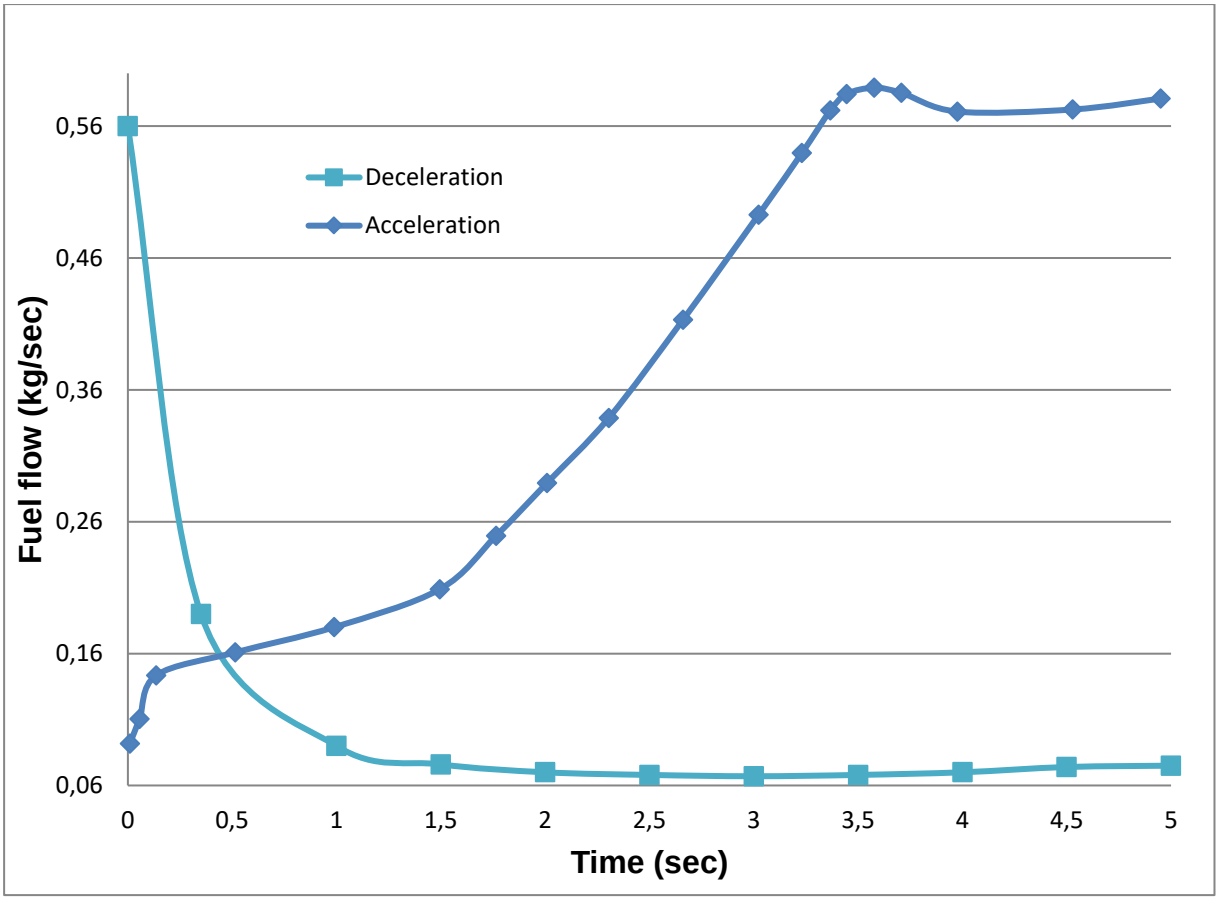

Figure 19: Fuel flow in acceleration and deceleration from Bauerfeind's simulation

4.2. Heat soakage

During transient operation, there are significant heat fluxes between the gas and the engine carcass. In a transient acceleration from Idle to Design, about 30% of the excess fuel energy is transferred to the metal [6], [7]. This heat transfer is called heat soakage. As it has a huge effect on engine performance, it is very important to identify and quantify heat soakage. It provokes a loss in fuel conversion efficiency; it increases components operating temperatures which affects their durability.

Heat soakage calculation requires geometric data of the components, material masses and heat transfer coefficients. These coefficients are difficult to estimate and empirical considerations are often used to evaluate them.

The heat flux Q calculation consists in defining a differential equation for T(t) which is the temperature of the metal at time t. Calculation can also be seen in [7].

$$Q = h * A * \left(T(t) - T_{gas}\right) \tag{11}$$

With m being the metal mass and Cp its specific heat:

$$\frac{dT}{dt} = \frac{Q}{m * Cp} \tag{12}$$

Substituting Q in 11 with 12:

$$m * Cp * \frac{dT}{dt} = h * A * \left(T(t) - T_{gas}\right) \tag{13}$$

Defining the time constant τ:

$$\tau = \frac{m * Cp}{h * A} \tag{14}$$

Using τ in formula 11:

$$\frac{dT}{dt} + \frac{1}{\tau} T(t) = \frac{1}{\tau} T_{gas} \tag{15}$$

Finally the equation is:

$$\Delta T(t) = \Delta T_0 * e^{-\frac{t}{\tau}} \quad (16)$$

With ΔT_0 the temperature difference between the metal and the gas at t=0.

In NPSS, there is a subelement called ThermalMass that can perform the calculation. The user only must give the different parameters explained above. The solver uses the same process to calculate metal temperature at each step. It is important to note that the heat transfer coefficient can only be calculated based on a Design value, so the user must specify it. The heat flux is then obtained with formula 11.

In the RB153 engine, the most important heat fluxes are situated in the burner, the high pressure compressor and the turbines [1], [6]. Table 4 summarizes the variables needed by the solver for heat soakage calculation in NPSS. Data from [1].

Variable	Unit	LPC	HPC	HPT	LPT	Burner
Ah	m²	5.45	4	0.75	1.2	0.78
h	W/m²K	1050	3350	3150	850	3150
Cp	J/kgK	950	520	520	520	520
k	W/mK	150	150	150	25	25
m	kg	130	130.7	130.7	17.5	139

Table 4: Variables for heat soakage calculation

With the values in this table, the ThermalMass subelement has been added in the main components. Figure 20 represents the implementation of the subelement in the low pressure compressor. The "*setOption("switchLagIn", "PHYSICAL")*" instruction determines that the design inputs are Ahx and massMat. wtdAvg_Fl is a factor used to average the fluid conditions between the inlet and outlet of the element. [8]

```
//Low pressure Compressor LPC
real Ah_SI_LPC {value =5.45; units="m2";}
real Ah_US_LPC {value = convertUnits("Ah_SI_LPC", "in2");
units = "in2";}

real h_SI_LPC {value =1050; units="W/(m2*K)";}
real h_US_LPC {value = convertUnits("hx_SI_LPC",
"Btu/(sec*in2*R)"); units = "Btu/(sec*in2*R)";}

real Cp_SI_LPC {value =950; units="J/(kg*K)";}
real Cp_US_LPC {value = convertUnits("Cp_SI_LPC",
"Btu/(lbm*R)"); units = "Btu/(lbm*R)";}

real K_SI_LPC {value =150; units="W/(m*K)";}
real K_US_LPC {value = convertUnits("K_SI_LPC",
```

```
"Btu/(sec*in*R)"); units = "Btu/(sec*in*R)";}

real M_SI_LPC {value =130; units="kg";}
real M_US_LPC {value = convertUnits("M_SI_LPC", "kg"); units =
"kg";}

Element Compressor CmpL {
     #include "Boosterlpc.map";
     PRdes=2.4;
     effDes=0.85;
     Subelement ThermalMass S_Qhx {
          setOption ("switchLagIn", "PHYSICAL");
          Ah = Ah_US_LPC;
          h = h_US_LPC;
          Cp = Cp_US_LPC;
          k = K_US_LPC;
          m = M_US_LPC;
          wtdAvg_Fl = 0.55; }
     }
```

Figure 20: ThermalMass subelement in NPSS

If T_{in} is the inlet temperature and T_{out} the outlet temperature, $\mathrm{T_{gasPath}}$ is the weighted flow temperature:

$$T_{gasPath} = wtdAvg_{Fl} * T_{in} + (1 - wtdAvg_{Fl}) * T_{out} \tag{17}$$

NPSS calculates the material temperature $\mathrm{T_{mat}}$ recursively with the temperature at the previous time step $\mathrm{T_{matPrev}}$ and the time step timeStep:

$$T_{mat} = T_{matPrev} + \left(T_{gasPath} - T_{matPrev}\right) * \left(1 - e^{-\frac{timeStep}{\tau}}\right) \tag{18}$$

With τ:

$$\tau = \frac{m * Cp}{h * Ah} \tag{19}$$

Data for the heat fluxes in a case of an acceleration in the different components has also been found in [1] and is presented in figure 21. The acceleration is set by the fuel flow represented in figure 19.

