Development and Validation of a Civil Aircraft Engine Simulation Model for Advanced Controller Design

Thesis submitted for the degree of

Doctor of Philosophy

at the University of Leicester

by

Sonny Martin

Department of Engineering

University of Leicester

June, 2009

Abstract

Sonny Martin. Development and Validation of a Civil Aircraft Engine Simulation Model for Advanced Controller Design.

This thesis is concerned with the results of a joint academic and industrial study on the development of a detailed nonlinear dynamic model of a turbofan jet engine to be used for research into advanced control strategies for civil turbofan aircraft engines.

The model is representative of a dual shaft engine with variable bleed, variable stator vanes, turbine cooling, heat transfer, and a duct and exhaust nozzle. A switched, gain-scheduled, feedback control system incorporating bumpless transfer and antiwindup functionality has been designed and implemented according to current industrial practice. This baseline implementation permits realistic transient operation of the simulation and may act as a reference design for further control work.

The simulation computes a non-iterative solution, by progressing calculations in the direction of the gas stream flow. Where possible the underlying physics are used and empirical approximations are avoided so that the model requires minimum data. This approach also makes a future inclusion of component failure easier to implement.

The simulation is modular in nature so that engine or control modules can be easily replaced or modified if an improved design becomes available. The Simulink implementation of the control architecture has been redesigned to permit the addition or removal of control loops, also during the simulation's operation, to allow testing of advanced control strategies. The entire controller can also be easily replaced.

A detailed description of the modeling process, the various simulation issues that arise with a model of this complexity, and the results of the overall aero-engine system are presented. The design of the switched, gain-scheduled aero-engine controller with bumpless transfer and antiwindup which achieves dynamic performance that closely matches that of a real aero-engine is also discussed.

Acknowledgements

Grateful acknowledgement is due to the UK Engineering and Physical Sciences Research Council and to Alstom Aerospace for financial assistance.

Contents

1. Introduction **1**

1.1. Overview 1

1.2. Background 3

1.3. Contributions of this research 4

1.4. Structure of the thesis 9

1.5. Summary 11

2. Civil aircraft gas turbines **12**

2.1. Introduction 12

2.2. Gas turbines - an overview 16

2.3. Gas turbine performance 32

2.4. Summary 36

3. Modular construction of the mathematical model **37**

3.1. Gas turbine architecture 37

3.2. Overview of the aero-engine simulation 39

3.3. Comparison with current state-of-the-art engine simulation models 43

3.4. Component models 46

3.5. Summary 66

4. Controller specifications, architecture and design **67**

4.1. Introduction 67

4.2. Controller architecture 73

4.3. Preventing integrator windup 80

4.4. Controller continuous analysis 84

4.5. Reference tracking: requirements and consequences 85

4.6. PI control and gain schedule 86

4.7. Summary 87

5. Model implementation and simulation **89**

5.1. The simulation environment 90

5.2. Initial development 91

5.3. Simulation initialisation 93

5.4. Turbomachinery characteristic maps 96

5.5. Characteristic map scaling 98

5.6. Algebraic loops 109

5.7. Running the simulation 110

5.8. Summary 123

6. Full envelope closed loop model validation **124**

6.1. Introduction 124

6.2. Simulation time step 125

6.3. Closed-loop validation: performance plots 128

6.4. Summary 147

7. Conclusions **148**

7.1. Review of contents 148

7.2. Contributions of this dissertation 150

7.3. Recommendations for future work 153

Appendix **160**

A. Key simulation scripts **160**

B. Key simulation scripts (code) **163**

C. Engine key parameters and corresponding model variables **171**

Bibliography **174**

List of Figures

1.1. Twin shaft turbofan simulation components. 5
1.2. Simulation dashboard 8

2.1. Diagram of the current model's closed-loop system. 15
2.2. Twin shaft turbofan 17
2.3. Twin shaft turbofan schematic. 18
2.4. Twin shaft turbofan simulation components. 18
2.5. Compressor characteristic curve 22
2.6. Axial compressor characteristics 25
2.7. Phenomena at off-design operation 28
2.8. Transient trajectories on compressor characteristic map 30
2.9. Effect of blow-off and increased nozzle area 31

3.1. The main components of a turbofan engine. 39

4.1. Diagram of the current model's closed-loop system. 70
4.2. Selection logic architecture. 75
4.3. Active regulator output during acceleration. 76
4.4. Active regulator output during deceleration. 76
4.5. Controller fuel demand. 77
4.6. Active regulator 78

4.7. Thrust response to large throttle demand. 79
4.8. PI controller structure with integrator anti-windup. 81
4.9. Controller structure as reference tracking 84

5.1. Fan duct characteristic map. 100
5.2. Fan core characteristic map. 100
5.3. LPC characteristic map. 101
5.4. HPC characteristic map. 101
5.5. HPT characteristic map. 102
5.6. LPT characteristic map. 102
5.7. Outline of the map lookup dataflow and scaling 109
5.8. Top level of the Simulink model 112
5.9. Simulink governor section 113
5.10. Simulink engine section 114
5.11. Simulink engine section - detail 115
5.12. Simulink engine section - detail 116
5.13. Simulink engine section - detail 117
5.14. Simulink engine section - detail 118
5.15. Simulink engine section - detail 119
5.16. Simulink engine section - detail 120
5.17. Simulink engine section - detail 121
5.18. Simulink engine section - detail 122

6.1. Relative thrust and demand - case 1 130
6.2. Relative thrust and demand - case 2 131
6.3. Relative thrust and demand - case 3 131
6.4. Regulators upon acceleration - case 1 132
6.5. Regulators upon acceleration - case 2 132

6.6. Regulators upon acceleration - case 3 133
6.7. Regulators upon deceleration - case 1 133
6.8. Regulators upon deceleration - case 2 134
6.9. Regulators upon deceleration - case 3 134
6.10. Engine fuel demand: case 1-3 135
6.11. Active regulator - case 1 135
6.12. Active regulator - case 2 136
6.13. Active regulator - case 3 136
6.14. Fan massflow - core section 138
6.15. Fan massflow (inclusive of bleed air) - duct section 138
6.16. Engine massflow at exit 139
6.17. Variable bleed valves (VBV) - percentage of massflow extracted 139
6.18. VBV position 140
6.19. VSV position 140
6.20. Cooling flow to HPT 141
6.21. Total output power of the turbines 142
6.22. Fan power consumption - percentage of total power output 143
6.23. HPC power consumption - percentage of total power output 143
6.24. LPC power consumption - percentage of total power output 144
6.25. Engine net power 144
6.26. Total engine massflow 145
6.27. HPC pressure ratio 145
6.28. Overall compression ratio 146
6.29. HPT inlet temperature 146

Nomenclature

PR	pressure ratio
LP	low pressure
HP	high pressure
LPC	low pressure compressor
HPC	high pressure compressor
VSV	variable stator vane
ISA	international standard atmosphere
FAR	fuel-air-ratio
BPR	bypass ratio
SLS	sea level static
N1	LP shaft speed
N2	HP shaft speed
EGT	engine exhaust gas temperature
CFD	computational fluid dynamics
T	temperature $[K]$
V	volume $[m^3]$
w	mass of gas stream $[kg]$
w_0	inital value of mass present in plenum $[kg]$
$\dot{w}$	gas stream mass flow $[kg\ s^{-1}]$
R	gas constant $[J\ K^{-1}\ mol^{-1}]$

$\bar{R}$	specific gas constant of the gas stream $[J\ kg^{-1}\ K^{-1}]$
c_p	specific heat at constant pressure $[J\ kg^{-1}\ K^{-1}]$
γ	ratio of specific heats
τ	heatsoak time constant $[s]$
M	metal mass $[kg]$
q	heat flow $[W]$
$\bar{h}$	average heat transfer coefficient over a surface $[W\ m^{-2}\ K^{-1}]$
T_m	metal component average temperature $[K]$
$\bar{T}$	gas stream mean temperature $[K]$
A	heat transfer area $[m^2]$
c_{pm}	metal specific heat capacity $[J\ kg^{-1}\ K^{-1}]$
τ_d	heatsoak time constant established under experimental conditions $[s]$
η_i	isentropic efficiency
η_t	turbine isentropic efficiency
π	pressure ratio
P_x	power consumption of component x $[W]$
p	total pressure $[Pa]$
	(all pressures are total pressure unless otherwise stated)
$p^{'}$	static pressure $[Pa]$
I	moment of inertia $[kg\ m^2]$
ω	angular velocity $[rad\ s^{-1}]$

1. Introduction

This chapter contains a review of previous work on aero-engine modelling, and a summary of the novel contributions of this research.

1.1. Overview

Computer simulation is a powerful tool for the mechanical and control system design of gas turbines. A high fidelity computer simulation can be used as a substitute for a real engine in many applications. For example, it is possible to simulate critical transients that must be avoided on the actual plant due to the risk of damage. Turbine engines can be modelled at various levels of detail, from full 3-D descriptions of the gas path (e.g. NASA's NPSS simulation [23]) that can require distributed computers or supercomputers, to simplified algebraic equations [4] and even simple overall transfer functions. It is generally accepted that a 1-D simulation is sufficient for accurate dynamic performance modeling and therefore controller design.

This thesis presents a full aero-thermodynamic model of a 2-spool, high-bypass turbofan engine with an unmixed exhaust together with a switched, gain-scheduled aero-engine controller with bumpless transfer and anti-windup. The engine simulation in conjunction with this controller achieves dynamic performance representative of that of a real aero-engine. Model implementation is in the Matlab-Simulink® environment.

The simulation method used in this study is known as an aerothermal transient perfor-

mance model [42]. This method avoids iterative calculations by arranging the component equations to follow the direction of the gas path and introducing storage volumes between components to account for the unsteady balance of mass at compressor discharge, combustion chamber and between the turbines [8]. Relative to an iterative model there is some loss of accuracy, but this is negligible for the purpose of engine control system development, particularly if a small time step is used, and is offset by a superior execution time. If a constant time-step is chosen for numerical integration, then this method can provide a model with a predictable run-time. If the simulation is designed to run in realtime then it can also be used with real hardware, although clearly the computer program outputs have to be generated at least as fast as the predicted physical phenomena for the model to run side-by-side with an engine [17].

Features of the model developed in this study are its modularity, ease of implementation and adherence to underlying physics. Empirical approximations have been avoided wherever possible with the aim of improving the model's flexibility and providing physical justification for the approximations used (as opposed to empirical approximations whose boundary of validity is often unknown). Most of the model's implementation details are provided by initialisation scripts, and therefore tuning the model is a simple matter of changing script details without the need to extensively modify Simulink modules, i.e. most variables are dynamically initialised from the Matlab workspace. For example, the gain schedule is provided by variables in the workspace and the parameters used for scaling the performance maps of the turbomachinery components (this is performed online while the simulation runs) are also provided by external scripts. The model is also compatible with the Real-Time-Workshop, a toolbox available in the Matlab-Simulink environment that is able to automatically generate a source code in C language from the Simulink scheme.

The purpose of the model is the development of advanced control strategies, therefore a

baseline controller that closely mirrors industrial practice has been designed to validate the closed-loop performance properties of the model. The controller has itself a modular approach, allowing easy extension of the number of parameters regulated by the controller. The model is provided with a "dashboard" that permits inspection during the simulation's progress of significant parameters such as massflow, temperature and pressure at each engine station.

Full flight-envelope validation of both the model and controller has been performed with the assistance of Alstom Aerospace, with the exception of engine start-up as this is not within the scope of the model.

1.2. Background

There are relatively few papers in the literature that deal specifically with civil aircraft engines [4]. Several papers provide an architecture for simulating gas turbines [1, 5–8, 16, 34, 40], but most are appropriate only to industrial turbines and few provide details of implementations suitable for civil aircraft aero-engines [4].

There are also relatively few papers that discuss in detail the architecture and design of industrially applicable control schemes for civil aircraft engines. Several papers discussing advanced control of military aircraft engines may be found in the literature because these engines have more controller inputs and thus require multivariable controllers; some examples are: [11, 22, 30, 35, 44]. A high-level general introduction to aircraft engine feedback control is contained in [38]. Another useful resource is [34]. The latter has many useful details regarding engine and actuator dynamics, and provides detailed performance figures that are backed up directly by industry, but like [38] this reference also falls short of providing a comprehensive overview of the controller architecture and practical methods of implementation such as would allow replication of the design.

The current model has therefore reimplemented from scratch a cascade-type architecture. The Simulink implementation of this has been completely redesigned into a modular architecture that permits to easily add further cascade branches or additional controls to each branch.

1.3. Contributions of this research

The key contributions are as follows:

- The engine's fan is modelled as two separate sections (a core section and a duct section) with different pressure ratios and efficiencies (see figure 1.1). This permits fan power consumption to be calculated on the basis of pressure ratio rather than as a simple percentage of total power consumption. Fan power consumption as a percentage of total varies with altitude and Mach Number and has a significant transient effect (see figure 6.22). This detail would be absent with a simpler representation. The fan absorbs a significant proportion of the power output by the engine, therefore an accurate model is highly desirable, and keeps the model consistent with the design objective of remaining true to core physics.

- A nozzle module suitable for an aerothermal model (i.e. that does not require iterative procedures) and a model of the duct (as a nozzle) has also been developed - the latter is an inherent requirement to accurately calculate fan power consumption. The conventional approach is to model the nozzle and duct based on empirical compressible flow charts (Q-curves) and efficiency factors, if at all: in some cases the turbine exhausts directly to atmospheric pressure [2], neglecting to take into account the increased pressure at turbine discharge due to nozzle friction.

  A duct nozzle is a necessary requirement if fan power consumption is to be calculated as a function of pressure ratio and not simply as a percentage of power

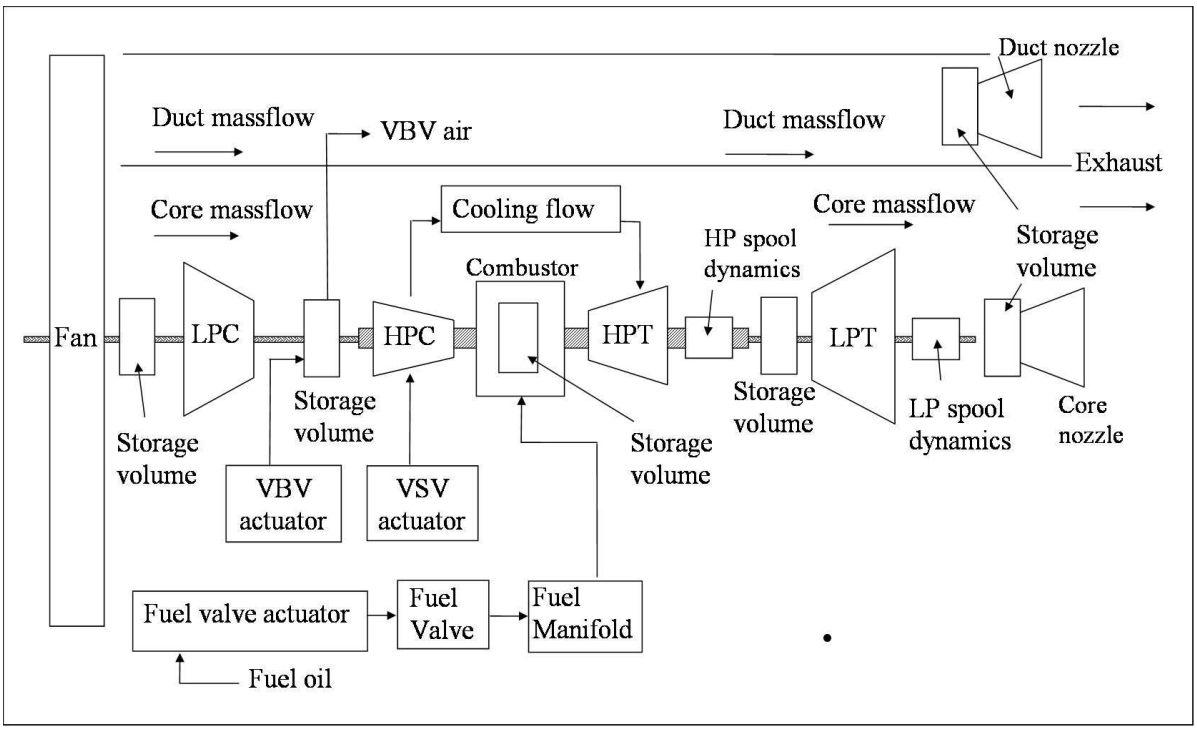


Figure 1.1.: **Twin shaft turbofan simulation components.**

output. The nozzle module is based on first-principles modelling of flow through a convergent nozzle.

- Cooling flow is directed from the high pressure compressor into the high pressure turbine. An accurate representation of this element is important since the total percentage of engine inlet mass extracted before the combustor may be up to 25% for a high technology aero or industrial engine [36, 42], and cooling flow will represent a significant proportion of this (see figure 6.20). Cooling is often modelled simply as a percentage of the engine's massflow, however the current simulation uses a relation that, although partly empirical (the calculation of high speed fluid dynamics is too big an undertaking for the current simulation), is also based on radius of the cooling air extraction vent and the ratio of the air source pressure (the high pressure compressor) to that of the sink (the high pressure turbine) [8, 27] -

and thus has a foundation in physics. This expression allows the model to perform well under a range of conditions, even in the relative absence of data: it is sufficient to provide one reasonable cooling flow data point; the formula can then be used to determine the cooling flow of the engine's entire operational envelope.

- The traditional scaling formulae for characteristic maps have been reformulated: because the simulation's validation requires matching the performance of a target engine, an alternative formulation can better suit our needs, i.e. matching a map position to target pressure, temperature and massflow conditions. The advantage of this scheme is that in these alternative expressions, the map positions and target values are used directly, making it easy and intuitive to tailor the map scaling to a target engine. Furthermore, this new expression is suitable for scaling data at both component inlet and outlet by simply reversing the map and target operating points (further details are in Chapter 5, Section 5.5).

- A feature of the simulation is the transfer of heat between the gas stream and the engine components. This is taken into consideration by providing heat storage modules for each major component. Heat soak has a significant effect upon the dynamics of the engine, particularly at high altitude where the massflow through the engine is greatly reduced (see Chapter 3, Section 3.4.2).

- Industry can benefit from this engine model in several ways: the simulation can be used to implement failure scenarios and because of its modular design based on scripts and the avoidance of empirical approximations, the model can be easily adapted to match the performance of specific engines and used as a validation tool. Indeed, Alstom aerospace has used the model during the course of the past year to evaluate and refine condition monitoring algorithms. The model was used to generate many steady state running points at varying thrust settings, altitudes and

mach numbers. This data was used to train a novel detection tool and then inject various off-normal operating conditions to evaluate the detection algorithms.
A further use, suitable for both industry and academia, is to use the model to assess and prototype novel control schemes: i.e. as a test bed for control design.

- This model is, to the best of the author's knowledge, one of the most fully featured non-proprietary aero-engine models implemented in the Matlab/Simulink environment. This is a familiar platform for control engineers and the graphical nature of Simulink makes it easy to tap into the engine's data streams, without requiring an understanding of the overall simulation. The modular nature of the simulation enables modules to be easily removed and replaced should more accurate designs become available, and the modules themselves can be reused for alternative engine configurations. Scripts enable the model to start at several operating points, and run configurable test profiles. The simulations can be paused, saved and resumed at any time.

- A significant issue during the development of the model was the initialisation of the simulink elements that have a prior state, e.g. all integrators and single time-step delays. The model states are saved in the order that they are required by the simulation, however should the simulation change and elements be added or removed, this order may no longer be valid because Simulink frequently rearranges the order of the states when the simulation is recompiled. This requires a manual rearrangement of intialisation values - which is a matter of some consequence for a development process that can required hundreds of changes and many tests. Therefore a new initialisation script was implemented to keep track of model states via their block address - in this way the states are no longer linked to an order kept internally by the Simulink compiler but are conveniently linked to the appropriate block via an address.

- Another useful addition to the simulation is a dashboard (figure 1.2) that allows the user to view key engine and controller parameters while the simulation is running and adjust them directly from the dashboard. A submodule of the dashboard allows the user to adjust the characteristic map scaling parameters in realtime and view the coordinates of a component's operating point on both the original and scaled turbomachinery map. By default map scaling adjustments are reset upon the next simulation run, but can be retained if required.

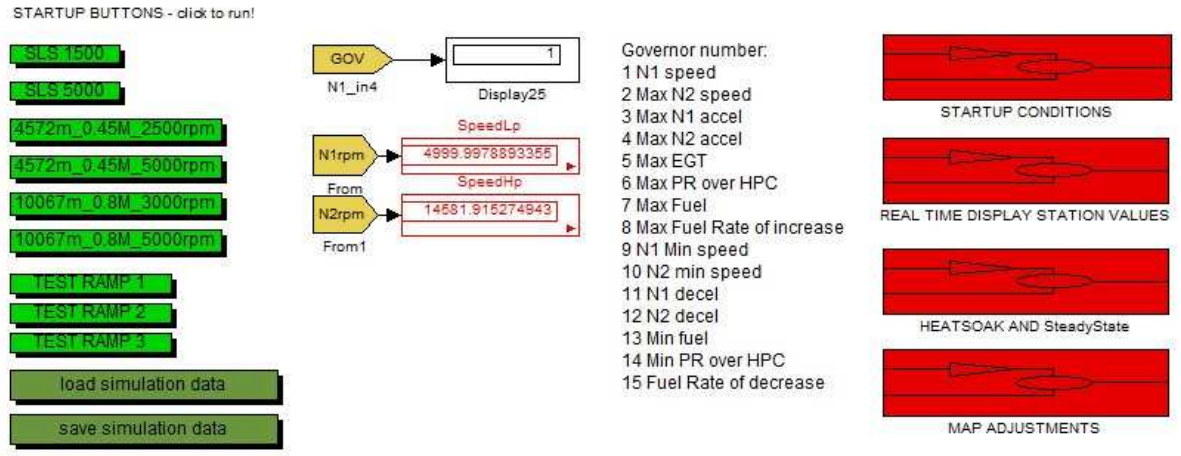


Figure 1.2.: **Top level of the simulation's dashboard.**
The modules in red allow the user to view and modify key engine and controller parameters while the simulation is running.

- A switched, gain-scheduled aero-engine controller with bumpless transfer and anti-windup has been implemented. While the architecture is an industry standard, its implementation in Simulink has been completely redesigned so that it is now easy to add additional control loops and organize a hierarchy of controls. This baseline implementation permits realistic transient operation of the simulation and may act as a reference design for further control work. This architecture is analyzed and plots of its performance and tracking are shown.

  Tests and key plots of the model's steady state and dynamic performance were

reviewed and approved by industrial control engineers at Alstom Aerospace over the course of the project, and published in the leading journal of the field: the ASME Journal of Engineering for Gas Turbines and Power [25]. The overall control system architecture and results were presented at the 2008 American Control Conference [24].

In summary, the contents of the thesis will describe the development of a flexible, feature-rich, fully dynamic model of a relatively generic civil aero engine. The model is particularly suitable for control system design as it enables proper solution of the nonlinear differential equations which describe the engine behaviour, allowing transient behaviour to be simulated accurately. In other available models, dynamic simulation appears to be approximate in nature (i.e. comprising a number of steady state "snapshots" rather than true dynamic behaviour) or else is difficult to enable. In the proposed model, dynamic behaviour is fully catered for, which thus provides the control system designer accurate and important information about the closed-loop system's transient response.

1.4. Structure of the thesis

This dissertation consists of 7 chapters. The contents of each chapter are outlined below.

1.4.1. Chapter 2: Civil aircraft gas turbines

In this chapter, the basic aero-thermodynamic principles of gas turbines in general, and of civil aircraft aero-engines in particular, are introduced. This includes the key phenomenae of surge, surge line displacement and environment effects. Engine

performance requirements are also introduced.

1.4.2. Chapter 3: Modular construction of the mathematical model

Here the architecture of a gas turbine is analyzed in greater detail, presenting the key equations that can be used to model each component, showing how they can be put together, and describing the principles of the simulation architecture. The target aero-engine, a high-bypass separated flow turbofan as used on high subsonic commercial aircraft, is also discussed.

1.4.3. Chapter 4: Controller specifications, architecture and design

This chapter describes the design and implementation of the control system for the gas turbine engine model. The engine controller serves the dual purpose of assisting with simulating engine transients and also as a benchmark for future controls development. The architecture is a switched, gain-scheduled, feedback control system incorporating bumpless transfer and anti-windup functionality.

1.4.4. Chapter 5: Model implementation and simulation

Issues related to the practical implementation of the simulation itself require some explanation. Here we discuss some of the most important practical aspects of translating the theory into a simulation architecture, starting with a brief discussion of the simulation environment and simulation initialisation before proceeding

to one of the more elaborate simulation modules: the real-time scaling and implementation of the turbomachinery characteristic maps.

1.4.5. Chapter 6: Full envelope closed loop model validation

This chapter presents the performance results of some key dynamic and steady-state tests that validate both the engine performance and that of the controller. These tests and key plots were reviewed and approved by industrial control engineers at Alstom Aerospace over the course of the project.

1.4.6. Chapter 7: Conclusions

In conclusion, a summary of the main results presented in the dissertation and an outline of future directions for research in the area of aero-engine simulation, with particular reference to areas relevant to the discipline of control engineering.

1.5. Summary

This chapter provided an overview of the model's key features and contributions. The simulation proceeds by progressing the solution along the direction of the gas stream. The calculated outputs of each component are used as the input variables of the component downstream. A key advantage of this method is that the model has a predictable runtime if a constant time-step of integration is used.
The model is a dual shaft turbofan with a dual section fan, inlet recovery, HPT cooling, heatsoak, and a core and duct nozle. The implementation is modular and the implementation of key functionality is via scripts, permitting the model to be rapidly adapted to new designs.

2. Civil aircraft gas turbines

In this chapter the basic aero-thermodynamic principles of gas turbines in general and in particular that of civil aircraft aero-engines are introduced. This includes the key phenomenae of surge, surge line displacement and environment effects. Engine performance requirements are also introduced.

2.1. Introduction

Almost all of today's jet airliners, for reasons of both fuel economy and reduced noise, are powered by high-bypass turbofans. The engine model developed in this study has general characteristics and is non-proprietary, however a suitable reference is the high-bypass CFM 56-5B(3) engine, the most powerful engine in the CFM 56-5B family. The CFM56-5B is capable of providing up to 32000 lbs of thrust and is the engine of choice for the A320 family, having been selected to power nearly 60 percent of the A318/A319/A320/A321 aircraft ordered. Specifically, the CFM 56-5B(3) powers the A321 which has 185 seats and a range of 5,600 Km - a popular medium range aircraft. The A320 family is a leader in the single-aisle jetliner marketplace and the model is therefore representative of one the most common type of jet engines flying today.

The engine model discussed in this dissertation used as an initial template a

Matlab-Simulink model provided by Alstom Aerospace - although very little of this model now remains, namely the fuel flame temperature lookup table! The Alstom model simulated the Alstom GTX100 gas turbine, a single shaft turbine used for power generation. The current model, having undergone a comprehensive overhaul, is representative of a 2-shaft, high-bypass turbofan engine with an unmixed exhaust. It is able to reproduce the characteristic behaviour of a modern turbofan engine, to include transients, heatsoak, and altitude and frontal velocity effects. Gas dynamic effects are neglected since their time constants are very fast (in the range of kHz). Reverse thrust (used on landing) is not included in the model and neither is start-up. Currently the model is set to initialise at a steady state point (there is a choice of several).

The major engine components included in the model (see figure 2.4 and 2.3) are: inlet duct, fan (modelled as separate duct and core sections), bypass duct, booster (low pressure compressor), inter-compressor bleeds, high pressure compressor with variable stator vanes and cooling bleeds, service bleed valve(s), fuel metering valve and fuel manifold, combustion chamber, high pressure turbine with cooling, low pressure turbine (without cooling), discharge nozzle.

The engine sensors included in the model are: ambient temperature, inlet temperature, N1 (LP shaft speed), N2 (HP shaft speed), EGT (exit gas temperature), P0 (fan inlet pressure), P20 (booster discharge pressure), P30 (HPC discharge pressure), P50 (HPT discharge pressure), FF (mass flow), variable stator vanes position sensor, variable bleed valve position, fuel metering valve position, power lever angle.

The model's control loops are: max N1, max N2, min N1, min N2, N1 acceleration (to monitor for engine stall and surge), N2 acceleration, N1 deceleration, N2 decel-

eration, max EGT[1], min HPC pressure ratio, min fuel flow, min idle, ground idle, approach idle. The control loops are controlled variables only and are fed back from the engine to the control system (often not directly but as derived parameters, e.g. shaft acceleration is derived from shaft speed). These controlled variables in all cases can be considered as scheduled limits. The single control variable is fuel, which alone regulates the entire engine - with the exception of the bleed and variable stator vanes, which are open loop scheduled solely to avoid, by design, engine surge conditions. Each control loop consists of a PI controller that ensures the engine does not exceed an operating limit (e.g. maximum HP shaft acceleration), under a range of environmental conditions and upon the pilot's throttle request for engine acceleration or deceleration. A diagram of the current model's closed-loop control system is provided in Fig. 2.1.

[1]EGT is an indirect measure of HPT inlet temperature, since this is difficult to measure in the harsh conditions present at the combustor exit. The model however is able to use this value directly.

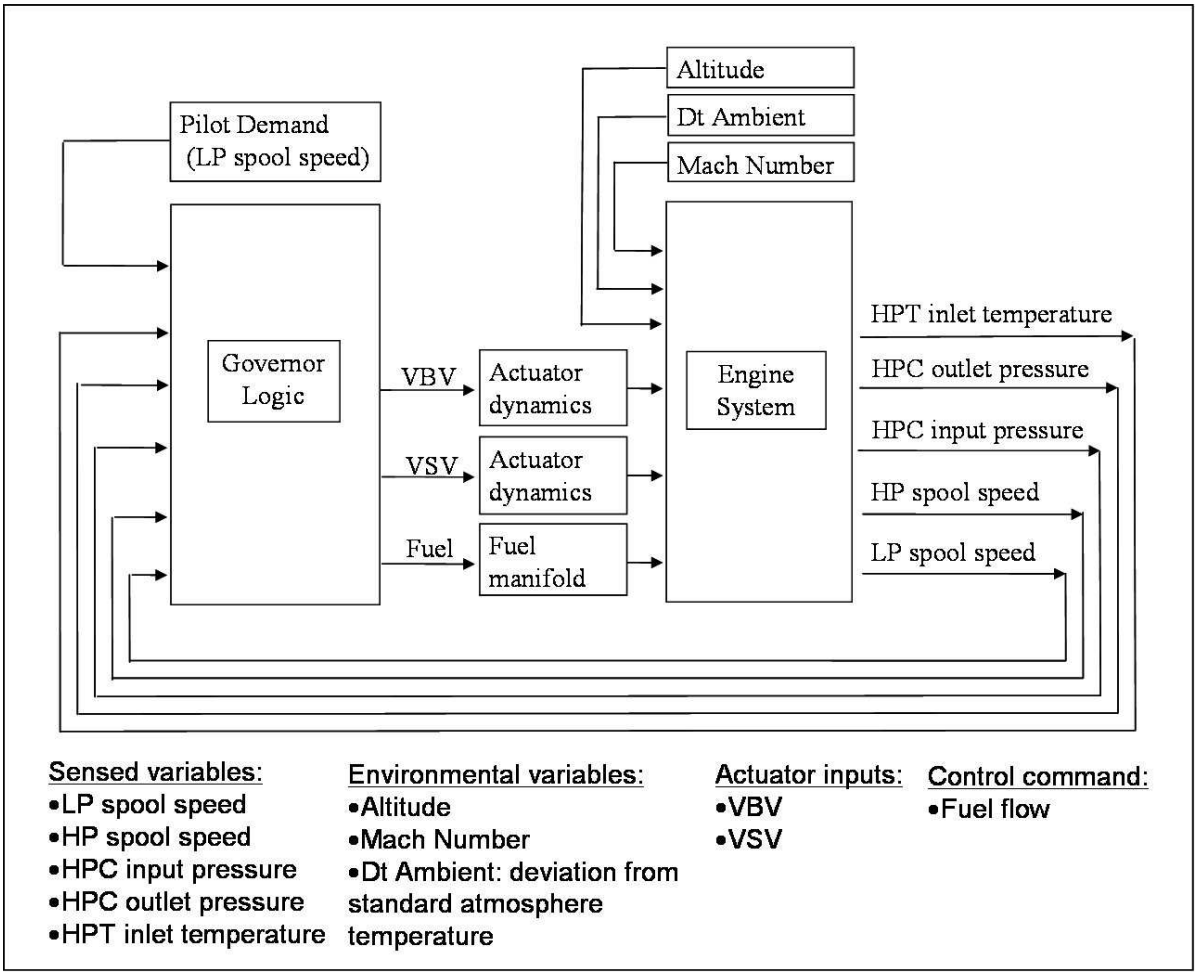


Figure 2.1.: **Diagram of the current model's closed-loop system.**

2.2. Gas turbines - an overview

What follows refers specifically to civil aircraft turbine engines, in particular a dual-shaft turbofan engine. While much of what is discussed is relevant to single-shaft engines, this should not be assumed unless it is explicitly stated.

The majority of a turbofan's thrust is provided by a large ducted fan, usually placed at the engine's inlet as shown in figures 2.2, 2.3 and 2.4. A turbofan is the preferred turbine configuration on most civilian jets due to the fact that it is more efficient to move a large mass of air slowly than a small mass of air more rapidly [3]. This also significantly reduces the noise produced by the engine. Furthermore, the diameter of a ducted fan is approximately 70% of that of an unshrouded propeller of equal static thrust [3]. The air stream at the inlet to the engine is "smoothed out" by the design of the engine inlet duct in such a manner that the velocity of the air through the fan blades is not greatly affected by the speed of the aircraft. Air that passes through the fan is split into two streams: flow to the engine inner section (known as the core) and bypass air. Primary flow is air that passes through the central part of the engine for combustion, (i.e. into the "core"); secondary flow is air that is discharged via an annular discharge duct that surrounds the engine core. The ratio of secondary airflow to primary airflow is known as the bypass ratio. Mixing characteristics of the secondary flow with the exhaust gases of the engine depend on the type and length of the duct used and the mixing process is often assisted to reduce noise - mixing of the hot and cold gas streams is the greatest source of noise at cruise conditions. Gas stream mixing and engine noise control is not relevant for our purposes and is therefore not implemented in the engine model.

Immediately following the fan is the engine's compressor. The compressor is made up of stages and each stage consists of rotating vanes, and stationary stators.

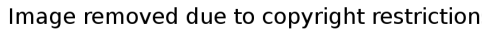


Figure 2.2.: **Twin shaft turbofan [36].**

As air is drawn deeper through the compressor, its heat and pressure increases. While the duct section of the engine provides the majority of the engine's thrust, it is in the core section that the process of compression, combustion and expansion occurs. These three processes do not occur in a single component as they do in a reciprocating engine's piston chamber, but are instead split amongst the compressor, combustor and turbine respectively. These are shown in diagrams 2.3 and 2.4. Work is provided by the expansion of gas through the turbine - but for this to occur, a pressure ratio is required, and this is the task of the compressor. The role of the combustor is to add energy to the gas stream by raising the temperature prior to expansion, this is achieved via combustion of fuel in the compressed air. Expansion of the hot working fluid then provides a greater power output from the turbine and this can be used to both turn the compressor and do additional work

Image removed due to copyright restrictions.

Figure 2.3.: **Twin shaft turbofan schematic.**

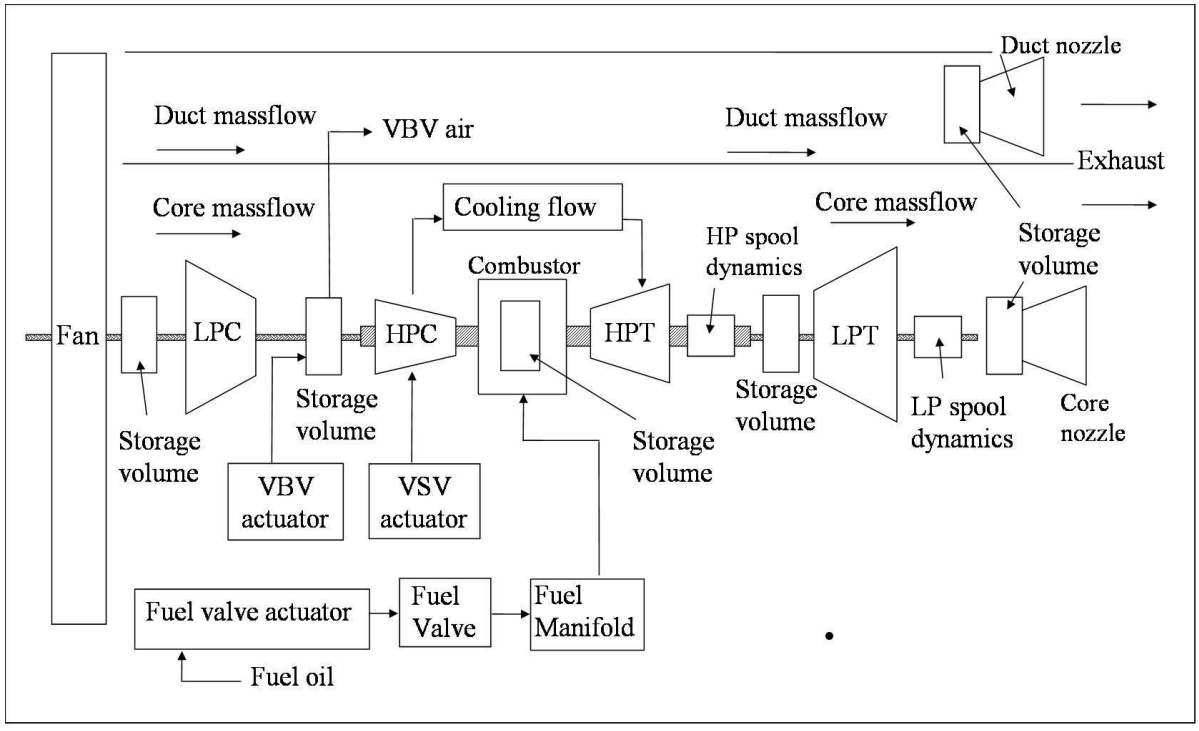


Figure 2.4.: **Twin shaft turbofan simulation components.**

such as rotating the engine's fan and propelling air out through the bypass duct to provide thrust. Note that the air passed through the core section adds to the thrust provided by the fan.

The maximum fuel/air ratio that may be used is limited by the maximum temperature that can be tolerated by the turbine blades. This depends upon thermally activated creep (elongation) of the highly stressed turbine blades. These are therefore usually cooled to permit an increased operating temperature. The cooling air is extracted from the compressor.