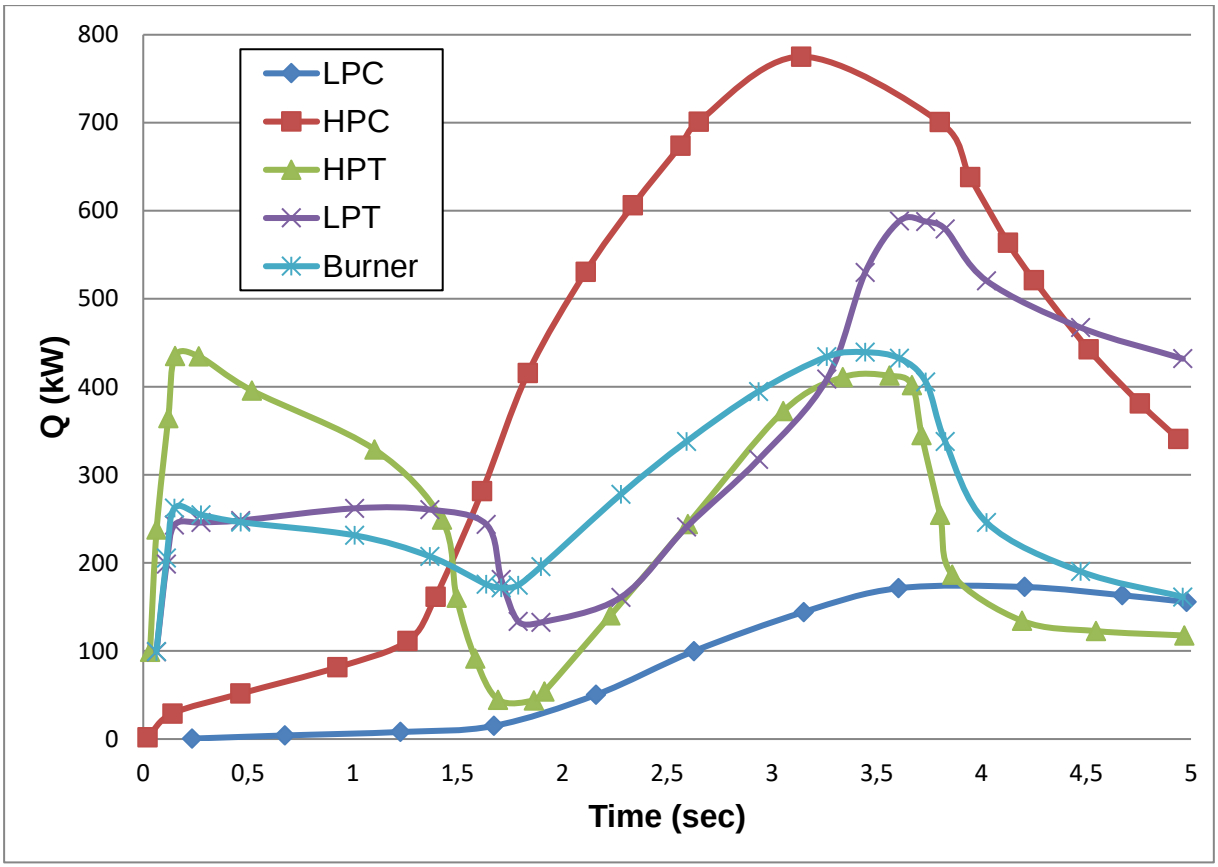

Figure 21: Heat soakage in the main components, [1]

5. Results and discussion

5.1. Design and Off-Design

Once the Design calculation has been done, results can be analyzed. The data presented in 3.3. (Table 2) is compared to the Design results. The comparison is in figure 22. If the temperatures match, it might be suggested that the model is correct. More details can be seen in Appendix A, Table 6.

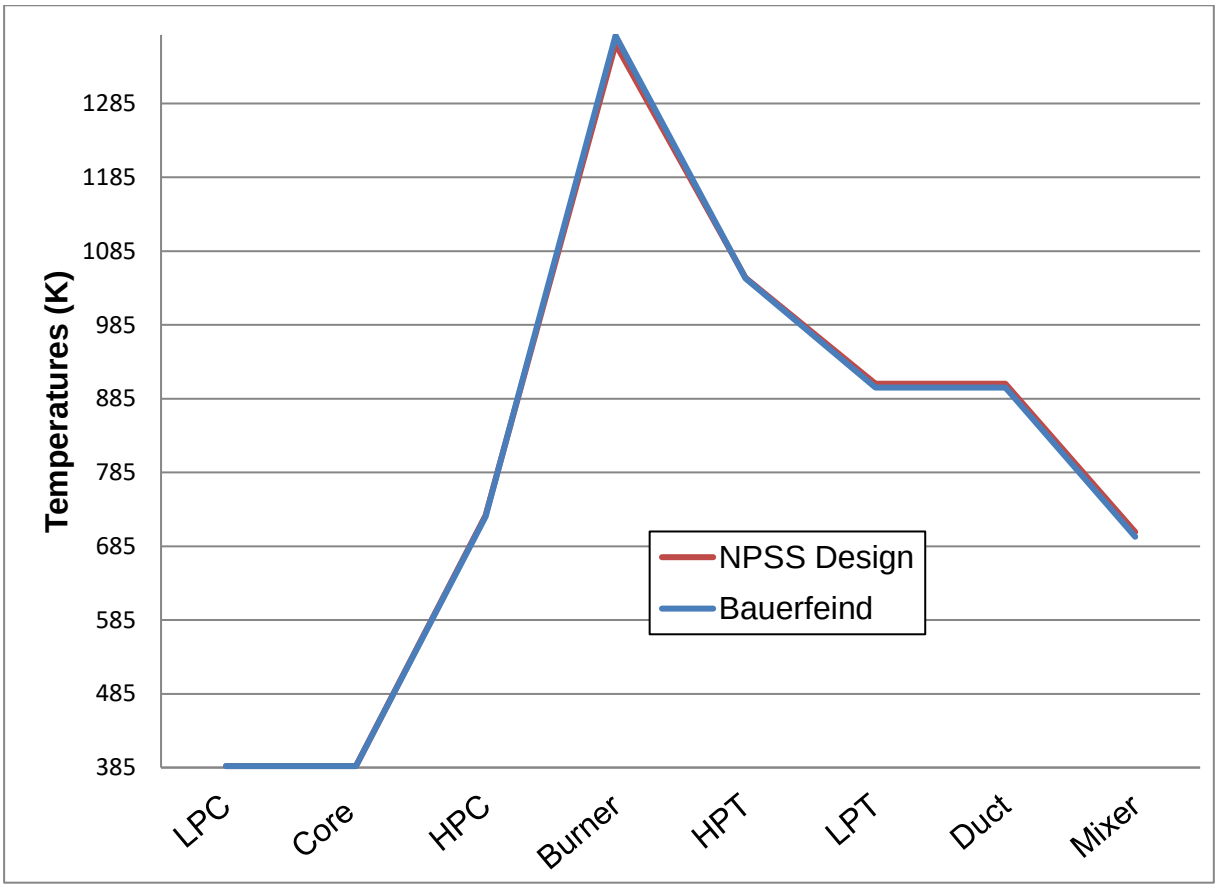

Figure 22: Temperature deviation in Design calculation

Variable	Unit	Bauerfeind	NPSS Design	Relative delta
Burner.FAR	[-]	0.019	0.0189748	-0.13%
Burner.Wfuel	[kg/sec]	0.588	0.5893348	0.23%
Engine.FN	[N]	31400	31399.985	0.00%

Table 5: Parameters deviation in Design

The temperatures match the model almost perfectly. Besides, the fuel flow and the fuel-air ratio are correct, too, as it can be seen in table 5. There is a good alignment between the NPSS model and the result of [1], which indicates that the assumptions made might be appropriate. Moreover, having a reliable stationary Design model is essential to run Off-Design and Transient calculation: The solver is correctly set and the engine performance will be more realistic.
The second analysis that needs to be done is the idle comparison. There is only information about the temperatures and not about the fuel flow and the fuel-air ratio at this operating point.
More details in Appendix B, Table 7.

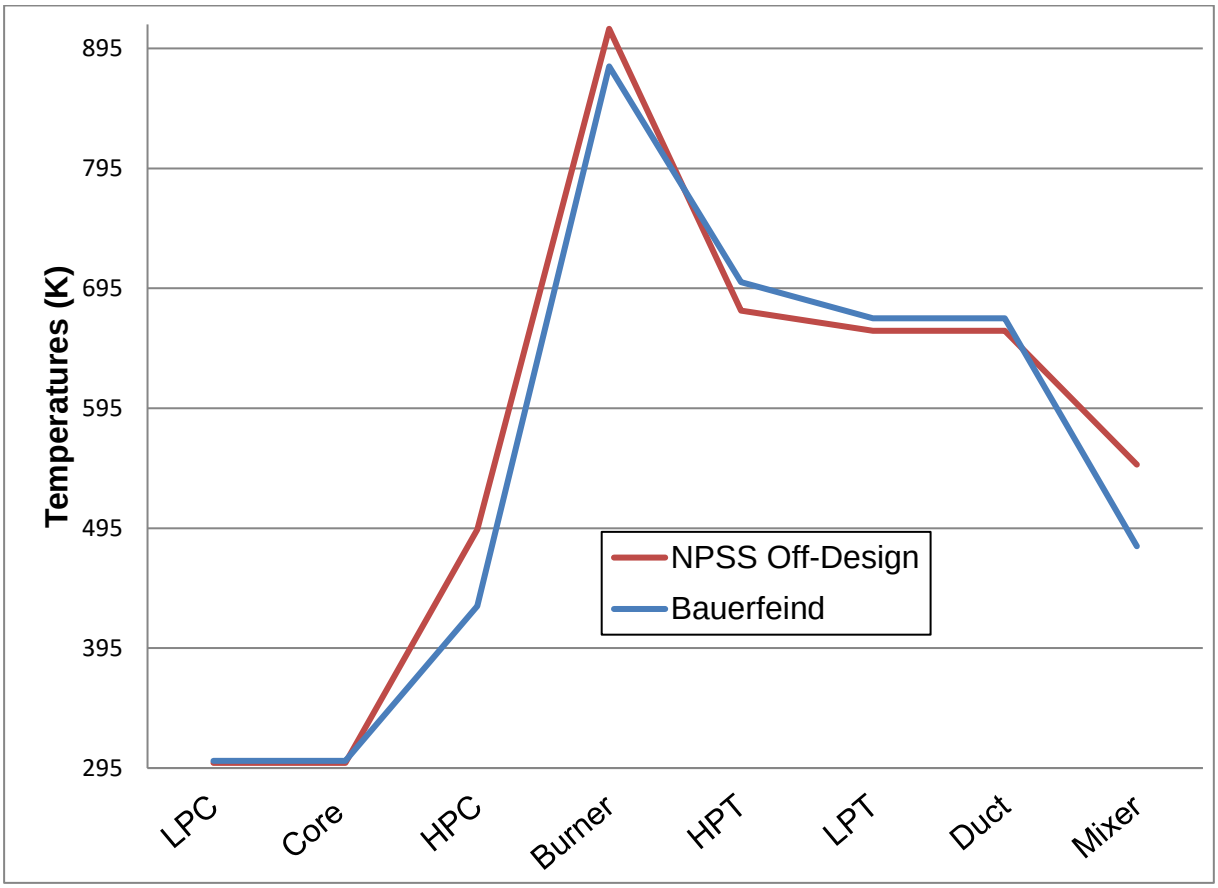

Figure 23: Temperature deviation in Off-Design

The temperatures are matching the model to some extent. There is a difference but the curve profiles are similar. The NPSS simulation temperatures match perfectly the data given in [1] in the low pressure compressor and in the core channel; they are too high in the high pressure compressor, the burner, and the mixer but too low in the turbines and duct.

The current handling bleed represents 18.65% of the core engine air flow. A test has been made with 0, 10 and 20% of handling bleed to see the influence of this on the temperatures. Figure 24 presents these results.

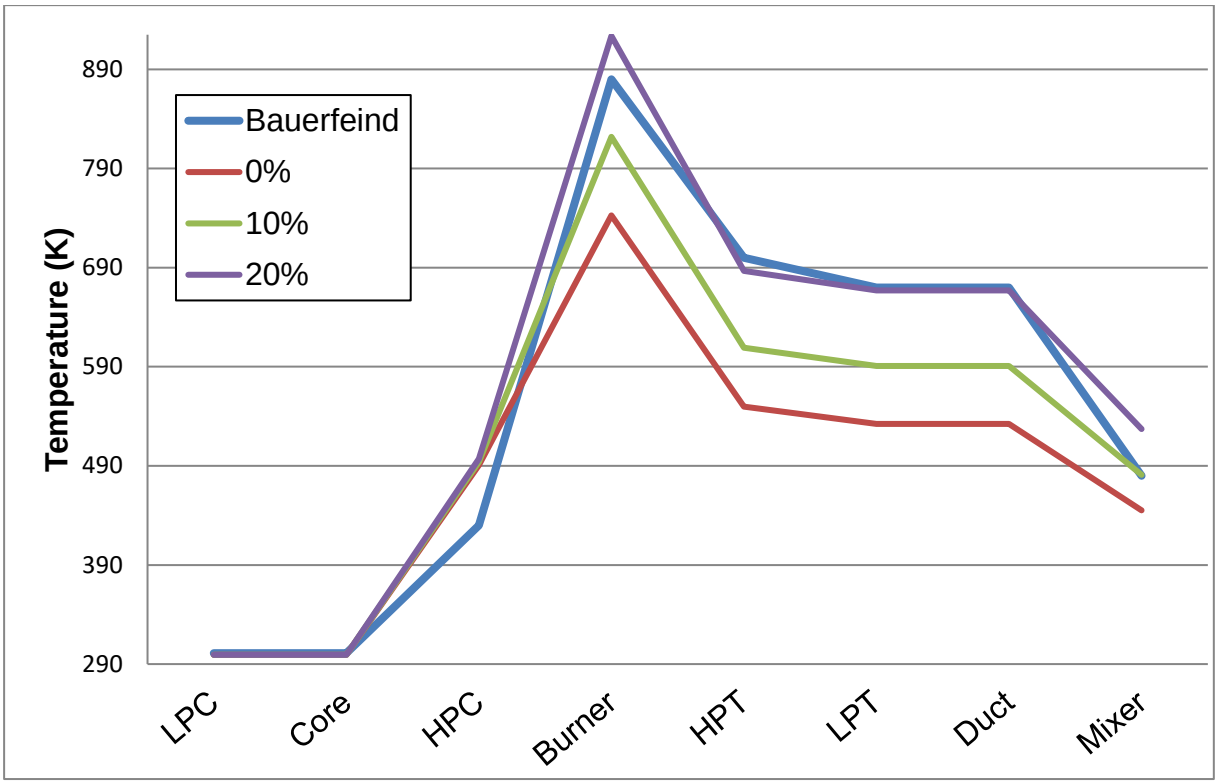

Figure 24: Temperature deviation in Off-Design, with bleeds

The handling bleed has a huge influence on the temperatures at Idle. Without bleed, the difference can be up to 150 K; at 10% the difference is still big. With 20% bleed, the burner temperature overpass [1] but the turbines and the duct are correct. Whatever the percentage applied, the temperature after the high pressure compressor remains the same. In the current configuration, the extracted flow has a relative enthalpy and a relative pressure to the high pressure compressor of 0.9. Those two parameters determine where the flow is taken in the turbomachine, where 0 corresponds to the entrance and 1 to the exit. Different solutions have been tested to see the influence of the extraction pressure and enthalpy on the temperature. The results can be seen in figure 25. Three configurations are presented; the first at 0.3 which corresponds to one third of the compressor length; the second at 0.5 which corresponds to the middle of it and the last at 0.9 which corresponds to the end of the device. Once again, the temperature after the high pressure compressor remains the same and this is not expected. Otherwise, the influence of the extraction position is clear: At 0.3 and 0.5 of enthalpy and pressure the temperatures are too low.

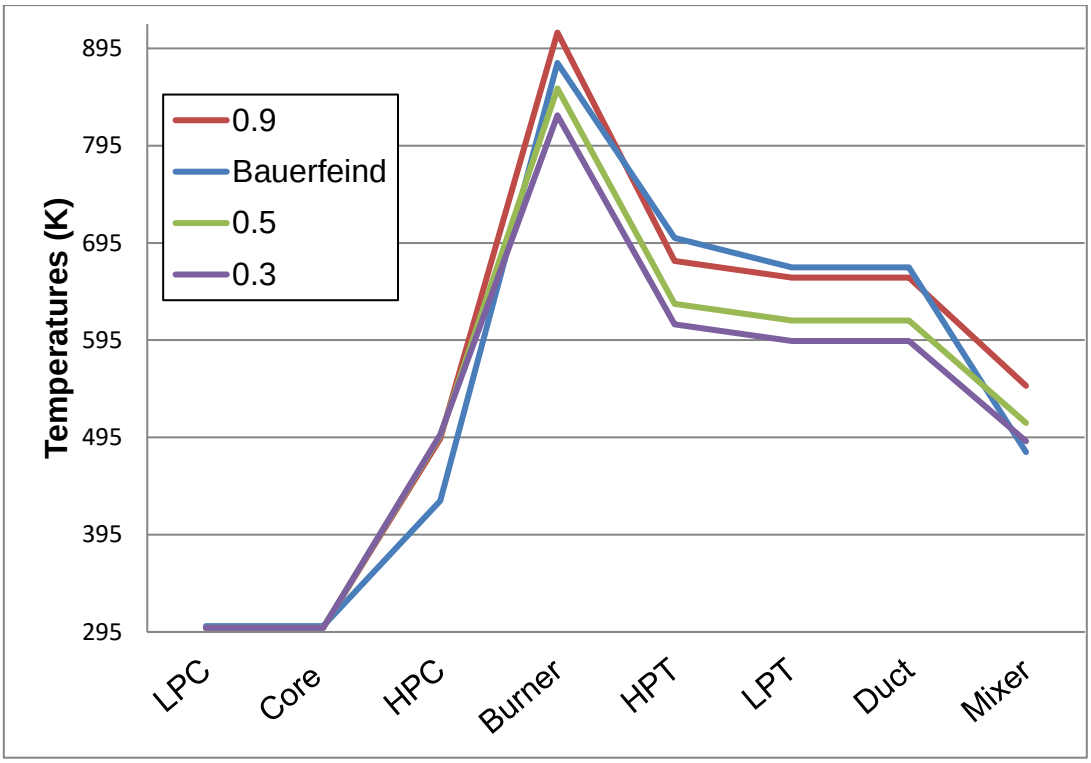

Figure 25: Handling bleed extraction at different relative enthalpy compared to data from [1].

Eventually, the handling bleed at 18.65% and 0.9 of relative enthalpy and pressure represents the best compromise: The burner temperature is not too high; turbines and ducts temperatures are still close to the model. Firstly, the Off-Design calculations were run without handling bleed, which corresponds to the 0% curve; then this parameter has been found and implemented. It is a huge improvement and it is far better this way. However, the remaining difference has not been understood and further research should be made in the future about that in particular.

5.2. Transient

Transient operation results are to be presented in the next pages. The simulation has been run at first without heat soakage and then it has been implemented in the model. Transient maneuvers are driven by a scheduled fuel flow and the net thrust responds to it among other parameters like rotor speed for instance. The next figures represent the net thrust in transient acceleration and deceleration with and without heat soakage.