As shown in figure 2.3 each major engine section (the core and duct section) has an exhaust nozzle. The hot gases leaving the engine exhaust to atmospheric pressure via a nozzle, with the objective being to produce a high velocity jet (thrust is mass multiplied by velocity). In most cases, the nozzle is convergent and of fixed flow area, as is the case for the present model. Variable area nozzles are typically found on military turbojets. The encasement that follows the duct section of the fan is also modelled as a nozzle, this is to provide a means of calculating a pressure ratio over the fan duct section (see also Section 3.4.3).

As can be seen in figures 2.4 and 2.3 the engine's core section is split into a low pressure (LP) and high pressure (HP) region. This is because it is very difficult to achieve a high pressure ratio over the compressor without splitting it into stages (this is explained further in the following sections after introducing the phenomenum of surge, which is a prerequisite for a more complete discussion). For now it may suffice to say that the compression of air is divided between two major compressor components: a low pressure compressor (LPC) and a high pressure compressor (HPC). The two are on separate shafts but are linked aerodynamically. The low pressure compressor is powered by the low pressure turbine (LPT) and the two are linked by the low pressure shaft. The high pressure compressor

is powered by the high pressure turbine (HPT) and the two are similarly linked via the high pressure shaft. These two shafts are concentric, with the low pressure shaft rotating inside the hollow high pressure shaft.

Figure 2.3 has an extra element between each turbomachinery component: a "plenum", also known as a "storage volume". This module (discussed further in Section 3.4.1) is placed between the turbomachinery elements and is a volume where the mass balance of air flowing in and out is used to calculate the pressure at that location in the engine.

The two elements "LP spool dynamics" and "HP spool dynamics" are used to calculate the engine's acceleration or deceleration. Once the engine's net torque has been obtained, the angular acceleration of the compressor rotor can be calculated; it may then be assumed that this will be constant for a small time interval and a resulting change in speed can be found. This simple process (of Euler integration) can then be repeated many times to provide a transient running line starting from some convenient equilibrium running point.

The variable bleed valve (VBV) placed between the compressors and the variable stator vanes (VSVs) on the high pressure compressor (HPC) are both used to maintain the engine's operating point away from regions where a dangerous phenomenum called surge may occur. This is an instantaneous reversal of the flow of the gas stream that can greatly damage the engine. However before the VBV and VSV elements can be discussed further it is necessary to discuss surge in more detail. This will also serve to clarify the reason for a multi-stage compressor and why the engine is divided into a low pressure and a high pressure region.

2.2.1. Compressor surge, choking and characteristic maps

Turbine inlet temperature has been previously mentioned as a limiting factor for a turbine engine. Surge is the other factor that most limits an engine's performance. Surge is a complex aerodynamic phenomenon - far too complex to be simulated in this model (it would require a full 3-D finite-element simulation of the engine and its core components), so what has to be done is to provide some means of establishing a condition of surge, although surge itself is not explicitly simulated. The task is to then ensure that the engine model does not approach operating conditions where surge occurs. Therefore surge is a forbidden region of the engine's operating regime and, as previously mentioned, the dedicated function of certain engine elements, e.g. variable bleed valves (VBVs) and compressor variable stator vanes (VSVs) is to keep the engine operating point away from surge. This is particularly an issue during engine transients (sudden deviations from steady state operation), such as rapid accelerations or decelerations.

Surge and compressor maps are intimately linked and a discussion of surge will serve to introduce the principles required to understand compressor maps. Each compressor's operating characteristics can be represented by a "compressor map". This element will be discussed further in a few paragraphs (and in detail in section 3.4.3) but for now suffice it to mention that surge can be avoided by plotting a "surge line" on the compressor map as a means of establishing a forbidden operating region.

Surging is associated with a sudden drop in delivery pressure, and with violent aerodynamic pulsation which is transmitted throughout the entire engine. First consider figure 2.5: this is a plot of mass flow versus pressure ratio for a compressor rotating at constant speed and is called a "compressor characteristic curve". To explain this figure, imagine that a valve, placed in the delivery line of a compressor

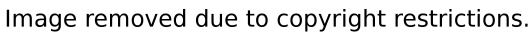


Figure 2.5.: **Compressor characteristic curve [36].**

running at constant speed is slowly opened. What is plotted above is the variation of the pressure ratio across the fan with mass flow.

When the valve is shut and the mass flow is zero the pressure ratio will have some value A, corresponding to the centrifugal pressure head produced by the action of the impeller on the air trapped between the vanes. As the valve is opened and flow commences, the diffuser begins to contribute its quota of pressure rise and the pressure ratio increases. At some point B, where the efficiency approaches its maximum value, the pressure ratio will reach a maximum, and any further increase in mass flow will result in a fall of pressure ratio. For mass flows that greatly exceed the design mass flow, the air angles will be widely different from the vane angles, breakaway of the air will occur, and efficiency will fall off rapidly. In this hypothetical case the pressure ratio drops to unity at C, when the valve is fully opened and all the power is absorbed in overcoming internal frictional resistance. At some point B where the fan efficiency approaches its maximum value, the pressure ratio will also reach a maximum, and any further increase in mass flow will result in a fall of pressure ratio.

The above plot could in theory be obtained for a fan considered in isolation - with the exception of, in direction of increasing mass flow, points after E. However in practice, and particularly in the case of multiple rows of compressors, most of the curve between A and B will not be feasible due to surging. Suppose that the compressor is operating at some point D on the part of the characteristic having positive slope, then a decrease in mass flow will be accompanied by a fall in delivery pressure. If the pressure of the air downstream of the compressor does not fall quickly enough, the air will tend to reverse its direction and flow back in the direction of the resulting pressure gradient, which causes the pressure ratio to rapidly drop. This reversed airflow means that the pressure downstream of the compressor will fall until the compressor is able to recover ordinary operation and correct mass flow direction will resume - until the cycle of events reoccurs and the airflow is again reversed. This succession of events repeats itself at a high frequency. Surging may not occur immediately upon movement of the operating point to the left of B in Fig. 2.5, because the pressure downstream of the compressor may be made to fall at a greater rate than the delivery pressure. Sooner or later, as the mass flow is reduced, the reverse will apply and the conditions are inherently unstable between A and B. However as long as the operating point is on the part of the characteristic map having a negative slope, decrease of mass flow is accompanied by a rise in delivery pressure and stable operation is assured. In a gas turbine, the actual point at which surging occurs depends upon the mass flow through the components downstream of the compressor (the combustor and turbines) and the way in which this mass flow varies during transients over the range of operating conditions.

Returning now to consider the hypothetical constant speed curve ABC in Fig. 2.5, there is an additional limitation to the operating range, this time between B and C.

As the mass flow increases and the pressure decreases, the density is reduced and the radial component of velocity must increase [10]. At constant rotational speed this must mean an increase in resultant velocity and hence in angle of incidence at the diffuser vane leading edge. Sooner or later, at some point E say, the position is reached where no further increase in mass flow can be obtained and choking is said to have occurred. This point represents the maximum delivery obtainable at the particular rotational speed for which the curve is drawn. Other curves may be obtained for different speeds, so that the actual variation of pressure ratio over the complete range of mass flow and rotational speed will be shown by curves such as those in Fig. 2.5. The left-hand extremities of the constant speed curves may be joined up to form what is known as the surge line, while the right-hand extremities represent the points where choking occurs.

The performance of a compressor may be specified by curves of delivery pressure and temperature plotted against mass flow for various fixed values of rotational speed. Charts of multiple compressor characteristics such as those in figure 2.6, superimposed at different speeds, are often simply called “compressor maps”. Note that these characteristics are also dependent upon other variables such as the conditions of pressure and temperature at entry to the compressor and the physical properties of the working fluid.

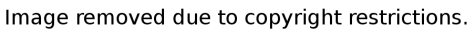


Figure 2.6.: **Axial compressor characteristics [36].**
Note:
$m\sqrt{T_{01}}/p_{01}$: non-dimensional mass flow; p_{02}/p_{01} : stagnation pressure ratio;
$N\sqrt{T_{01}}$: shaft speed (relative to design value).

By using dimensional analysis [42] the variables involved may be combined to form

a smaller and more manageable number of dimensionless groups. These characteristic curves can then be plotted on a non-dimensional basis, i.e. stagnation pressure ratio and isentropic[2] efficiency η against the non-dimensional mass flow

$$m\sqrt{T_{inlet}/P_{inlet}} \tag{2.1}$$

for fixed values of the non-dimensional speed

$$N\Big/\sqrt{T_{inlet}} \tag{2.2}$$

where m is the massflow, T_{inlet} and P_{inlet} are the temperature and pressure at the inlet respectively and N is the shaft speed.

The use of these parameters is specific to gas turbines and can be somewhat confusing. For example "non-dimensional" mass flow, as it is known in the gas turbine literature, does in fact have dimensions (e.g. mass flow, $m \cdot \sqrt{T/P}$, has units of $[kg \cdot \sqrt{K/Pa}]$), and is accordingly also called a "quasi-dimensionless" group. A full discussion of these parameter groups and their derivation via the "Buckingham PI theorem" is beyond the scope of this thesis and for further details the reader is referred to [42].

It can be seen that at high rotational speeds the constant speed lines become very steep. The surge point is normally reached before the curves reach a maximum value, and hence the design operation point, which is always near the peak of the characteristic, would ideally be kept very near the surge line. Consequently the range of stable operation of axial compressors is narrow.

The temperature ratio is a simple function of the pressure ratio and isentropic efficiency, therefore the form of the curves for temperature ratio will be similar to

[2] Adiabatic and reversible.

those in Fig. 2.6(a) when plotted on the same basis; there is no need to give a separate diagram here. From these two sets of curves the isentropic efficiency may be plotted as in Fig. 2.6(b) or, alternatively, contour lines for various values of the efficiency may be superimposed upon Fig. 2.6(a).

The efficiency at a given speed varies with mass flow in a similar manner to the pressure ratio, but the maximum value is approximately the same at all speeds.

To avoid surge it is desirable to keep the flow axial velocity approximately constant throughout the compressor. With density increasing as the flow progresses through the machine, it is therefore necessary to reduce the flow area by decreasing the blade height. When the machine is running below design speed, the density in the rear compressor stages will be far from the design value, resulting in incorrect axial velocities which will cause blade stalling and compressor surge. Manufacturers have used several methods to overcome this problem: Rolls-Royce and Pratt and Whitney have used multi-spool configurations, whereas General Electric has favoured the use of variable stator blades. Another possibility is the use of blow-off valves (these are discussed later), and on advanced engines it is sometimes necessary to include all these schemes (as is the case in our model).

The axial flow compressor itself consists of a series of stages, each of which has its own characteristic: stage characteristics are similar to the overall characteristic but have much lower pressure ratios. The mass flow through the compressor is limited by choking in the various stages - under some conditions this will occur in the early stages and under others in the rear stages. Off-design characteristics are summarised in figure 2.7. We have noted that if an axial flow compressor is designed for a constant axial velocity through all stages, the annulus area must progressively decrease as the flow proceeds through the compressor, because of the increasing density. The annulus area required for each stage will be determined

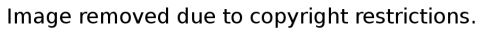


Figure 2.7.: **Phenomena at off-design operation [36].**
Note:
$m\sqrt{T_{01}}/p_{01}$: non-dimensional mass flow; p_{02}/p_{01} : stagnation pressure ratio.

at the engine design point therefore at any other operating condition this fixed area will result in a variation of axial velocity through the compressor. When the compressor is run at a speed lower than design, the temperature rise and pressure ratio will be lower than the design value and the effect of this reduction in density will be to increase the axial velocity in the rear stages, where choking will eventually occur and limit the mass flow. Thus at low speeds the mass flow will be determined by choking of the rear stages, as indicated in Fig. 2.7. As the speed is increased, the density in the stages is increased to the design value and the rear stages of the compressor can pass all the flow provided by the early stages. Eventually, however, choking will start to occur at the inlet; the vertical constant speed line in Fig. 2.7 is due to choking at the inlet of the compressor.

2.2.2. Surge control

Following the previous introduction to surge, the use of a staged compressor can now be introduced. Surge avoidance is the reason for the compressor's division into a low pressure and high pressure element, and is also the reason for the presence of a variable bleed valve (VBV) between these two elements, and the presence of variable stator vanes (VSVs) in the HP compressor.

First consider equilibrium conditions, i.e. a compressor running at constant speed.As a compressor's design pressure ratio is increased, the difference in flow density between design and off-design conditions will be increased, and the probability of blades stalling due to incorrect axial velocities is much higher. The effects of increased axial velocity towards the rear of the compressor can be alleviated by means of blow-off, where air is discharged from the compressor at some intermediate stage through a valve to reduce the mass flow through the later stages. Blow-off is wasteful, but is sometimes necessary to prevent the engine running line intersecting the surge line. A more satisfactory solution is to use a twin-spool compressor.

In the common twin-spool configuration the low pressure (LP) compressor is driven by a low pressure turbine and the high pressure (HP) compressor by a high pressure turbine. The speeds of the two spools are mechanically independent but have a strong aerodynamic coupling that serves to maintain compatibility of flow when the gas turbine is not operating at the design point. Off design performance can also be addressed by using several rows of variable stators (VSVs) at the front of the compressor - the effect is to decrease the axial velocity and mass flow for a given rotational speed. At low rotational speeds this delays stalling of the first few stages and choking in the last stages. Therefore one of the major benefits is an overall improvement in the surge margin at low engine speeds, which is particularly

Figure 2.8.: **Transient trajectories on compressor characteristic map [36].**
Note:
$m\sqrt{T_{01}}/p_{01}$: non-dimensional mass flow; p_{02}/p_{01} : stagnation pressure ratio;
T_{03}/T_{01} : ratio of outlet to inlet temperature.

important during starting and at idle speeds.

Surge however presents an even greater problem during engine acceleration: figure 2.8 shows the transient trajectory of the engine's operating point on a compressor characteristic map. Upon engine acceleration there is an initial movement towards surge. This is due to the rise in temperature that follows an increase in fuel flow, before the rotor has had time to increase its speed, thereby increasing mass flow.In addition, many high performance axial compressors exhibit a deleterious kink in the surge line at low rotational speeds (see figure 2.9). Therefore surge is less of a problem at high rotational speeds and with most modern compressors surge is likely to be encountered at low values of non-dimensional speed. In figure 2.9 a running line intersecting the surge line at low speed and at the kink is visible. To overcome this problem it is necessary to lower the running line locally in dangerous regions of operation. This figure also shows the effect of a variable area exhaust

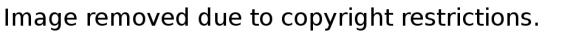


Figure 2.9.: **Effect of blow-off and increased nozzle area [36].**
Note:
$m\sqrt{T_{01}}/p_{01}$: non-dimensional mass flow; p_{02}/p_{01} : stagnation pressure ratio.

nozzle - this can also be used to lower the engine's operating point, however the current model uses a fixed area nozzle as variable area nozzles are mostly found on military turbojet engines. Note that on figure 2.8 during deceleration the operating point is moving away from surge and the turbine inlet temperature decreases; the only problem that may occur is a "flame out" of the combustion chamber because of very weak mixtures. This can be overcome by scheduling the metering of fuel flow as a function of rotor speed to prevent too rapid a reduction in fuel flow.
In summary, if the compressor's running line intersects the surge line at equilibrium conditions it will not be possible to bring the engine up to full power without taking remedial action. However even when the engine's equilibrium operation is clear of the surge line, if the running line approaches it too closely then the compressor may surge upon rapid acceleration. Therefore too rapid an increase in

fuel flow would cause the compressor to surge, resulting in very high temperatures that could destroy the turbine, and an abrupt decrease in fuel flow would result in a "flame out" of the combustion chamber. The provision of the correct fuel flow during engine transients is the responsibility of the designer of the fuel control system.

The shape of the transient running line will be determined by both the choice of a limiting turbine inlet temperature (which could well be placed some 50K higher than the design point value) and the location of the surge line. If the equilibrium running line intersects the surge line, either blow off or variable geometry can be used to lower the running line, but it should be noted that neither would have much effect on the transient running line. Consequently the provision of fuel controls the correct operation of the engine during transients, and this fuel scheduling is the task of the engine control system. Note that the engine rate of acceleration will be slowed down by blow-off via the variable bleed valve, because of the consequent reduction in mass flow through the turbine.

2.3. Gas turbine performance

The performance of a twin-shaft engine can be assessed in terms of either its low pressure (LP) or high pressure (HP) rotor speeds. This is justified because it is necessary to satisfy compatibility of flow between the shafts. This compatibility requirement gives rise to a strong aerodynamic coupling, which determines the ratio of the rotor speeds, even though the rotors are mechanically independent of each other. Because of the fixed relation between the speeds of the two shafts, for fixed geometry engines, the variation of turbine inlet temperature and fuel flow could be plotted versus either N1 (LP shaft speed) or N2 (HP shaft speed), but

because of the rapid variation of both with N2, the latter may be more suitable as abscissa.

It should be noted that although the HP shaft speed will always be higher than the LP shaft speed and the HP turbine blades are at a higher temperature, they may not be the mechanically critical components of the engine because the LP turbine blades are substantially longer and therefore suffer stronger rotational stresses.

From the previous discussion of surge, it is clear that transient behaviour is critical for aircraft engines: the prime requirement for civil aircraft is for a rapid thrust response to cope with an aborted landing. An in-depth understanding of the aero-engine dynamic behaviour at the design stage is therefore essential for the design and development of control systems.

Requirements for improved performance have led to multi-shaft rotor systems (e.g. Rolls-Royce uses three shafts on its engines, GE uses two), variable geometry in the compressor and use of blow-off valves - all of which complicate the prediction of both steady-state and transient performance. The acceleration of gas turbines is obviously dependent on such factors as the polar moment of inertia (a measure of resistance to torsional stresses) of the rotor system and the maximum temperature that the turbine blades can withstand for short periods, however the limiting factor for engine acceleration is usually the proximity of the surge line to the equilibrium running line. This is particularly critical at the start of acceleration from low speeds. Clearly, given the considerations above, a twin-shaft engine will respond quite differently to a single shaft engine.

2.3.1. The engine environment - variation of thrust with rotational speed, forward speed and altitude

Gas turbine performance depends significantly upon external pressure, temperature, and the speed (measured as a Mach number) at which the engine is travelling. Although it is convenient to express the engine's performance in terms of non-dimensional engine speed, it is actual mechanical speed that is limited by turbine stresses and which must be governed by an engine controller. The strong dependence of thrust upon engine speed indicates that accurate control is essential. The situation is very serious in the case of overspeed: centrifugal stresses increase with the square of the speed, and there is also a rapid increase in turbine inlet temperature. Typically an increase in rotor speed of only some 2% above the design limit (this is called engine overspeed) may result in an increase in T41 (turbine inlet temperature) of some 50° Kelvin. Since blade life is determined by thermally activated creep, the time for which high speeds and high temperatures are permitted must be strictly controlled. This is achieved by "rating" the engine for several operation regimes, each with a strict time constraint of maximum duration. The maximum permissible speed is normally restricted to periods of less than 5 minutes - this value is used for the engine take-off rating. The climb rating is obtained with a small reduction in fuel flow and hence rotor speed, and can usually be maintained for 30 minutes. The cruise rating requires a further reduction in fuel flow, resulting in stress and temperature conditions that permit unrestricted indefinite operation.

2.3.2. Ambient temperature effects

While the deleterious effects of engine overspeed will not be immediately noticed by the user, the effect of ambient temperature is important, and noticeable, to both

the manufacturer and the user. A change in ambient temperature will modify the engine's operating point and this has a number of consequences. As an example of this, consider that when an engine is running at its maximum mechanical speed, an increase in ambient temperature will result in an effective performance loss that is equivalent to a decrease in mechanical speed. This occurs because the actual mass flow entering the engine is reduced as ambient temperature increases due to the decrease in air density. Furthermore, for a given mechanical speed, the turbine inlet temperature will be higher than that reached with a cooler ambient temperature. Therefore to prevent the maximum turbine inlet temperature being exceeded on a hot day, the engine speed would have to be reduced, giving a further reduction in thrust.

2.3.3. Ambient pressure effect

Thrust changes in near direct proportion to the ambient pressure. No change in engine operating point occurs, but the mass flow is reduced as the ambient pressure is reduced.

2.3.4. Altitude effect

Both pressure and temperature drop with increasing altitude. While a temperature drop is beneficial, because thrust is dependent upon the first power of the pressure, the decrease in thrust due to the decrease in ambient pressure more than outweighs the increase in thrust due to the reduction in inlet temperature.
In summary, a worst case scenario is an airport located at high altitude and in the tropics (e.g. Mexico City)! Such a situation may require a significant reduction in payload. The role of the engine's control system is therefore to ensure that the

engine operates under a large range of environmental conditions without exceeding the engine's operating limits, e.g. the HPT inlet temperature, or entering a condition of surge. As mentioned in the section above, these limits may not be fixed and which limit caps the engine's operation will also depend upon the environmental conditions. This is further discussed in Section 4.1.
The role of the controller will be discussed further in the following chapters, and will be analysed in detail in Chapter 5.

2.4. Summary

This chapter discusses the engine's design reference and introduces key characteristics of gas turbine engines in general and turbofans in particular.
The key engine limitations of surge and HPT inlet are discussed along with the use of blow-off and control schemes to keep the engine within its multiple operating limits. The effect of the environment on the engine and the need for a gain-scheduled control scheme to maintain the engine within its multiple operating limits was introduced. Engine performance requirements were outlined.
The purpose of this chapter was to foster a basic understanding of a gas turbine as relevant to this project - and the reason it needs to be controlled.

3. Modular construction of the mathematical model

Here the architecture of a gas turbine is analyzed in greater detail, presenting the key equations that can be used to model each component - and how these elements can be put together: the principles of the simulation architecture. The target aero-engine, a high-bypass separated flow turbofan as used on high subsonic commercial aircraft is also discussed.

3.1. Gas turbine architecture

A gas turbine engine is a highly complex non-linear system requiring an accurate dynamic model for controller design and evaluation. For the purpose of development and testing of advanced aero-engine controllers, a full aero-thermodynamic simulation of a jet engine has been developed and implemented in the Matlab-Simulink® environment. The plant controlled is a generic high-bypass separated flow turbofan. This has been found to be the optimum configuration for high subsonic commercial aircraft [36]. The model is therefore representative of the jet engine of a mid-range aircraft, such as the highly successful Airbus A320 family. The model has two shafts, variable bleeds and stator vanes, actuator dynamics,

heatsoak dynamics, cooled turbines, a model of the duct, and a fixed convergent exhaust nozzle. It has been developed in a modular fashion so as to permit the easy implementation of failure scenarios and adaptation to multiple engine configurations. The model, in its current implementation, develops a maximum thrust of above 30000 lb_f (133 kN) and has been validated, in collaboration with the industrial partner, via comparison with steady state and dynamic data for a comparable engine.

A turbofan's airflow is split into two main sections (as shown in Fig. 3.1). The outer, "duct" or "bypass" airflow is accelerated by a fan situated at the front of the engine. This section provides most of the engine's thrust by moving a large mass of air at a relatively low speed. The inner or "core" section provides the power to drive the fan. Airflow through this section is compressed via two sequential compressors: a low pressure (LP) compressor and a high pressure (HP) compressor. The high pressure flow at the exit of the high pressure compressor is in part combusted, expands through the HP turbine (this powers the HP compressor), and then flows into the LP turbine (this powers the LP compressor and the fan). A further expansion to atmospheric pressure is via a fixed convergent nozzle placed at the rear of the LP turbine. The shaft connecting the fan, LP compressor and LP turbine is called the "low pressure shaft", and similarly the shaft connecting the HP compressor and the HP turbine is known as the "high pressure shaft". These two shafts are concentric - the LP shaft extends beyond and rotates within the HP shaft. Figure 3.1 outlines how these turbofan components are interconnected. The variable bleed valve placed between the low pressure compressor (LPC) and the high pressure compressor (HPC) improves the surge characteristics [16, 34] by purging air directly into the duct. For the same reason the HPC module is provided with a simple model of variable stator vanes (VSVs). The operation of

both these components is open loop scheduled with corrected shaft speed.

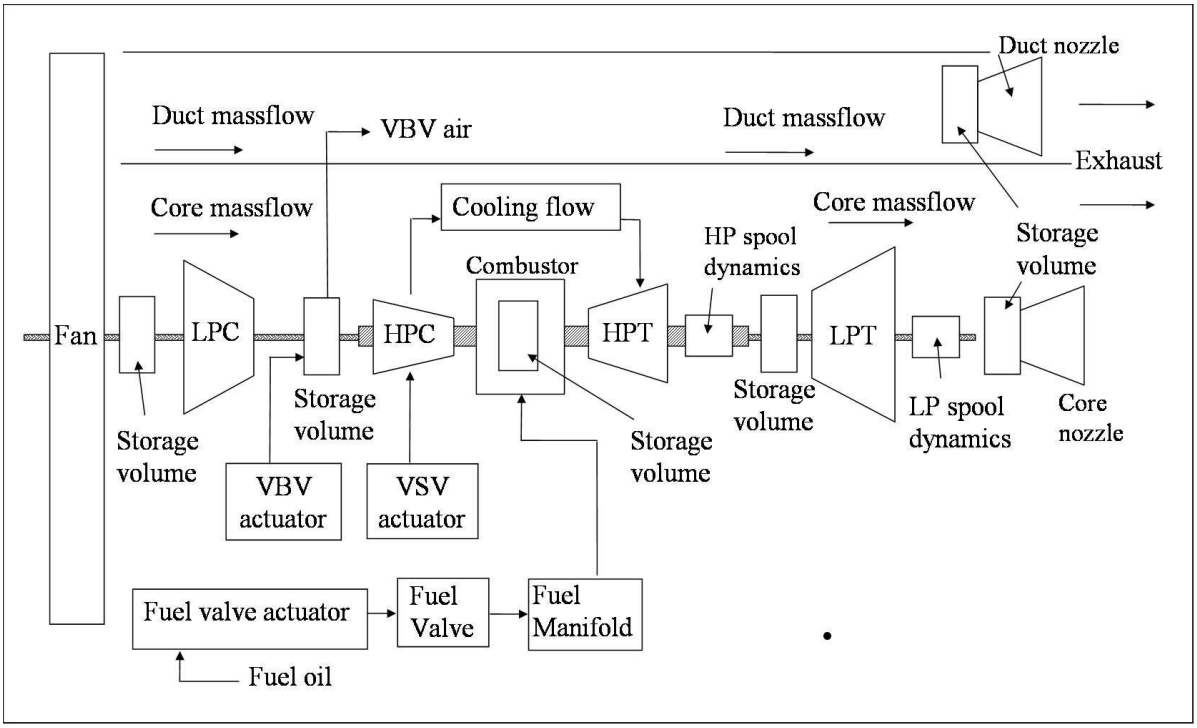


Figure 3.1.: **The main components of a turbofan engine.**

3.2. Overview of the aero-engine simulation

The simulation uses "lumped" elements: the engine components are simplified to volumeless elements, thereby reducing the partial differential equations that describe their distributed properties to ordinary differential equations that describe the evolution of key properties over time. The aim is to derive a set of explicit, first order differential equations that can be solved using an integration algorithm to accurately describe the dynamic characteristics of the modelled components. The unsteady mass balance between components is taken into account via storage volumes (plena).

Simulation of a gas turbine's components requires the application of the following thermodynamic conservation laws or continuity equations:

- conservation of mass (basic physical principle)
- conservation of energy (first law of thermodynamics)
- conservation of momentum (Newton's second law)

Additionally, relationships describing heat transfer processes and fluid mechanics must also be applied. Finally, shaft dynamics are added by making mechanical connections between the component models. Engine acceleration is provided by integrating the shafts' power imbalance over time.

There currently exist several approaches to modeling gas turbines: the most accurate are iterative models based on thermodynamic conservation laws and continuity equations. An engine equilibrium point is calculated (this is the component matching phase) and from this all engine parameters are derived via an iterative solution of the system of equations (initial guesses are usually provided). These models assume as constant the mass flow for the required matching of the compressors and turbines.

The type of model presented here, called an "aerothermal transient performance model" [42] or "aero-thermodynamic model" [8] avoids iteration by arranging the equations to follow the direction of the gas path, and introducing storage volumes between components to account for the unsteady balance of mass at fan and compressor discharge, combustion chamber, between the turbines, and between the turbines and nozzle [8]. This is the most accurate of the many non-iterative models used for modeling engine performance transient parameters [42]. Other models are more suitable for flight simulators where lower accuracy is permissible - these generally use transfer functions relating fuel flow and parameter outputs [42].

Relative to iterative models all other approaches suffer from some loss of accuracy,

but in the case of the aerothermal model considered here this is negligible for the purpose of engine control system development, particularly if a small time step is used. This loss is also offset by a superior execution time. The size of the volumes (plena) included between the turbomachinery is important [42], as they have their own time constant [8] and this will affect the maximum permissible time step. Clearly the update time step should stay below the plena time constant. For small plenum volumes there is a very steep rate of change in pressure for any variation in mass influx or efflux. Should the time step of the system be large, this would lead to situations whereby the system would respond to large changes in pressure rather than to the gradual change that would occur for a smaller sample time. An unphysical situation would arise when these changes in pressure would result in the engine exceeding the pressure ratio boundaries of the compressor or turbine maps either downstream or upstream of the plenum. From this perspective, the smallest possible time step is desirable, however one should also consider that the turbomachinery characteristic maps are in themselves approximations, and an extremely small time step will thus not necessarily improve the simulation accuracy in practice, although it will indeed improve the solution accuracy.

Because the model incorporates numerous component maps, the tradeoff between solution accuracy and time step is not readily quantifiable analytically, and determination of an overall limiting time step is not easy. Plenum size is therefore a limiting factor to the permissible time step and solution accuracy needs to be carefully tested should very small volumes be used. This is however likely to become an issue only when simulating multi-stage turbomachinery in a stage-by-stage fashion (this is not the case for the present simulation as the model uses "lumped" components). A practical way of testing for this is to repeatedly halve the time-step until no difference in solution accuracy is noticed for an engine rapid transient

such as would occur during a pilot slam request [1]. A good, conservative, constant time-step for the current simulation is 10^{-4} seconds. This works comfortably even with basic Euler numerical integration (Matlab ode1).

Other simulation methods, such as the iterative "thermodynamic matching model" presented in [16] and also discussed in [42] are much slower and usually do not have a predictable run time as the solution time to convergence may vary. Furthermore, the iterative method produces noise on the output parameters, due to solutions at each time step falling at random places within the permitted tolerance band. This noise may prevent assessment of system stability due to the perturbations produced [42]. For real-time applications, in order to reduce the computational time, techniques based on a linearisation of the model have often been adopted [8, 17, 36]. However simulations based on linearised models should be used only in the neighborhood of the working point about which the linearisation has been carried out and are therefore particularly unsuitable for aero-engine simulation as an aero-engine will be expected to undergo a large variation in operating conditions.

In summary, calculations in the current model are performed in a sequential manner and in the direction of the gas flow, thus allowing for a direct non-iterative calculation of the engine cycle and performance. Once atmospheric conditions, namely altitude, Mach Number, temperature deviation from the International Standard Atmosphere (ISA), and the value of the control variables, e.g. fuel flow and other control parameters that are open loop scheduled (e.g. bleed valve position) have been specified, then the operating point follows by propagating the calculations along the engine's direction of flow. A few notable exceptions are addressed via use of Simulink "memory blocks" (more details are in Section 5.6) - this is so that thermodynamic properties can be based on a component's mean temper-

[1] A pilot slam request is an abrupt acceleration from ground idle to maximum thrust. This is a standard test of engine transient performance.

ature and the compressors can be provided with pressure data from downstream components.

3.3. Comparison with current state-of-the-art engine simulation models

The two reference state-of-the-art models openly available are GasTurb [19] and GSP [28]. GSP has been in development at the Netherland's National Aerospace Laboratory (NLR) since the seventies. GasTurb is developed by a gas turbine expert previously at MTU aerospace and has been available since 1995.

Both are licensed programs but GasTurb provides a limited version for free, while GSP enables all features but disables saving in the free version. These models are advanced gas turbine simulations but are closed-source systems. They are provided as compiled code, i.e. the design is open to customisation only to the degree permitted by the developers. The customisation options provided - without additional involvement of the model developers - are however insufficient for control design or implementing failure scenarios, as will be clarified below.

Both GSP and GasTurb are iterative models: a deviation from the design point is calculated by solving a set of non-linear differential equations. The equations are determined by the mass balance, the heat balance (GSP only), the equation for conservation of momentum and the power (energy) balance for all components. In the case of a transient simulation, the differential equations also include time-derivatives. Then, in each time step, dynamic effects are calculated and the solution represents a quasi-steady state operating point.

GasTurb omits heatsoak. It is stated explicitly in its manual that some parameters (fuel flow is specifically mentioned), will not be correctly calculated for a

transient simulation. This fact alone immediately rules out its use for control design purposes. GSP on the other hand states that it does incorporate heatsoak but surprisingly then adds that "Volume and heat soakage effects may be calculated for each gas path component but often are relatively small and therefore disabled". This contrasts starkly with the gas turbine literature ([42],[36]) that clearly states that heatsoak is significant. GSP's statement would benefit from further clarification as to the frame of reference (e.g. are large transients explicitly disallowed in the simulation?) but this is not discussed further in the GSP manual, so this point is left unclear.

What is however clear is that heatsoak is indeed significant ([42],[36]). This is also affirmed by the GasTurb manual: "The simplifications have the consequence that the calculated value for certain parameters, especially fuel flow, are not realistic during transients (this is because the considerable amount of heat that is exchanged between the gas and the components is not considered)".

Modules of both these models cannot be replaced by the user without direct involvement of the developers. GasTurb does includes a limited form of control - this is implemented as a "black box" containing PID controllers with adjustable gains, but the architecture is not clarified or open to inspection or further adjustment. Due to the lack of heatsoak the control system can only be used for rough control of a transient simulation, i.e. to approximately maintain the engine within physically realistic operating conditions.

With regards to a control scheme GSP's manual states that "For detailed models of complex control systems, usually custom components are required. Also for accurate transient simulation of modern multi-spool jet engine (e.g. turbofan engines) control systems, controlling both gas generator and fan rotor speeds, custom components are required". Later it is further stated that: "Custom components

are provided in separate custom libraries and require additional coding. Using the GSP component developers package, custom components can easily be derived from the standard component models using object inheritance. NLR has a large number of custom libraries available, developed for detailed performance analysis of specific engine designs and engine control systems. New custom libraries are usually developed at NLR. For advanced use of GSP, custom components can be developed outside NLR using the additional GSP Component developers package. Contact NLR for additional information on Custom components or the new GSP component developers package.".

Many key parameters of the these simulations are inaccessible. For example GasTurb's most recent version 11 (which was not available until the latest stages of our current work) enumerates as a new development that the gas stream parameters are now accessible. This fact alone should serve to highlight the rather closed nature of these simulations.

By comparison, the model herein discussed permits access to all and any parameters, heatsoak is included, a reference control architecture is implemented and is fully open to inspection. GasTurb allows implementation of a flight envelope via a fixed number of 49 points that must start at ground level. The current simulation can perform transient simulation without limit of operation starting from any operating point of a previously saved test case. The control system can be inspected, modified, and completely replaced should the user so desire - as can indeed all components.

The non-iterative nature of the current model also means that the simulation can run at a predictable and constant speed by using a constant time step. Indeed the simulation may run side-by-side with a real engine if required and a suitably powerful computer is provided (this should be possible on any reasonably powerful

dual core machine produced within the last few years).

3.4. Component models

Each modular component can be viewed as an operator, the purpose of which is to compute the thermodynamic state of the fluid (typically mass flow $\dot{w}$ $[kg\ s^{-1}]$, total temperature and pressure) at the outlet of the module based on the inlet conditions and some additional parameters. Each component model relies on the equations for mass, momentum and energy balances and on empirical information derived from rig tests or advanced CFD [2] calculations, e.g. compressor and turbine characteristic maps. The stagnation conditions at the engine inlet are computed with Saint-Venant-Wantzel relations [36] as a function of altitude, variation from ISA day temperature and Mach Number. This accounts for ram recovery [3] at the engine's inlet.

The thermodynamic properties of the air stream and combustion gases will vary due to the range of environmental and operating conditions at which the engine must operate. This makes an accurate evaluation of the gas stream's thermodynamic properties highly desirable for an aero-engine. Lookup tables may be used (tables containing the values of the specific heats have been published in many works [1, 3]) or, as in the present case, algebraic curve-fitting expressions [42] may be used. Therefore, in the present work the working fluids are not considered as perfect gases of constant specific heats, as this is mainly appropriate for preliminary design calculations [36]. This has the advantage of both improved accuracy and preserves a computational architecture that allows for complex scenarios to be simulated e.g. ingestion of water vapour and dramatic changes in inlet temper-

[2] Computational fluid dynamics.

[3] Ram recovery is a measure of the conversion of high velocity air at the duct inlet to static pressure around the engine's inlet.

ature. The specific heat at constant pressure, c_p, was derived from the polynomial fits provided in [42] and is temperature and fuel-air-ratio (FAR) dependent. The gas constant, R, is temperature independent and FAR dependent. The ratio of specific heats, γ, is temperature and FAR dependent and can be expressed as a function of the specific heat capacity, c_p, and the gas constant R. It should be noted that for formulae using c_p and γ it is most accurate to base these values on the mean temperature for each component, i.e. the arithmetic mean of the inlet and outlet values [42].

3.4.1. Plena

Because turbomachinery, compressor and turbine units are considered as volumeless elements, a plenum is placed at the compressor outlet in order to take into account the unsteady mass balance at compressor discharge, within the combustion chamber, and between the turbines and the low pressure turbine and the nozzle. Mass conservation implies:

$$w = \int (\dot{w}_{in} - \dot{w}_{out}) \cdot dt + w_0, \tag{3.1}$$

where w is the mass present in the casing, $\dot{w}$ represents gas stream mass flow $[kg\ s^{-1}]$ and w_0 is the initial value of the mass present in the plenum. Pressure inside the plenum is then calculated via the ideal gas law:

$$p_{out} = w \cdot \frac{T \cdot \bar{R}}{V}, \tag{3.2}$$

where $\bar{R}$ is the specific gas constant of the gas stream $[J\ kg^{-1}\ K^{-1}]$. Each plenum also includes a module to calculate the heat soak of the component upstream. If

this was not included the plenum's outlet temperature would be equal to the inlet temperature. Energy accumulation due to transient effects such as volume packing [42] is neglected. Pressure losses are also not considered although these are easily implemented if desired [19, 36].