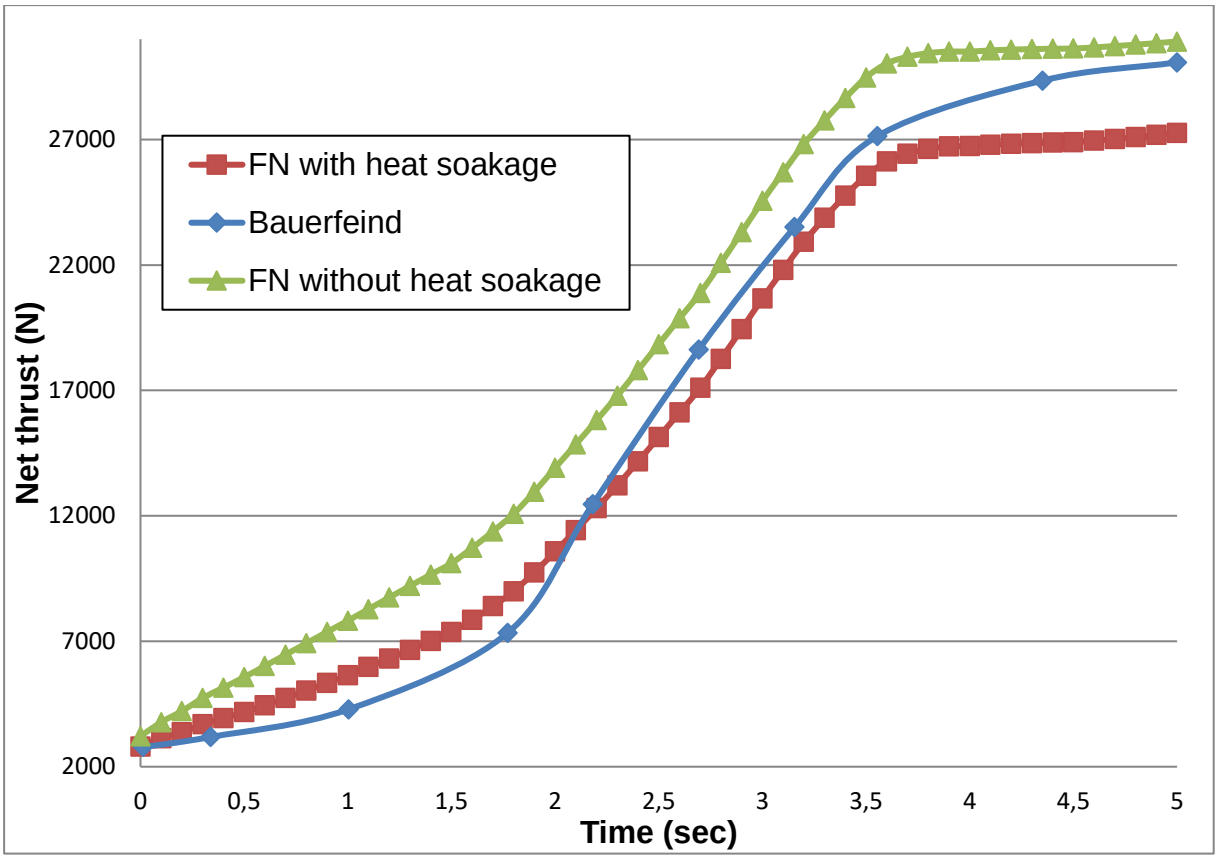

Figure 26: Net thrust in acceleration

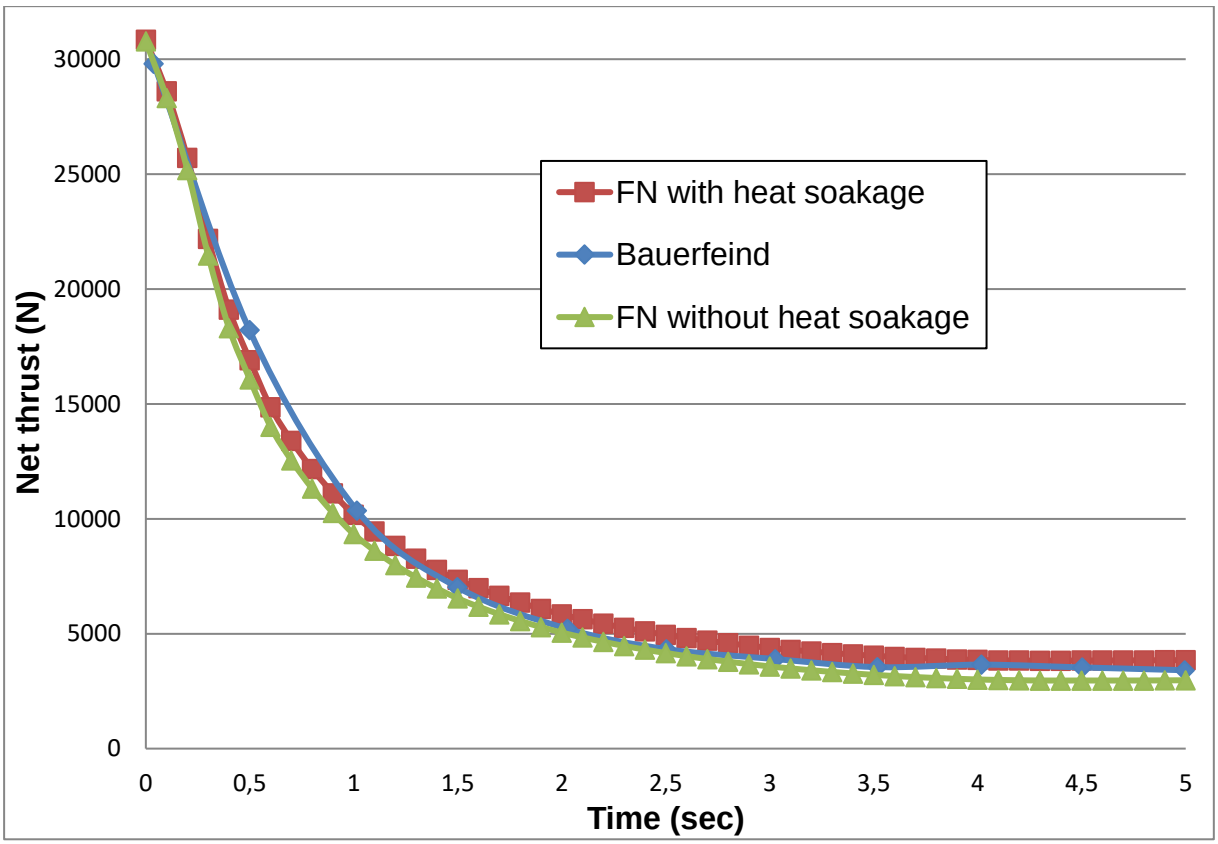

Figure 27: Net thrust in deceleration

In transient deceleration, the net thrust follows almost perfectly the theoretical graph but in transient acceleration there is a slight deviation. The maps used for the compressors and turbines are not the right ones but they have been selected according to the design pressure ratio. This might be the reason of the deviation. Influence of heat soakage can be clearly seen in acceleration operation. Once it has been implemented, the model seems to match the theory better. However the difference is not as significant in deceleration operation. The two NPSS curves in transient acceleration are converging and they eventually meet. This comparison highlights the fact that, in order to build a correct model, the heat soakage phenomenon has to be taken into account. In acceleration, the deviation between t =1 and t = 3.5 sec is around 30% which corresponds to the assumption made in the introduction to heat soakage.

As it has been explained, the solver calculates the heat flux Q in NPSS. The heat soakage has the most influence in acceleration; the heat flux in transient acceleration is represented in the next figures with [1] data (see part 4.2.).

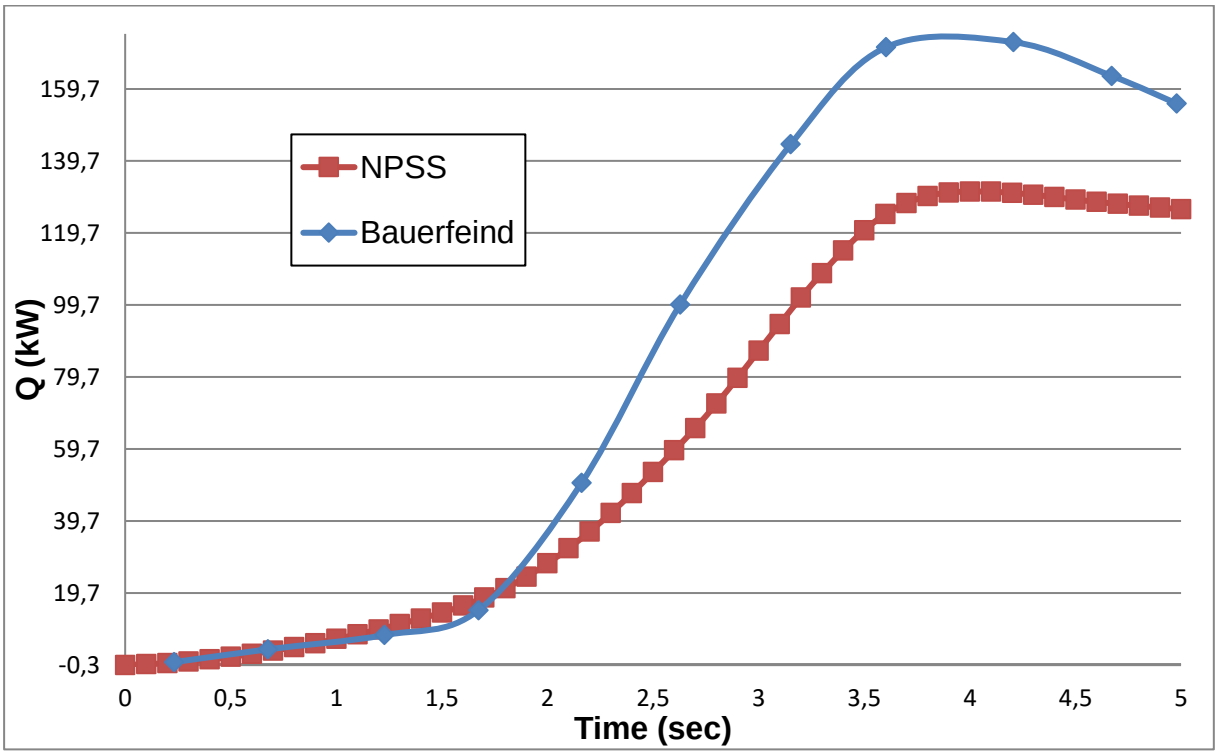

Figure 28: Heat soakage in the LPC

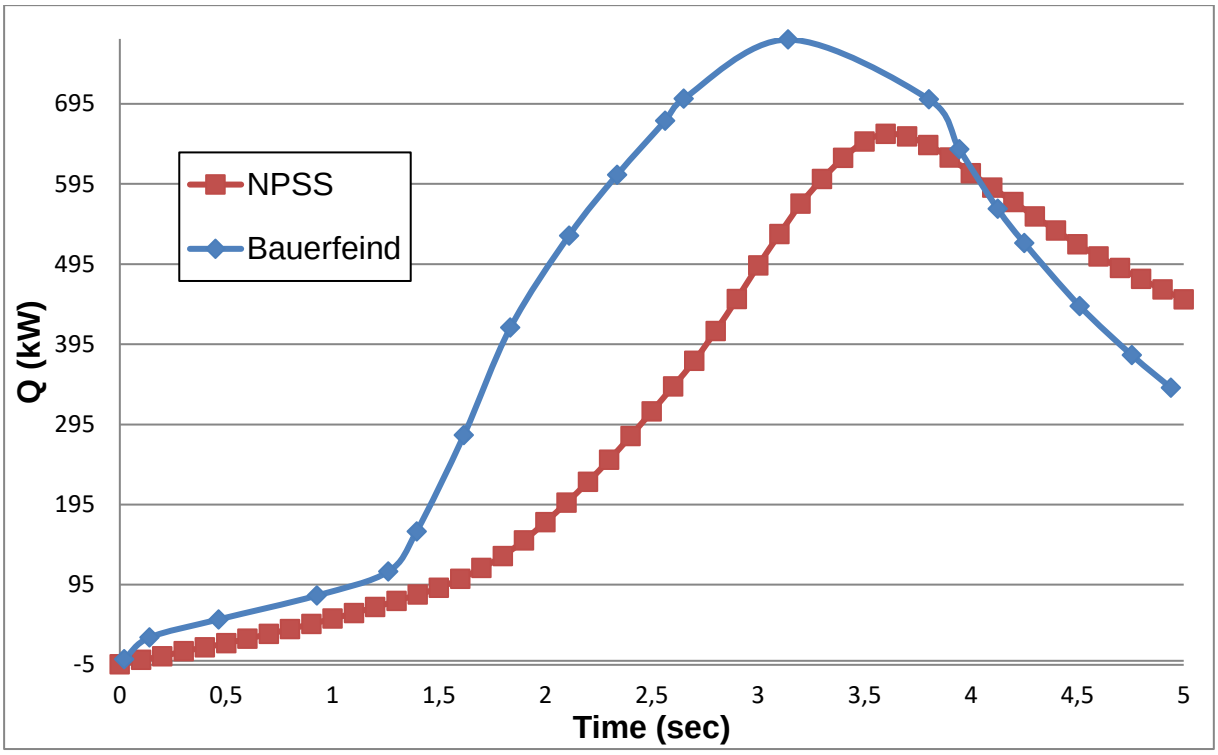

Figure 29: Heat soakage in the HPC

There is a mismatch in heat soakage graphs between NPSS and [1]. The compressors have a similar profile: Before t = 1.5 sec, the simulation follows the theory, after that time they split up. That means there are less heat exchanges in NPSS simulation. Mistakes in assumption for the constants might be a source of deviation. In fact, transfer areas, masses and heat transfer coefficients of the components are only guesses based on [1]. Moreover, the engine dimensions are known but not the components dimensions; it is then impossible to guess a transfer area. Formula 11 gave Q proportional to h, A, T(t) and Tgas so changings have been done in the constants to see if the results were influenced. As a result, the curves are slightly driven up, which indicates that there might be mistakes in the assumptions made. The low pressure compressor is situated in the primary cold section. It is the components where there is least heat soakage and as it is situated at the beginning of the engine, assumption mistake chances are reduced for the mass flow for instance. It can be seen that the curve of this component is closer to [1] than the high pressure compressor curve.