Table 3.1.: **Summary of the plenum module's variables and parameters**

model parameter	**description**	**units**
Inputs		
T_{in}	temperature of inlet air	$[K]$
$\dot{w}_{in}$	mass flow of inlet air	$[kg \cdot s^{-1}]$
$\dot{w}_{out}$	mass flow of outlet air	$[kg \cdot s^{-1}]$
Outputs		
T_{out}	temperature of outlet air	$[K]$
p_{in}	pressure at inlet	$[Pa]$
p_{out}	pressure at outlet	$[Pa]$
Boundary conditions		
$T1_0$	previous turbomachinery component inlet air temperature	$[K]$
T_0	initial plenum inlet air temperature	$[K]$
M_0	initial plenum mass	$[kg]$
Dynamic states		
M	plenum mass	$[kg]$
T_m	average metal temperature of previous turbomachinery component and plenum casing	$[K]$
Constant parameters:		
- lookup tables for converting between temperature and specific enthalpy - reference values for calculation of pressure drop - volume of plenum - heatsoak design point time constant and heat transfer coefficient		

3.4.2. Heat soakage

Each turbomachinery component includes heat transfer effects, such as the heat transfer to turbine blades and casings. Only convective heat transfer is considered based on simplified equations for turbulent flow over a flat plate and assuming a constant Prandtl number [45].

By considering a lump of metal (e.g. a blade) in a hot gas flow a simple, first order heat soak equation can be developed. The time constant, τ, can be calculated from the heat transfer coefficient and the mass and specific heat capacity of the metal. The heat transfer coefficient is calculated for the design point conditions and modified at off-design conditions depending on mass flow and temperature, as both alter the flow's Reynold [4] number. It has been demonstrated that heat soak effects play an important role in determining a gas turbine's dynamic performance [17], yet they are extremely difficult to predict in the absence of good test data. In the absence of such data, the heat soak released upon an abrupt deceleration of an industrial engine [29] was scaled down for the current model. The time constant was similarly reduced. The overall heat soak quantity was then distributed amongst the components according to their mass and temperatures. Key equations for the implementation of a heatsoak module follow. The most fundamental equation is Newton's law of cooling:

$$q = \bar{h}A \cdot \Delta T = \bar{h}A \left(T_{\mathrm{m}} - \bar{T}\right) , \tag{3.3}$$

[4] The Reynolds number is used to characterize different flow regimes, such as laminar or turbulent flow: laminar flow occurs at low Reynolds numbers, where viscous forces are dominant, and is characterized by smooth, constant fluid motion, while turbulent flow occurs at high Reynolds numbers and is dominated by inertial forces, which tend to produce random eddies, vortices and other flow instabilities. These flow characteristics affect heat transfer.

where q is the heat flow $[W]$, $\bar{h}$ is the average heat transfer coefficient over the surface $[W\ m^{-2}\ K^{-1}]$, T_m is the component average temperature, $\bar{T}$ is the gas stream's mean temperature and A is the heat transfer area $[m^2]$. The heat transfer coefficient is temperature and flow dependent, and this dependency can be approximated with the following relationship [15]:

$$Y \propto T^{0.23}\dot{w}^{0.8}, \tag{3.4}$$

where $\dot{w}$ is mass flow rate and

$$Y = \bar{h}A \tag{3.5}$$

In experimental conditions the easiest parameter to measure will be the time constant τ. This can be obtained by rapidly increasing or decreasing the gas stream temperature from a starting condition with the gas and metal mass in thermodynamic equilibrium at a known temperature:

$$\frac{dT_m}{dt} = -\frac{q}{Mc_{pm}}, \tag{3.6}$$

where T_m is the metal temperature, M is the metal mass and c_{pm} is the metal specific heat capacity. The two equations (3.6) and (3.3) above can be combined to give:

$$\frac{dT_m}{dt} = -\frac{\bar{h}A(T_m - \bar{T})}{Mc_{pm}} \tag{3.7}$$

Since the time constant is

$$\tau = \frac{Mc_{pm}}{\bar{h}A}, \tag{3.8}$$

it then follows that

$$\frac{dT_m}{dt} = -\frac{1}{\tau}\Delta T \tag{3.9}$$

Therefore the time constant of the system can be experimentally determined. Once this is known, for a system of similar mass and area the following equations can be applied:

$$Y_d = \bar{h}A = \frac{1}{\tau_d} M c_{pm}, \tag{3.10}$$

where Y_d is the value of Y established under the experimental conditions T_d, $\dot{w}_d$, and τ_d. From this it follows that:

$$q = Y \cdot \Delta T = \Delta T \cdot Y_d \cdot \frac{\tau_d}{\tau} = \Delta T \cdot Y_d \cdot \left(\frac{\tau}{\tau_d}\right)^{-1} = \Delta T \cdot Y_d \cdot \left[\left(\frac{\bar{T}}{T_d}\right)^{-0.23} \left(\frac{\dot{w}}{\dot{w}_d}\right)^{-0.8}\right]^{-1} \tag{3.11}$$

This is because:

$$\tau = \tau_d \left[\left(\frac{\bar{T}}{T_d}\right)^{-0.23} \left(\frac{\dot{w}}{\dot{w}_d}\right)^{-0.8}\right] \tag{3.12}$$

This last relationship arises from the following considerations: since

$$\tau = \frac{M c_{pm}}{\bar{h}A} \approx \frac{k}{\bar{h}} \ , \tag{3.13}$$

where k is a constant and from (3.4):

$$\bar{h} \approx k_1 \cdot T_d^A \cdot \dot{w}_d^B \tag{3.14}$$

it then follows that:

$$\tau_d = \left(\frac{k}{k_1}\right) \cdot T_d^{-A} \cdot \dot{w}_d^{-B} = k_d^* \cdot T_d^{-A} \cdot \dot{w}_d^{-B} \tag{3.15}$$

Considering a new time constant τ at different temperature and massflow conditions:

$$\tau = k^* \cdot T^{-A} \cdot \dot{w}^{-B} \tag{3.16}$$

and therefore:

$$\frac{\tau}{\tau_d} = \frac{k^* \cdot T^{-A} \cdot \dot{w}^{-B}}{k_d^* \cdot T_d^{-A} \cdot \dot{w}_d^{-B}} \tag{3.17}$$

and if

$$k_d^* = k^*, \tag{3.18}$$

which holds for components of similar geometries [45], then:

$$\frac{\tau}{\tau_d} = \frac{T^{-A} \cdot \dot{w}^{-B}}{T_d^{-A} \cdot \dot{w}_d^{-B}} = \left(\frac{T}{T_d}\right)^{-A} \left(\frac{\dot{w}}{\dot{w}_d}\right)^{-B} \tag{3.19}$$

Therefore if Y_d and τ_d have been established experimentally, then the formula (3.11) can be applied to get the heat transferred q $[W]$, over a range of temperatures and massflows. This heat can then be added or subtracted to the enthalpy of the gas stream.

Table 3.3.: **Summary of the heatsoak module's variables and parameters**

model parameter	**description**	**units**
Inputs		
$T_{average}$	average temperature [5]	[K]
$\dot{w}_{average}$	average massflow [6]	$[kg \cdot s^{-1}]$
Outputs		
Q	heat transferred to or from from the gas stream	[W]
Boundary conditions		
T_0	initial plenum inlet air temperature	$[K]$
$T1_0$	previous turbomachinery component inlet air temperature	$[K]$
Dynamic states		
T_m	average metal temperature of previous turbomachinery component and plenum casing	$[K]$
Constant parameters:		
- lookup tables for converting between temperature and specific enthalpy - heatsoak design point time constant and heat transfer coefficient		

[5] this is the average of the input and output temperature of the component for which the heatsoak is calculated e.g. the HPT input and output temperature.

[6] this is the average of the input and output massflow of the component for which the heatsoak is calculated e.g. the HPT input and output massflow, the two can differ due to an increase in air pressure, caused by a change in engine operating point.

3.4.3. Fan and compressors

The fan placed at the front of the turbofan provides the majority of the engine's thrust. The fan simulation module is divided into two sections: a core and a duct section. The ratio between the duct and core massflows is known as the bypass ratio (BPR):

$$BPR = \frac{\dot{w}_{duct}}{\dot{w}_{core}} \tag{3.20}$$

Each section is modelled via a characteristic map that describes the steady state performance of the component. The performance can be specified by curves of delivery pressure and temperature, plotted against mass flow for various fixed values of rotational speed. These characteristic curves are however dependent upon other variables such as the conditions of pressure and temperature at entry and the physical properties of the working fluid. By using dimensional analysis, the variables involved may be combined to form a smaller and more manageable number of dimensionless groups and these characteristic curves can then be plotted on a non-dimensional basis, i.e. stagnation pressure ratio and isentropic efficiency η_i against the non-dimensional mass flow rate $\dot{w}_c$ for fixed values of the non-dimensional speed $(n/\sqrt{\theta})$ [42].

The fan's corrected massflow is provided by the characteristic map once the fan's pressure ratio and corrected shaft speed have been provided. Applying the conservation equations, the temperature increase over the fan is described by:

$$T_{out} = T_{in} \cdot \left[1 + \frac{1}{\eta_i} \cdot \left(\pi^{\frac{\gamma-1}{\gamma}} - 1\right)\right], \tag{3.21}$$

where π is the pressure ratio over the core or duct section of the fan, γ is the ratio of specific heats and η_i is the isentropic efficiency. The power required to drive the

fan is then:

$$P_{fan} = c_p \cdot \dot{w} \cdot (T_{out} - T_{in}) \tag{3.22}$$

and is positive since it is supplied to the air.

The air that flows through the core section of the engine is compressed via two compressors in series. These are modelled in a similar fashion to the fan, except that there is no splitting of the mass flow. Characteristic maps are again at the core of the compressor models. More complex models would split the compressor into stages separated by small plena and solve gas flow equations based on a knowledge of blade angle and stage performance, but the approach adopted here is more practical, particularly for a generic turbofan model where data requirements are not too onerous [32].

Between the two compressors there is an open loop scheduled bleed valve (VBV). This extracts air from the core flow and exhausts it directly into the duct. The percentage of massflow extracted is scheduled with the corrected LP shaft speed. For more information on the predicted effect of VBVs, the reader is referred to [16].

The second of the compressors in series, the high pressure compressor (HPC) also includes customer bleeds (these power aircraft accessories) and variable stator vanes (VSVs). The latter are to improve the surge margin of the HPC and are modelled by a percentage reduction in corrected mass flow. This reduction is open loop scheduled against corrected HP shaft speed. The reader is referred to [16] for plots of the effects of VSVs on compressor maps. The customer bleeds are modelled by extracting a percentage of the inlet air flow. This air will not be available for work in the compressor and this is reflected in the formula for HPC

power:

$$HPC_{power} = c_p \cdot \dot{w} \cdot (T_{out} - T_{in}) \cdot [k_1 x_1 + k_2 x_2 + (1 - k_1 - k_2)] \tag{3.23}$$

where k_1 and k_2 are the proportion of air removed via bleeds 1 and 2 respectively and x_1 and x_2 are scalars from 0 to 1 that represent the proportion of the total temperature rise to be expected at that stage.

Cooling air is also extracted prior to the combustor, at the outlet of the HPC. Cooling flow is an important element of the model because the total percentage of engine inlet mass flow extracted before the combustor may be up to 25% for a high technology aero or industrial engine [42], and cooling flow will represent a significant proportion of this. Cooling flow is often modelled simply as a percentage of the engine's massflow [36] but in this case it was preferred to use a relation that, although empirical, is based on the ratio of the pressure of the air source (the HPC) and that of the sink (the HPT) [8, 27]:

$$\dot{w}_{cool} = K \cdot \sqrt{1 - \frac{p'_{out}}{p_{in}}} \cdot \frac{p_{in}}{\sqrt{T_{in}}}, \tag{3.24}$$

where K is the discharge coefficient, p_{in} and T_{in} are respectively the pressure and temperature at the bleed point and p'_{out} is the static pressure at the exit of the cooling circuit. In this work, this is approximated by the pressure value at the cooling flow exit. This is reasonable, since the velocity of the gas stream is relatively low at this stage and dynamic pressure remains a low proportion of the total up to approximately Mach 0.4 [42].

Table 3.5.: **Summary of the fan and compressor modules' variables and parameters**

model parameter	**description**	**units**
Inputs		
N [7]	shaft speed	$[rpm]$
p_{out}	outlet pressure	$[Pa]$
VSV	variable stator vanes' position [8]	$[degrees]$
Outputs		
T_{out}	outlet temperature	$[K]$
$\dot{w}_{out}$	outlet mass flow	$[kg \cdot s^{-1}]$
P	power consumed	$[W]$
Boundary conditions: none		
Dynamic states: none.		
Constant parameters:		
- lookup tables for calculating mass flow, isentropic efficiency, bleed flows and surge line pressure ratio. - linear function coefficients for calculating specific heat at constant pressure - gains associated with variable conversion between relative, normalised and actual values.		

[7] N_1 for the LP shaft and N_2 for the HP shaft.
[8] High pressure compressor (HPC) only.

3.4.4. Combustor

The air at the outlet of the HPC is passed into a combustion chamber. This increases the enthalpy of the working fluid via the combustion of fuel. The flame temperatures in the model are obtained from 2-D lookup tables that were calculated using NASA program SP273. This data was extracted from [2]. Therefore the exit temperature of the combustor is provided as a function of excess air factor, λ, and inlet temperature. To account for the unsteady mass balance between the HPC, combustor and HPT, a storage volume is included in the combustor.

Table 3.7.: **Summary of the combustor module's variables and parameters**

model parameter	**description**	**units**
Inputs		
T_{in}	inlet temperature	$[K]$
R	inlet ideal gas constant	$[J \cdot kg^{-1} \cdot K^{-1}]$
$fueloil$	fuel oil flow	$[kg \cdot s^{-1}]$
Outputs		
T_{out}	outlet temperature	$[K]$
p_{out}	outlet pressure	$[Pa]$
Boundary conditions		
M_0	initial combustor mass	$[kg]$
T_0	initial inlet gas temperature	$[K]$
Dynamic states		
M	combustor mass	$[kg]$
T	combustor temperature	$[K]$
Constant parameters:		
- stoichiometric fuel air ratios for gas and oil - 2D lookup tables for burner exit gas temerature calculation - lookup tables for converting between temperature and specific enthalpy - reference values for calculation of pressure drop - heatsoak design point time constant and heat transfer coefficient		

3.4.5. Turbines

The hot gas stream exits the combustor and is expanded via two turbines in series. A characteristic map is used to represent each turbine. The constitutive equation for the temperature drop across the turbine is:

$$T_{inlet} - T_{outlet} = \eta_t \cdot T_{in} \left[1 - \left(\frac{1}{p_{in}/p_{out}} \right)^{\frac{\gamma-1}{\gamma}} \right], \tag{3.25}$$

where η_t is the turbine isentropic efficiency. The HPT, although not modelled as a multi-stage cooled turbine, does include injection of cooling air from the HPC. Cooling air is injected into the main stream at the turbine inlet, therefore a mixing procedure has to be included in the module. For calculations of work the whole flow is used i.e. both the hot gases and the cooling air do work in the turbine. The mixing block takes the mass fraction of cooling air and calculates the temperature of the mixed flow:

$$T_{mix} = \frac{x \cdot T_{cool} \cdot c_{p,\,cool} + (1-x) \cdot T_{inlet} \cdot c_{p,\,hot}}{x \cdot c_{p,\,cool} + (1-x) \cdot c_{p,\,hot}}, \tag{3.26}$$

where x is the mass fraction of cooling air. The temperature of the mixture of hot and cold gas, T_{mix}, is then substituted for T_{inlet} in (3.25).

Table 3.9.: **Summary of the turbine module's variables and parameters**

model parameter	**description**	**units**
Inputs		
T_{cool}	cooling air temperature [9]	$[K]$
T_{in}	temperature of inlet gas stream [10]	$[K]$
p_{in}	inlet pressure	$[Pa]$
p_{out}	outlet pressure	$[Pa]$
N	shaft speed	$[rpm]$
FAR	fuel-air ratio [11]	
Outputs		
T_{out}	outlet temperature	$[K]$
$\dot{w}_{in}$	inlet mass flow	$[Pa]$
P	power produced	$[W]$
Boundary conditions: none		
Dynamic states: none		
Constant parameters:		
- lookup tables for calculating massflow and isentropic efficiency - linear function coefficients for calculating the gas streams' specific heat at constant pressure		

[9] HPT only
[10] In the case of the high pressure turbine, this is the combustor gas.
[11] This is used to calculate the thermodynamic properties of the gas stream.

3.4.6. Exhaust system

Exhaust air exits the turbofan engine in two separate unmixed streams: duct air and core section air. Both the duct and core exhaust are modelled as convergent nozzles, each of a fixed area. At current levels of cycle pressure ratio, virtually all turbojets (note that these engines do not have a duct section) operate with the nozzle choked during take-off, climb and cruise and the nozzle only becomes unchoked when thrust is significantly reduced. Thus the nozzle is liable to be unchoked only when preparing to land or when taxiing [36]. The situation is rather different for a turbofan and the core nozzle can be assumed to be unchoked over the operating range of the engine. The duct nozzle, due to the inherently low speed of the duct gas stream is also unchoked. The core and bypass flows are therefore expanded through the core and bypass nozzles to the pressure of ambient air. A converging nozzle is used because the exhaust flow is subsonic and for subsonic flow, a convergent nozzle accelerates the gas. The "throat" or minimum area of the nozzle will regulate the amount of flow that can be exhausted through the nozzle.

The object of a turbine nozzle is to produce as much kinetic energy as possible from given inlet conditions and pressure drop. There will inevitably be some frictional losses - maximum efficiency will be when there is no friction and the process is perfectly isentropic. The efficiency of the nozzle is the ratio of the actual kinetic energy produced to this theoretical maximum. As the nozzle efficiency falls below unity, the expansion will be a mixture of isentropic and isothermal [39]. Conditions at the inlet of the nozzle shall be denoted with subscript "0", e.g. p_0, v_0, T_0 for the pressure, specific volume and temperature respectively. Similarly conditions at the exit of the nozzle are denoted with subscript "1". Conditions at the nozzle throat are denoted with subscript "t".

The nozzles of a gas turbine receive a gas that already possesses an appreciable velocity, therefore the equations used to model the nozzle must incorporate factors to account for these significant inlet velocities. The non-negligible gas velocity may be accounted for by using the concepts of stagnation pressure, p_{0T}, and stagnation temperature, T_{0T}. For a gas undergoing an isentropic expansion:

$$pv^{\gamma} = constant, \tag{3.27}$$

where γ, the ratio of specific heats, is known as the adiabatic index because this is an isentropic (i.e. adiabatic and reversible) process. For a frictionally resisted, adiabatic expansion this becomes:

$$pv^{m} = constant, \tag{3.28}$$

where the polytropic exponent, m, is a function of the adiabatic index, γ, and nozzle efficiency η_N:

$$m = \frac{\gamma}{\gamma - \eta_N(\gamma - 1)} \tag{3.29}$$

The nozzle throat will pass flow at speeds up to and including sonic but cannot support supersonic flow (a convergent nozzle slows down supersonic fluids). Sonic flow will be reached when the ratio of throat pressure p_t to inlet stagnation pressure p_{0T} has reached a critical value (the nozzle is "choked"):

$$\frac{p_{tc}}{p_{0T}} = \left(\frac{2}{\gamma + 1}\right)^{\frac{m_c}{(m_c - 1)}}, \tag{3.30}$$

where p_{tc} and m_c are respectively the throat pressure and polytropic exponent at these critical conditions.

Furthermore, assuming that nozzle efficiency is constant over the length of the

nozzle (the variation, for a convergent only nozzle, has been found in practice to be small up to and including sonic velocity [39]) and under the assumption that pressure and specific volume in the stagnation state are related to temperature by:

$$p_{0T} v_{0T} = \bar{R} T_{0T}, \tag{3.31}$$

where $\bar{R}$ is the specific gas constant, the nozzle's massflow is then:

$$\dot{w} = A \sqrt{2 \frac{\gamma}{\gamma - 1} \frac{p_{0T}}{v_{0T}} \left(\left(\frac{p_1}{p_{0T}} \right)^{\frac{2}{m_0}} - \left(\frac{p_1}{p_{0T}} \right)^{\frac{(m_0+1)}{m_0}} \right)} \tag{3.32}$$

for

$$\frac{p_1}{p_{0T}} \geq \left(\frac{2}{\gamma + 1} \right)^{\frac{m_c}{(m_c-1)}}, \tag{3.33}$$

i.e. for subsonic flow. A is the throat area and is the same as the exit area for a convergent-only nozzle.

An important issue is that there is no value for the specific volume at stagnation conditions, v_{0T}, in the model at this stage, but this can easily be calculated from the relationship in (3.31). Under the assumption that $m_c = m$, i.e. the polytropic index is not greatly varied at critical conditions [39], the model can be provided with an architecture that can also cope with sonic flow if the following equation is also included in the nozzle module:

$$\dot{w} = A \sqrt{\left(\frac{2}{\gamma + 1} \right)^{\frac{(m_c+1)}{(m_c-1)}} \gamma \frac{p_{0T}}{v_{0T}}} \tag{3.34}$$

for

$$\frac{p_1}{p_{0T}} < \left(\frac{2}{\gamma + 1} \right)^{\frac{m_c}{(m_c-1)}} \tag{3.35}$$

This raises the question of how to determine the throat pressure given only the inlet and outlet pressure for the nozzle, which will be the usual case. Because for a convergent only nozzle the nozzle throat and outlet are adjacent, the throat pressure will be the same as the outlet pressure until the ratio of the outlet to inlet pressure falls below the critical value (i.e. the nozzle becomes choked), and thereafter the throat pressure will remain at the critical value.

Table 3.11.: **Summary of the exhaust module**

model parameter	**description**	**units**
Inputs		
T_{in}	LPT exhaust temperature	$[K]$
P_{amb}	ambient pressure	$[Pa]$
FAR	fuel-air ratio	
Outputs		
$\dot{w}_{out}$	nozzle massflow	$[kg \cdot s^{-1}]$
Boundary conditions: none		
Dynamic states: none		
Constant parameters:		
- nozzle diameter - nozzle efficiency		

3.4.7. Shafts

Speeds at time t are calculated using component power values, speeds from the previous point and the spool inertias. The rotational acceleration of the shaft can be found from the shaft dynamic balance. For example, for the LP shaft:

$$\frac{d\omega}{dt} = \frac{1}{I\omega}(P_{LPT} - P_{LPC} - P_{Fan} - P_{losses}), \tag{3.36}$$

where P is power, I is shaft moment of inertia and ω is the shaft angular speed. Power losses are caused by friction and engine accessories and can be modelled as a simple percentage of the total power provided to the shaft, or as a loss that is proportional to shaft speed.

Table 3.13.: **Summary of the shaft module**

model parameter	**description**	**units**
Inputs		
P_{out}	power out [12]	$[W]$
P_{in}	power in [13]	$[W]$
$\dot{w}_{in}$	mass flow of inlet air	$[kg \cdot s^{-1}]$
Outputs		
N	shaft speed	$[rpm]$
$\dot{N}$	shaft acceleration	$[rpm \cdot s^{-1}]$
Boundary conditions: none		
Dynamic states		
N	shaft speed	$[rpm]$
Constant parameters:		
- shaft moment of inertia		

[12]This is the power consumed by the HPC in the case of the HP shaft, and the power consumed by the fan and the LPC for the LP shaft.
[13]This is the power provided by the HPT in the case of the HP shaft, and the power provided by the LPT for the LP shaft.

3.4.8. Actuators

Models for mechanical actuators such as those of the fuel system valve, the variable bleed valves (VBVs) and the stator vanes (VSVs) are also included. These are modelled in terms of first and second order transfer functions [34]. The temperature sensor (transducer), also has its own dynamics and these too are represented

in the model via transfer functions according to the representation in [13].

3.5. Summary

Chapter 3 analysed in in greater detail than Chapter 2 the architecture of a gas turbine, presenting the key equations that can be used to model each component, showing how they can be put together, and describing the principles of the simulation architecture.

The chapter starts with a brief summary of the engine's components. These, having been discussed in chapter two are now given a whistle-stop tour: the purpose is to show how the components are interconnected, with the assistance of a diagram. The simulation's basic principle, the use of "lumped" elements was then discussed: the engine components are simplified to volumeless elements, thereby reducing the partial differential equations that describe their distributed properties to ordinary differential equations that describe the evolution of key properties over time. The aim is to derive a set of explicit, first order differential equations that can be solved using an integration algorithm to accurately describe the dynamic characteristics of the modelled components. The unsteady mass balance between components is taken into account via storage volumes (plena).

The key elements of each component are then described. Some can be described algebraically, others require the use of lookup tables. This is the case for the turbomachinery components, whose airflow and relationship between is far too complex to be simulated in a model of this scale - experimental maps are used.

4. Controller specifications, architecture and design

The engine controller serves the dual purpose of assisting with simulating and validating engine transient performance and also as a benchmark for future controls development. This chapter discusses the controller architecture: a switched, gain-scheduled, feedback control system incorporating bumpless transfer and anti-windup functionality.

4.1. Introduction

One of the key objectives of an aircraft engine is to achieve maximum thrust with minimum engine weight. Therefore although a gas turbine is inherently in itself a stable system [17], feedback control is an essential part of jet engines because of the requirement to run close to the engine's operating limits. Transient conditions do exist for which operation is unstable, but because these are conditions that may seriously damage the engine (an example of this is surge, a violent oscillatory reversal of the gas stream's flow) they are considered as areas of the operating envelope that are to be avoided. This is achieved via open loop scheduled controls on some of the engine components (e.g. bleed valves and stator vanes) and by closed-loop

limiting of both shaft acceleration and the rate of change of fuel flow. Therefore these unstable conditions are not part of the feedback control problem for civil aircraft engines. The main aim of the engine controller is therefore to guarantee that the engine will be maintained within its operating limits at all times, regardless of how the operator may move the throttle lever or of inlet conditions (e.g. altitude and Mach number). The engine's operating limits may be of a mechanical, thermal or aerodynamic nature (e.g. surge). In addition to these constraints the overall engine performance is also subject to both regulatory standards and the requirements of the airframe manufacturer.

The highly constrained and variable environment, in which the engine must operate, requires a controller that can cope with such conditions. Due to the range of environmental conditions, it is unlikely that a single linear controller would be able to achieve adequate performance across the operating envelope, although a simple PI controller is well-known to yield good performance at a given operating point. Therefore, gain-scheduling was used in order to vary the PI controller's gain as a function of varying operating condition. Furthermore, the constraints on the various engine variables mean a single PI controller may be unable to deliver adequate performance whilst adhering to these limits. For this reason switching logic was used to choose a PI controller yielding the highest level of performance without leading to violation of constraints. Due to the way the switching logic was implemented, anti-windup protection was implemented on all off-line regulators to prevent integrator run-away. The resulting controller architecture is thus a gain-scheduled (with operating point), switched (as a function of active constraints) PI controller. As a useful aside, the anti-windup logic used for preventing integrator run-away, also provided a simple bumpless transfer mechanism in order to reduce the magnitude of the so-called "bump" which may occur during controller substi-

tution.

What a pilot wants to achieve when moving the thrust lever is a certain percentage of the maximum thrust available at the current flight conditions, not a specific net thrust from the engine. Since thrust itself is not measurable in-flight, the relative thrust command given by the PLA (power lever angle) setting must be translated into a command change of a measured variable. Thrust corresponds well with the low pressure shaft speed (after taking into account aircraft speed and environmental conditions such as pressure and temperature), particularly in the case of a high bypass turbofan, where the majority of the thrust is provided by the fan. Therefore it is common practice to choose fan speed as the controlled parameter (this, for example, is the choice adopted by General Electric [17], although other vendors may and do choose other indirect measurements of thrust). Fuel flow is the sole control input available, as other engine components that may modify the engine's operating point, such as bleed valves or variable stator vanes, are in this instance open loop scheduled. On advanced civil aircraft these may be "tweaked" according to the operating condition and past history of the engine, but they are not, in any case, truly feedback controlled as would occur in multivariable military jet engine control schemes. Furthermore, the exhaust nozzle on a civil aircraft is fixed and as such cannot be used to modify the engine's operating point, as occurs on military engines. A diagram of the current model's closed-loop control system is provided in Fig. 4.1.

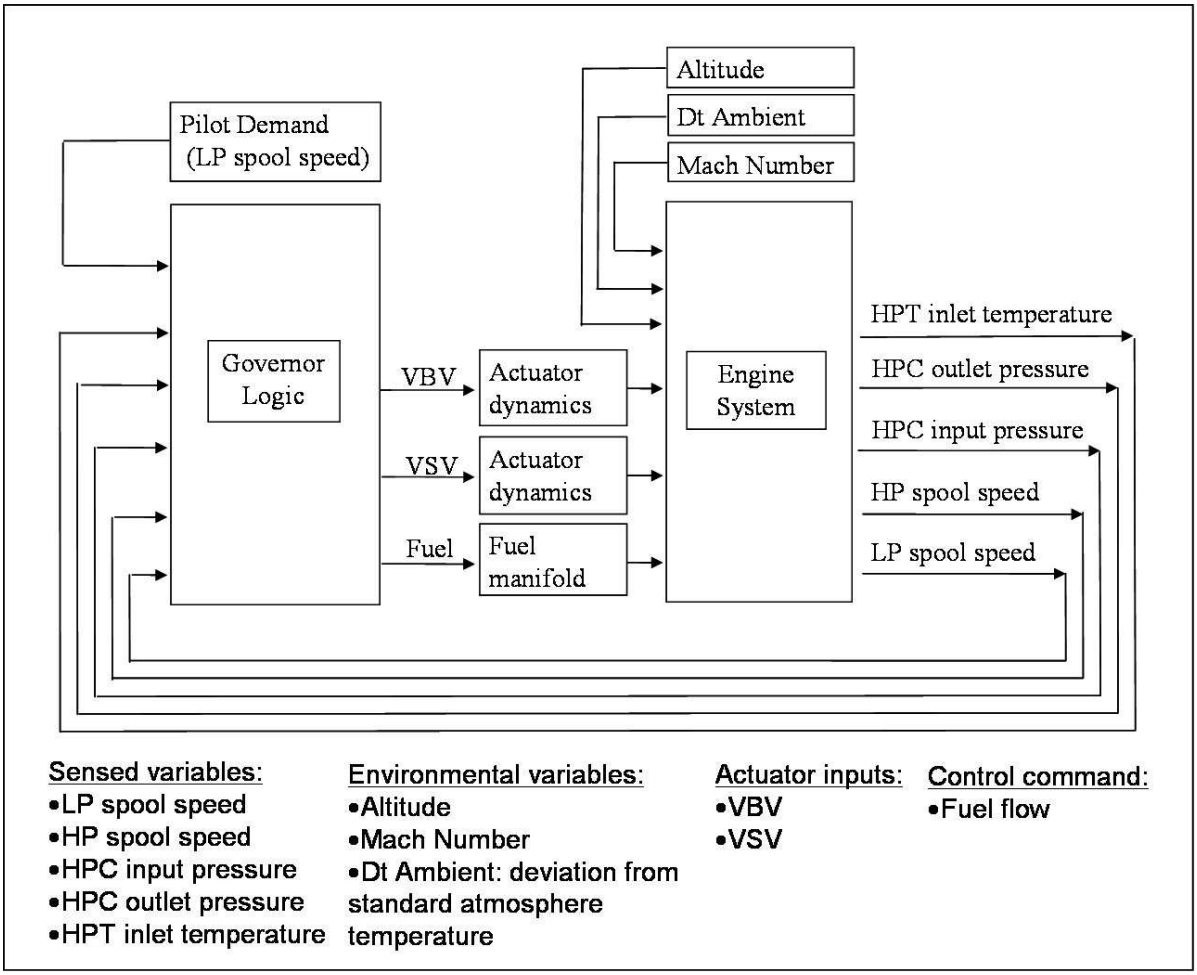


Figure 4.1.: **Diagram of the current model's closed-loop system.**

The engine's operating limits may be divided into two groups: maximum limits are those that must not be exceeded from below and minimum limits are those that must not be exceeded from above.

Maximum limits are as follows: fan speed (i.e. LP spool speed), fuel flow increase rate, HP spool speed (the shaft that joins the HP turbine and the HP compressor), HP spool acceleration, HP turbine inlet temperature, HP compressor discharge pressure and fuel flow.

Minimum limits are: LP spool speed, fuel flow decrease rate, HP spool speed, HP spool deceleration, and fuel flow.

With the exception of fan speed demand, minimum LP spool speed, maximum fuel and fuel flow rate, all of the above are currently set to fixed values - however the architecture would be able to cope with these varying according to operating conditions or different engine requirements following, say, a failure. The engine's idle condition may be any of the minimum limits on LP spool speed, HP compressor delivery pressure or fuel, according to the engine's operating condition and environmental factors.

The engine is maintained within its operating limits by fuel regulators, these can also be split into two categories: steady state control regulators and transient control regulators. The steady state regulators consist of fixed limits that prevent the engine operating outside fixed bounds, for example, to limit against over-temperature, over-speed, fuel metering valve high and low limits, minimum idle limits and flame out limits. These are maximum and minimum HP shaft speed, fan speed control, maximum turbine inlet temperature, maximum and minimum fuel flow and maximum and minimum fuel flow rate. The aim of the transient control regulators is to control the HP shaft rate of acceleration and fuel rate during sharp acceleration demands, and to control minimum core speed and fuel flow rate of

reduction during sharp deceleration conditions.

There are several regulators within the designed controller and some are there as a back-up to the others, for example, the duplication of acceleration and deceleration regulators (thereby allowing control of the acceleration rate of both shafts). An important motivation for the development of the model is to allow investigation of robust and adaptive control solutions to specific failure modes. Should a failure mode affect the fuel system, then the acceleration profiles would need to be controlled via the speed rate regulators. Should a failure mode affect the variable geometry or bleed systems, the compressor delivery pressure would be significantly affected along with the relative pressure regulators. The performance of the engine under these failure modes needs to be assessed and so several regulators have been implemented. The aim is to keep the controller design as open as possible to accommodate future research objectives.

Indeed, the engine's operational limits also need to be considered in the context of engine degradation or failure. As an example, the acceleration of a healthy engine is usually limited by the maximum permissible rate of acceleration of any of the engine's shafts (note: this is not a mechanical limit but is to avoid surge and therefore the maximum rate of acceleration of the HP shaft tends to dominate the acceleration profile), the allowed fuel flow rate of increase, and possibly the maximum permitted fuel flow (for an aggressive acceleration profile). A healthy engine can therefore successfully reach its maximum rated power without encountering any further limits. However for a condition of low ambient pressure with extreme bleed and high ambient temperature, the maximum turbine inlet temperature may instead become the operating envelope's limiting factor, and the engine may not reach maximum power but will instead settle at the maximum turbine inlet temperature. Furthermore, transient turbine inlet temperature is higher than that

permitted during steady state operation, therefore the maximum power achievable would be further reduced after a short period of time. Another consideration is that these operating limits may not always be set at a fixed value. For example, the engine's minimum LP spool speed is increased at high altitude or during landing (to allow for a rapid increase in thrust in case of an aborted landing) and fuel flow rate of change is scheduled according to the engine's operating point. A further example of this is a software upgrade by the engine manufacturer: it is current practice that an engine's performance may be capped to increase lifespan and fuel economy. However should the aircraft owner have a requirement for improved performance he may pay for a software upgrade to remove some of these limitations. In addition to the engine operating limits, there are also the regulatory requirements on the overall behaviour of the engine. These standards are issued by an international or national authority, for example the US Federal Aviation Authority (FAA). Typical requirements for a civil aircraft [17], are that upon receiving a demand for an increase in thrust, the engine must not overshoot its target by more than 2%, with a maximum steady state deviation of 1%. The engine's time to accelerate to maximum thrust is also regulated; this is a specification of the airframe manufacturer and often takes the form of a requirement that the engine must be able to accelerate at Sea Level Static (SLS) from within 15% of idle to full throttle within a specified timeframe, e.g. 10 seconds and preserve this acceleration profile over a given altitude range.

4.2. Controller architecture

The controller's input is relative thrust demand (a percentage of the maximum thrust available at the engine's operating point, here considered proportional to

fan speed) and the output is fuel flow demand. Performance requirements are satisfied while maintaining the engine within its operating limits by assigning a gain-scheduled Proportional-Integral (PI) controller to each of the individual constraints. A selection logic then chooses amongst these so that at any point of its operating envelope the engine is governed by an appropriate controller. The gains of each PI controller are scheduled as a function of engine LP shaft speed and altitude using lookup tables that are dynamically initialised from the workspace, over a flight envelope from Sea-Level to 35000 ft and over values of engine shaft speed from idle to maximum thrust. Bumpless transfer and anti-windup are also provided by the controller architecture. For the purpose of clarity, for the remainder of this dissertation each individual PI loop will be referred to as a "regulator", with only the overall architecture and resultant control system referred to as "the controller".

All of the regulators that control a "maximum" limit (e.g. fan speed, HP turbine inlet temperature) are routed into a "minimum" selection block that selects the smallest output of these regulators. The resultant signal is then passed into a "maximum" selection block along with the output of all the regulators assigned to "minimum" limits. This "maximum" selection block then chooses the greatest of these signals. A diagram of this architecture is shown in Fig. 4.2. A useful feature of this minimum/maximum selection architecture is that it inherently provides bumpless transfer since a switchover between regulators may only occur at a crossover point: when one of the inactive regulators' outputs intersects that of the active regulator. What this means in practice is best illustrated by a diagram: Fig. 4.3 shows the individual fuel demands of the active regulators during a pilot demand - in this case an abrupt increase in thrust at Sea Level Static (SLS), to 98% of maximum thrust, in a time span of half a second. It is immediately ap-

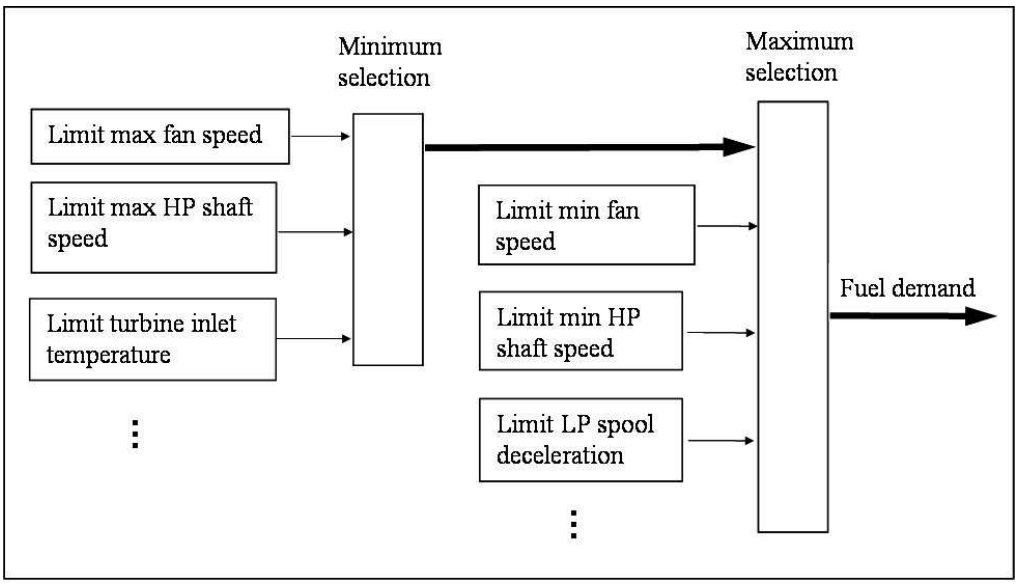


Figure 4.2.: **Selection logic architecture.**

parent that it is the "maximum" regulators (i.e. limits not to be exceeded from below) that govern the engine's behaviour during an acceleration. Conversely, it is the "minimum" regulators that intervene during a deceleration as is shown in Fig. 4.4. Figure 4.5 shows the final controller demand after the selection logic has been applied to the output of all regulators. By comparing this to Fig. 4.3, it is apparent that the controller has selected the smallest of the regulators' demands during acceleration, and that the transfer between these is smooth. Conversely it has selected the largest of the regulators' demands during deceleration. Note that the controller's output is converted to fuel flow units ($kg\ s^{-1}$) in a module external to the controller, to allow for an easy implementation of different fuels should the model be adapted in the future to a different configuration, e.g. an industrial aero-derivative engine.