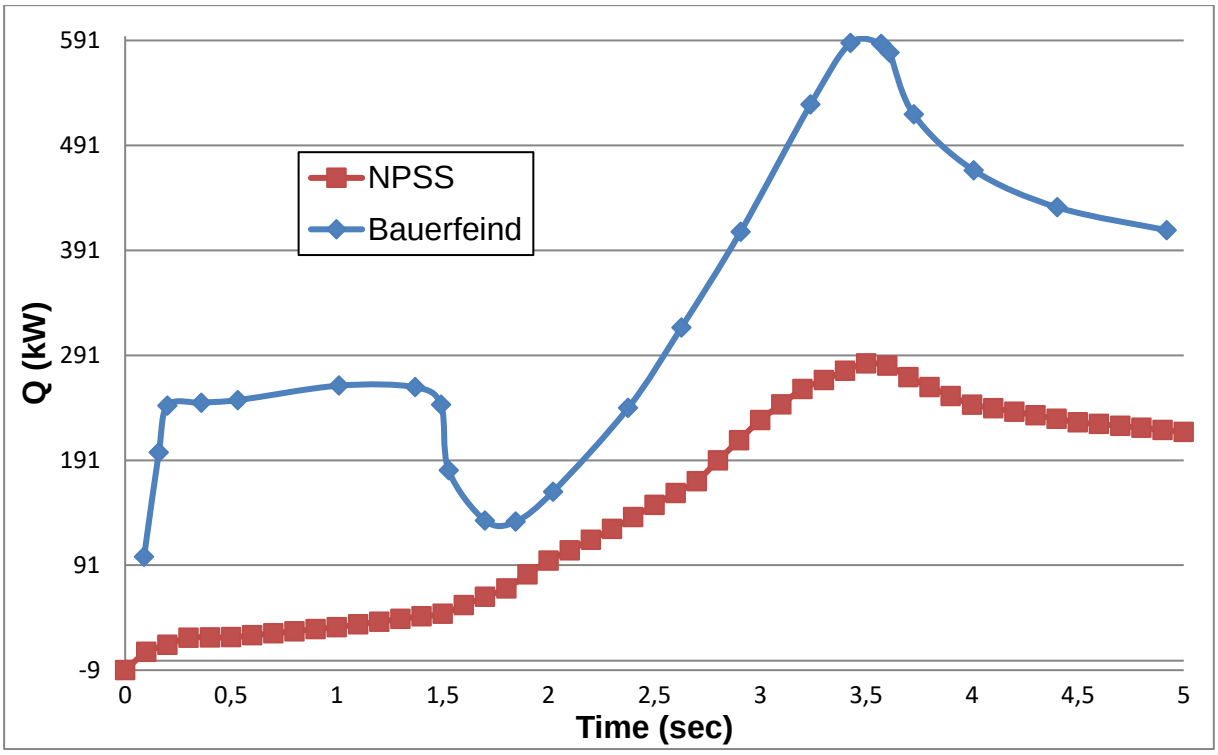

Figure 30: Heat soakage in the LPT

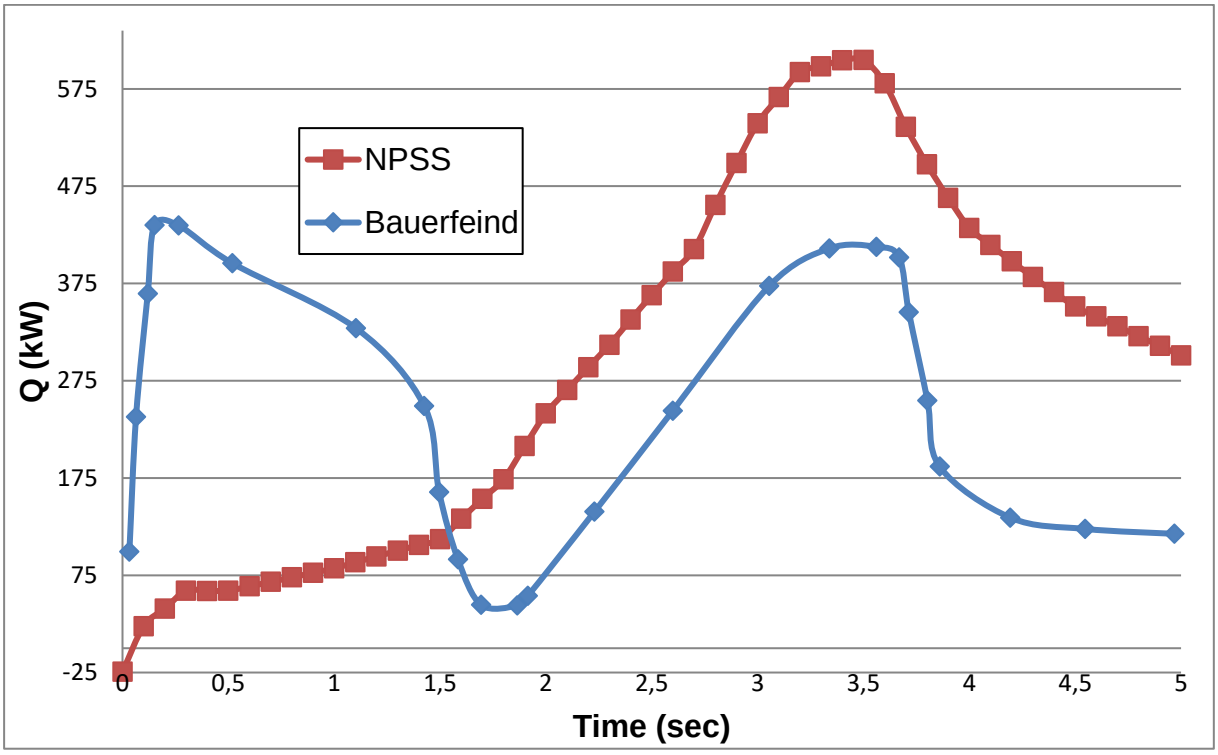

Figure 31: Heat soakage in the HPT

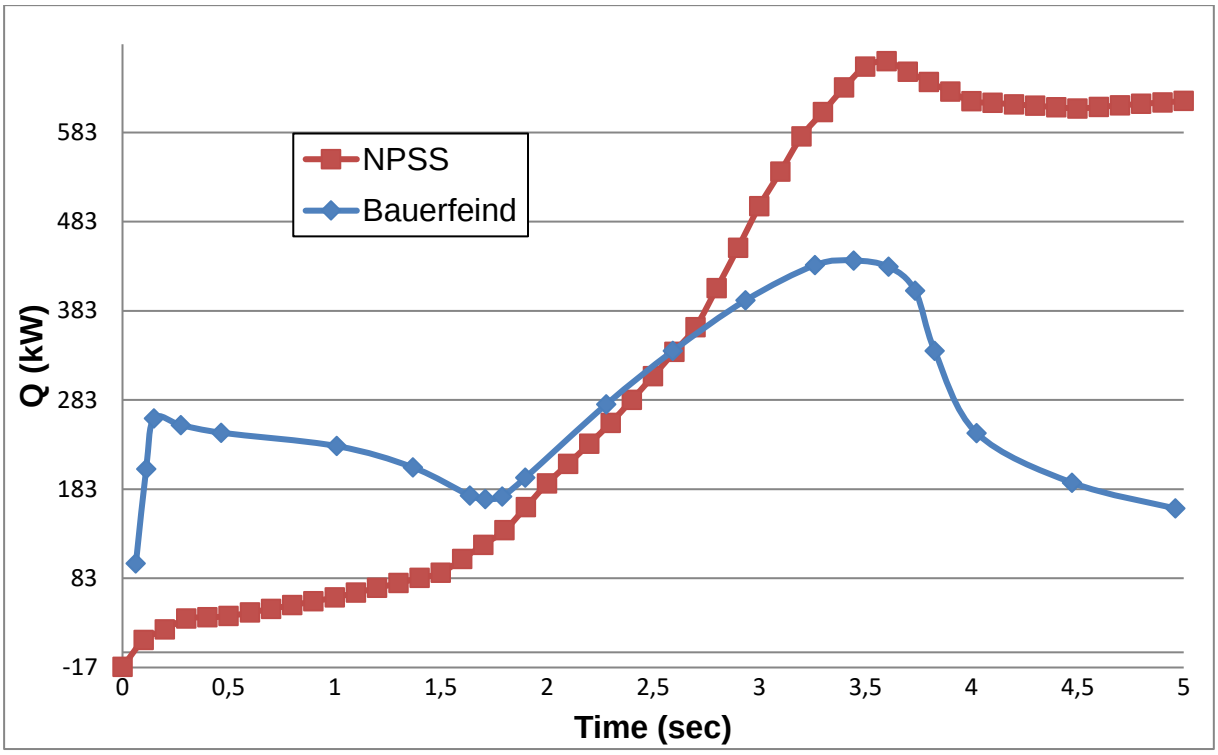

Figure 32: Heat soakage in the Burner

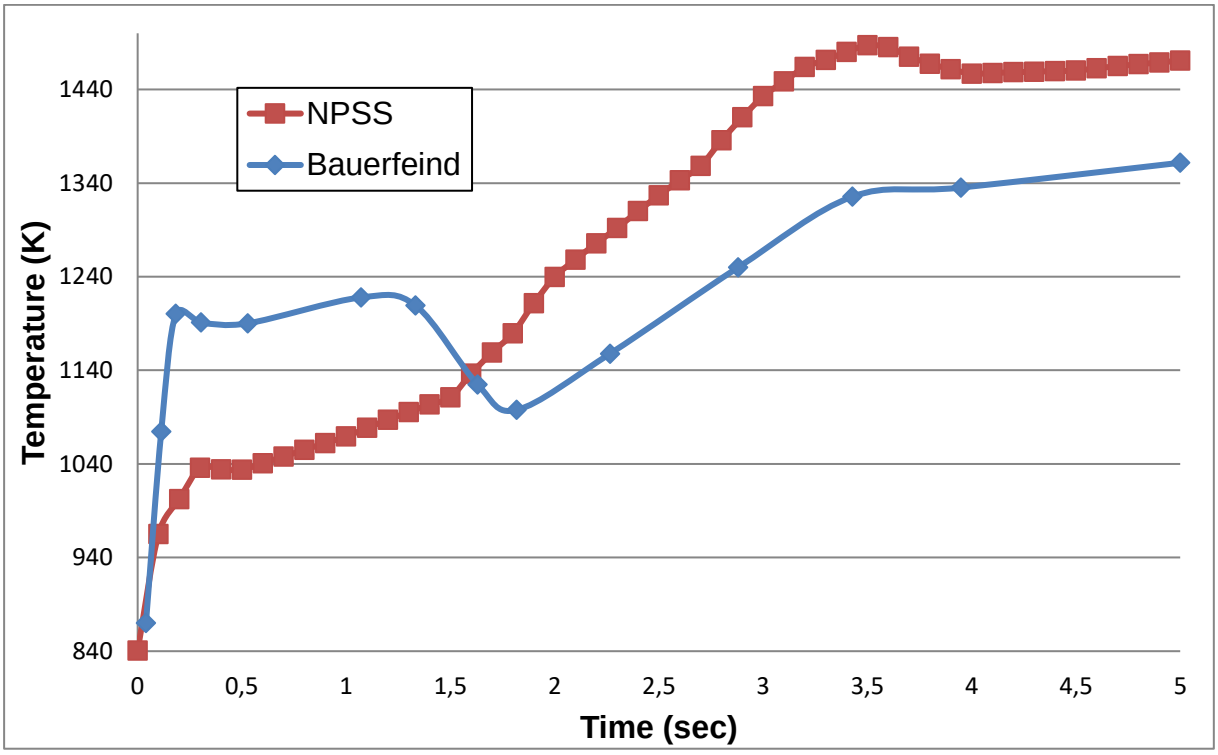

Figure 33: Temperature T4

Concerning the heat soakage in the turbines and the burner, there is an even bigger mismatch than the compressors' curves. As it can be seen, blue curves present two peaks and red curves only one. The NPSS simulation curves increase from t = 0 second to t = 3.5 seconds while [1] curves increase for 0.2 seconds, then decrease until t = 1.8 seconds, and then the second peak is situated at t = 3.5 seconds like NPSS curves. The presence of the peak in [1] between t = 0 and t = 1.8 seconds is unexplained. It is also not understood why the NPSS simulation is below the curve from [1] in the low pressure turbine and above it in the high pressure turbine. The two components presented in figures 31 and 32 are situated in the hot section of the engine. Both of them present a heat flux Q around 580 kW at the peak; the low pressure turbine, though, would have been expected to present a higher heat flux. There is 4% of the core engine incoming air flow that pass through the air cooling system and 18.65% of the air flow that is going to the bypass channel after the high pressure compressor. That means that there is globally 22.65% less air mass flow in the burner and 18.65% less after the high pressure compressor. This might lead to deviation in the NPSS model as it is an assumption and the air mass flow in those components is not specified in [1].

Nevertheless, it can be noticed that heat soakage in the burner in figure 31 looks very similar to the fuel flow profile (figure 19) and this is expected: The gas temperature rises with the fuel flow increase and so are the heat transfers. This temperature increase is seen well in figure 32 which represents the temperature after the burner (Station 4). The differences between the model and the NPSS simulation raise the question of the boundary conditions for the two cases. If they are not the same, it might explain these differences. In [1] are some measures and then simulation of global behaviors for the different parameters. Assumptions might have been done as it has been done in NPSS. Therefore, the two models could easily drift apart from each other.

Beyond that, the transient results in NPSS are coherent between them. Temperature increases, net thrust follows [1] model. In heat soakage graphs, the peak in NPSS situated around time t = 3.5 sec is matching the peak of the fuel flow (figure 19) which is also situated at t = 3.5 sec. The high pressure compressor, the high pressure turbine and the burner have the most important effect in heat soakage as it is expected.

Conclusion

This internship consisted in studying a specific engine, the Rolls-Royce/MAN Turbo RB153. For that, the commercial tool NPSS has been used. The literature investigations represented a great part of the work. Bauerfeind's document has been the basis of the internship. Without that dissertation, the engine model could not have been set as there is a very few data available about this motorization due to the cancellation of the project VJ101D. Moreover, the fact that the engine study is complete is a valuable thing; it is complicated to find transient data and Bauerfeind gathered calculation and results in his document.

First, the stationary model has been built and the different parameters have been adjusted to match the theory. Design calculation fits well in the model and Off-Design calculation presents a slight difference although it is acceptable.

Then, using the model, transient calculation has been run. This consisted in changing the fuel flow and analyzing the engine response. The solver had to be set differently for an acceleration or a deceleration: a low thrust steady-state point for the first and a high thrust steady-state point for the second. Some elements needed adjustments to run properly in transient operation: The shafts needed inertia for example. The pressure losses were now taken into account in the simulation.

Finally, heat soakage has been a great part of the study. Theory has been investigated and so has been the way to add it in the NPSS model. Once again, Bauerfeind's document helped to set the parameters needed for the module that NPSS uses to determine heat effects. The influence of heat soakage has been demonstrated, it is not correct to neglect it in a complete engine study. The fact that the heat transfers should represents around 30% of the excess energy in acceleration has been experimentally verified.

Transient results and particularly heat soakage results are valuable. Even if they are a bit distant from Bauerfeind's values, they show that the model works and that the RB153 engine model is fully functional. The results are coherent, a complete stationary and transient study has been performed and the model constitutes a complete database.

Bibliography

[1] Klaus Bauerfeind. *Die exakte Bestimmung des Übertragungsverhaltens von Turbostrahltriebwerken unter Berücksichtigung des instationären Verhaltens seiner Komponenten.* Fakultät für Maschinenwesen und Elektrotechnik der Technischen Hochschule München. PhD thesis. 1968.

[2] E.H. Hirschel, H. Prem, G. Madelung. *Aeronautical research in Germany – From Lilienthal until today*. Berlin, 2004.

[3] K. von Gersdorff, K. Grassmann, H. Schubert. *Flugmotoren und Strahltriebwerke*. Bonn, 1995.

[4] R.-G Becker, F. Wolters, M. Nauroz, T. Otten. *Development of a Gas Turbine Performance Code and its Application to Preliminary Engine Design*, DGLRK 2011, 27.-29. Sept, Bremen, Germany.

[5] F. Wolters, R.-G. Becker. *Engine performance simulation of the integrated V2527-Engine Fan*. 54th AIAA Aerospace Sciences Meeting, 2016.

[6] P. P. Walsh, P. Fletcher. *Gas turbine performance - second edition*. Oxford, 2004.

[7] Joachim Kurzke. *Design and Off-Design performance of gas turbines*. GasTurb 12 User Guide. Germany, 2015.

[8] NPSS consortium. *NPSS User Guide.* USA, 2013.

[9] IEA/OECD. *Transport, Energy and CO2: Moving toward sustainability*. Paris, 2009.

Appendix

A- Design calculation

Variable	Unit	Bauerfeind	NPSS Design	Relative delta
S2.T	[K]	387	387.057	0.01%
S20.T	[K]	387	387.057	0.01%
S3.T	[K]	725	726.933	0.27%
S4.T	[K]	1377	1365.542	-0.83%
S40.T	[K]	1048	1048.911	0.09%
S5.T	[K]	900	905.318	0.59%
S51.T	[K]	900	905.318	0.59%
S6.T	[K]	698	704.38	0.91%

Table 6: Temperature deviation in Design

B- Off-Design calculation

Variable	Unit	Bauerfeind	NPSS Off-Design	Relative delta
S2.T	[K]	301	299.31	-0.56%
S20.T	[K]	301	299.31	-0.56%
S3.T	[K]	430	493.653	14.80%
S4.T	[K]	880	910.297	3.44%
S40.T	[K]	700	675.931	-3.44%
S5.T	[K]	670	659.049	-1.63%
S51.T	[K]	670	659.049	-1.63%
S6.T	[K]	480	547.784	14.12%

Table 7: Temperature deviation in Off-Design

C- NPSS model

RB153.mdl

```
setThermoPackage("GasTbl");
#include <bleed_macros.fnc>

//Ambient Amb
Element Ambient Amb {
	alt_in=0.0;
	Ps_in {value=14.7; units="psia";};
	Ts_in {value=288.15; units="K";};
}
//InletStart InletStart
real W_in_SI {value =55; units="kg/sec";}
real W_in_US {value = convertUnits("W_in_SI", "lbm/sec"); units = "lbm/sec";}

Element InletStart InletStart{
	AmbientName = "Amb";
	W_in = W_in_US;
}
//Inlet Inl
Element Inlet Inl {
}
//Low pressure Compressor CmpL
real Ahx_SI_CmpL {value =5.45; units="m2";}
real Ahx_US_CmpL {value = convertUnits("Ahx_SI_CmpL", "in2"); units = "in2";}
real Chx_SI_CmpL {value =1050; units="W/(m2*K)";}
real Chx_US_CmpL {value = convertUnits("Chx_SI_CmpL", "Btu/(sec*in2*R)"); units =
"Btu/(sec*in2*R)";}
real Cpmat_SI_CmpL {value =950; units="J/(kg*K)";}
real Cpmat_US_CmpL {value = convertUnits("Cpmat_SI_CmpL", "Btu/(lbm*R)"); units =
"Btu/(lbm*R)";}
real Kcmat_SI_CmpL {value =150; units="W/(m*K)";}
real Kcmat_US_CmpL {value = convertUnits("Kcmat_SI_CmpL", "Btu/(sec*in*R)"); units
= "Btu/(sec*in*R)";}
real Massmat_SI_CmpL {value =130; units="kg";}
real Massmat_US_CmpL {value = convertUnits("Massmat_SI_CmpL", "kg"); units = "kg";}

Element Compressor CmpL {
	#include "Boosterlpc.map";
	PRdes=2.4;
	effDes=0.85;

	Subelement ThermalMass S_Qhx {
		setOption ("switchLagIn", "PHYSICAL");
		Ahx = Ahx_US_CmpL;
		ChxDes = Chx_US_CmpL;
		CpMat = Cpmat_US_CmpL;
		kcMat = Kcmat_US_CmpL;
		massMat = Massmat_US_CmpL;
		wtdAvg_Fl = 0.55;
	}
}
//Splitter Splt
Element Splitter Splt {
	BPRdes=0.7;
}
//High Pressure Compressor CmpH
real Ahx_SI_CmpH {value =4; units="m2";}
real Ahx_US_CmpH {value = convertUnits("Ahx_SI_CmpH", "in2"); units = "in2";}
real Chx_SI_CmpH {value =3350; units="W/(m2*K)";}
real Chx_US_CmpH {value = convertUnits("Chx_SI_CmpH", "Btu/(sec*in2*R)"); units =
```

```
"Btu/(sec*in2*R)";}
real Cpmat_SI_CmpH {value =520; units="J/(kg*K)";}
real Cpmat_US_CmpH {value = convertUnits("Cpmat_SI_CmpH", "Btu/(lbm*R)"); units =
"Btu/(lbm*R)";}
real Kcmat_SI_CmpH {value =150; units="W/(m*K)";}
real Kcmat_US_CmpH {value = convertUnits("Kcmat_SI_CmpH", "Btu/(sec*in*R)"); units
= "Btu/(sec*in*R)";}
real Massmat_SI_CmpH {value =130.7; units="kg";}
real Massmat_US_CmpH {value = convertUnits("Massmat_SI_CmpH", "kg"); units = "kg";}

Element Compressor CmpH {
	#include "5st_hpc.map";
	PRdes=7.5;
	effDes=0.85;

	Subelement ThermalMass S_Qhx {
		setOption ("switchLagIn", "PHYSICAL");
		Ahx = Ahx_US_CmpH;
		ChxDes = Chx_US_CmpH;
		CpMat = Cpmat_US_CmpH;
		kcMat = Kcmat_US_CmpH;
		massMat = Massmat_US_CmpH;
		wtdAvg_Fl = 0.81;
	}
}
Element Bleed BldDct {
}
Element FuelStart FusEng {
}
//Burner BrnPri
real Ahx_SI_BrnPri {value =0.78; units="m2";}
real Ahx_US_BrnPri {value = convertUnits("Ahx_SI_BrnPri", "in2"); units = "in2";}
real Chx_SI_BrnPri {value =3150; units="W/(m2*K)";}
real Chx_US_BrnPri {value = convertUnits("Chx_SI_BrnPri", "Btu/(sec*in2*R)"); units
= "Btu/(sec*in2*R)";}
real Cpmat_SI_BrnPri {value =520; units="J/(kg*K)";}
real Cpmat_US_BrnPri {value = convertUnits("Cpmat_SI_BrnPri", "Btu/(lbm*R)"); units
= "Btu/(lbm*R)";}
real Kcmat_SI_BrnPri {value =25; units="W/(m*K)";}
real Kcmat_US_BrnPri {value = convertUnits("Kcmat_SI_BrnPri", "Btu/(sec*in*R)");
units = "Btu/(sec*in*R)";}
real Massmat_SI_BrnPri {value =139; units="kg";}
real Massmat_US_BrnPri {value = convertUnits("Massmat_SI_BrnPri", "kg"); units =
"kg";}

Element Burner BrnPri {
	effBase=0.98;
	FAR=0.0196;
	dPqP_dmd = 0.04;
	switchBurn="FAR";

	Subelement ThermalMass S_Qhx {
		setOption ("switchLagIn", "PHYSICAL");
		Ahx = Ahx_US_BrnPri;
		ChxDes = Chx_US_BrnPri;
		CpMat = Cpmat_US_BrnPri;
		kcMat = Kcmat_US_BrnPri;
		massMat = Massmat_US_BrnPri;
		wtdAvg_Fl = 0.5;
	}
}

//High Pressure Turbine TrbH
real Ahx_SI_TrbH {value =0.75; units="m2";}
real Ahx_US_TrbH {value = convertUnits("Ahx_SI_TrbH", "in2"); units = "in2";}

real Chx_SI_TrbH {value =3150; units="W/(m2*K)";}
real Chx_US_TrbH {value = convertUnits("Chx_SI_TrbH", "Btu/(sec*in2*R)"); units =
```

```
"Btu/(sec*in2*R)";}

real Cpmat_SI_TrbH {value =520; units="J/(kg*K)";}
real Cpmat_US_TrbH {value = convertUnits("Cpmat_SI_TrbH", "Btu/(lbm*R)"); units =
"Btu/(lbm*R)";}

real Kcmat_SI_TrbH {value =25; units="W/(m*K)";}
real Kcmat_US_TrbH {value = convertUnits("Kcmat_SI_TrbH", "Btu/(sec*in*R)"); units
= "Btu/(sec*in*R)";}

real Massmat_SI_TrbH {value =18.2; units="kg";}
real Massmat_US_TrbH {value = convertUnits("Massmat_SI_TrbH", "kg"); units = "kg";}

Element Turbine TrbH {
       #include "hptE3.map";
       PRbase=3.5;
       effDes=0.88;

       Subelement ThermalMass S_Qhx {
              setOption ("switchLagIn", "PHYSICAL");
              Ahx = Ahx_US_TrbH;
              ChxDes = Chx_US_TrbH;
              CpMat = Cpmat_US_TrbH;
              kcMat = Kcmat_US_TrbH;
              massMat = Massmat_US_TrbH;
              wtdAvg_Fl = 0.5;
       }
}
//Low Pressure Turbine TrbL
real Ahx_SI_TrbL {value =1.2; units="m2";}
real Ahx_US_TrbL {value = convertUnits("Ahx_SI_TrbL", "in2"); units = "in2";}

real Chx_SI_TrbL {value =850; units="W/(m2*K)";}
real Chx_US_TrbL {value = convertUnits("Chx_SI_TrbL", "Btu/(sec*in2*R)"); units =
"Btu/(sec*in2*R)";}

real Cpmat_SI_TrbL {value =520; units="J/(kg*K)";}
real Cpmat_US_TrbL {value = convertUnits("Cpmat_SI_TrbL", "Btu/(lbm*R)"); units =
"Btu/(lbm*R)";}

real Kcmat_SI_TrbL {value =25; units="W/(m*K)";}
real Kcmat_US_TrbL {value = convertUnits("Kcmat_SI_TrbL", "Btu/(sec*in*R)"); units
= "Btu/(sec*in*R)";}

real Massmat_SI_TrbL {value =17.5; units="kg";}
real Massmat_US_TrbL {value = convertUnits("Massmat_SI_TrbL", "kg"); units = "kg";}

Element Turbine TrbL {
       #include "lptE3.map";
       PRbase= 3;
       effDes=0.87;

       Subelement ThermalMass S_Qhx {
              setOption ("switchLagIn", "PHYSICAL");
              Ahx = Ahx_US_TrbL;
              ChxDes = Chx_US_TrbL;
              CpMat = Cpmat_US_TrbL;
              kcMat = Kcmat_US_TrbL;
              massMat = Massmat_US_TrbL;
              wtdAvg_Fl = 1;
       }
}
//Primary Duct DctPri
Element Duct DctPri {
       switchDP = "INPUT";
       dPqP_in=0.01;
}
//Mixer Mix
```

```
Element Mixer Mix {
	switchDesStream="1";
	Fl_I1.MN = 0.6;
}
//Secondary Duct DctSec
Element Duct DctSec {
	switchDP = "INPUT";
	dPqP_in=0.015;
}
//Nozzle NozPri
Element Nozzle NozPri {
	PsExhName= "Amb.Ps";
	switchType="CONIC";
}
//FlowEnd FeAir
Element FlowEnd FeAir {
}
//Low Pressure Shaft
real LowInertia_SI {value =2; units="kg*m2";}
real LowInertia_US {value = convertUnits("LowInertia_SI", "slug*ft2"); units =
"slug*ft2";}

Element Shaft ShL {
	Nmech=14550;
	ShaftInputPort MeCmpL, MeTrbL;
	fracLoss=0.01;
	inertia= LowInertia_US;
}
//High Pressure Shaft
real HighInertia_SI {value =1.5; units="kg*m2";}
real HighInertia_US {value = convertUnits("HighInertia_SI", "slug*ft2"); units =
"slug*ft2";}

Element Shaft ShH {
	Nmech=16260;
	ShaftInputPort MeCmpH, MeTrbH;
	fracLoss = 0.02;
	inertia= HighInertia_US;
}
Element EngPerf Perf {
}

//Fluid Links
//Ambient to Inlet
linkPorts("InletStart.Fl_O", "Inl.Fl_I", "S0");

//Primary Cold Section
linkPorts("Inl.Fl_O", "CmpL.Fl_I", "S1");
linkPorts("CmpL.Fl_O", "Splt.Fl_I", "S2");
linkPorts("Splt.Fl_O1", "CmpH.Fl_I", "S20");
linkPorts("FusEng.Fu_O", "BrnPri.Fu_I", "S3f");
linkPorts("CmpH.Fl_O", "BrnPri.Fl_I", "S3");

//Primary Hot Section
linkPorts("BrnPri.Fl_O", "TrbH.Fl_I", "S4");
linkPorts("TrbH.Fl_O", "TrbL.Fl_I", "S40");

//Mixer Section
linkPorts("TrbL.Fl_O", "DctPri.Fl_I", "S5");
linkPorts("DctPri.Fl_O", "Mix.Fl_I1", "S51");
linkPorts("Splt.Fl_O2", "BldDct.Fl_I", "S50");
linkPorts("BldDct.Fl_O", "Mix.Fl_I2", "S52");

//End Section
linkPorts("Mix.Fl_O", "DctSec.Fl_I", "S6");
linkPorts("DctSec.Fl_O", "NozPri.Fl_I", "S7");
linkPorts("NozPri.Fl_O", "FeAir.Fl_I", "S8");
```

```
//Shaft Links
//High Pressure Components
linkPorts("CmpH.Sh_O", "ShH.MeCmpH", "MeCmpH");
linkPorts("TrbH.Sh_O", "ShH.MeTrbH", "MeTrbH");

//Low Pressure Components
linkPorts("CmpL.Sh_O", "ShL.MeCmpL", "MeCmpL");
linkPorts("TrbL.Sh_O", "ShL.MeTrbL", "MeTrbL");

//Air cooling system
//From CmpH to TrbH
linkBleedCT("CmpH", "TrbH", 0.04, 1, 1, 1, 1, "Bld1");

linkBleedCB("CmpH", "BldDct", 0.0, 0.9, 0.9, "Bld2");
//linkBleedCB("CmpH", "BldDct", 0.01, 0.5, 0.5, "Bld2");

#include "controls.cmp";
```