The switchover between regulators is evident in the next plot: Fig. 4.6 shows which regulator is controlling the engine at any moment of time. The numbers along the ordinate of the plot indicate the active regulator: 1 is the fan speed

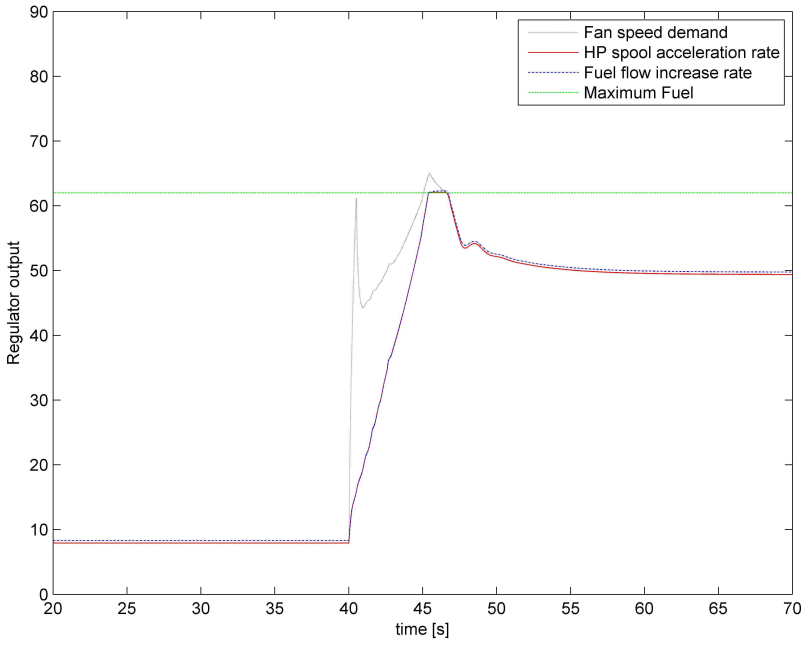


Figure 4.3.: **Active regulator output during acceleration.**

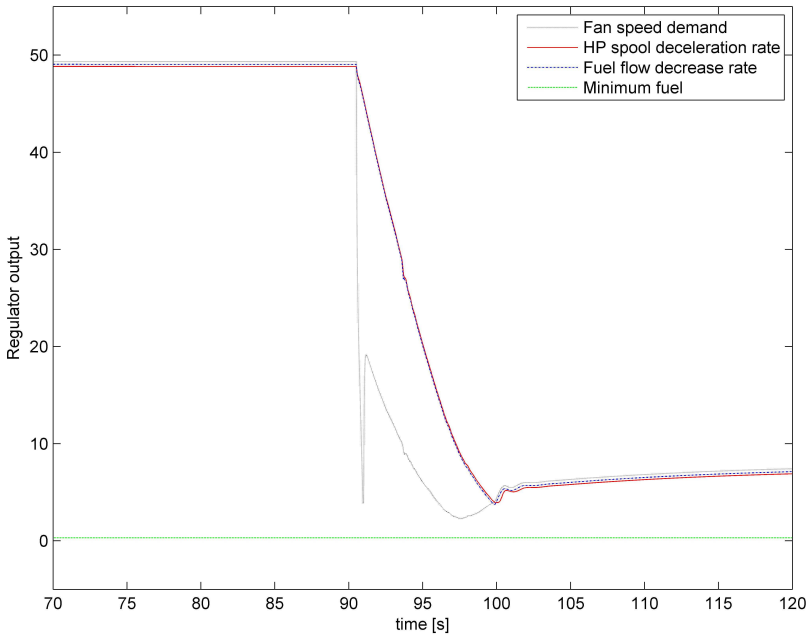


Figure 4.4.: **Active regulator output during deceleration.**

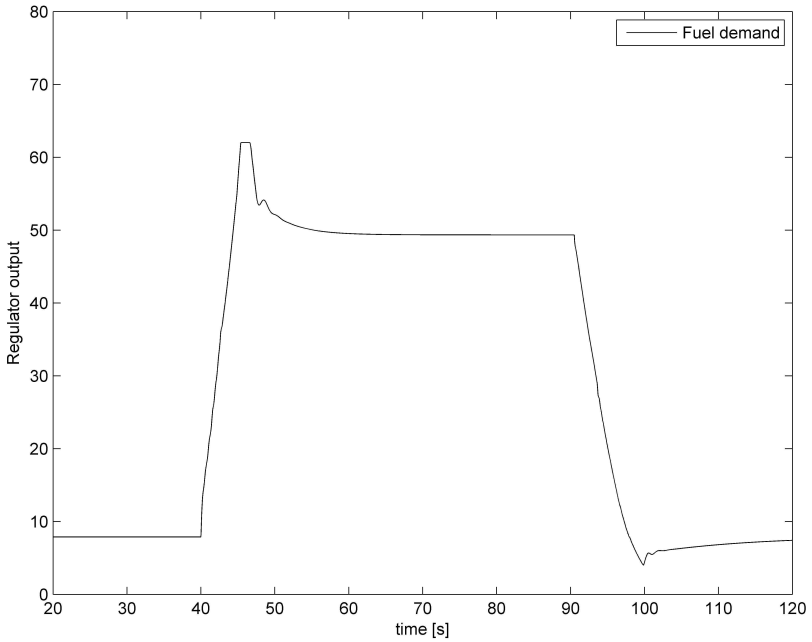


Figure 4.5.: **Controller fuel demand.**

(LP shaft) regulator, 8 is the fuel flow increase rate regulator, 4 is the HP shaft rate of acceleration regulator, 7 is the maximum fuel regulator, 15 is the fuel flow decrease rate regulator, 12 is the HP shaft rate of deceleration regulator and 13 is the minimum fuel regulator. At steady state, up until time t=40, the engine is controlled by the fan speed demand regulator, then upon a pilot request at t=40 for an abrupt acceleration, the fuel demand of the fan speed regulator rapidly ramps up. At approximately t=40.01, the active regulator briefly switches over to regulator 8 - fuel flow increase rate. During the remainder of the acceleration the engine is controlled by the HP shaft maximum acceleration rate regulator (number 4), the fuel flow increase rate regulator (number 8) and the maxiumum fuel regulator (number 7, at time t=45-46), as the target speed is reached control is switched back to the fan speed regulator - which requires a shaft deceleration as a small overshoot occurs, thereby activating the high pressure shaft deceleration regula-

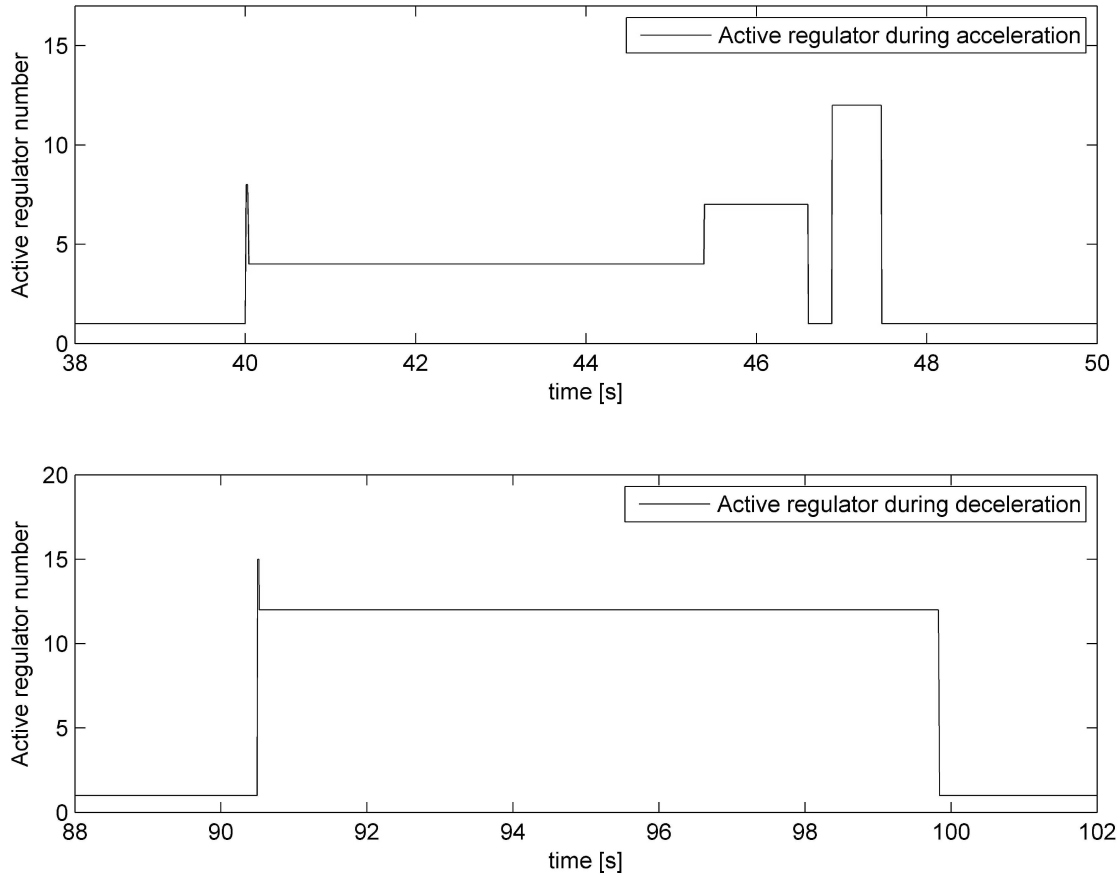


Figure 4.6.: **The ordinate gives the number of the active regulator: 1 is the fan speed (LP shaft) regulator, 4 is the HP shaft rate of acceleration regulator, 8 is the fuel flow increase rate regulator, 7 is the maximum fuel flow regulator, 15 is the fuel flow decrease rate regulator, 12 is the HP shaft rate of deceleration regulator.**

tor (number 12, at time t=47-47.5) before control finally returns to the fan speed regulator. The inverse process occurs after t=90.5, upon engine deceleration: the fuel flow decrease rate regulator is the first regulator to take over from fan speed but is subsequently replaced by the HP shaft deceleration regulator. The overall controller output is a smooth ramp and engine acceleration proceeds accordingly until maximum thrust has been achieved. This can be seen clearly in Fig. 4.7.

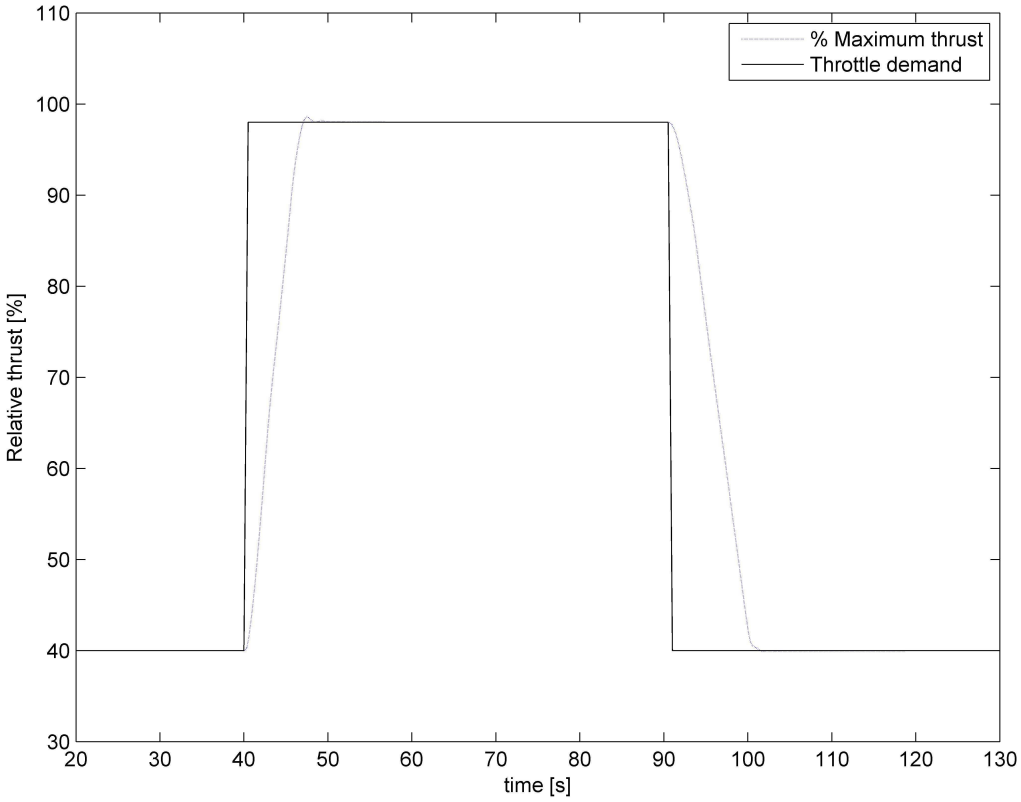


Figure 4.7.: **Thrust response to large throttle demand.**

4.3. Preventing integrator windup

The bumpless transfer logic in the controller works well, as indeed it must - it is an inherent property of the architecture of the controller: a minimum and maximum selection logic applied successively to groups of individual regulators. However, although this bumpless transfer is inherent to the minimum/maximum selection logic, this switching between controllers could not in practice occur with just a PI control for each regulator - each PI regulator must have a procedure in place to prevent the integrator's output from growing indefinitely when that particular regulator is not active. This would otherwise break the logic of the architecture. Consider if the maximum LP shaft acceleration regulator did not have an integrator reset in place - then while the engine is at steady state, its integrator's output would grow without bounds and switchover to the acceleration limit (upon a demand for increased thrust) would only occur after the maximum acceleration limit had already been greatly exceeded - or not at all if the engine had been running at steady state for a long enough period of time. An effective anti-windup architecture that ensures that the integrator's output does not grow without bounds is shown in Fig. 4.8. When a regulator is active (it is notified of this by a module that tracks which is the active regulator) the switch selector (as seen in Fig. 4.8) will pass the regulator's own signal around the feedback loop, and at point B the output will be zero and the regulator will see an unmodified error signal. When the controller is inactive, the switch will pass the value of the current active regulator. This is then fed back via point B, after being subtracted from the signal at point A and then multiplied by the gain W_G - the result is then subtracted from the error, thereby effectively reducing the error seen by the inactive regulator and thus reducing its value.

When the output of the active controller is below the regulator's target, the track-

ing will reach a value slightly above that of the active regulator, and vice versa - when the output of the active controller is above the regulator's target it will rest at slightly below the active signal (i.e. the regulator's output will track the active signal from above and below respectively). This property, in conjunction with a maximum and minimum selection logic, is what allows the overall architecture to function.

The value of the gain W_G underpins the operation of the anti-windup loop. For

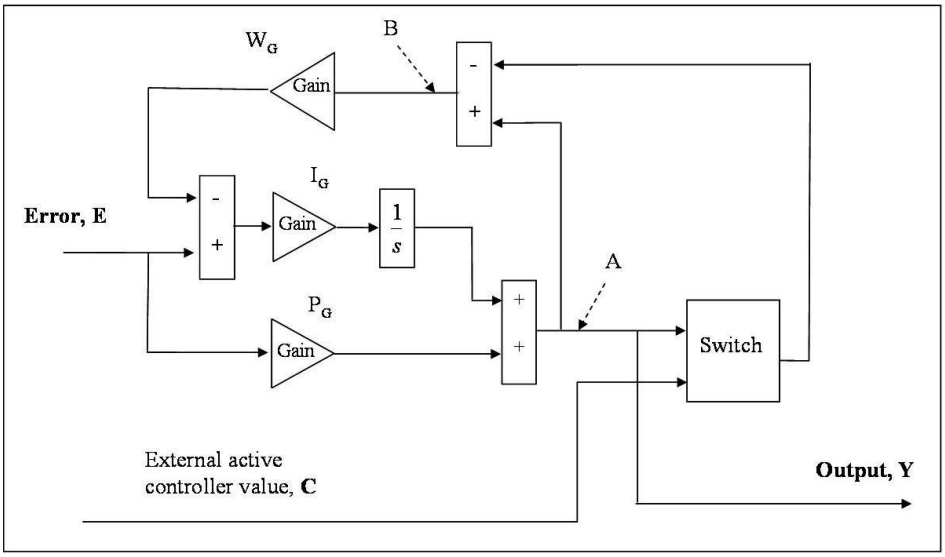


Figure 4.8.: **PI controller structure with integrator anti-windup.**

example, one may wish to know the value of a regulator when it is inactive and the engine has reached a steady state - e.g. the condition before an abrupt acceleration. Simple inspection leads to the following conclusion - the output of an inactive controller will reach a fixed value when:

$$(A - C) \cdot W_G = E,$$

which if rearranged gives the steady state value of A as:

$$A = E/W_G + C \tag{4.1}$$

It would seem from this that during steady state operation it is desirable to have a large W_G, so that E/W_G tends to zero and A≈C, where C is the value to be tracked. Under these conditions a maximum limit regulator would track the active regulator's output asymptotically from above, whilst a minimum limit regulator would track the active regulator's output asymptotically from below. The requirements for this to occur are readily highlighted under the assumption of a discrete model (as all simulated models are in practice, due to numerical integration) and a single, small, time step update Δt. For a "maximum" limit, assuming constant E and C, the change in value of A over one time step is approximately:

$$I_G(E - (A - C)W_G)\Delta t \tag{4.2}$$

Since A > C for a "maximum" limit, the major requirement is that the value subtracted from A during one time step must be less than A-C. Consider the worst case scenario of a negligible error value (since a large error value helps in satisfying the above condition),

$$(A - C) \cdot W_G \cdot I_G \cdot \Delta t < (A - C) \tag{4.3}$$

simplifies to:

$$\Delta t \cdot W_G \cdot I_G < 1 \tag{4.4}$$

Similarly for a "minimum" limit, since now C >A, the requirement is that the value added to A during one time step must be less than C-A, i.e. :

$$-(A-C)\cdot W_G \cdot I_G \cdot \Delta t < (C-A), \tag{4.5}$$

which again reduces to the expression in Eq. (4.4):

$$\Delta t \cdot W_G \cdot I_G < 1 \tag{4.6}$$

These considerations may tempt one to arrive at the conclusion that W_G should be as large as possible subject to the above constraint - but first we need to consider what happens during a single time step update for a time-varying controller output C, assuming a constant error E. Under such a circumstance:

$$A(t+\Delta t) = P_G E + I(t) + I_G \cdot \{E - [A(t) - C(t+\Delta t)]W_G\}\Delta t, \tag{4.7}$$

where

$$I(t) = I_G\{E - [A(t-\Delta t) - C(t)]W_G\} \tag{4.8}$$

The requirement for a "maximum" limit to retain a margin of safety, i.e. how much its signal rides above the signal to be tracked, is then:

$$I_G \cdot \{E(t) - [A(t) - C(t+\Delta t)]W_G\}\Delta t \geq \Delta C \tag{4.9}$$

and similarly for a "minimum" limit :

$$I_G \cdot \{E(t) - [A(t) - C(t+\Delta t)]W_G\}\Delta t \leq \Delta C, \tag{4.10}$$

under all conditions. This is aided by a small Δt, but too large a value of W_G can indeed lead to the value of A dipping below that of C when C should rise very rapidly. For the system under consideration, it has been found that this would require very large values of W_G - of the order of 10^3 or more - and that excellent tracking of C is achievable with values in the order of 10^2, without fear of the tracking regulator's output crossing over C. However, in general one must be aware that a safe value of W_G depends on the system considered - and in particular on the maximum rate of increase of C and E - and that possible variations in the time step, Δt, are also of consequence. The implications of a time-varying error during transients are a logical extension of the above.

In summary, it is not possible to provide an absolute guarantee that a certain value of W_G will be appropriate for all conditions unless clear boundaries on E, C and their rate of increase are established. Therefore the value of W_G must in general be verified for the range of conditions expected.

4.4. Controller continuous analysis

The control structure in figure 4.8 can be rephrased as a reference tracking problem as shown in figure 4.9.

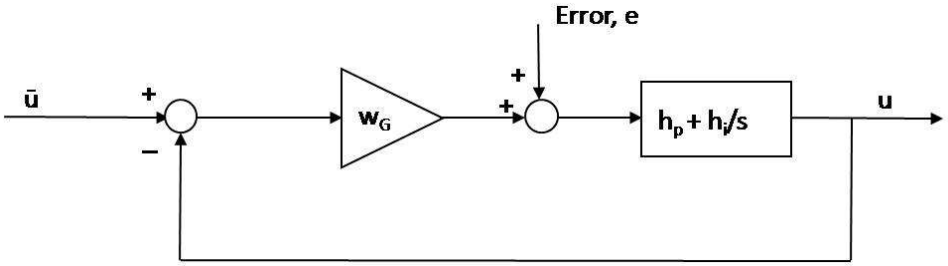


Figure 4.9.: **Controller structure as reference tracking**

Accordingly, defining K_s as:

$$K_s = \frac{h_p s + h_i}{s}$$

then,

$$u = K_s[e + W_G(\bar{u} - u)] \tag{4.11}$$

$$= (I + K_s W_G)^{-1}[K_s e + K_s W_G \bar{u}]$$

$$= \frac{\frac{h_p s + h_i}{s} e + \frac{h_p s + h_i}{s} W_G \bar{u}}{1 + (\frac{h_p + h_i}{s}) W_G}$$

and finally:

$$u = \frac{h_p s + h_i}{s + (h_p s + h_i) W_G} e + \frac{(h_p s + h_i) W_G \bar{u}}{s + (h_p s + h_i) W_G} \tag{4.12}$$

Therefore when W_G is large compared to s, then $u \approx \bar{u}$. Thus there is good tracking when $\bar{u}$ contains predominantly low frequency and W_G is large. There is poor tracking when $\bar{u}$ contains higher frequencies. For large W_G there is better tracking and improved bandwidth, but a decreased stability margin. Note however, that the analysis conducted here is local to the PI-control loop and does not account for the (hybrid) dynamics of the overall system, an analysis of which is significantly more difficult.

4.5. Reference tracking: requirements and consequences

How closely an inactive regulator must track the active regulator depends on how rapidly switch-over must occur. It must not, however, track the active regulator too closely, thereby providing a 'safety margin' in case of a rapid change in the value of C. Time for switch-over also depends on the size of the gains P_G and I_G - with

sufficiently large gains it has been found that the time for switch-over is negligible and a good safety tracking margin can be achieved with no fear of switch-over being delayed enough for any of the regulator limits to be exceeded by more than 1%. It is clear then that this controller architecture requires extensive validation and tuning to the dynamics of the controlled plant: the architecture does not guarantee that no limits will be exceeded should the plant's dynamics be significantly faster than those predicted during testing, although this does not pose a serious problem for gas turbines, whose dynamics are well known and predictable. Furthermore, the presence of multiple regulators also provides a degree of redundancy.

4.6. PI control and gain schedule

A diagram of the implementation of the PI regulators, with anti-windup, is shown in Fig. 4.8. The gains are implemented via the script "PI Gainschedule.m", which stores all values of the controller's gain-schedule and places them in the workspace in a format suitable for the Simulink lookup tables. In accordance with the flexible architecture of the controller's implementation, this script is designed to permit the number of variables controlled to be easily extended.

The tuning of each of the PI loops was done manually in the standard fashion: the proportional gain is raised until there is oscillation and then reduced somewhat. Despite the steady-state offset, both tuning theory and industrial practice indicate that it is the proportional term that should contribute the bulk of the output change, however in this instance, due to the switching and cascaded architecture of the overall controller, the integral gain may be significant.

The proportional and integral gains are both scheduled versus fan speed and altitude, over the range of 1000 to 5000 rpm and from sea level to 35000 ft. Mach

number would be a good additional scheduling parameter, but the increased complexity is not required for our purposes. The script "PI Gainschedule.m" allows the user to set gain values for each combination of fan speed and altitude or alternatively, as an aid to quick and easy tuning, the function "PIgainscheduleInterpolation.m" can be called from within the script: once gains are chosen for the extrema of each range, this script will then linearly interpolate between the outlying values, thereby completing the matrix of gain values. This script can be seen in the appendix, section B. It would be more appropriate to interpolate based on the sum of fan and HPC power, but this is data that will change upon a new target design and while the engine is being tuned, so it is best to work with what does not change significantly: a target fan speed and altitude. It is a compromise, but one that works well.

4.7. Summary

Chapter 2 introduced the basic principles of aero-engine gas turbines and Chapter 3 discussed the mechanical components of the engine in greater detail. This chapter then proceeded to discuss the engine's control system.

The engine controller serves the dual purpose of assisting with simulating engine transients and, as representative of a baseline industrial implementation, also as a benchmark for future controls development. The architecture is a switched, gain-scheduled, feedback control system incorporating bumpless transfer and anti-windup functionality. The appeal of this architecture was its transparent structure to the control system designer and its ability to accommodate variation in operating condition while ensuring various constraints are not violated. Moreover, sample simulation results have indicated that the controller performs well in realistic sce-

narios.

There are several regulators within the designed controller and some are there as a back-up to the others, for example, the duplication of acceleration and deceleration regulators (thereby allowing control of the acceleration rate of both shafts). An important motivation for the development of the model is to allow investigation of robust and adaptive control solutions to specific failure modes. Should a failure mode affect the fuel system, then the acceleration profiles would need to be controlled via the speed rate regulators. Should a failure mode affect the variable geometry or bleed systems, the compressor delivery pressure would be significantly affected along with the relative pressure regulators. The performance of the engine under these failure modes needs to be assessed and so several regulators have been implemented. The aim is to keep the controller design as flexible as possible to accommodate future research objectives.

The entire controller can be easily replaced (or switched off) by the user, to evaluate novel control designs.

5. Model implementation and simulation

Up until this point we have given an overview of aero-engine characteristics and discussed the theory behind the aero-engine simulation and the logic of its controller. The practical implementation aspect - the simulation itself, also requires some explanation. Here we discuss some of the most important practical aspects of translating this theory into a simulation architecture: the simulation environment, its initialisation to steady state conditions and the implementation of the aerothermal lookup tables. The nature of the feed-forward solution also has significant implications for the simulation, and can lead to algebraic loops when a component requires data from downstream. These are discussed towards the end of this chapter.

The values of engine parameters and tuning values for the characteristic maps are provided in the model and the startup scripts. There is little value in listing them lengthily here. A description of the startup scripts, their uses and the parameters they initialise can be found in the appendix (section A).

5.1. The simulation environment

The model is implemented within the Matlab-Simulink® environment, largely as an industry requirement. The use of Simulink for large simulations implies by necessity the use of Matlab code to supplement the Simulink environment, and the choice of Simulink itself is dictated by the fact that it is an industry standard and appears to be, at a first glance, an easily accessible platform. A key feature is that it provides a graphical interface that allows the designer to visually subdivide a simulation into block modules that correspond to physical components or logical groupings. Whether in the long run this remains the most viable method of implementing large simulations (and in particular those that do not benefit from one of Simulink's own toolkits) is open to debate: code that would be simple to implement in a conventional programming language, both procedural or object oriented, can lead to intricate nested blocks and entanglements of weaved interconnects in Simulink. In the author's opinion, this can often obfuscate any residual intelligibility. As is often said: "the devil's in the details!", and although the overall simulation structure may be apparent at the top level, the details of the individual modules can be devilishly intricate. However the presence of a large widget toolkit for displaying values and implementing controls does make rapid prototyping easy, and it is perhaps here that its strength lies. As shown in figure 5.10, due to the graphical nature of Simulink the engine components are clearly visible on the top layer of the simulation, making it easy to understand the parallel between the simulation and the physical apparatus. Furthermore it is easy for anyone to copy, paste and interconnect the individual modules without any programing knowledge. The aero-engine model was developed for the purpose of investigating future strategies for robust fault tolerant aero-engine control, hence we seek to provide an implementation architecture that permits the model to be easily altered - to both

simulate faults and to adapt it to different engine configurations if so wished. The development process itself has these same requirements and this prompts the use of Matlab scripts to load data sets that contain variables to be assigned to the main workspace at startup. As an example of this, where possible all Simulink tables are initialised using vectors loaded from the workspace. In this way, scripts can assign initialisation values to all lookup tables, enabling core performance characteristics of the model to be rapidly changed if required. This is the reason characteristic map scaling is performed online while the model is running: each of the scaling parameters is a workspace variable that can be changed at startup via script (and in this particular case also while the simulation runs, via the model's "dashboard"). Each script is usually tied to a single logical aspect of the simulation. Some examples of logically independent scripts are the initialisation of environment conditions, the controller's gain schedule, loading the engine's characteristic parameters (to define properties specific to this engine such as bypass ratio, nozzle diameter etc.) and the loading and saving of the simulation's states. Assigning scripts to specific tasks also helps to keep track of the resources required for the simulation. Note that all parameters can also be made time dependent if their values are provided via "from workspace" blocks.

5.2. Initial development

The simulation is designed along the lines of the engine components - i.e. a fan section, a high pressure compressor section, combustor, turbines and so forth. Although the mathematical basis for each section may be in place, good values are required at the boundaries of each module, otherwise the simulation will rapidly degenerate into unphysical conditions. This is further exacerbated by the feed-

forward nature of the simulation - an iterative solver would be more robust to discrepancies at the interfaces of the modules.

Gas stream temperature, pressure and massflow data at several engine operating points was provided by Alstom. From these values quite a few others can be deduced e.g. component efficiency, power consumption (under assumptions for fuel flow and gas stream properties) etc. - however a number of important variables remain unknown, notably the bleed extraction values and schedule, cooling flow % and pressure, nozzle diameter and efficiency, rotor inertia, component heat-soak values etc. Most of these have a considerable effect on both the steady state and dynamic performance of the engine, therefore an iterative tuning process is required to adjust the values of all variables until the engine performance is acceptable. The initial step towards this is to first tune each module in isolation. This can be achieved by isolating each element via use of adjustable gains at all connecting points. The advantage of this is that each module can be run independently if so desired but can also be gradually integrated into the system to evaluate the effect of interaction between the elements (i.e. a tendency towards instability, rapid acceleration/deceleration, unphysical conditions at the boundaries due to incompatibilities etc.), without inducing catastrophic failure and requiring a full simulation restart. Fortunately, the self-stabilizing nature of the gas turbine works to our advantage: if the values between the components are approximately correct, the engine will settle into a steady state after some initial instability. However this does depend on what "approximately correct" may be... and considering the small size of the storage volumes between the engine components and the enormous massflow, there is in reality little margin for error.

The physics of the engine can be used to our advantage for the engine tuning: during engine acceleration the engine components absorb heat from the gas stream

and during deceleration they release it. However at steady state the components are "heatsoaked", i.e. in thermal equilibrium with the gas stream and no heat is transferred (note that in practice a small quantity of the gas stream's heat would still be transferred to the components due to heat loss through the engine's casing). Because the objective of initial tuning is to run the engine to steady-state (at which heat transfer is nil), the quantity of heatsoak, which effectively acts as a damper, can be momentarily increased to tame any oscillatory behaviour or tendency to abrupt transients that the engine may undergo due to maladjusted tuning during the initial startup (as an aside, note also that heatsoak itself plays a significant role in controller tuning - allowing for much higher gains than would otherwise be possible).

5.3. Simulation initialisation

Simulation initialisation is an aspect that is easily overlooked, however the choice of an inappropriate method for saving and restoring the model's states can lead to massively time-consuming procedures at a later date (as we discovered to our expense), not least because of Simulink's compilation procedure. When first developed the model adopted a startup routine in use at the time at Alstom, but the lengthy procedures required to restore state ordering when adapting and changing the model - clearly a common activity during development, eventually prompted a complete overhaul of this aspect of the simulation. The consequences for model redevelopment (and linearisation) are far-reaching: the initial implementation is difficult, but the longer-term time savings are dramatic. This new method is supported by Simulink, but appears to be largely undocumented.

5.3.1. Simulink and the state array

The way in which Simulink saves and restores the simulation's states between sessions can be a major inconvenience when developing and maintaining a large simulation. The Simulink default is to save the simulation's states to the workspace as an array. This can be used to initialise a model to a previously saved state (see Simulink's "Configuration parameters/Data Import-Export" section). This ordered array contains the model's states in the specific order needed to start the simulation: by default Simulink does not save information that links the states to the simulation elements for which they are required, relying instead upon the ordering of the saved array being identical to that needed by the Simulink compiler upon simulation startup.

The problem with Simulink's default method is that the user has no control over the order in which the states are required at initialisation, and that this order is strictly enforced by the Simulink compiler (from this point of view, Simulink effectively works as a black box). The model states are saved in the order that they are required by the simulation, however should the simulation change and elements be added or removed, this order may no longer be valid because Simulink frequently rearranges the order of the states when the simulation is recompiled. This can be the source of some confusion: for example, simply adding a block that integrates an output value of the model (i.e. one upon which the simulation's progress is not dependent) - for example integrating fuel flow to calculate total fuel consumption - can cause a reordering of states. The consequence of this is that if the user saves a simulation and then subsequently adds a block to the model, the previously saved simulation is unlikely to be able to resume. The user is then

forced to reassign the states manually by assigning the correct state value to each of the blocks required for startup. If the changes are minor and the model is by some good luck still capable of starting up (although its results may be garbage), the user can evaluate the ordering of the simulation states via the command [sizes, x0, xstord] = vdp([ ], [ ], [ ], 0), and then manually adjust the initial vector. However, should this fail (as it frequently does), there is no choice but to manually replace the initialisation values in each of the Simulink blocks. When there are a large number of states (the current simulation had up to 68 at one point!), and the state blocks are situated in multiple sublayers of the simulation, this can be an extremely tedious and error prone process. Clearly this is a matter of some consequence when developing a model that requires frequent changes and hundreds of tests, as is the case here.

5.3.2. Assigning initial states by block address

The simulation's dependency upon Simulink's internal order of states is clearly a source of problems. Fortunately there is a way around this, although with some provisos of its own: a little known method of simulation initialisation is to set the model's states at startup via each block's "address", i.e. the path within the simulation to each block that has an internal state. In this way, the states are no longer linked to an order kept internally by the Simulink compiler but are conveniently linked to the appropriate block via an address.

The onus is now upon the user to keep track of the simulation's states: because an absolute block address is used (unfortunately one cannot provide an address relative to a top level main simulation block), Simulink is unable to find the state blocks should the model's name change. This can be resolved by parsing the model's filename at startup and inserting the current model's name into the state

structure for each block by using the "bdroot" command. For an example of how this can be done see section B in the Appendix. This script manages the blocks' addresses and also erases automatically any states linked to Simulink blocks that are no longer present. A drawback to this method is that when a block that requires an initial condition is added to the model, its address and values needs to be manually added to the state structure prior to simulation startup. A script to simplify this procedure can be found in section B in the Appendix: first highlight the required block, copy its address (using the "gcb" command) and then run the script. If the simulation is already able to start, the easiest way to begin using a state block address approach is to simply have the structure created by running the simulation and saving the final states to a structure. This is done through the "Workspace I/O" tab in Simulink's "Configuration parameters". However it may not be possible to get the model to run without correct initial state values (or at least good guesses), particularly if it is a complicated model or if there have been changes: it may be necessary to create the state structure programmatically from the simulation. An example of how this can be achieved can be found in the Appendix, section B.

5.4. Turbomachinery characteristic maps

The core aero-thermodynamic behaviour of the turbomachinery components is defined by their "characteristic maps". Each characteristic map is effectively a 4-D plot of pressure ratio, referred speed, referred massflow and efficiency that encompasses the characteristics of the modelled turbomachinery component(see sections 2.2.1 and 3.4.3 for more details). Although simulation is used during the design of most turbomachinery components, in practice these components show discrepan-

cies from the theoretical, ideal behaviour and it is therefore experimental data that is used in final design - the reason for characteristic maps. Furthermore, surge is also verified experimentally as it is too complicated a phenomenon to be accurately simulated [42]. With the inclusion of surge data an additional dimension is added to the map - bringing the dimensions up to 5.

It is worthwhile taking a little time to discuss how these characteristic maps are implemented in the simulation because they are a key element in the aero-thermodynamic performance of the engine's turbomachinery and therefore its overall performance. Their implementation, tuning, and limitations are discussed here. The simulation uses "generic" turbine maps that have been adapted to be appropriate to the performance requirements of the modelled engine. While it is common practice to use a map from a component of similar design and apply "factors" and "deltas" to align its design point to that required, it should be noted that this is a technique to provide only an approximate map for early engine off design performance [18, 21, 33]. There are more sophisticated methods available, but most are beyond the scope of the current simulation and require data that was not available to the author. The suitability and detractions of characteristic map scaling are discussed in [18, 21, 33]. Other methods of map adaptation and relative considerations regarding component matching can also be found in [20]. A point to note is that, as mentioned in [21]: "The traditional methods of scaling maps can lead to significant error in component representations. This was recognised many years ago and led to the development of programs which generate from only a few design points typical maps for fans, boosters and high pressure compressors". Such a detailed analysis is however beyond the scope of our current objective which is to develop a detailed simulation that exhibits the complex dynamic performance of a gas turbine and models its elements accurately (so that failure can be examined)

- but without the additional complexity required for simulating the details of the complex aerodynamic phenomena that can occur within the engine such as for example, surge or combustion chamber conditions.
Such a simulation would also be well beyond the capacity of a single workstation, particularly should the simulation be desired to run at real-time or near real-time speed.
Therefore a clear note should be made of the fact that the characteristic maps used cannot simulate extreme off-design conditions. Although the simulation itself will not fail, this is because limits have been hard coded into the simulation and therefore the simulation will run stably at the boundary of the simulation maps. As noted in [33], under extreme engine partload conditions, as for example subidle or windmilling, the operating points in the compressor map are located in a region that is usually not covered by rig tests and the parameters generally used in compressor maps are no longer appropriate. In short, as with all simulations, the old adage: "garbage in - garbage out", applies here.

5.5. Characteristic map scaling

An example of turbine maps that are suitable for adaptation are those shown in figures 5.1-5.6. These are the characteristic maps used in the NLR GSP simulation [28] and they are also used here after being appropriately scaled. However any suitable characteristic map may be used (i.e. one that is not too dissimilar to the component being modelled). Note that should the true characteristic map of a modelled component be available then of course this scaling procedure would not be necessary.

The engine extremum operating points used for tuning and scaling are shown on figures 5.1-5.6: desired operating positions can be plotted on the maps and then matched to those of a given target engine via map scaling [28]. The map scaling procedure requires temperature, pressure and massflow data at the component inlet and exit over a range of engine speeds (the temperature difference and pressure ratio over the component can then be used to calculate efficiency). Corrected shaft speed is calculated and used to plot the engine's operating point on the simulation map (disregarding the dynamic element of pressure, which is relatively small until 0.4 Mach [42]).

A point worth mentioning is that, as can be seen in the turbine maps (figures 5.5-5.6), beyond a certain pressure ratio the massflow hardly increases: this phenomenon is called "choking". Notice also that the shaft speed shown is "corrected speed" that has in addition been normalised to the maximum design rotor speed of the shaft and therefore 1 corresponds to maximum (corrected) shaft speed for a particular design of engine.

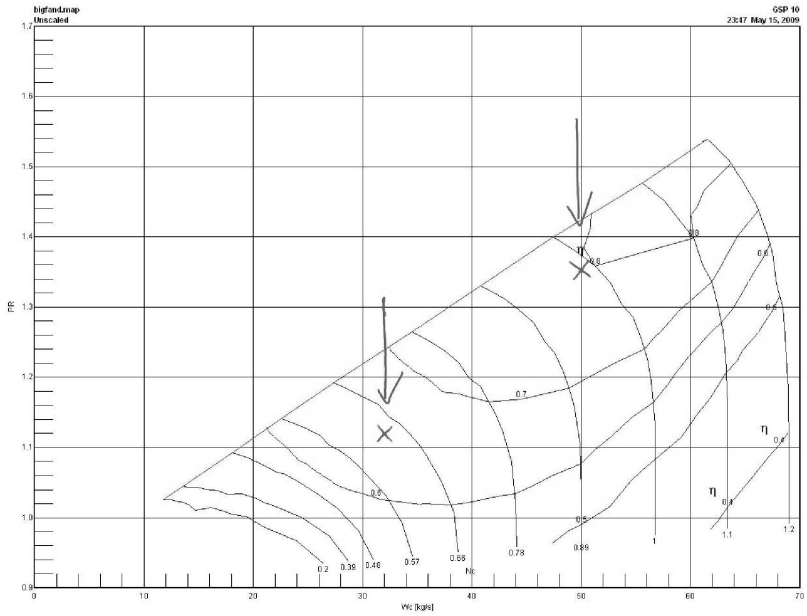


Figure 5.1.: **Fan duct characteristic map. The left-hand arrow indicates the engine's low-regime operating point; the right-hand arrow indicates the high-regime operating point.**

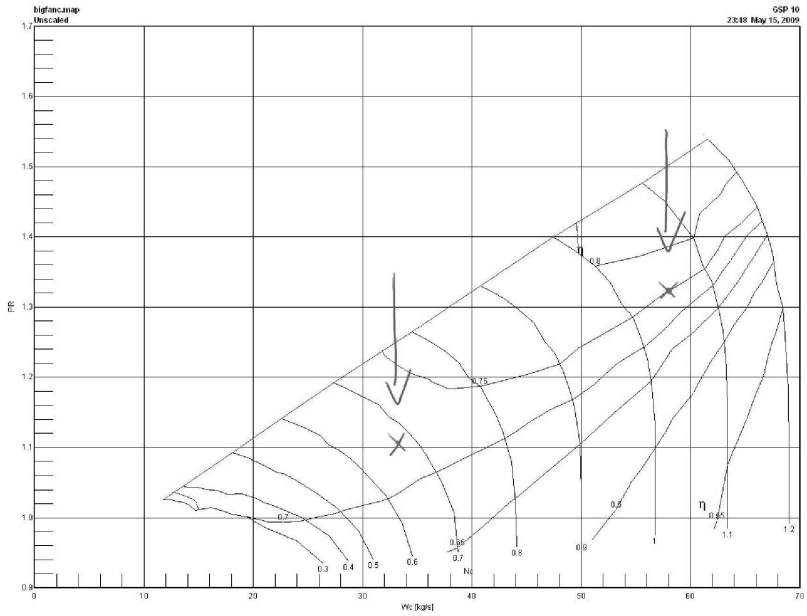


Figure 5.2.: **Fan core characteristic map. The left-hand arrow indicates the engine's low-regime operating point; the right-hand arrow indicates the high-regime operating point.**

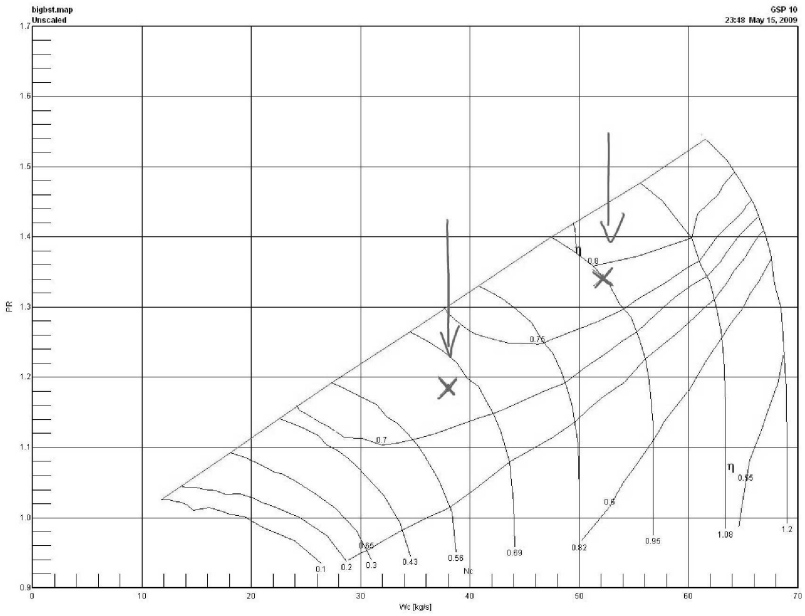


Figure 5.3.: **LPC characteristic map. The left-hand arrow indicates the engine's low-regime operating point; the right-hand arrow indicates the high-regime operating point.**

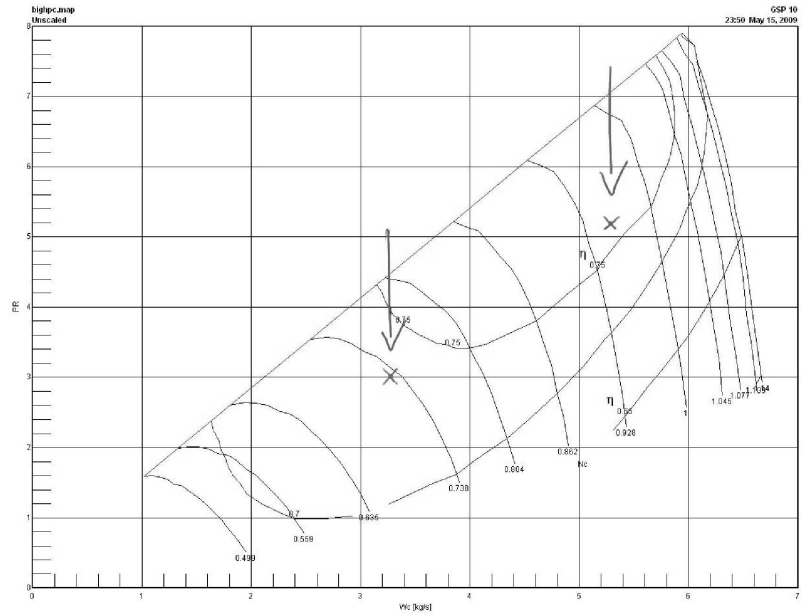


Figure 5.4.: **HPC characteristic map. The left-hand arrow indicates the engine's low-regime operating point; the right-hand arrow indicates the high-regime operating point.**

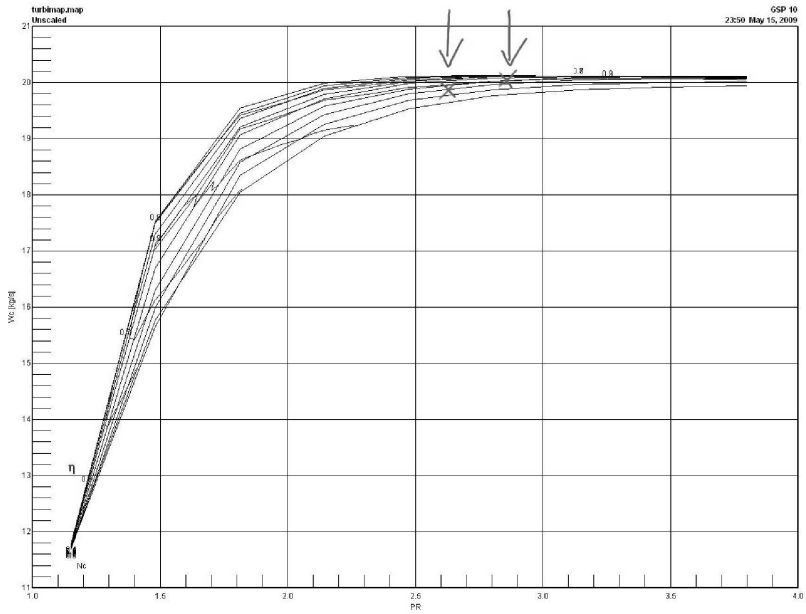


Figure 5.5.: **HPT characteristic map. The left-hand arrow indicates the engine's low-regime operating point; the right-hand arrow indicates the high-regime operating point.**

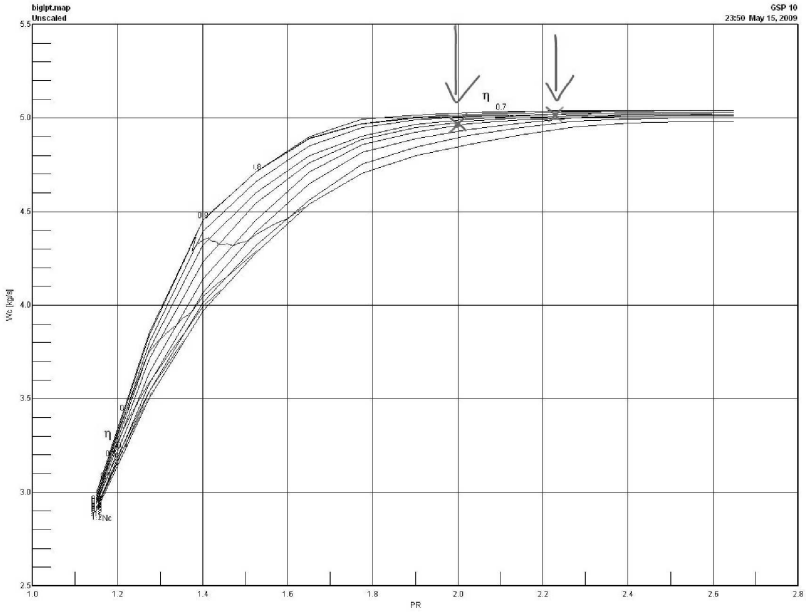


Figure 5.6.: **LPT characteristic map. The left-hand arrow indicates the engine's low-regime operating point; the right-hand arrow indicates the high-regime operating point.**

5.5.1. Map scaling formulae

For a compressor (or a fan) the conventional scaling formulae are as given below.

Massflow scaling:

$$\dot{w}_c = factor_1 \cdot w_{map} + \Delta_1 \tag{5.1}$$

Efficiency scaling:

$$\eta_i = factor_2 \cdot \eta_{map} + \Delta_2 \tag{5.2}$$

Pressure ratio scaling:

$$\pi = [(\pi_{map} - 1) \cdot factor_3 + \Delta_3] + 1 \tag{5.3}$$

Corrected speed scaling:

$$n/\sqrt{\theta} = n_{map} \cdot factor_4 + \Delta_4 \tag{5.4}$$

These same expressions apply to scaling a turbine characteristic map (as shown in Figures 5.5-5.6), with the exception of the formula for pressure ratio, which is not usually scaled for a turbine.

For each characteristic map a linear scaling value (e.g. $factor_1$, $factor_2$) and an offset (Δ) is required. $\dot{w}_c$ is the non-dimensional mass flow rate, η_i is isentropic efficiency, $n/\sqrt{\theta}$ is corrected shaft speed and π is the pressure ratio over the compressor. $factor_2$ is usually set to 1 and Δ_4 to 0, i.e. the efficiency data may have an offset but is not scaled linearly (and therefore care must be taken to not get unreasonable values of efficiency very close to - or even greater than 1!) while the corrected speed is scaled linearly, without an offset.

Target inlet and discharge values of pressure, temperature and massflow are available over a range of engine speeds for each component. However only the data

points at the lowest and highest value of engine speed are used to adapt the map, with the values in between being used to verify that the scaling works well at off-design conditions.

Map scaling is usually undertaken when a new engine is being designed and is achieved by choosing a scaling factor: a value that matches the reduction or increase in size of the target component. Our task here, to design an engine simulation that performs in agreement with real-world data, is not the standard scenario. Usually maps are scaled when doing initial designs for an engine that is not yet in existence, and the original formulae are then quite suitable. However, because our objective is to match the performance of a target engine, an alternative formulation of the above has been derived to suit our needs - i.e. matching a map position to target pressure, temperature and massflow conditions. The advantage of this scheme is that in these alternative expressions, the map positions and target values are used directly, making it easy and intuitive to tailor the map scaling to a target engine.

Each of the formulae above can be mapped to a single expression (note, however, that for the scaling of efficiency a simple offset is often used, so this may not be necessary):

$$x^* = (x - a) \cdot \frac{b^* - a^*}{b - a} + a^* \tag{5.5}$$

where a and b are the operating points chosen on the unscaled map for the low rpm and high rpm setting respectively, and a^* and b^* are the target values for the data set in consideration, i.e. one of pressure, temperature, massflow or efficiency. x and x^* are the independent and dependent variables respectively, i.e. x is data to look up in the scaled map, for example looking up the corrected massflow or efficiency for a given corrected shaft speed and pressure ratio), and x^* is the scaled data output. As an example, the conventional formula for scaling massflow can be

expressed as equation 5.5 under the conditions that:

$$factor_1 = \frac{b^* - a^*}{b - a}$$

and

$$\Delta_1 = -a \cdot (factor_1) + a^*$$

This follows from:

$$x^* = (x - a) \cdot \frac{b^* - a^*}{b - a} + a^*$$

$$= x(\frac{b^* - a^*}{b - a}) - a(\frac{b^* - a^*}{b - a}) + a^*$$

whereupon it therefore follows that:

$$factor_1 = \frac{b^* - a^*}{b - a}$$

and

$$\Delta_1 = -a(\frac{b^* - a^*}{b - a}) + a^* = -a \cdot (factor_1) + a^*$$

Similarly the scaling formula for pressure ratio can also be expressed in the form of equation 5.5. The conventional formula for map scaling for pressure ratio is:

$$\pi = [(\pi_{map} - 1) \cdot factor_3 + \Delta_3] + 1$$

which is equivalent to:

$$\pi = x \cdot (factor_3) + (-factor_3 + \Delta_3 + 1)$$

and again the expression is of the form:

$$x \cdot (factor) + \Delta$$

This can be expressed using starting and target map points a, b, and a^* and b^* as before.

This alternative expression of the traditional formulae is more suitable for scaling the characteristic maps to target design points because all that is required are suitable design points on the compressor and turbine maps, and target performance values (for each of pressure ratio, massflow and corrected speed). Another benefit is that this same expression can be used for all conversions, turning the conversion element itself into a simple simulation module.

As mentioned previously, the gas stream data at the lowest and highest speeds were used to scale the maps: the assumption is that if the maps are suitable for the target engine and the design points are well chosen, the scaled map will perform acceptably also between these two extrema and the output data will match well the target performance data. By using several design points, it is in theory possible to adapt the performance maps to fit exactly the requirements of any model for which sufficient data is provided, however this is in effect equivalent to recreating characteristic maps entirely from the data provided (which is indeed reasonable should there be sufficient data, but this is not usually the case as such information is usually closely guarded by the engine manufacturer).

Equation 5.5 is applied in real-time to the characteristic maps. This might be considered an excessive use of processing resources but it is of great use during engine tuning and allows the model to be easily adapted by simply changing the scaling parameters - which are set via scripts upon simulation initialisation and can be further adjusted in the simulation's "dashboard". This online adaptation

also allows verification of the engine operating point at any moment upon both the original and scaled maps.

Although the maps can be adjusted to the design point of a target engine, without a complete and thorough dataset there will be discrepancies between the operating point on the simulation and the target engine. There are several possible factors that can contribute to this: discrepancies between the efficiencies calculated, different fuel calorific content which would lead to a different fuel/air mixture, VBV and VSV schedule, metering of cooling flow, pressure losses etc. Therefore although the tuned maps may match precisely the operating points of the received data, due to even minute discrepancies the operating point of the simulation will drift away from these ideal values - and in general the discrepancies will not be so minute (consider the huge amount of power generated by a turbine - a discrepancy of a fraction of a percent in efficiency will have a noticeable effect upon the engine operating point). It must also be remembered that the characteristic maps used are not strictly equivalent to those of the reference engine: because generic characteristic maps are used, although the simulation is tuned to the extrema of the data provided and will match well at these points, the regions between these outlying points are essentially interpolated data that is unlikely to match the desired performance (for example efficiency might decrease/increase at a steeper/shallower rate). All of these factors will contribute to the overall engine's operating point drifting immediately upon the start of the simulation until a new steady state - different to that of the target engine, is reached. The aim is for this drift to be minimal and with a good understanding of the factors involved and the principles of turbomachinery it is possible, to a degree, to compensate for this and achieve a simulation that has similar characteristics to those of the reference engine. Without sufficient data and the original characteristic maps, identical performance will

not be achieved. The process of adjusting design values to achieve good matching could be automated but this would be a major design effort in itself and is not particularly useful for control design, where the aim is to achieve a good representation of a typical turbofan engine - any standard turbofan -, and to design a simulation architecture that is easily adjustable and modular. Our aim is not to identically model a particular design of engine, although we have sought to match a reference design for the purpose of validation.

5.5.2. Simulink implementation

Figure 5.7 outlines how the real-time map scaling is implemented in Simulink. The simulation does not use an iterative solver and therefore does not require use of "beta-lines" placed on the characteristic maps for compressor, turbine and derived components (e.g. the fan). The beta parameter is used to avoid numerical convergence problems during iterations towards the operating point solutions. With the typical relations between corrected mass flow and pressure ratio for constant rotational speed for these turbo-machinery components, either one of the variables can become independent of the other. For example, the constant speed curve in a compressor map can be nearly horizontal or nearly vertical. This causes numerical problems, since for one rotor speed and pressure ratio, multiple values for mass flow are possible. To avoid these numerical problems the beta parameter is added: this is represented by arbitrary equidistant lines that range in value between 0 and 1 and are parallel to the surge line. The position along a beta line serves simply as an array address.

Note that at the component inlet, PR and normalised speed are scaled in the inverse direction - i.e. we are mapping data from the model onto desired positions

on the map, while for efficiency and massflow, the opposite occurs: we are mapping positions on the map to desired values of efficiency and massflow. Another advantage of the scaling expression as defined above is that one simply has to reverse a and a^* and b and b^* to get an expression suitable for scaling data from the simulation to the map - it can be used as defined for data provided from the map. In summary, data from the component's inlet - i.e. the shaft referred (and normalized) speed and pressure ratio are (inversely) scaled, and used to look up the operating point on the original map. This returns values of efficiency and massflow that are then scaled back up to to the values they would have on an adapted map. Note the use of referred speed and massflow for the map lookup - massflow then needs to be restored to its-non referred value for use in the simulation.

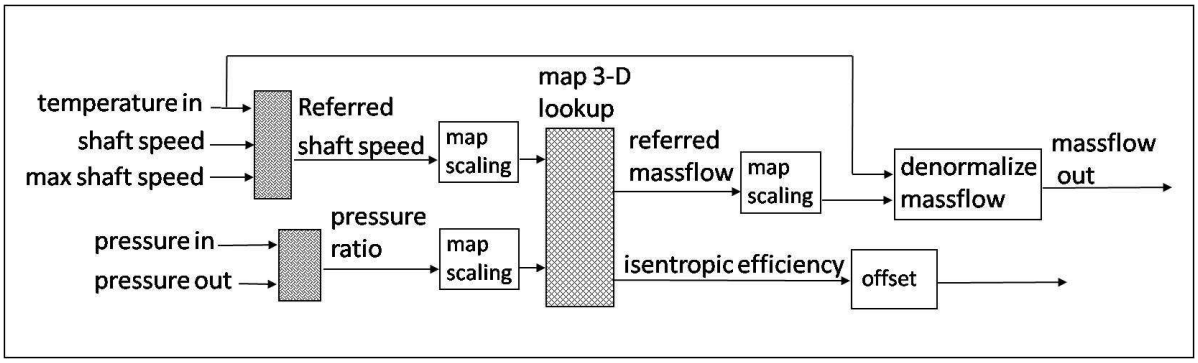


Figure 5.7.: **Outline of the map lookup dataflow and scaling**

5.6. Algebraic loops

Algebraic loops occur if a module requires data from elements downstream. This is a major issue for the compressors and turbines because outlet pressure data is required to calculate the pressure ratio across the component (see figure 5.7)

and thus, via the characteristic map, the component's massflow. It is also good practice to calculate thermodynamic parameters based on a component's average temperature but this requires knowledge of the component's outlet temperature prior to its calculation by the simulation module.

If these algebraic loops are not specifically eliminated the Simulink loop solver uses Newton's method to iteratively find a solution [26]. Although the method is robust, it is possible to create loops for which the loop solver will not converge without a good initial guess for the algebraic states. An initial guess for a line in an algebraic loop can be specified by placing an IC block (used to specify an initial condition for a signal) on that line. Another way to specify an initial guess is to use an Algebraic Constraint block.

An alternative method that is particularly advantageous for the current model is to specify a one-time-step delay by using a "memory" block, thereby avoiding the need for iterative calculations that would slow down the simulation considerably. Thanks to the use of these delays the solution can be reached without the iterative procedure that is generally required for the solution of the nonlinear equation system. This is made possible by means of the approximations that are applicable to real-time gas turbine dynamic models in consideration of the very short time step used in the simulation.

5.7. Running the simulation

The top level of the simulation, as shown in figure 5.8, largely mirrors the diagram of the closed-loop system (figure 4.1): the section at the left is the controller (figure 5.9), the right-hand section is the engine (figure 5.10), and the top section is the "dashboard". This last shows key engine and controller parameters, and contains

the controls for running test cases.

5.7.1. Standard test cases

Within the dashboard block, the first group of blocks on the left triggers steady-state test runs. These are labelled according to altitude, Mach Number and fan speed. To the right are three key transient test cases:

1. Case 1. Sea Level Static, from 40% to 98% of maximum thrust.
2. Case 2. Environment conditions: 15000 ft (4572 m), Mach Number 0.45, from 50% to 98% of maximum thrust.
3. Case 3. Environment conditions: 35000 ft (10067 m), Mach Number 0.8, from 60% to 96% of maximum thrust.

Double clicking on one these blocks will start the relevant simulation run. Each block calls a script: to see its name right click on a block, choose "block properties", move to the "callback" tab and browse to the "openfcn" section. The script is then listed on the right-hand side. Note that scripts are placed in subdirectories of the main simulation directory to permit related scripts to be grouped together. Matlab would not usually be able to find these because they are not directly on the working path, however in this instance the Simulink model automatically sets all subfolders of its main folder to be on the search path. This is done via the command "addpath(genpath(pwd))" and can be changed by locating the menubar of any of the model's windows and then browsing to "File\.. model properties \.. callbacks \.. PreLoadfunction.

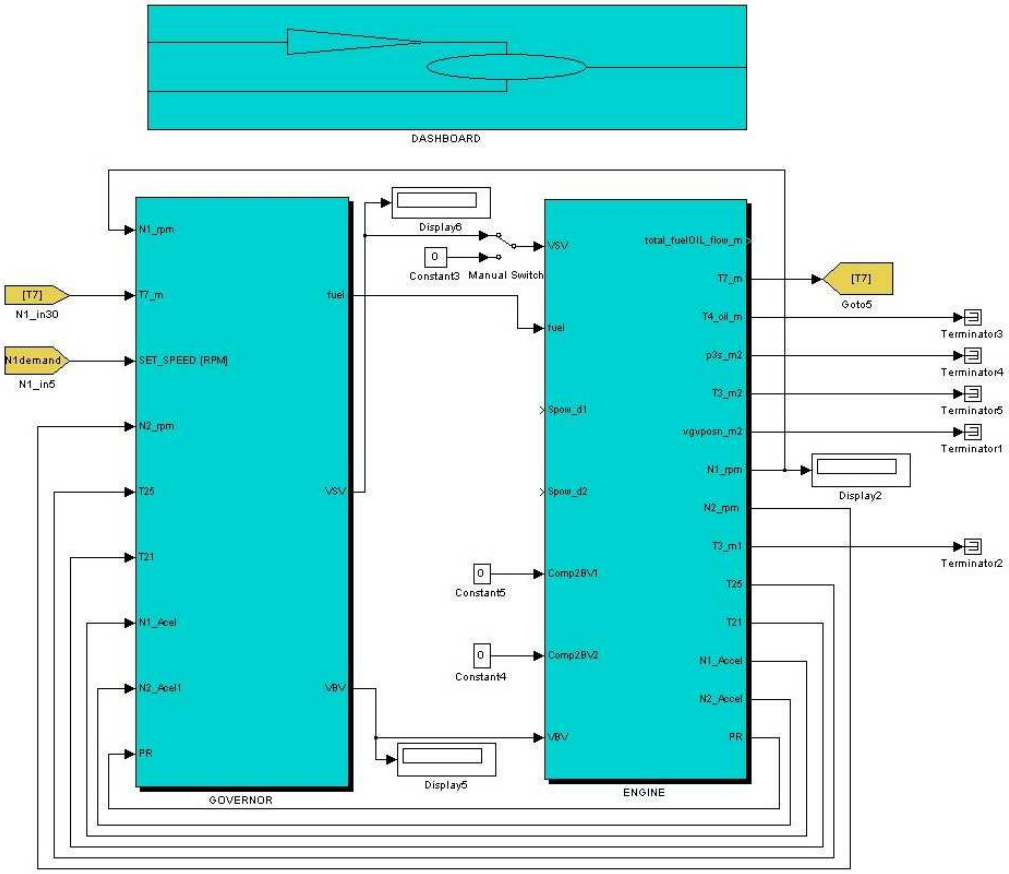
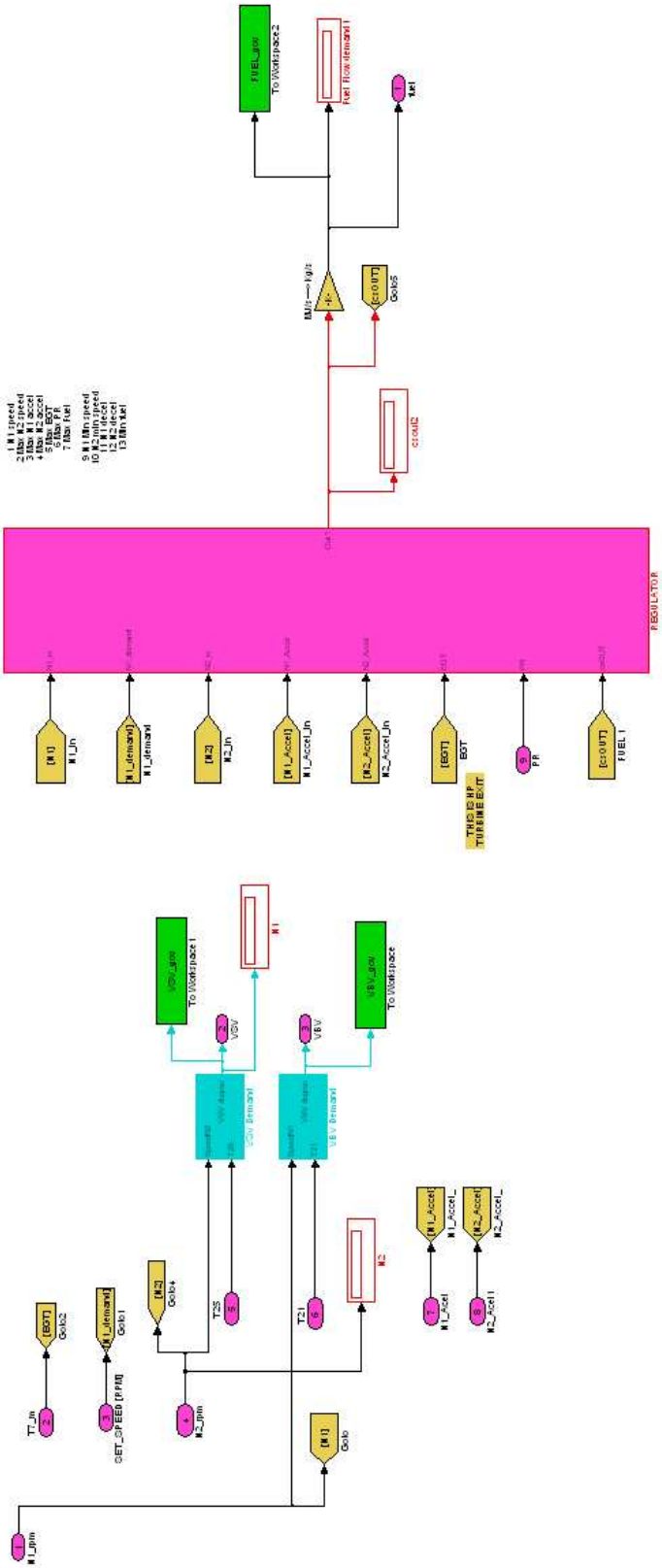