RB153.run

```
#include "RB153.mdl"
#include "printRB153.view"
#include "RB153solverparams.inc"

//Switch to DESIGN calculation
setOption( "switchDes", "DESIGN" );
autoSolverSetup();

//Add dependents to the solver
solver.addDependent("dep_FN");
solver.addDependent("dep_effPolyH");
solver.addDependent("dep_effPolyL");

//Add independents to the solver
solver.addIndependent("ind_FAR");
solver.addIndependent("ind_effDesH");
solver.addIndependent("ind_effDesL");

CASE++;
run();
printRB153.update();

//Switch to OFF-DESIGN calculation
setOption ("switchDes", "OFFDESIGN");
autoSolverSetup();

//Add dependent to the solver
solver.addDependent("dep_FNOD");

//Add independent to the solver
solver.addIndependent("ind_FAR_OD");


real FN[] = {5700, 5800, 7060}; // in lbf
int FNLoop=0;

for (FNLoop=0; FNLoop<FN.entries();FNLoop++) {
       FN_OD=FN[FNLoop];

       CASE++;
       run();
       printRB153.update();
}

//Low thrust steady-state point
real FNSI {value= 3000; units="N";}
real FNOD {value=convertUnits("FNSI", "lbf"); units="lbf";}
```

```
FN_OD = FNOD;

CASE++;
run();
printRB153.update();

//Remove the solver Off-Design variables
solver.removeIndependent("ind_FAR_OD");
solver.removeDependent("dep_FNOD");


//Switch to TRANSIENT calculation
setOption("solutionMode", "TRANSIENT");
autoSolverSetup();

//Include CaseViewer file
#include "printTransient.view"
solver.postExecutionSequence.append("CaseView");

//Set the dependent variable RunCondition
RunCondition.eq_rhs = "TB_FuelSchedule(time)";

//Add independent to the solver
solver.addIndependent("ind_FAR_TR");

//Add dependent to the solver
solver.addDependent("RunCondition");

transient.stopTime = 5;
transient.baseTimeStep = 0.1;
time = -0.1;

CASE++;
run();
```

RB153solverparams.inc

```
//Independent parameters

Independent ind_FAR {
	varName="BrnPri.FAR";
	indepRef = "0.019";
	dxLimit = 0.2;
	dxLimitType = "FRACTIONAL";
	perturbation = 0.01;
	perturbationType = "FRACTIONAL";
	description = "vary the fuel flow to achieve the desired net thrust";
}

//Dependent parameters

real FN_in_SI {value= 31400; units="N";}
real FN_in {value=convertUnits("FN_in_SI", "lbf"); units="lbf";}

Dependent dep_FN {
	eq_rhs = "FN_in";
	eq_lhs = "Perf.Fn";
	eq_Ref = "FN_in";
	toleranceType = "FRACTIONAL";
	description = "desired FN";
}

//Independent

Independent ind_FAR_OD {
	varName="BrnPri.FAR";
	indepRef = "0.019";
```

```
        dxLimit = 0.2;
        dxLimitType = "FRACTIONAL";
        perturbation = 0.01;
        perturbationType = "FRACTIONAL";
        description = "vary the fuel flow to achieve the desired afterburner
temperature";
}

//Dependent

real FN_in_OD {value= 2950; units="N";}
real FN_OD {value=convertUnits("FN_in_OD", "lbf"); units="lbf";}

Dependent dep_FNOD {
        eq_rhs = "FN_OD";
        eq_lhs = "Perf.Fn";
        eq_Ref = "FN_OD";
        toleranceType = "FRACTIONAL";
        description = "desired afterburner temperature";
}

Independent ind_FAR_TR {
        varName="BrnPri.FAR";
        indepRef = "0.006";
        dxLimit = 0.2;
        dxLimitType = "FRACTIONAL";
        perturbation = 0.01;
        perturbationType = "FRACTIONAL";
        description = "vary the fuel flow to achieve the desired afterburner
temperature";
}

//Table for the fuel flow in transient acceleration
Table TB_AccSchedule( real myTime) {
    myTime = {0, 0.1, 0.3, 0.5, 1, 1.5, 1.8, 2, 2.3, 2.7, 3, 3.2, 3.4, 3.5, 3.6,
3.7, 4, 4.5, 5, 6, 7, 8, 9, 10}
    WfuelTab = {0.20, 0.28, 0.34, 0.35, 0.40, 0.46, 0.55, 0.64, 0.75, 0.91, 1.09,
1.19, 1.26, 1.29, 1.30, 1.29, 1.26, 1.26, 1.28, 1.28, 1.28, 1.28, 1.28, 1.28}

    myTime.interp = "linear" ;
    myTime.extrap = "linear" ;
}

//Table for the fuel flow in transient deceleration
Table TB_DecSchedule( real myTime) {
    myTime = {0.0, 0.35, 0.6, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5, 6, 7, 8,
9, 10}
    nHTab = {1.2348, 0.41895, 0.27, 0.19845, 0.16758, 0.15435, 0.14994, 0.147735,
0.14994, 0.15435, 0.16317, 0.165375, 0.165375, 0.165375, 0.165375, 0.165375,
0.165375}

    myTime.interp = "linear" ;
    myTime.extrap = "linear" ;
}

Dependent RunCondition {
        eq_lhs = "BrnPri.Wfuel";
}
```

Controls.cmp

```
//Element Control CONTROL {
Element CONTROL {
  // Variables and limits
  real HPCXNredStdRel;
  real HPCHandlingWQWin;
  real WQWinCalculated;
  real HPCXNredStdRel_Transient;
```

```
real HPCHandlingWQWin_Transient;
real WqWinCalculated_Transient;

void variableChanged(string name, string value) {
}

Option switchDes {
  allowedValues = { "DESIGN", "OFFDESIGN" }
  value = "DESIGN";
}

Option solutionMode {
  allowedValues = { "TRANSIENT", "STEADY_STATE", "ONE_PASS" }
  value = "STEADY_STATE";
}

// Get data from engine to control
void mapIn() {
  HPCXNredStdRel = CmpH.NcqNcDes;
}

void mapInTransient() {
     HPCXNredStdRel_Transient = CmpH.NcqNcDes;
}

// Get data from control to engine
void mapOut() {
  HPCHandlingWQWin = WQWinCalculated;
  Bld2.fracBldW = HPCHandlingWQWin;
}

void mapOutTransient() {
     HPCHandlingWQWin_Transient = WqWinCalculated_Transient;
     Bld2.fracBldW = HPCHandlingWQWin_Transient;
}

void calculate() {
  //steady state mode
  if ( solutionMode == "STEADY_STATE" ) {
    mapIn();
    steadyState();
    mapOut();
  }

     else if ( solutionMode == "TRANSIENT" ) {
       mapInTransient();
       Transient();
       mapOutTransient();
     }
}
// Steady state
// read from schedule
void steadyState() {
  WQWinCalculated = TableSASHPCHandlingWQWin( HPCXNredStdRel );
}

void Transient() {
     WqWinCalculated_Transient = TableSASHPCHandlingWQWin( HPCXNredStdRel );
}

// Schedule of comp speed vs power code
Table TableSASHPCHandlingWQWin( real XNredStdRel ) {
  XNredStdRel = { 0.9, 0.95, 0.96, 0.961, 1.0 }
  TableSASHPCHandlingWQWin = { 0.2, 0.18, 0.15, 0.0, 0.0 }
}
```