Figure 5.8.: **Top level of the Simulink model**

Figure 5.9.: **Simulink governor section**



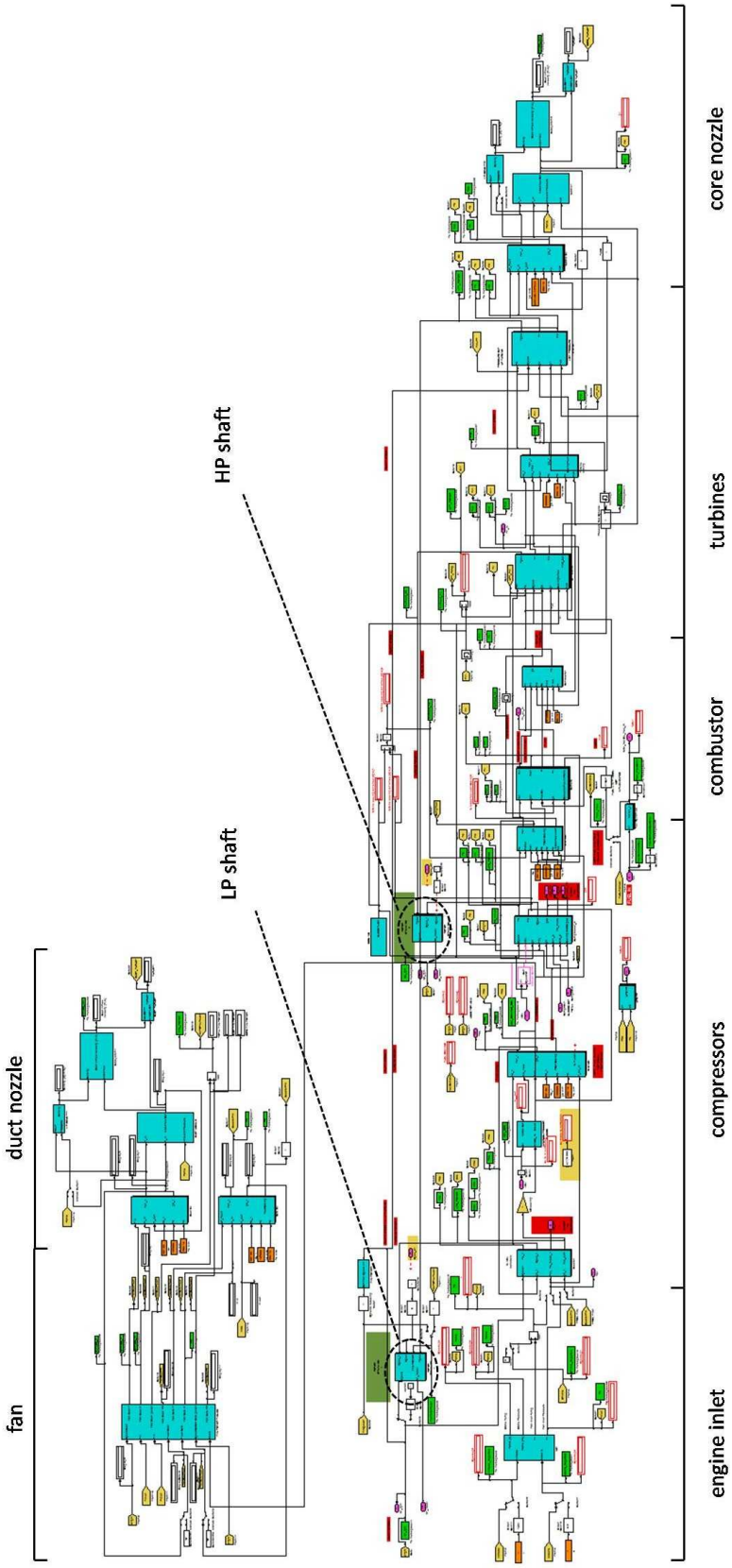


Figure 5.10.: **Simulink engine section**

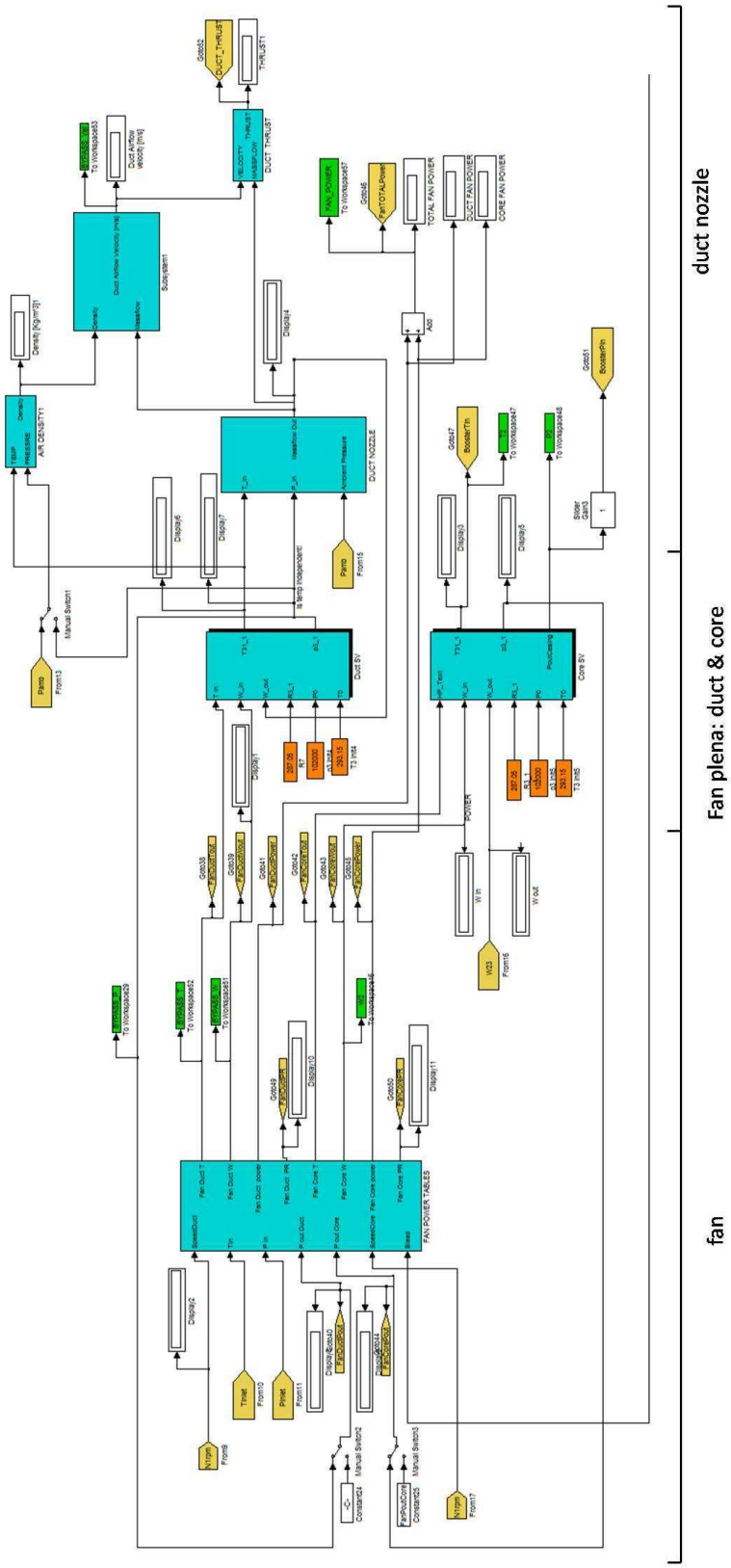


Figure 5.11.: **Simulink engine section - detail**

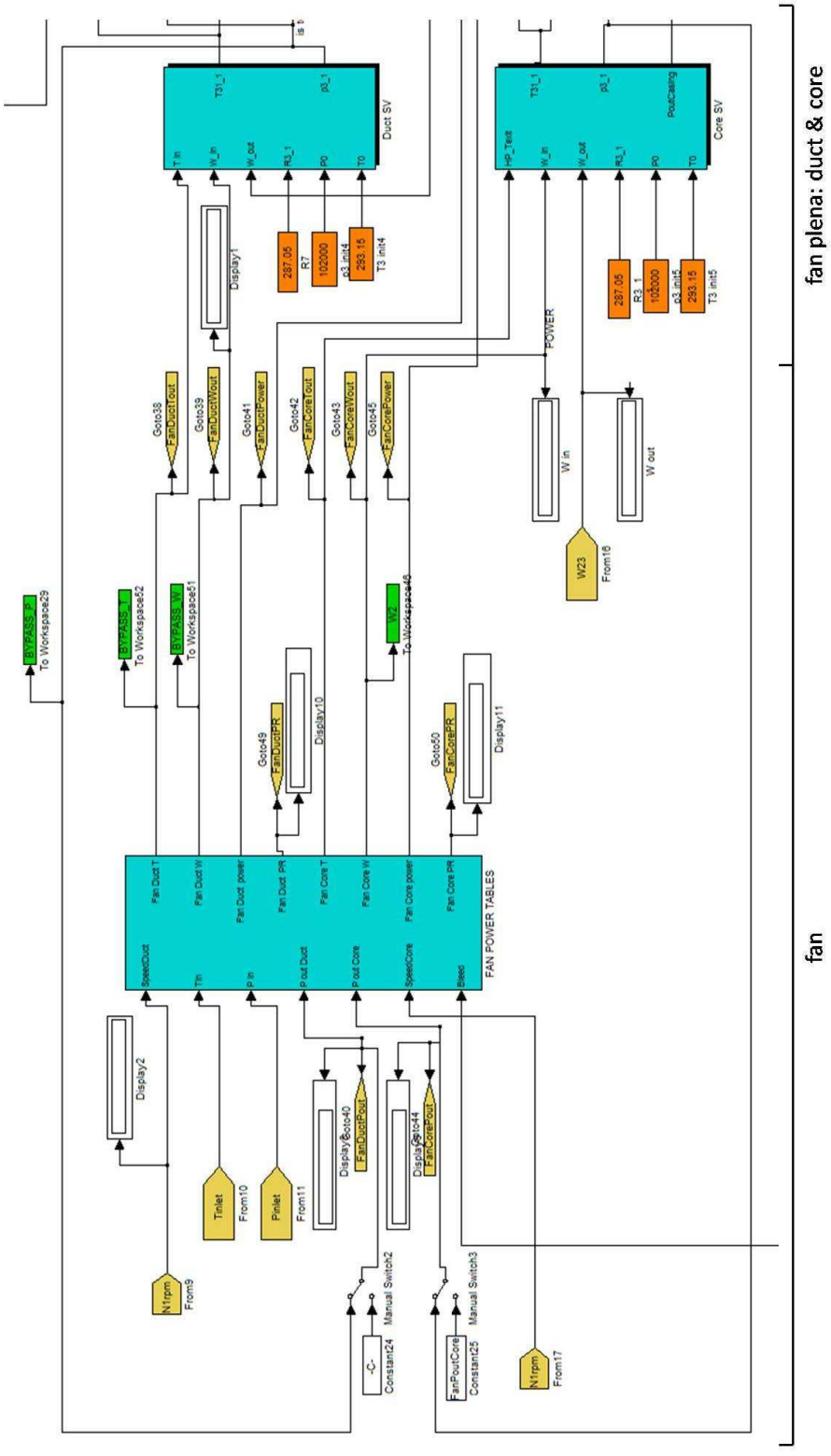


Figure 5.12.: **Simulink engine section - detail**

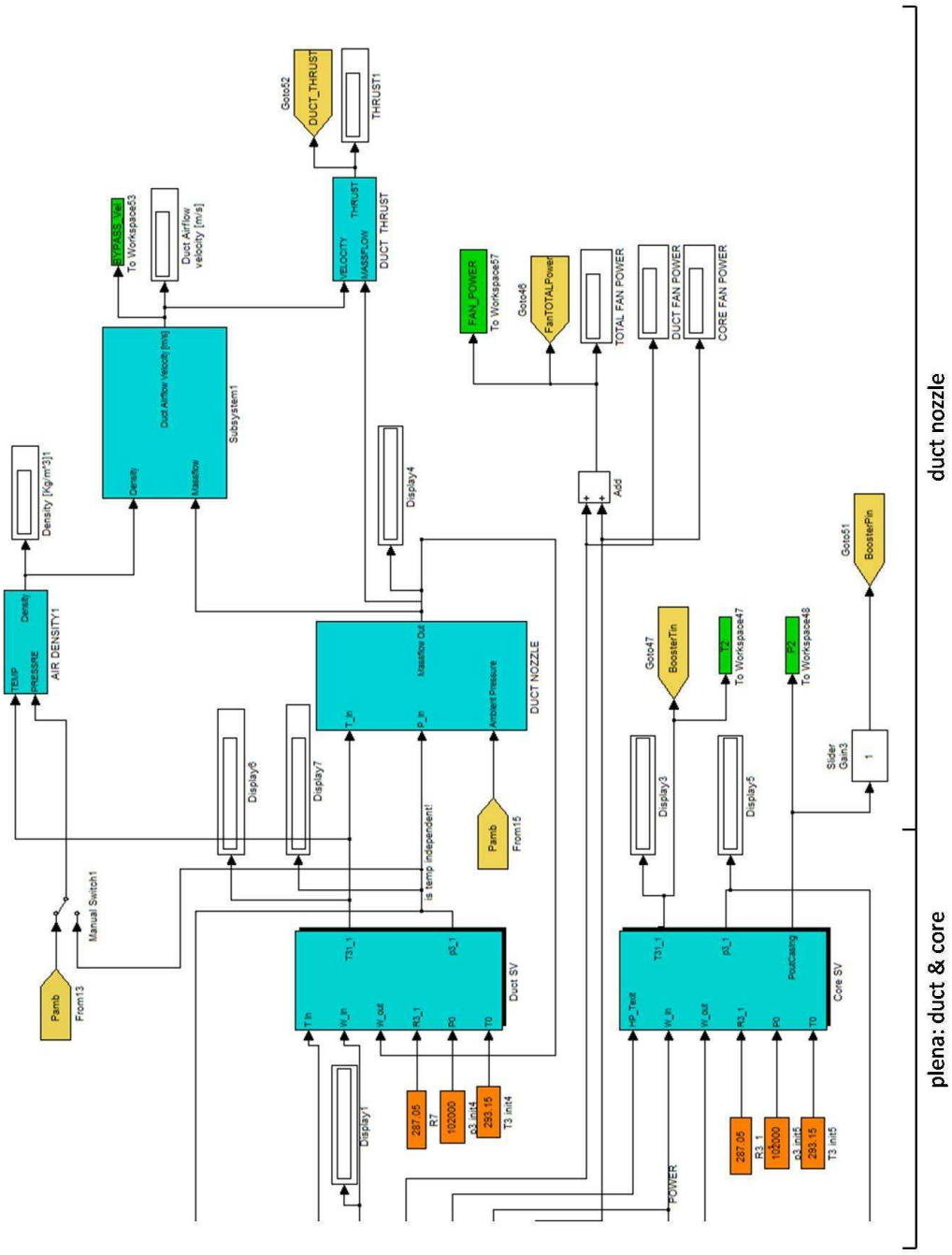


Figure 5.13.: **Simulink engine section - detail**

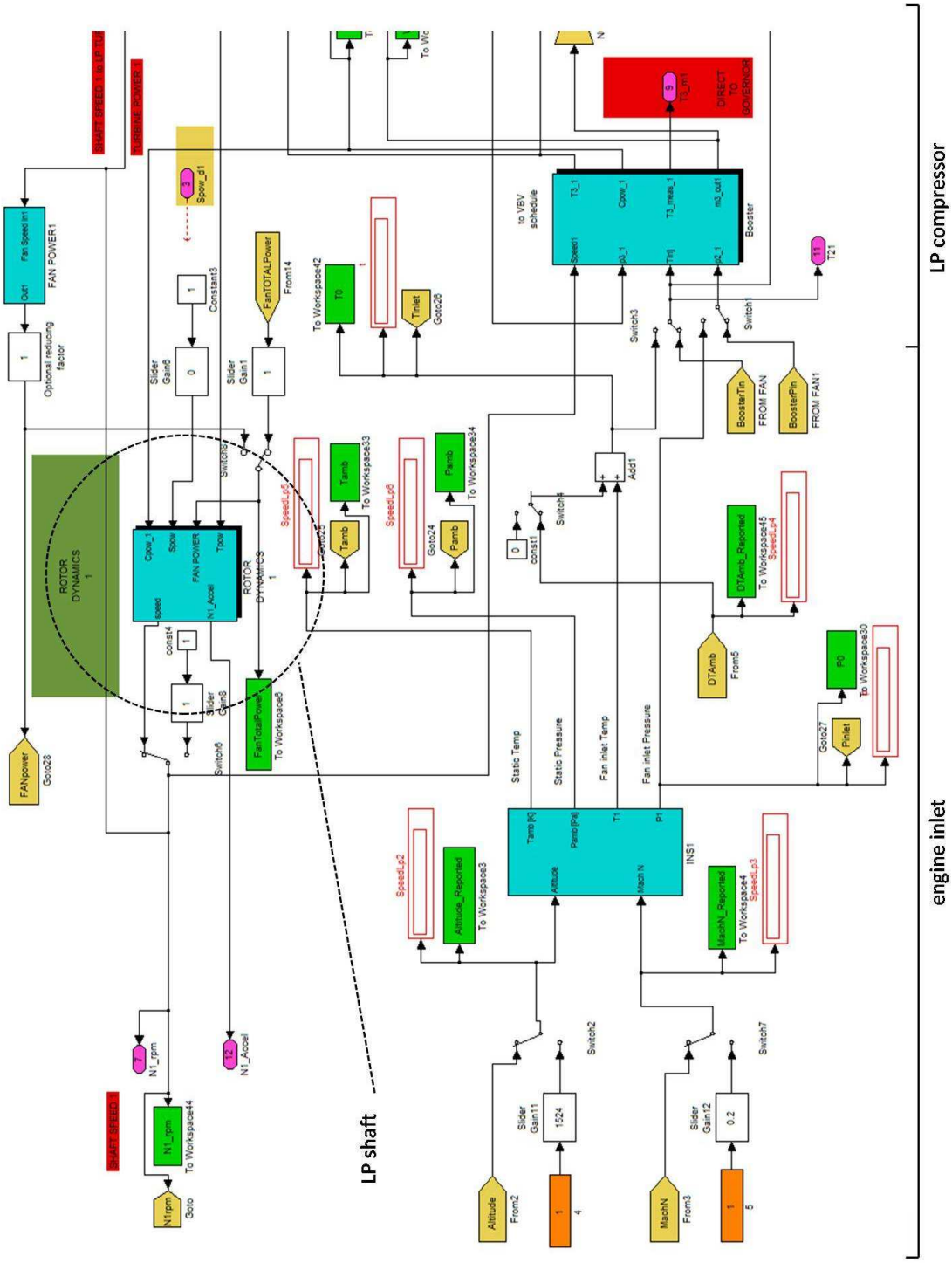


Figure 5.14.: **Simulink engine section - detail**

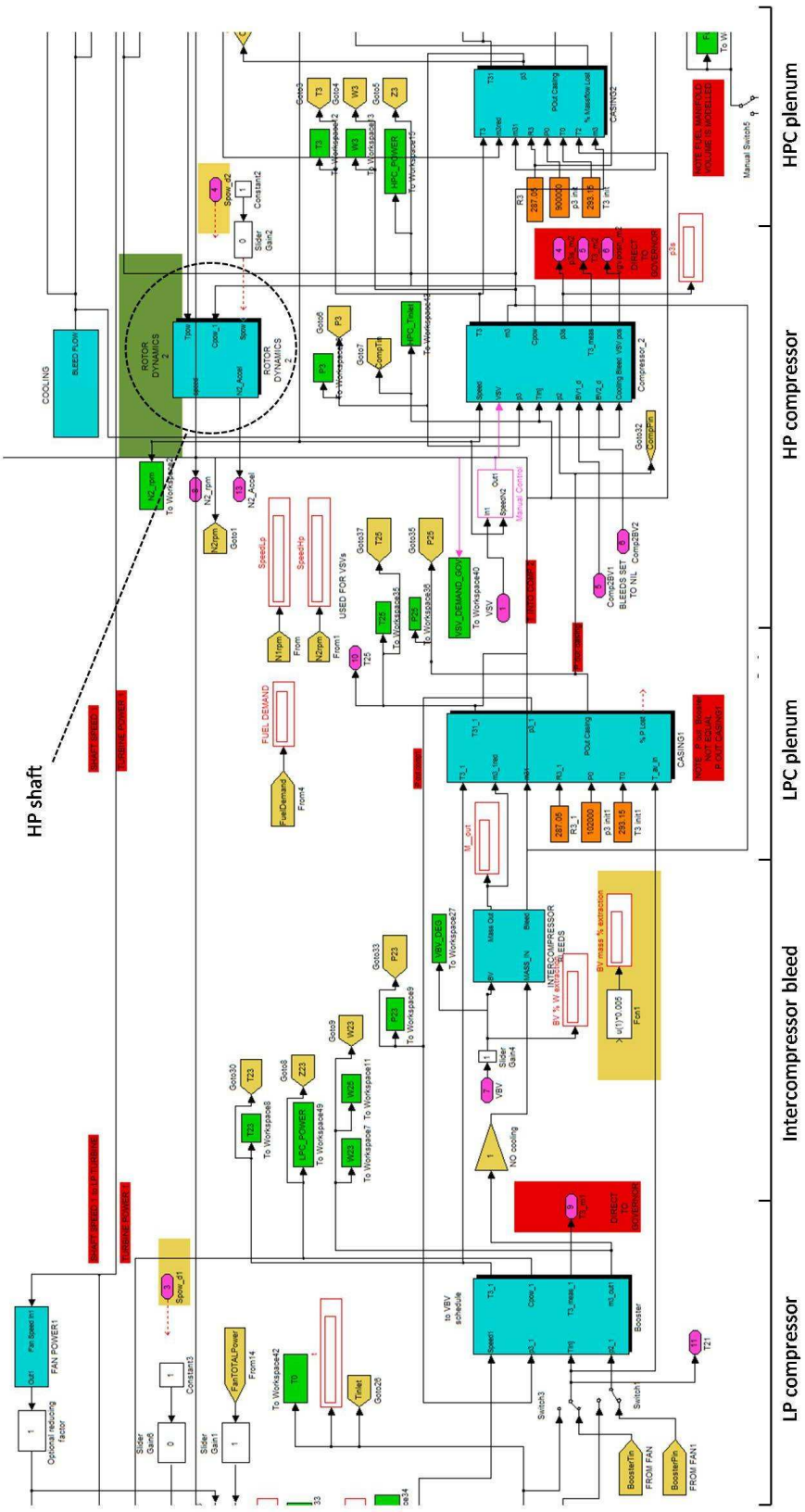


Figure 5.15.: **Simulink engine section - detail**

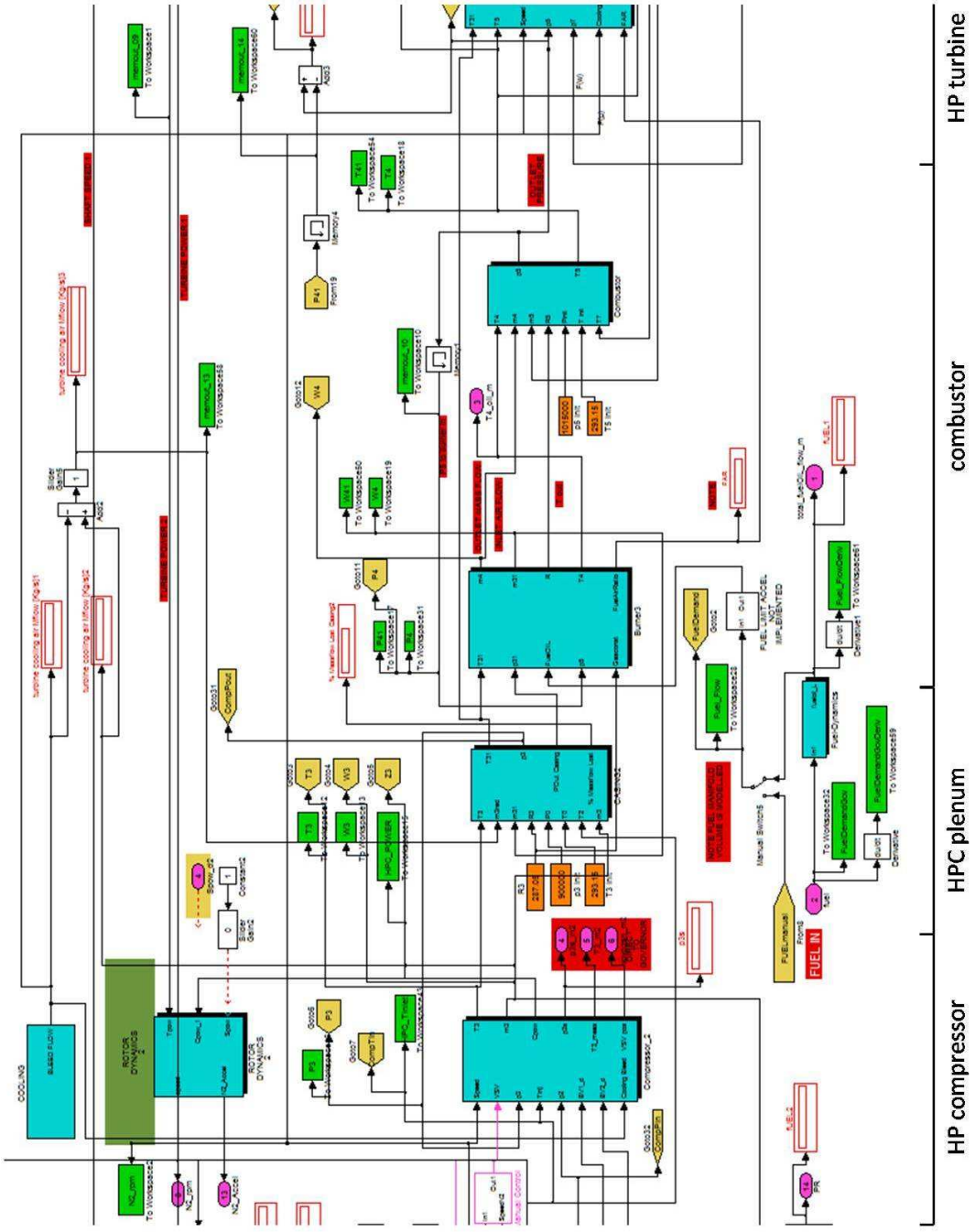


Figure 5.16.: **Simulink engine section - detail**

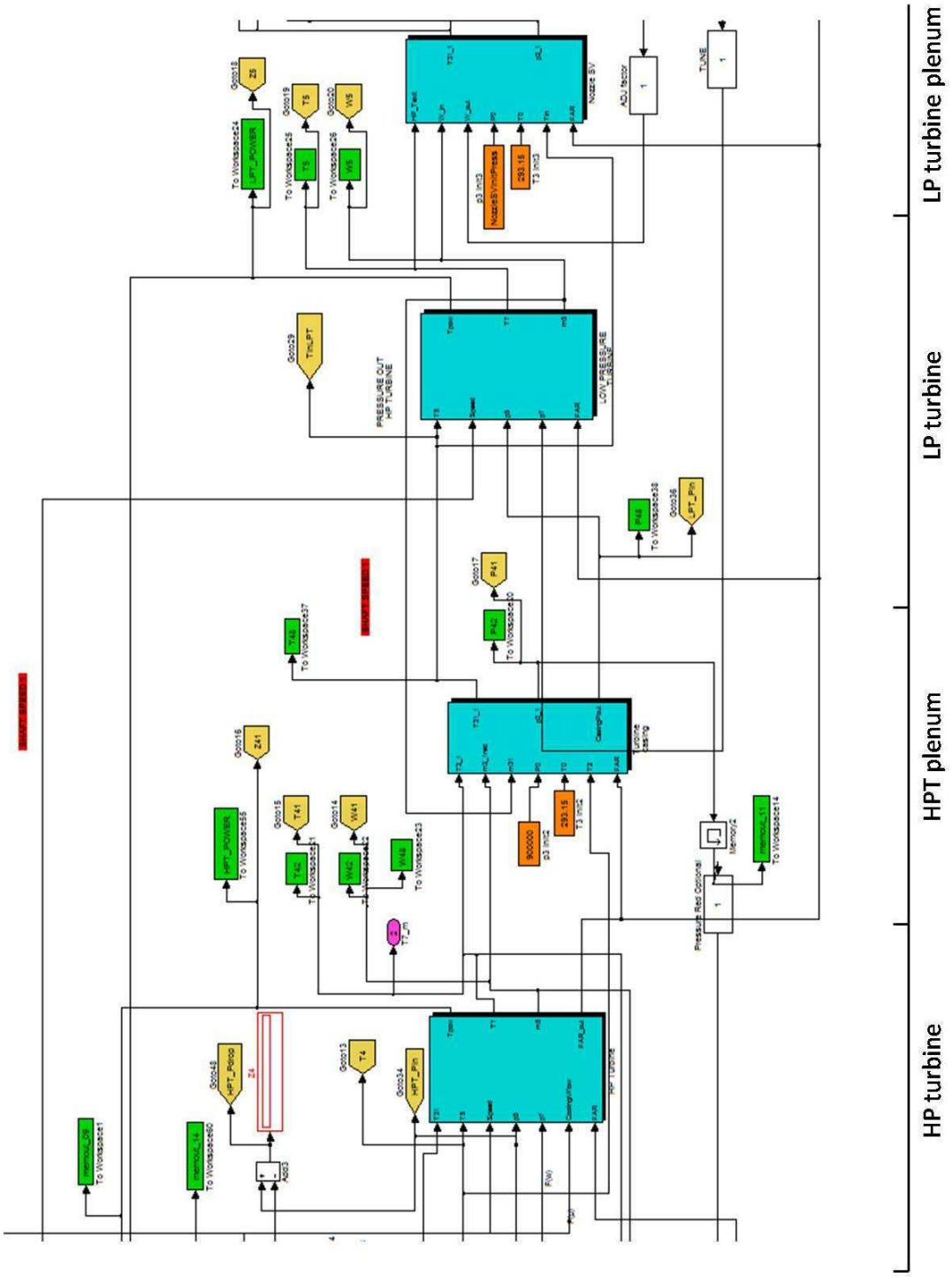


Figure 5.17.: **Simulink engine section - detail**

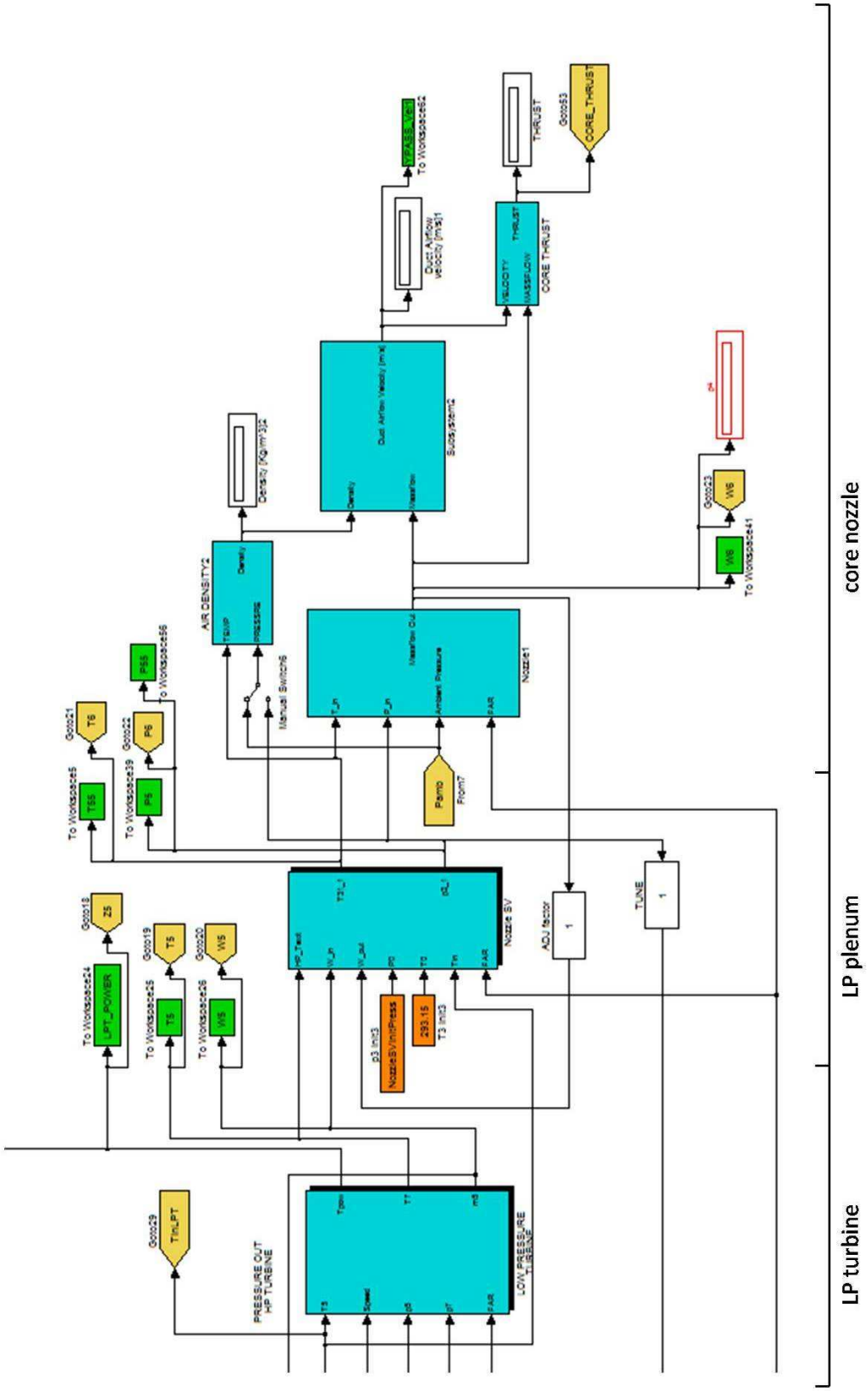


Figure 5.18.: **Simulink engine section - detail**

5.7.2. Custom test cases

The easiest way of running a custom test case is via the block "startup conditions" within the dashboard. Here the user can set the acceleration profile, ambient temperature, Mach Number and altitude for a custom test run.

5.7.3. Saving and restoring simulation tests

Test data can be saved by clicking on the "save simulation data" block. The "load simulation data" block allows the user to load a previous test case. A dialog box asks the user to provide the name of a data file (a .mat file), a start time, and finally the end time of the new simulation run: the simulation file stores the complete dataset of a previous run and not just the last operating point, therefore a saved run can be resumed from any point in time. If the requested start time is beyond the end time of the saved data, the simulation will resume from the end of the last run.

5.8. Summary

While the proceeding chapters looked at the modelling details of the simulation, the purpose of this chapter was to look at the practical aspects of implementing such a large-scale simulation: issues related to the practical implementation of the simulation also require some explanation.

Here were discussed some of the most important practical aspects of translating the theory into a simulation architecture, starting with a brief discussion of the Matlab-Simulink environment and the simulation initialisation, before proceeding to the implementation of one of the more elaborate simulation modules: the real-time scaling and implementation of the turbomachinery characteristic maps.

6. Full envelope closed loop model validation

This chapter presents the performance results of some key dynamic and steady-state tests that validate both the engine performance and that of the controller. These tests and key plots were peer-reviewed and published in the leading journal of the field: the ASME Journal of Engineering for Gas Turbines and Power [25]. The overall control system architecture and results were presented at the 2008 American Control Conference [24].

6.1. Introduction

True to the scientific process, the simulation's performance needs to be validated against some benchmark. Several methods to validate the dynamic performance of gas turbine simulations have been used in the past, e.g. [4] uses GasTurb [19] (commercial software dedicated to gas turbine modeling) and [5] matches calculated performance to previously published results [34].

The current model and controller have been validated against dynamic performance data for a comparable engine by analysing the resulting closed-loop performance properties for a range of different pilot thrust demands against the type of

responses required from a real turbofan engine.

Substantial work could be done to replicate exactly the performance of a specific engine, however the objective is to provide representative baseline turbofan performance for the purpose of testing overall controller architectures and not to match exactly a specific engine design.

At the request of Alstom Aerospace some of the data in the performance plots below has been normalized to remove the actual performance values used. However the dynamic behaviour is still clearly visible and shows the expected trends.

The engine was tuned by running multiple tests, starting from first-principle values of key parameters and gradually adjusting these until the engine performance closely matched that of our design target. As mentioned previously, one can conceive of a tuning framework that can adjust all parameters automatically - but this would be a very major undertaking in itself. The performance would never match exactly that of the target without identical characteristic maps, and therefore an averaging of the errors over all parameters, over all testing conditions, together with a ranking in importance of the errors would be required. Add to this that it is not simply a question of tuning parameters one at a time, but that the simultaneous adjustment of multiple parameters is often required for a beneficial effect to be seen... this puts the magnitude of the task into perspective. Parameters were instead adjusted manually using engineering insight into the roles they play in the simulation, a lot of tests, and a large dose of patience!

6.2. Simulation time step

The model runs deterministically if a constant time step is applied, i.e. the simulation results are repeatable. This means that with a powerful processor, the model

is able to run aside engine hardware because the time step is defined, and the model solution does not require an iterative procedure whose convergence time may vary. This type of simulation has previously been referred to as a "real time transient performance model" to emphasize this quality [42]. The model is also compatible with the Real-Time-Workshop, a toolbox available in the Matlab-Simulink environment that is able to automatically generate C language source code from the Simulink scheme. This feature can be useful for developing a code for hardware-in-the-loop applications.

While there are many "real time" transient performance models, the choice of an aerothermal model is preferable to other methods involving crude approximations of major turbomachinery elements via transfer functions. "Real time" is used here to mean those models that do not involve an iterative procedure [42] and are therefore, subject to sufficient processing power, most suitable for running alongside hardware. However should the convergence time for an iterative model be small and guaranteed, such a model would also be suitable. This is often however not the case [42]. When, in a mass of gas at rest, a small disturbance results in a slight local rise of pressure, it can be shown that a pressure wave is propagated throughout the gas with a velocity which depends upon the pressure and density of the gas. This velocity is the speed of sound in the gas and is known as the sonic velocity. Therefore during rapid transients, pressures cannot change instantaneously because of the finite volume between the components, and the assumption of flow compatibility at all times is not exactly true, although it is a good approximation. These gas dynamic effects occur only at the beginning of a transient and have extremely small time constants in the region of kHz (time for the effect to propagate through the storage volume) and therefore these dynamics are often neglected as in the case of the present model. The choice of an explicit

solver means that phenomena with a time constant smaller than the characteristic time cannot be considered (we shall consider this to be one tenth of the time step size).

For small plenum volumes there is a very steep rate of variation in pressure for any variation in mass influx or efflux. Should the time step of the system be large, this would lead to situations whereby the system would respond to massive changes in pressure rather than to the gradual change that would occur for a smaller sample time. An unphysical situation would arise when the changes in pressure that occur result in the model exceeding the pressure ratio boundaries of the compressor or turbine maps either downstream or upstream of the plenum. From this perspective, the smallest possible timestep is desirable, however one should consider that the compressor maps are in themselves approximations and an extremely small time step will not necessarily realistically improve the simulation accuracy, although it will indeed improve the solution accuracy. Furthermore, because the model incorporates numerous component maps, the tradeoff between solution accuracy and time step is not readily analytically quantifiable, and determination of an overall limiting time step is not easy. The smallest plenum will be the limiting factor to the permissible time step and therefore solution accuracy needs to be carefully tested should very small volumes be used, as would occur for multi-stage component modelling. A way of testing for this is to repeatedly halve the time-step until no difference in solution accuracy is noticed for an engine rapid transient such as would occur during a pilot slam request. A good, conservative, constant time-step for the current simulation is 10^{-4} seconds. This works comfortably even with basic Euler numerical integration (Matlab ode1).

6.3. Closed-loop validation: performance plots

In this section we show the results of three key performance tests. These test cases are:

1. Case 1. Sea Level Static, from 40% to 98% of maximum thrust.
2. Case 2. Environment conditions: 15000 ft (4572 m), Mach Number 0.45, from 50% to 98% of maximum thrust.
3. Case 3. Environment conditions: 35000 ft (10067 m), Mach Number 0.8, from 60% to 96% of maximum thrust.

Overall performance of the model and controller during a slam acceleration and deceleration can be seen in figures 6.1-6.3.

Figures 6.4-6.6 show the individual fuel demands of the active regulators during a pilot request for an abrupt increase in thrust in the time span of half a second. Figures 6.7-6.9 show the active regulator demands during a similar deceleration.

Figure 6.10 show the final controller demand after the controller's selection logic has been applied to the output of all regulators. This is essentially the fuel demand - the controller output is converted into fuel demand in a separate module to permit the engine to use fuels with different calorific content. The results have been staggered for clarity. As can be seen in test case 3, at 35000 ft the engine consumes a fraction of the fuel it consumes at sea level - which is one of the reasons why aircraft fly at high altitude!

By comparing the controller output of the three tests (figures 6.4-6.9) to figure 6.10, it is apparent that the controller has selected the smallest of the regulators' demands during acceleration, and that the transfer between these is smooth. Conversely, it has selected the largest of the regulators' demands during deceleration. The overall controller output is a smooth ramp and engine acceleration proceeds

accordingly until maximum thrust has been achieved. This is apparent in figures 6.1-6.3, which show the engine's response to the pilot's demand: although the request is abrupt, the engine acceleration is smooth. As can be seen in the plot inlays, there is a slight overshoot and undershoot however the overshoot does not exceed 0.6% and the undershoot never exceeds 1.2% - this is within the FAA requirement of a deviation in thrust of not more than 2% [17].

Plots 6.11-6.13 show the active regulators for these test conditions. The numbers along the ordinate of the plot indicate the active regulator: 1 is the fan speed (LP shaft) regulator, 8 is the fuel flow increase rate regulator, 4 is the HP shaft rate of acceleration regulator, 7 is the maximum fuel regulator, 15 is the fuel flow decrease rate regulator, 12 is the HP shaft rate of deceleration regulator and 13 is the minimum fuel regulator.

For the first test, figure 6.11, the engine is at steady state until time t=40 and the engine is controlled by the fan speed demand regulator. Upon a pilot request for an abrupt acceleration, the fuel demand of the fan speed regulator rapidly ramps up. At approximately t=40.01, the active regulator briefly switches over to regulator 8 - fuel flow increase rate. During the remainder of the acceleration the engine is controlled by the HP shaft maximum acceleration rate regulator (number 4), the fuel flow increase rate regulator (number 8) and the maximum fuel regulator (number 7, at time t=45-46). As the target speed is approached, control is switched back to the fan speed regulator. This then requires a shaft deceleration as a small overshoot occurs, thereby activating the high pressure shaft deceleration regulator (number 12, at time t=47-47.5) before control finally returns to the fan speed regulator.

The inverse process occurs after t=90.5, upon engine deceleration: the fuel flow decrease rate regulator is the first regulator to take over from fan speed but is

subsequently replaced by the HP shaft deceleration regulator. The overall controller output is a smooth ramp and engine acceleration proceeds accordingly until maximum thrust has been achieved. The profile is similar for the other two cases, with the exception that the high pressure shaft deceleration regulator no longer needs to intervene upon correction for fan speed overshoot.

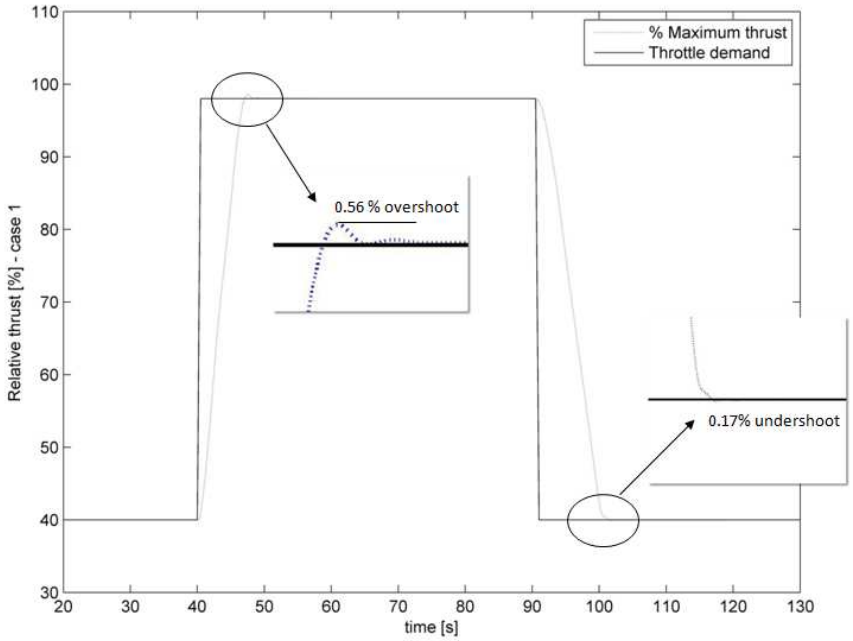


Figure 6.1.: **Relative thrust and demand - case 1**

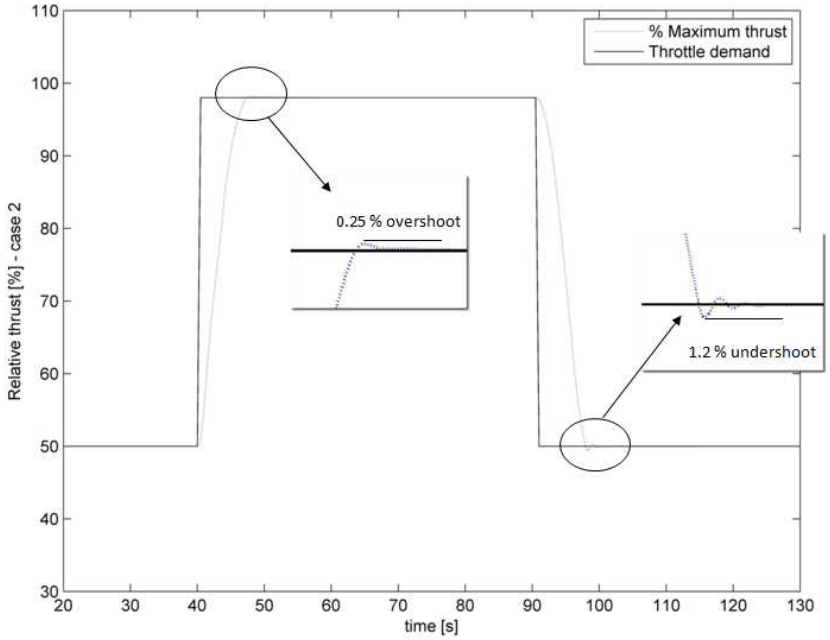


Figure 6.2.: **Relative thrust and demand - case 2**

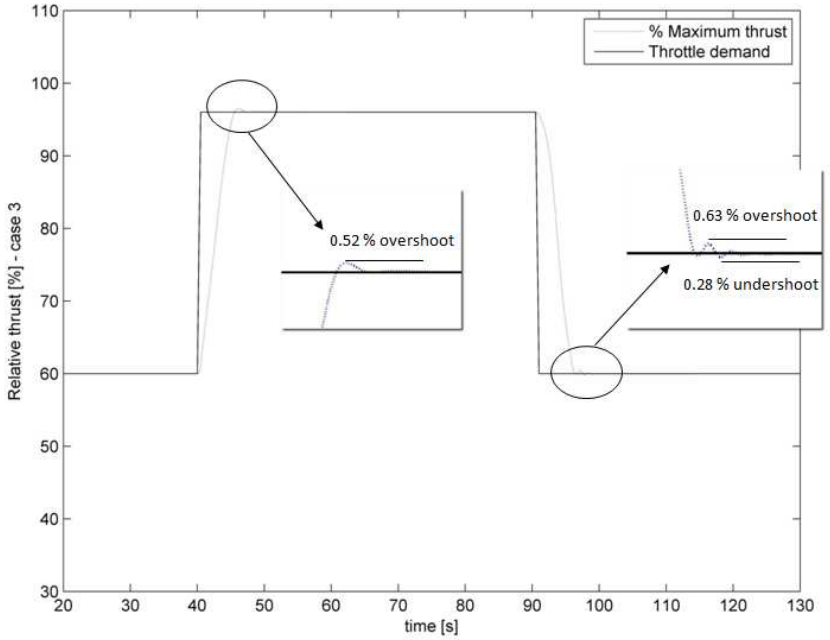


Figure 6.3.: **Relative thrust and demand - case 3**

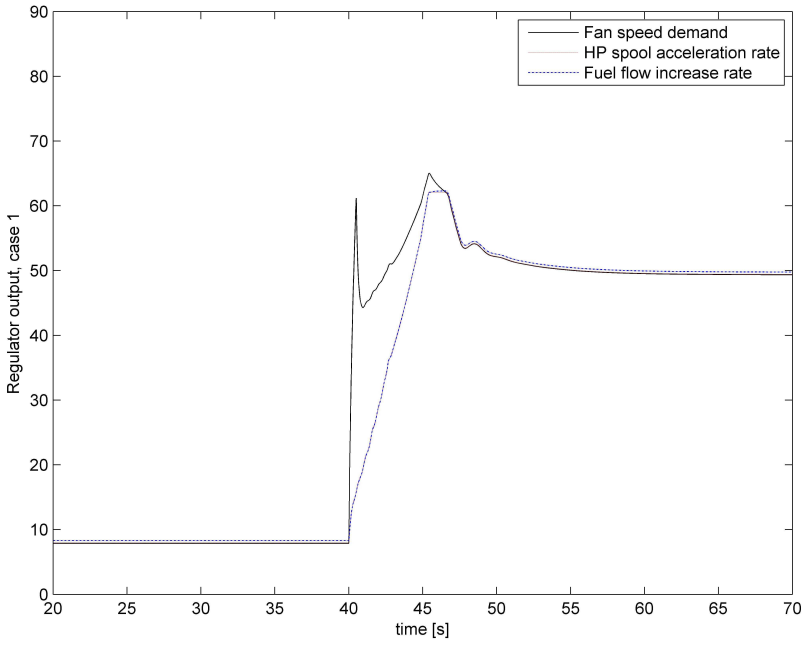


Figure 6.4.: **Regulators upon acceleration - case 1**

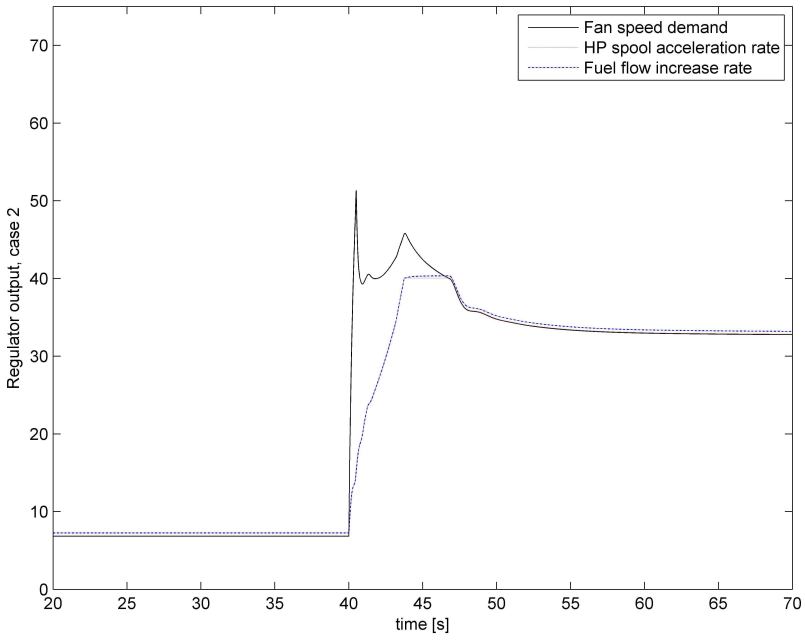


Figure 6.5.: **Regulators upon acceleration - case 2**

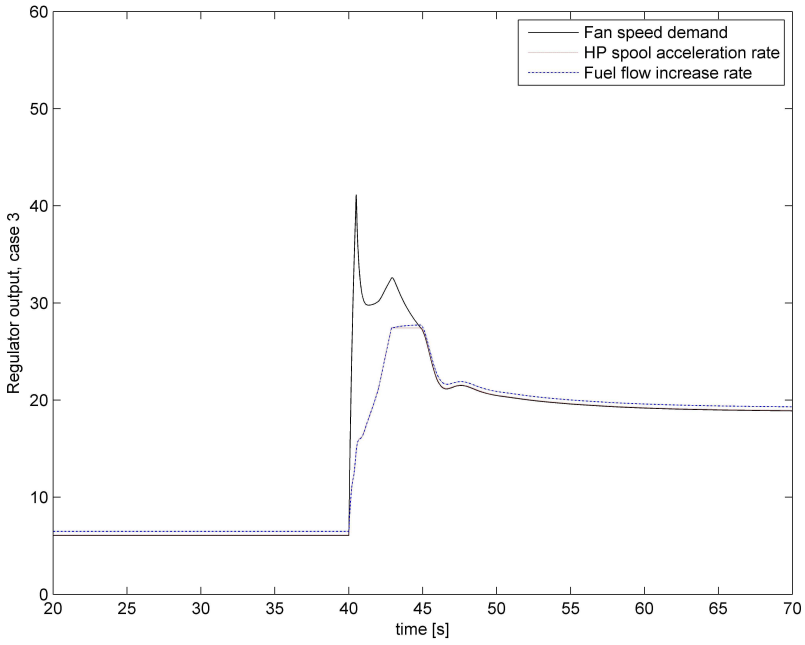


Figure 6.6.: **Regulators upon acceleration - case 3**

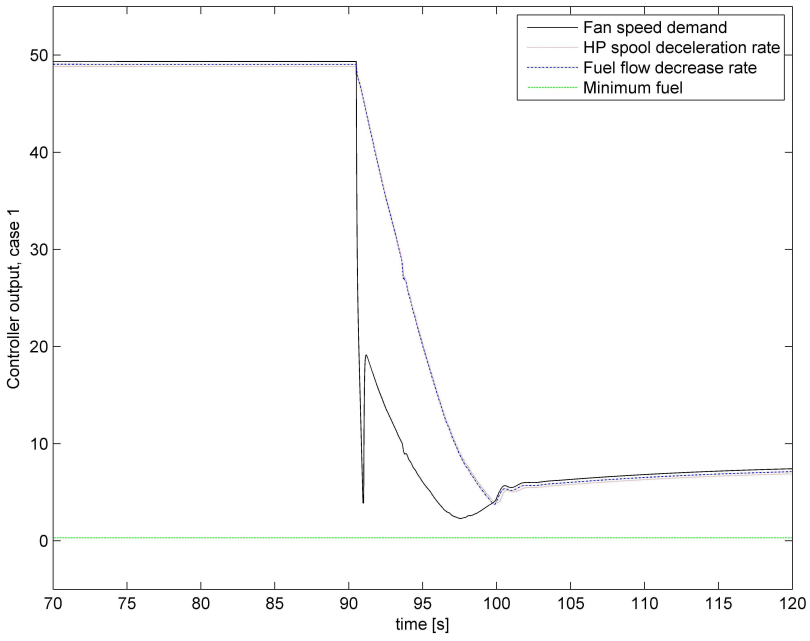


Figure 6.7.: **Regulators upon deceleration - case 1**

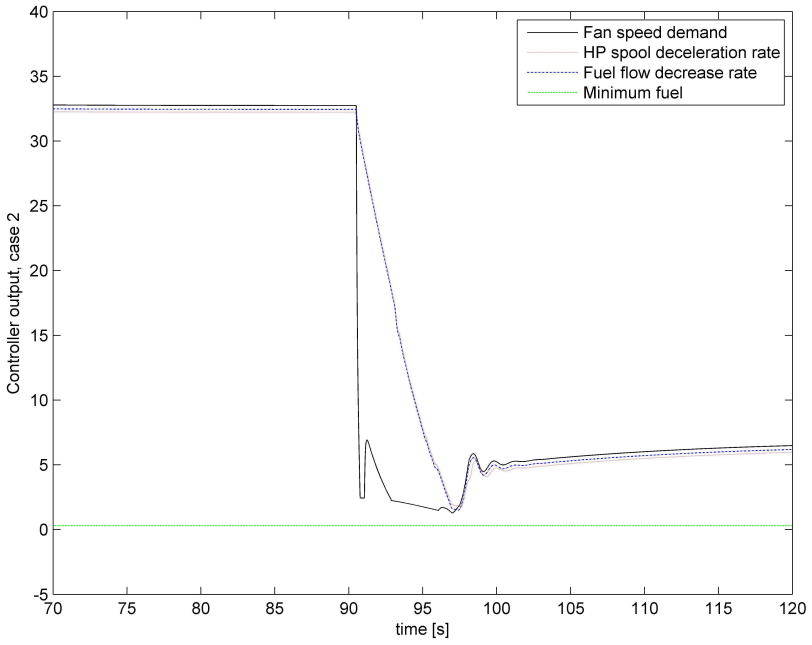


Figure 6.8.: **Regulators upon deceleration - case 2**

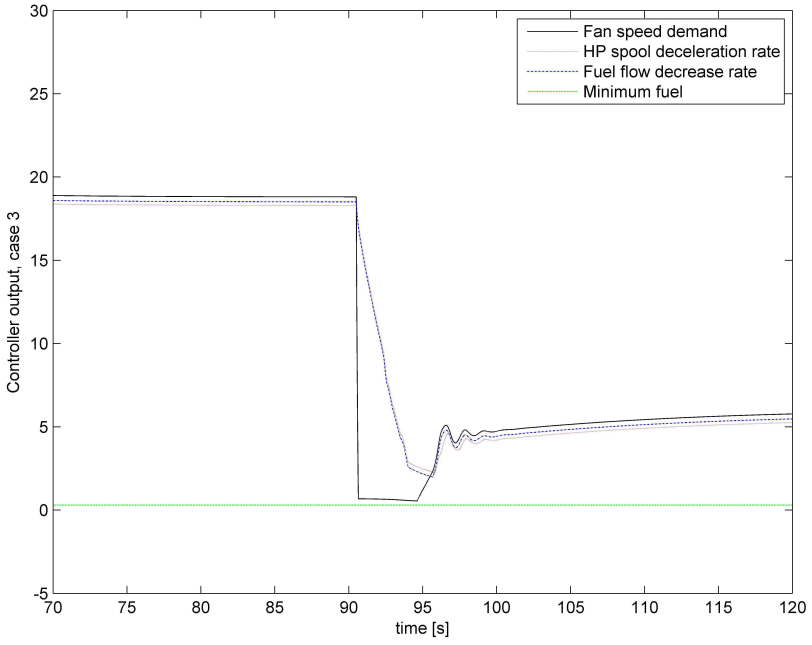


Figure 6.9.: **Regulators upon deceleration - case 3**

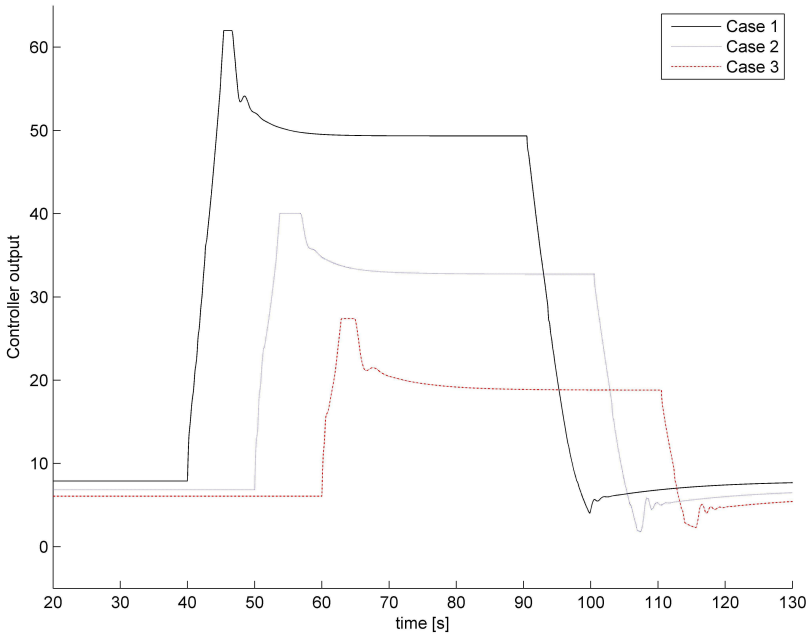


Figure 6.10.: **Engine fuel demand: case 1-3**

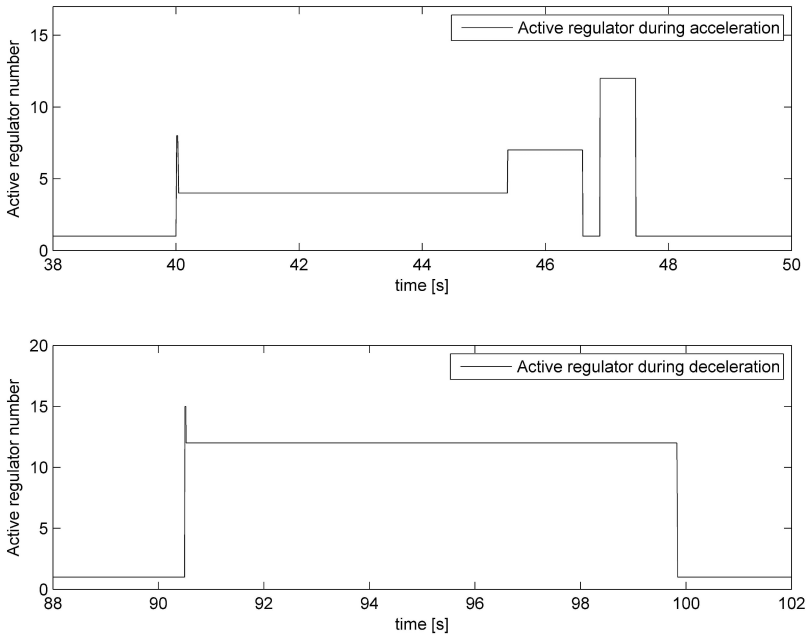


Figure 6.11.: **Active regulator - case 1**

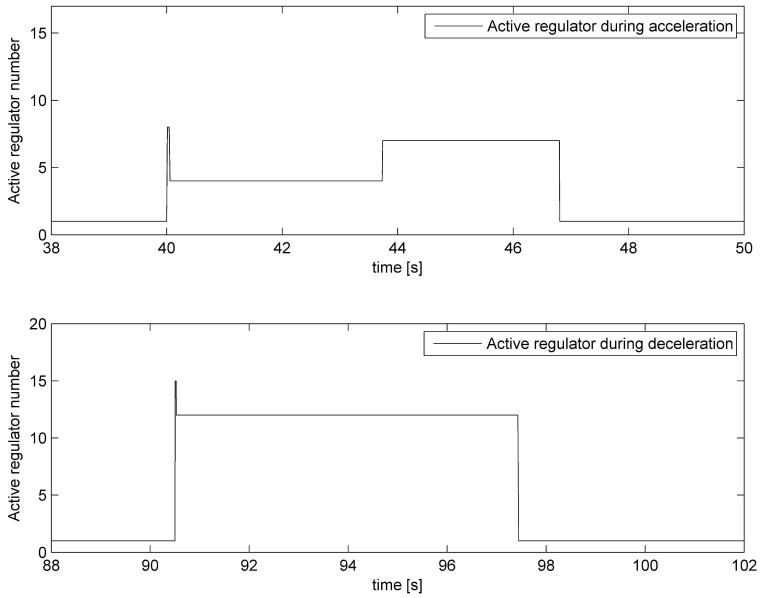


Figure 6.12.: **Active regulator - case 2**

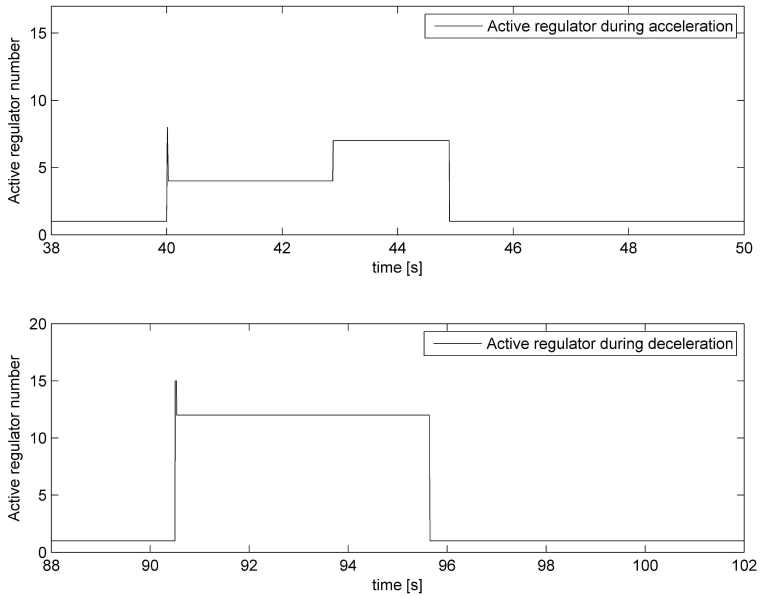


Figure 6.13.: **Active regulator - case 3**

Figures 6.14 and 6.15 show the massflow through the engine. The latter figure also includes the air from the bleed valve placed between the LPC and the HPC. As can be seen the massflow through the engine is less in test case 3 (35000 ft and 0.8 Mach), although the aircraft is travelling at a greater speed [mach 0.8]. This is due to the reduction in air density at high altitude. Figure 6.16 shows the massflow at the exit of the LPT - compare this to figure 6.14 to see the reduction in massflow that occurs due to engine bleeds.

The massflow passing through the engine core is significantly affected by the variable bleed valves and the variable stator vanes. Figure 6.17 shows the percentage of massflow extracted during the three cases and figure 6.18 shows the bleed valve position. As can be seen, the VBV position and percentage of massflow extracted are linearly proportional - this is the simplification used in the model. The VBV are scheduled versus corrected low pressure shaft speed - using corrected speed takes into account the variations in engine inlet conditions. A greater percentage of air is extracted at low engine speeds, due to the increased risk of surge at low speed. The purpose of the VBVs is to lower the operating point of the engine's high-pressure compressor (see also Section 2.2.2). The variable stator vanes (VSV) are similarly scheduled with corrected high-pressure spool speed.

Cooling flow to the HPT is discussed in Section 3.4.5. Figure 6.20 shows the massflow channeled to the the HPT.

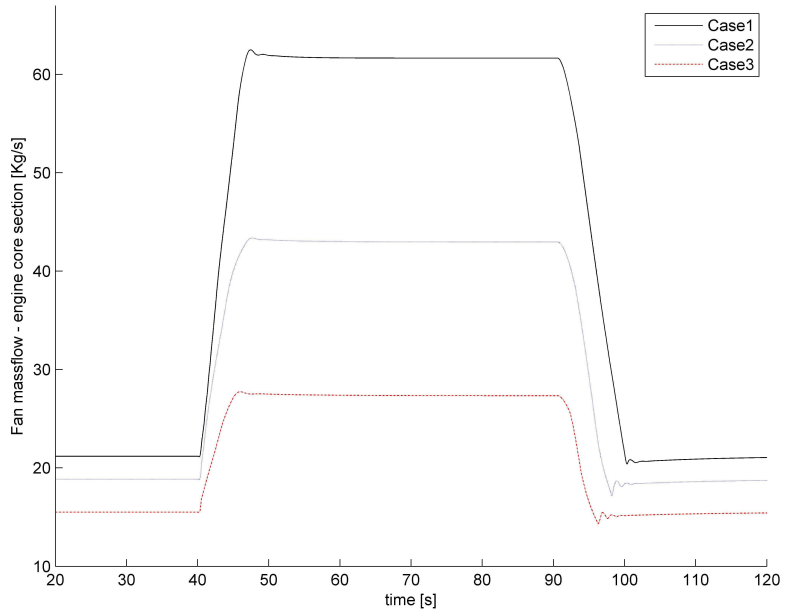


Figure 6.14.: **Fan massflow - core section**

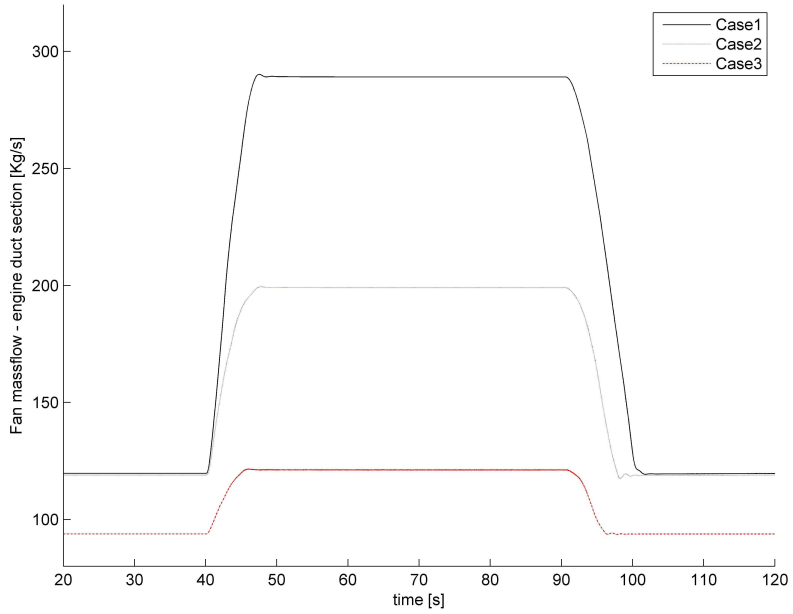


Figure 6.15.: **Fan massflow (inclusive of bleed air) - duct section**

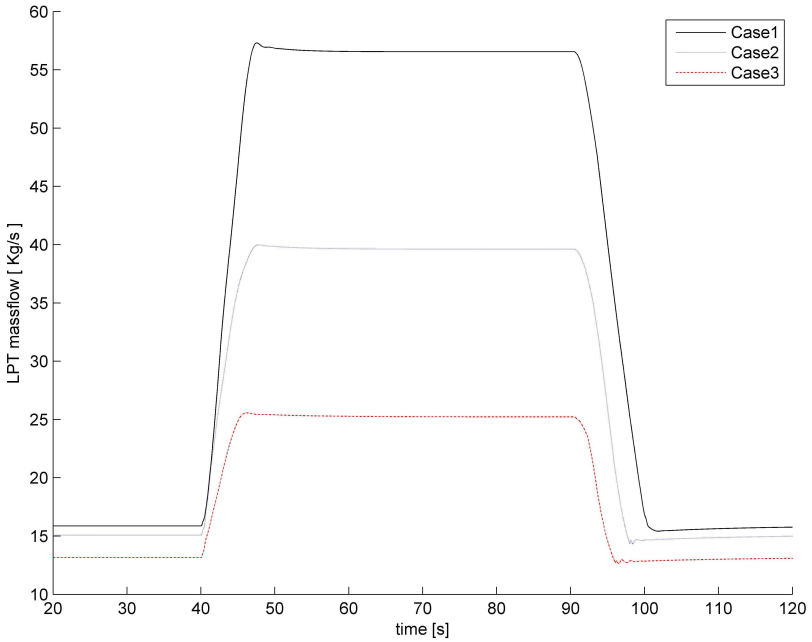


Figure 6.16.: **Engine massflow at exit**

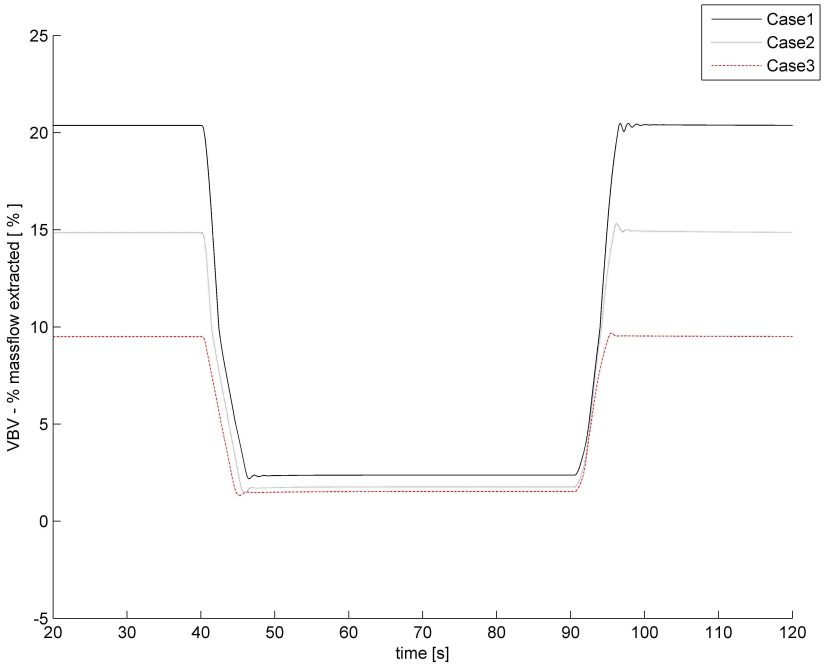


Figure 6.17.: **Variable bleed valves (VBV) - percentage of massflow extracted**

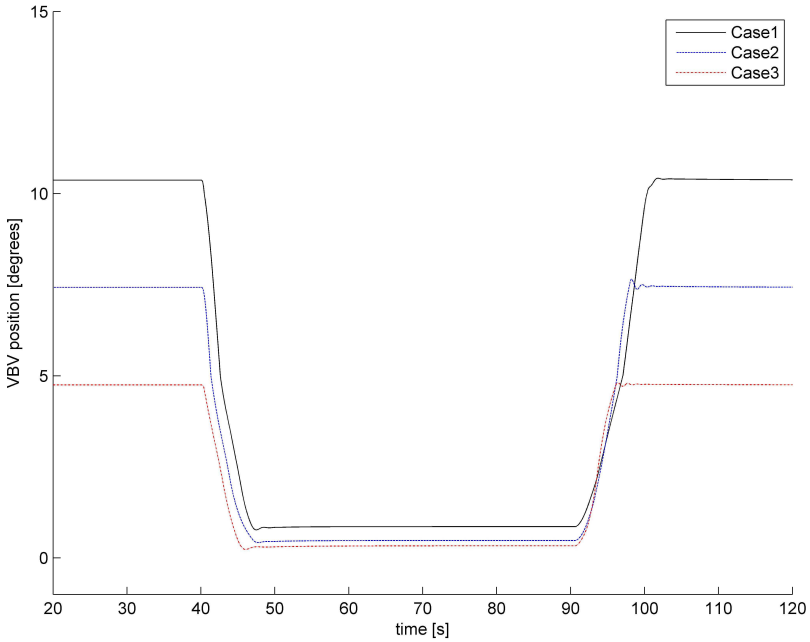


Figure 6.18.: **VBV position**

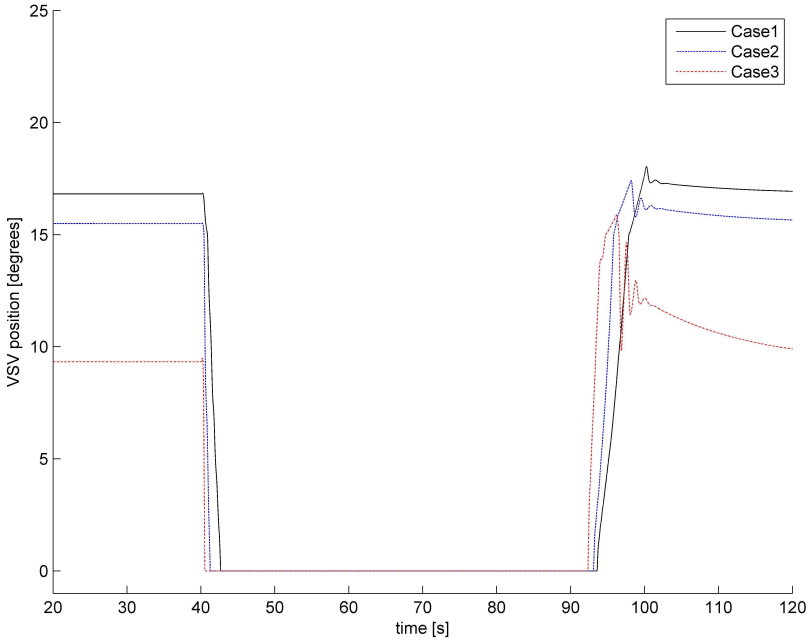


Figure 6.19.: **VSV position**

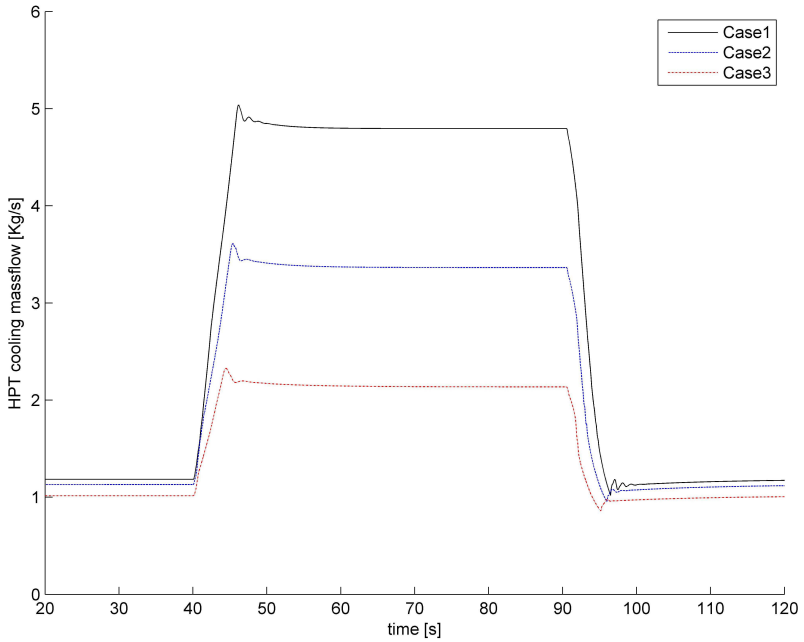


Figure 6.20.: **Cooling flow to HPT**

What makes turbines engines particularly suitable for aircraft is their very high power-to-weight ratio. Figure 6.21 shows the total power output by the engine's turbines. This massive power can also be the source of problems - it is not cost-effective to use a gearbox to obtain a higher bypass ratio because a gearbox sturdy enough to withstand such massive torque would offset the gains in efficiency (and would add a critical point of possible catastrophic failure to the engine's components). The following images show the percentage of total output power used by the fan (figure 6.22), HPC (figure 6.23), and LPC (figure 6.24) respectively. The HPC consumes the lion's share due to the high compression achieved over this stage, followed on closely by the fan. This barely compresses the gas stream, but moves a vast volume of air as shown in figures 6.15, 6.14 and 6.26.

The high pressure ratio of the HPC is the cause of its massive power consumption, this is shown in figure 6.27. The engine's overall pressure ratio is obtained

by multiplying the pressure ratios achieved over fan core, LPC and HPC and is shown in figure 6.28. As can be seen, the large surges in HPC pressure ratio during acceleration are small in comparison to the overall pressure ratio.

HPT inlet temperature is a limiting factor for turbine technology. However thermodynamics proves that the hotter the engine runs, the greater the achievable efficiency. Therefore engines aim to run with the highest possible HPT inlet temperature considering the metallurgical and mechanical limits of the turbine blades. The HPT inlet temperatures are shown in figure 6.29.

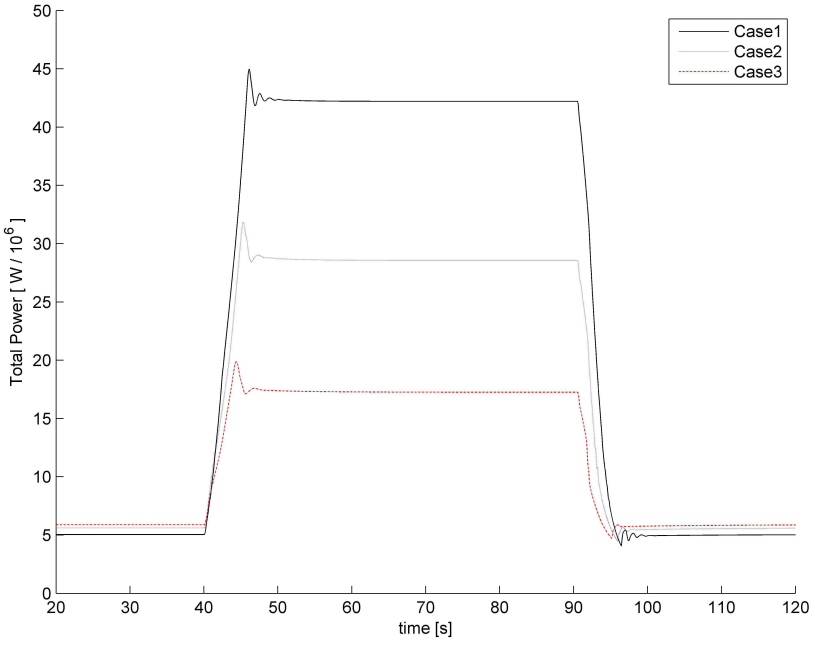


Figure 6.21.: **Total output power of the turbines**

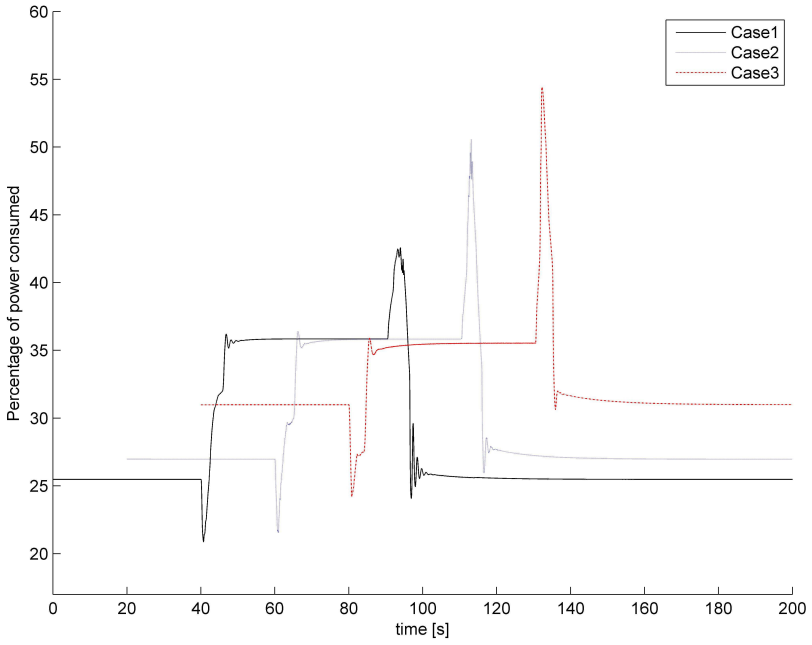


Figure 6.22.: **Fan power consumption - percentage of total power output**

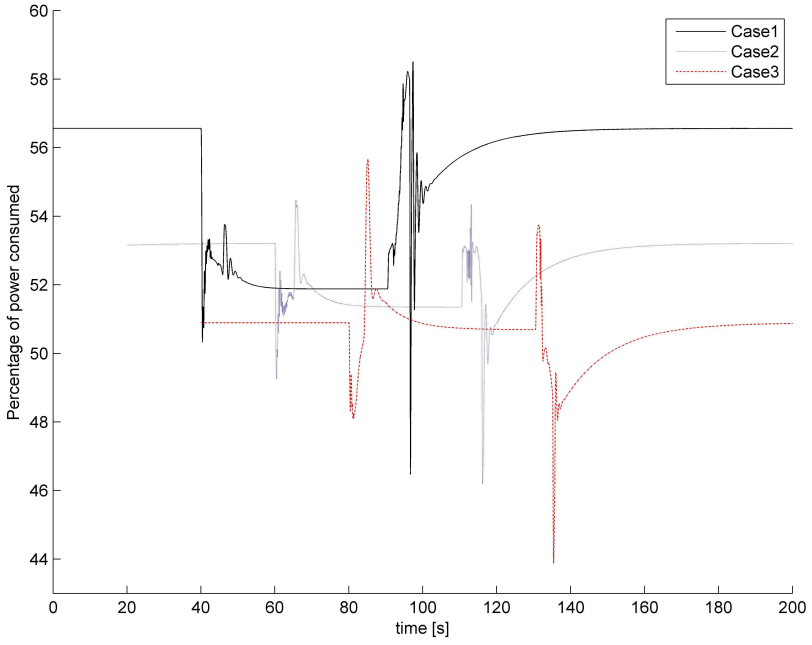


Figure 6.23.: **HPC power consumption - percentage of total power output**

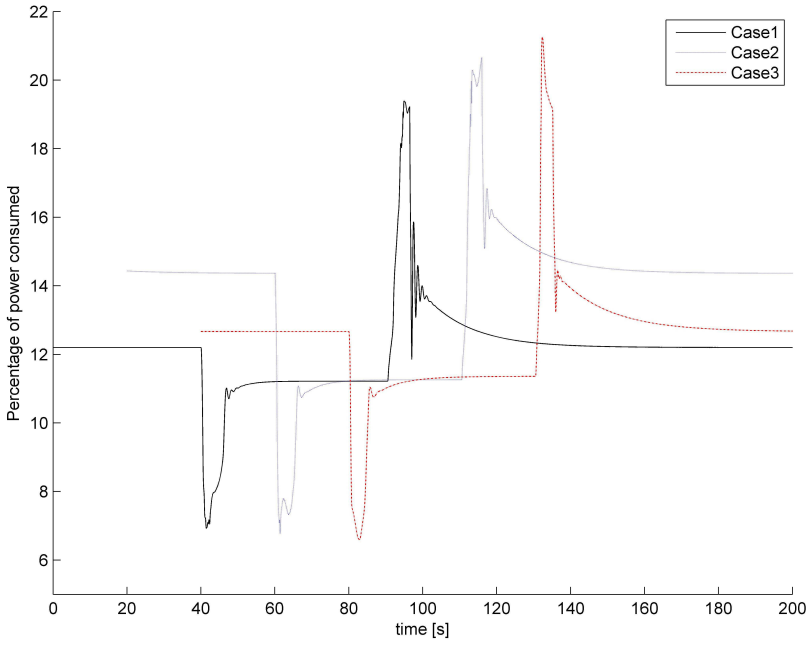


Figure 6.24.: **LPC power consumption - percentage of total power output**

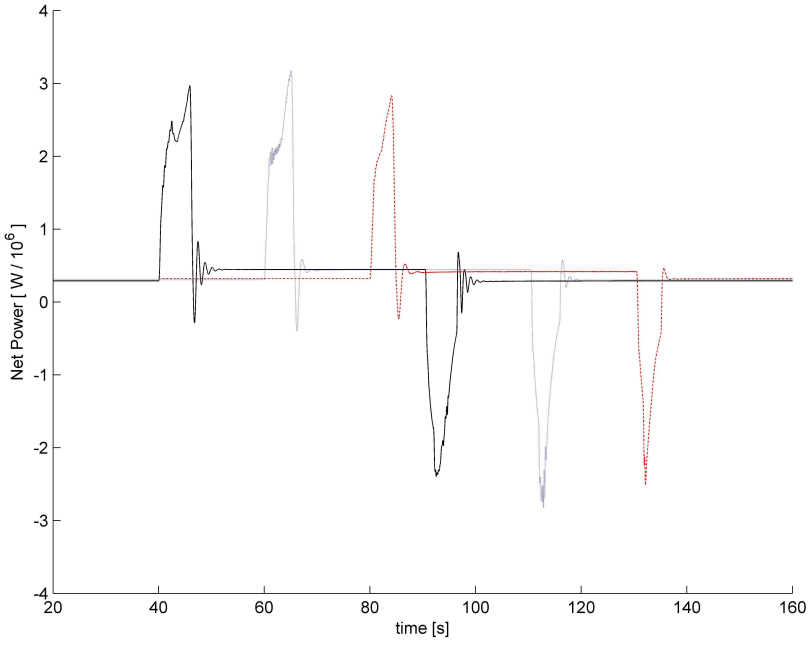


Figure 6.25.: **Engine net power**

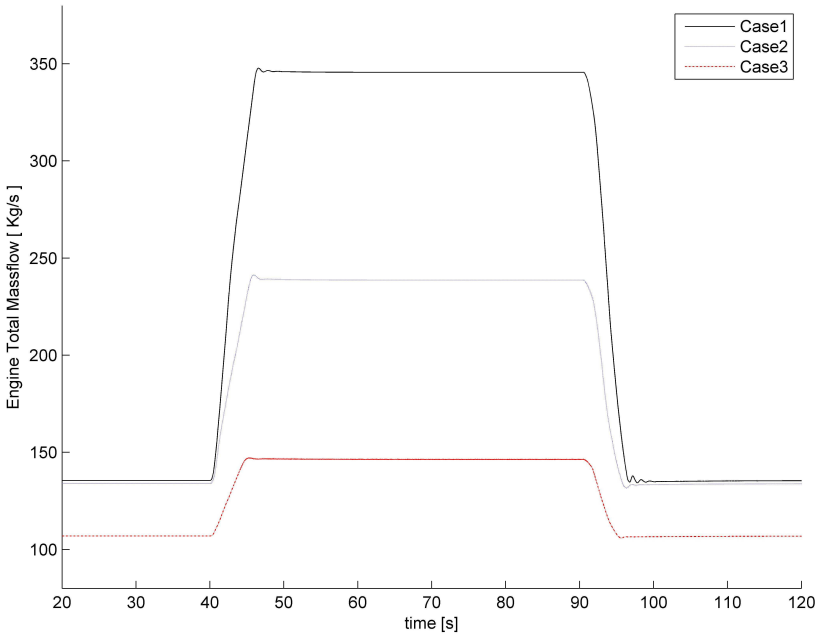


Figure 6.26.: **Total engine massflow**

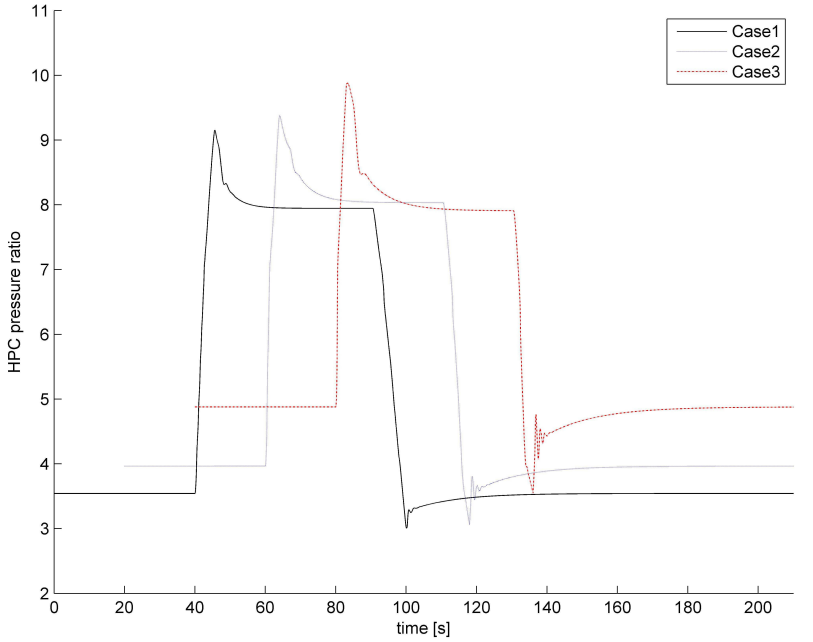


Figure 6.27.: **HPC pressure ratio**

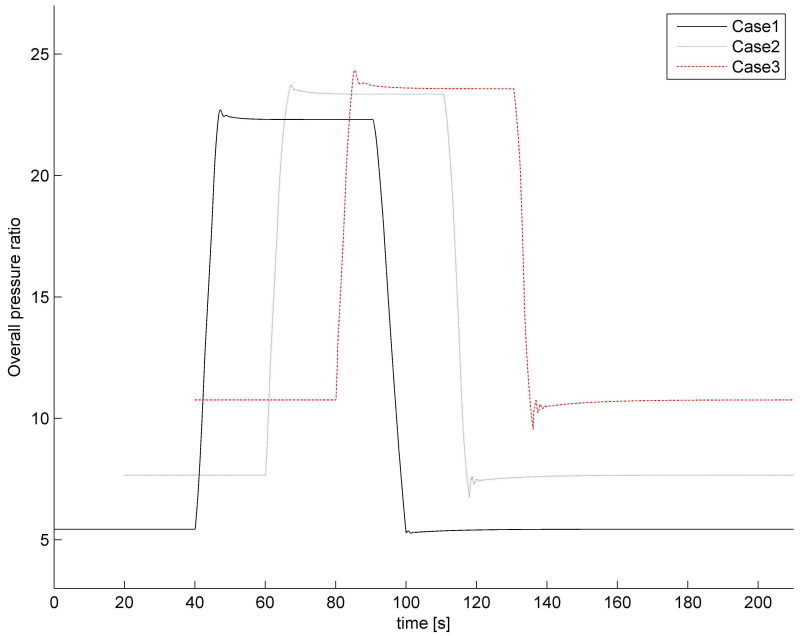


Figure 6.28.: **Overall compression ratio**

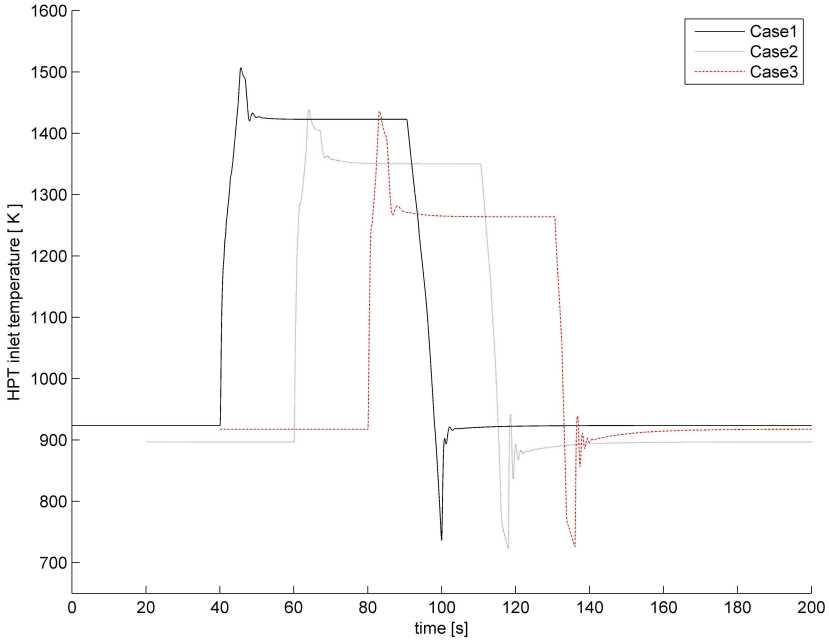


Figure 6.29.: **HPT inlet temperature**

6.4. Summary

This chapter presented the performance plots of key dynamic and steady-state tests. These tests occur under a wide range of operating conditions:

1. Case 1. Sea Level Static, from 40% to 98% of maximum thrust.
2. Case 2. Environment conditions: 15000 ft (4572 m), Mach Number 0.45, from 50% to 98% of maximum thrust.
3. Case 3. Environment conditions: 35000 ft (10067 m), Mach Number 0.8, from 60% to 96% of maximum thrust.

These figures show the dynamic performance of both the engine and the benchmark controller under rapid acceleration, decelaration and at steady state conditions.

7. Conclusions

By way of conclusion we now summarise the main contributions of this dissertation and outline some recommendations for future research in the area of aero-engine simulation.

7.1. Review of contents

First, a brief review of what has been presented:
Chapter 2 introduced the architecture and design principles of the turbofan aero-engine. The effect of the engine environment (namely frontal velocity, altitude and ambient temperature), and key aerodynamic phenomena such as surge and other transient effects were discussed along with performance requirements.
After the necessary introduction of chapter 2, the following chapter provides further details about the target engine, the engine components and how they work together. Then follows the details of how key components of the turbofan can be modelled. Key aerodynamic and physical phenomena are discussed further. Novel simulation elements are explained in detail: this includes heat soak, cooling flow and the core and duct nozzle. The overall simulation architecture - a lumped element model with a feedforward solution, known as an "aerothermal transient performance model" is outlined and compared with alternatives.

Chapter 4 outlines the control problem. A switched, gain-scheduled aero-engine controller with bumpless transfer and anti-windup has been implemented. While the architecture is an industry standard, its implementation in Simulink has been completely redesigned so that it is now easy to add additional control loops and organize a hierarchy of controls. This architecture is analyzed and plots of its performance and tracking are shown in order to explain its operation.

Chapter 5 moves onto the more practical aspects required for implementation of the engine model as outlined in chapters 3 and 4. The pros and cons of the Matlab-Simulink environment are discussed, along with some of the "tricks of the trade" required for initialisation of a large simulation.

Although the theory of the turbomachinery components was discussed in chapter 3, their implementation in Simulink requires some extra thought - it has been shown how the characteristic map scaling is implemented in a fashion suitable to the task at hand: matching the performance of a target engine. Via a simple reformulation of the conventional formulae we are able to succinctly implement a scaling module that is suitable for massflow, pressure and referred speed. To allow the scaling to occur and be adjusted in realtime, rather than scaling the component's characteristic map, the data is scaled instead: onto the characteristic map at the component's inlet, then at the outlet data is scaled from the map to the target values required in the simulation. A brief discussion of the issues of algebraic loops that occur when providing each map with pressure ratio and how these can be solved is given at the end of the chapter.

Chapter 6 shows the performance plots of the engine - the key plots that characterise the performance of this particular engine, and those that show key characteristics of aero-engines in general and turbofan engines in particular. Some plots of engine and controller performance were shown in chapter 3 to aid in understand-

ing the operation of the controller. Here additional test cases are introduced to detail the steady-state and dynamic performance of the engine at sea-level static conditions, then at 15000 feet and 35000 feet of altitude with respective frontal velocities of 0.45 and 0.8 Mach. The engine shows characteristic turbofan behaviour and the controller performs well, with minimal overshoot or undershoot that is well within the FAA requirements.

7.2. Contributions of this dissertation

This thesis has presented the details of the design, implementation and validation of a complex nonlinear realtime simulation model for a civil turbofan aircraft engine, with an associated gain-scheduled feedback control system.

For the control engineer and mechanical designer alike, some of the key features of the model are its flexibility, modularity and ability to easily handle the nonlinear dynamic features of the engine model. It seems that, in other available models, these features are either partially absent, difficult to enable or only available with input from the model developers. In contrast, these features are all inherent in the model described within this thesis and therefore, it is relatively easy to "plug-and-play" different components or control schemes and accurately assess their impact on the dynamic behaviour of the system as a whole. Such flexibility is clearly desirable for industrial practitioners and researchers alike. In fact, the model developed in this thesis has now been adopted by Alstom Aerospace for their routine use [41].

7.2.1. Engine developments

The model was developed in a modular fashion using where possible the underlying physics and avoiding empirical approximations. This allows us to achieve a greater degree of independence from empirical data from a manufacturer and allows the model to be easily adapted to a new engine design: the basic physics still holds. Similarly, failure cases would not be covered by standard operating range empirical data.

For this purpose, a model of the core nozzle that is suitable for a non-iterative model has been implemented from basic theory. Should the model change, it is sufficient to change the nozzle diameter (and perhaps adjust the efficiency, although that is unlikely to be required unless the changes are dramatic) - it is therefore not necessary to have a full performance dataset over the entire operating range.

The turbofan duct is also implemented as a nozzle - this allows fan power consumption to be accurately modelled based on fan duct and core pressure ratio, and not as a percentage of output power - this has the same benefits of not requiring a full data set over the entire operating envelope.

The fan consumes a major portion of the engine's power, therefore an accurate model of this element is highly desirable to obtain a good match with dynamic performance: although the pressure ratio achieved by the fan is small (depending on the engine design, between 1.5-1.9 for the duct section), the massflow is enormous, often well above 300 Kg/s so the consequences of inaccurate performance data would be large. The fan is implemented as two sections: a core and a duct section, each with its own characteristic map.

Cooling flow is also implemented using a formula that although empirical in origin allows physical parameters to be used to define cooling flow: namely the radius of the cooling flow outlet and the difference in pressure between the high pressure

compressor and the high pressure turbine. The benefit of this is again that for a changed design it is sufficient to scale a single characteristic value to get acceptable performance over the operating range. With just one datapoint (comprised of inlet and outlet pressure and cooling massflow) the cooling flow parameters can be determined for the entire operating range - rather than requiring a grid of cooling flow datapoints that covers the operating range.

Map scaling is made much easier with a reformulation of the key formulae: if one has target positions on the characteristic maps and target performance data, these can be used directly in the scaling formulae. Furthermore, this new expression is suitable for scaling data at both component inlet and outlet by simply reversing the map and target operating points (further details are in Chapter 5, Section 5.5). The scaling is performed in realtime (by scaling the data and not the maps) which allows the map scaling to be adjusted while the simulation is running which greatly simplifies engine tuning. Any momentary adjustments to the baseline scaling parameters can be recalled at startup or simply reset to test alternative adjustments.

Heatsoak is implemented in the simulation. Again, data for this was lacking so the heatsoak values of an industrial engine were scaled down to match the mass of a representative aero-engine. The theory behind its use is outlined in chapter 3, Section 3.4.2.

7.2.2. Controller developments

A switched gain scheduled feedback controller incorporating bumpless transfer and antiwindup functionality was designed and implemented on the engine model in accordance with current industrial practice. Together the engine and controller cover the full flight envelope (with the exception of startup and windmilling) and achieve dynamic performance that closely matches that of a real engine. The con-

trol scheme corresponds closely to current industrial practice and delivers high-performance tracking of pilot demands while ensuring that the operating constraints of the engine are met at all times. The architecture of the regulator is analysed in chapter 4.

The implementation of this controller has been completely redesigned from scratch. The gain-schedule is not hard-coded into the simulation but is provided by a script. To aid with rapid tuning the script allows the gains to be interpolated according to altitude and fan speed (it would be more appropriate to use the sum of fan and HPC power consumption - but this is data that will change upon a new target design and while the engine is being tuned, so it is best to work with what does not change significantly: a target fan speed and altitude). It is a compromise, but one that works well. The gains can also be manually scheduled over the range and not simply interpolated.

The architecture implementation was redesigned so that the output of all individual controllers that go into a "maximum" or "minimum" block is compared simultaneously rather than in a cascade fashion, while informing all controllers of the active controller. This allows the number of elements controlled by the engine regulator to be easily extended.

7.3. Recommendations for future work

7.3.1. Future modelling work

The engine in this study is considered in isolation. The model could benefit greatly from being coupled with a simple airframe simulation for the purpose of modelling a true correspondence between achievable frontal velocities and thrust and altitude. Our test cases do take this into account - the frontal velocity is within the

range of that achievable with a reference airframe, i.e an aircraft flying at the given altitude with a similar engine at full thrust would achieve approximately the test case velocity. However with even a simple model, it would be possible to greatly extend the number of test cases and reduce reliance on external data.

Indeed the author does not recommend further extensions to the engine simulation itself unless there are guarantees of obtaining the necessary data directly from an engine manufacturer: much of what can be done with generic data is already in place - and therefore further work in this direction would be extremely difficult (if not impossible) without extensive support. That said, on the premise of such data being available, the simulation could then be easily extended to include failure cases. However failure data (such as the effects of bird strike) is necessarily very engine specific, detailed, and expensive to produce and is (in the author's opinion) much less likely to be available than generic data - which can itself be difficult to get hold of.

Looking at a worst-case scenario of little to no further data being available, the first targets for investigating failure scenarios should be engine elements that are already implemented and that can fail in a fixed state: the obvious targets for this would be the variable bleeds - these could be modelled as stuck at the always on or off position or indeed anywhere in-between, and would have a dramatic effect upon the high pressure compressor.

It should however be noted that for civil aircraft the pilot's response to most serious failure cases would be to simply shut down the engine. The aircraft operator will not risk damaging a very costly investment by running the engine at conditions that could require an overhaul (such as high turbine inlet temperature) if the aircraft can limp home without the engine running: commercial aircraft have at least two engines, and can fly to the nearest landing point on the remaining

engine.

In summary, industry with its resource of test data can benefit from the engine model in many ways: the simulation can be used to implement failure scenarios and because of its modular design based on scripts and the avoidance of empirical approximations, the model can be easily adapted to match the performance of specific engines and used as a validation tool - indeed Alstom Aerospace has used the model during the course of the past year to produce engine operating points and failure conditions for the purpose of assessing condition monitoring tools.

A further use, suitable for both industry and academia, is to use the model to assess and prototype novel control schemes: i.e. as a testing tool for control design. This model is, to the best of the author's knowledge, one of the most fully featured non-proprietary aero-engine models implemented in the Matlab/Simulink environment. This is a familiar platform for control engineers and the graphical nature of Simulink makes it easy to tap into the engine's data streams, without requiring an understanding of the overall simulation.

7.3.2. Future control work

Should the existing controller architecture be retained, there is scope for improving the gain scheduled tuning of the individual PI loops and reexamining the present anti-windup strategy which is an ad-hoc scheme that performs well for the current range of the model but that is without sure guarantees of performance (as is PI gain scheduling itself).

With some provisos, the model can be extended to include various sources of uncertainty, faults, and failure scenarios. Control schemes that aim to achieve robust fault tolerant performance can therefore be tested on this platform.

Current gas turbine control algorithms are usually designed from a worst-case scenario perspective, to guarantee steady state operating points and transient performance in spite of phenomena such as engine deterioration over its lifecycle, engine scatter or heat effects, and faults/damage. As a consequence, present control systems have large safety margins that restrict the achievable overall engine performance. The ability of current generation engine control algorithms to cope with unexpected faults and/or damage to the engine is also limited, and in practice is often reliant on the conservatism of the original design providing robustness to such events.

Because of the safety implications of the nature of our application, the robustness of any applied controller is of vital importance [9]. The first step in control design is building a sufficiently accurate mathematical model of the plant - and that has been achieved here. The term sufficiently accurate is used because, apart from very simple systems, having an exact mathematical model of a physical plant is almost impossible. So by modelling we aim to adequately capture essential features of the plant for analysis and control synthesis. This abstraction necessarily introduces system uncertainty which is the difference between the nominal mathematical model and the actual physical plant. System uncertainty generally includes other concepts such as the imperfect knowledge of the system dynamics and unknown or uncertain system parameters. Hence, any effective control system should be, up to some degree, robust against uncertainties. In addition to system uncertainty, an applicable controller must be capable of dealing with environmental uncertainties such as noise and other (bounded) disturbances.

Currently, one of the most promising control techniques is gain scheduling achieved by interpolating controllers synthesized at different linearized operating points throughout the flight envelope [31]. As discussed in [31], during the past 15 years

work has been developed for gain scheduling, resulting in the so called LPV (Linear Parameter Varying) models. This method offers a theoretical framework to ensure stability, performance and/or robustness of the controlled system via convex optimisation over LMI (Linear Matrix Inequalities).

From a control engineering perspective, one of the main issues with the current model is that no linearisations have currently been developed. Virtually all standard control system design tools, aside from those based on empirical tuning rules or fuzzy-logic etc, begin from linear mathematical models. It is well-known that with this starting point, it is possible to design controllers with stability, robustness and performance guarantees, and that many powerful design techniques can be invoked. While, in this case, it was seen that the dynamics of the engine allowed successfull controller tuning to be achieved by directly using the nonlinear model, it would clearly be desirable for these controllers to be accompanied by classical robustness measures such as gain and phase margins, along with other performance metrics. It is also noted that virtually all gain-scheduled control algorithms (including so-called linear-parameter-varying (LPV) control methods) are based to some extent on linear models. This is also broadly true of most adaptive control methods as well.

However, linearisation of a complex model, such as that described in this thesis, is not a trivial task. While Matlab contains several dedicated functions to trim and linearise nonlinear models they are not always easy to work with and, indeed, it is standard practice for developers of complex nonlinear models to develop their own trim/linearisation routines which work specifically with a given model. Furthermore, finding linearisations around certain equilibria may be difficult or impossible. While some attempt at linearising the current model with the Matlab routines was made, it was found that `linmod` in particular failed to find accurate

linear models. This is a well documented problem; see [14] for further reading. Thus, while the lack of a set of linear models accompanying the nonlinear model is unfortunate, it was felt that the trimming and linearisation task was too great to be conducted during the duration of this research and is therefore left open for future study. It is noted that, as this model is purely open-source, any interested party could begin development of such routines.

Adaptive control (considered as other than gain-scheduling) could also have a strong role to play in future control design: the aim of adaptive control is to self-adapt automatically engine control algorithms in order to improve the capability to handle faults and engine damage, and improve overall engine performance by reductions in conservative safety margins. Other gains that could be expected from the introduction of adaptive control include an increase in engine component life and decreased specific fuel consumption through more accurate control of the surge margin.

To date, very few applications of adaptive control have been developed and marketed for gas turbine engines. For aircraft engines, adaptive control has been applied to restricted and specific cases such as torque adaptation on selected turboshafts, or adjustment of the LPC working line and thrust adaptation on some military gas-turbine engines. However, adaptive control of the overall engine is not currently implemented on any civil aero-engine (or even on military engines). The main reason for this is that gas turbine engines are classed as safety-critical systems, e.g. for aircraft engines, the safety of the overall aircraft and its occupants is critically dependent on the correct functioning of the engine. This places stringent analysis and certification requirements on aero-engine control systems. In particular, overall stability of the closed loop engine control system in the presence of uncertainty and disturbances must be clearly demonstrable, through both ana-

lytical analysis and computer simulation. This issue represents a major problem for many adaptive control schemes, which because of their time-varying nature, do not come with strong stability guarantees. In particular, methods to suitably constrain and monitor the adaptation of feedback controllers so that serious performance degradation or even instability cannot occur are crucial if such schemes are to be implemented on safety critical systems.

Optimal state space control [22, 43] and robust model-based control [12] have also been applied to gas turbine control. In recent years there has been considerable research on the application of multivariable gas turbine controllers. Most apply a linear fixed controller to the highly nonlinear plant which may not be the most suitable control strategy [37]. Artificial neural networks have also attracted a lot of attention because of their ability to approximate nonlinear relationships [37].

In summary, suggested future work would be to focus on extending the model to include various sources of uncertainty, fault and failure scenarios and on extending the control scheme to deliver robust fault tolerant performance.

A. Key simulation scripts

CASE1.m

Calls in the correct sequence all script files required for the predefined test case 1. CASE2.m loads the sequence for test case 2 and so forth.

SIM_Stop.m

Ensures the simulation is stopped. While not strictly necessary, this is a useful location for simulation maintenance code, e.g. to store workspace contents and clear workspace variables when running batches of test cases etc.

CounterScript.m

This file was used to call a function that loads into a sub-workspace the number of the current simulation run. This is useful for running multiple test cases - which requires the workspace to be cleared between runs. A sub-workspace can store the number of test cases run so far, allowing the next test case to be called up automatically. This is now legacy code and simply sets the counter variable to 1.

LoadInitialStatesCASE1.m

This file loads workspace data from a ".mat" data file, controls the simulation starting time and calls the function "CheckInitialStates" which is responsible for checking and, if required, reformatting the state blocks' addresses. It also loads the initial conditions of the simulation's memory blocks.

CheckInitialStates.m

The Matlab function responsible for ensuring that the states recalled are linked to the correct Simulink block by formatting the block address should the simulation name change. It also removes from the state structure any states that are no longer present in the model.

EnvironmentConditionsScriptCASE1.m

Recalls the parameters for the several steady-state and dynamic test cases.

EngineParameters.m

Recalls key engine parameters (that tune the engine to the target profile) and also the map scaling parameters.

PI_Gainschedule.m

Stores the values of the controller's gain-schedule and places them in the workspace in a format suitable for the Simulink lookup tables. In accordance with the flexible architecture of the controller's implementation, this script is designed to permit the number of variables controlled to be easily extended.

PIgainscheduleInterpolation.m

This function interpolates the gain values according to altitude and fan speed. It would be more appropriate to interpolate based on the sum of fan and HPC power, but this is data that will change upon a new target design and while the engine is being tuned, so it is best to work with what does not change significantly: a target fan speed and altitude. It is a compromise, but one that works well.

PI_Fixed_Limits.m

Sets the fixed limits of the controller.

SIM_Starter.m

The partner to "SIM_Stop.m".

B. Key simulation scripts (code)

Startup scripts - test case 1

```
%--------------------------------------------------------------------
run SIM_Stop

%clear all

MAPTUNINGSWITCHval=0 % when this is called the model will use the
                     % original map tuning parameters.
                     % See EngineParameters.m

run CounterScript
run LoadInitialStatesCASE1  % sim start time is also specified here
run EnvironmentConditionsScriptCASE1
run EngineParameters
run PI_Gainschedule
run PI_Fixed_Limits
run SIM_Starter
%--------------------------------------------------------------------
```

Initial states and memory block initial values

```
%--------------------------------------------------------------------
% model initialisation script
%--------------------------------------------------------------------

initfile = char('INITconds_SLS_1500')
eval(['load' initfile 'tout xout memout* M*0 H*0 T*0 conout GOV...
 p3s T3 xFinal']);

tstartip = char('0');
tstopip = char('10000');
```

```
tstart = round(min(eval(tstartip),max(tout)));
index1 = min(find(tout >= tstart));

if (isempty(index1))
   [index1,index2] = size(tout);
   tstart = round(tout(end));
end

tstop = eval(tstopip);

% uncomment to change simulation start time if needed
% tstart=0

xin = xout;
xin=CheckInitialStates(xin, index1);

%-----------------------------------------------------------------------
% memory block initialisation

meminit_01 = memout_01(index1)
meminit_02 = memout_02(index1)
meminit_03 = memout_03(index1)
meminit_04 = memout_04(index1)
meminit_05 = memout_05(index1)
meminit_06 = memout_06(index1)
meminit_07 = memout_07(index1)
meminit_08 = memout_08(index1)
meminit_09 = memout_09(index1)
meminit_10 = memout_10(index1)
meminit_11 = memout_11(index1)
meminit_13 = memout_13(index1)
meminit_14 = memout_14(index1)

meminit_015 = 0;
meminit_016 = 0;
meminit_017 = 2001;
%-----------------------------------------------------------------------
```

Saving a simulation run

```
%-----------------------------------------------------------------------
% save simulation data, ONLY WORKS IF SIMULATION PAUSED
%-----------------------------------------------------------------------

namesys = bdroot;
simstatus = get_param(namesys,'SimulationStatus');

if ((simstatus(1) == 'p')|(simstatus(1) == 's'))

```

```
disp(' ');

prompt = {'Enter name of *.mat file to save run data to:'};
title1 = 'Input for saving simulation data';
lines  = 1;
def    = {'runxxx'};
answer = inputdlg(prompt,title1,lines,def);

savefile = char(answer(1));

eval(['save ' savefile]);
disp(' ');
disp('simulation data SAVED');
%-----------------------------------------------------------------------
% continue the simulation if paused
if (simstatus(1) == 'p')
   set_param(namesys,'SimulationCommand','continue');
   disp(' ');
   disp('simulation CONTINUE');
   disp(' ');
end

else
   disp(' ');
   disp('WARNING simulation data NOT SAVED');
   disp('PAUSE or STOP the simulation to save the data');
end
%-----------------------------------------------------------------------
```

Absent states and block addresses

```
%-----------------------------------------------------------------------
function xin=correctinitialstates(xin, index1)
%-----------------------------------------------------------------------
xin2=xin;
forit1=length(xin.signals);
index02=0;
NumBlocksNotFound=0;
missingstatesindex=0;
%-----------------------------------------------------------------------
for a=1:1:forit1;
blockstatenameslist(a,1)={xin.signals(a).blockName};
%-----------------------------------------------------------------------
% this section replaces the old model name as found in the state
% list structure, e.g. xin, with the current model name.

seps=findstr(blockstatenameslist{a}, '/');% the "/" in the name!
firstseps=seps(1);
CurrModelName=bdroot;
```

```
SavedModelName=sscanf(blockstatenameslist{a}, '%c', (firstseps-1));
         if strcmp(CurrModelName,SavedModelName)==0
                %warn if the model name has changed!
                disp('model name has changed...');
         end
%replace saved model name with current name.
blockstatenameslist{a, 1}=regexprep(blockstatenameslist{a},...
SavedModelName,CurrModelName);

%-----------------------------------------------------------------------
% Section eliminates from state list any states not found in the
% current model.Note that the current model name needed to be
% inserted prior to this otherwise all states will be erased! Note
% this script can automatically REMOVE states, but cannot
% automatically discover new ones - these need to be added with the
% command "gcb" and the utility script "writeAnewState.m".

b=blockstatenameslist{a};
        try,
                d=get_param(b, 'Handle');
        catch,
                clear('d') % inserted for safety - might be redundant
                missingstatesindex=missingstatesindex+1;
                disp('this one not found!')
                indexmissingstates(missingstatesindex)=a;
                NumBlocksNotFound=NumBlocksNotFound+1;
        end
        %write a list of the existing block state names.
        if ~exist('d')==0 % i.e. does exist.
                 index02=index02+1;
                 existingblockstatesnames{index02,1}= ...
                 blockstatenameslist{a};
        end
end %end of [for a=1:1:forit1;]

if exist('indexmissingstates')==1 |...
                         strcmp(CurrModelName,SavedModelName)==0
    % if a)OR b): for ANY OF THE ABOVE CONDS (i.e. states are missing
    % or the model has changed name) THEN will remove missing states
    % if option a) is true and will reformat the xin model name to
    % the current model name. If a) exist('indexmissingstates')==1
    %--> i.e. indexmissingstates exists.
    % This occurs when there are missing states.
    % if b) strcmp(CurrModelName,SavedModelName)==0 --> i.e. when the
    % current model name and the SavedModelName
    % (as found by parsing xin) are not the same.
%-----------------------------------------------------------------------
    % uses indexmissing states to remove these from xin
    if exist('indexmissingstates')==1
                for eraser=(length(indexmissingstates)):-1:1
                %note this works only because we are erasing from
                % highest to lowest index number!!!
```

```
                xin2.signals(indexmissingstates(eraser))=[];
                end
        end

%-----------------------------------------------------------------------
   % change new xin2 to current model names
   for a=1:1:index02
   xin2.signals(a).blockName=regexprep(xin2.signals(a).blockName,...
   SavedModelName,CurrModelName);
   end
end %end of [if exist('indexmissingstates')==1 | strcmp...

for a=1:1:forit1
tempVal =  xin2.signals(a).values(index1);
xin2.signals(a).values = tempVal;
end

%-----------------------------------------------------------------------
xin=xin2 % a shiny new xin!
%-----------------------------------------------------------------------


%----------------------  garbage collection  -------------------------
clear('a','b','d')
clear('firstseps','forit1', 'index02', 'name1', 'seps', 'xin2')
clear ('blockstatenameslist','NumBlocksNotFound')
%-----------------------------------------------------------------------
```

New model states

```
%-----------------------------------------------------------------------
% When placing into the model a new block that has a state, use gcb
% command to get its path name. Getting an initial value can be
% tricky - an educated guess will need to be provided should the
% initial value be unknown. Another consideration is if it should
% be defined as a continuous or not continuous state - here we shall
% assume continuous.Use command "gcb" to get block address and
% substitute this for 'FOO' below.
%-----------------------------------------------------------------------
% to get this to work type at command, e.g. :
%  xin=writeanewstate(xin, 'FOO', '450')
%-----------------------------------------------------------------------
function xin= writeanewstate(xin, blockpath,statevalue)
%-----------------------------------------------------------------------

% xin structure has fields:
% xin.signals(a).values
% xin.signals(a).dimensions
% xin.signals(a).label
```

```
% xin.signals(a).blockName

% assume that all dimensions are 1 and that all are continuous states
% (label = CSTATE)

% ASSUME xin already loaded else
% load ('initialstates', 'xin')

statecurrlength = length(xin.signals);

xin.signals(statecurrlength+1).values=statevalue;
xin.signals(statecurrlength+1).dimensions=1;
xin.signals(statecurrlength+1).label='CSTATE';
xin.signals(statecurrlength+1).blockName=blockpath;

xin;
%-------------------------------------------------------------------
```

State structure preparation

```
%-------------------------------------------------------------------
function xstruct = makestatestruct(mdl)
%-------------------------------------------------------------------

%Get the initial condition and sample time info from the model
[sys,x0,stateblocks,ts,xts]=feval(mdl,[],[],[],0);

% Find the unique state names
[uniquestates,uind] = unique(stateblocks);
uniquexts = xts(uind);

% Create the structure
for ct = length(uniquestates):-1:1
    ind = find(strcmp(uniquestates(ct),stateblocks));
    if uniquexts(ct) == 0
        xsignal(ct) = struct('values',x0(ind),'dimensions',...
        length(ind),'label','CSTATE','blockName',uniquestates(ct));
    else
        xsignal(ct) = struct('values',x0(ind),'dimensions',...
        length(ind),'label','DSTATE','blockName',uniquestates(ct));
    end
end

if ~isempty(uniquestates)
    xstruct = struct('time',[],'signals',xsignal);
else
    xstruct = struct('time',[],'signals',[]);
end
%-------------------------------------------------------------------
```

PI gain schedule interpolation script

```
%-------------------------------------------------------------------------
function MatrixOut=PIgainscheduleInterpolation(a1,a5,a21,a25)
%-------------------------------------------------------------------------
% you can get these from the main workspace,
% but this is quicker for now.

GAIN_sched_vels=[1000 2000 3000 4000 5000]; %these are the x axis

GAIN_sched_alts=[0 2000 4572 7620 10668] ;  %these are the y axis


%-------------------------------------------------------------------------

TestMatrix =  {'a1'  'a2'  'a3'  'a4'  'a5';
               'a6'  'a7'  'a8'  'a9'  'a10';
               'a11' 'a12' 'a13' 'a14' 'a15';
               'a16' 'a17' 'a18' 'a19' 'a20';
               'a21' 'a22' 'a23' 'a24' 'a25'};

%-------------------------------------------------------------------------
%section creates values for a6, a11, a16 and then a10, a15, a20,
% i.e. fills out columns 1 and 5

for clmn = 1:4:5; %(i.e. 1 & 5)
    for i=2:4
        for j=1:4
        high = eval(TestMatrix{5,clmn});
        low =  eval(TestMatrix{1,clmn});
        %averaging expression
        val = low(j)+ ( (high(j)-low(j))/GAIN_sched_alts(5) )...
              *GAIN_sched_alts(i);
        eval( [ TestMatrix{i,clmn},'(j)=val' ] )
        end
    end
end
%-------------------------------------------------------------------------
%section creates values for a2, a7, a12 etc down column 2, then
% moves on to column 3 and 4 (column 1 and 5 are taken care of
% via previous block)
for k=2:4
    for i=1:5
        for j=1:4
        high = eval(TestMatrix{i,5});
        low =  eval(TestMatrix{i,1});
         %averaging expression
        val = low(j)+ ( (high(j)-low(j))/...
        (GAIN_sched_vels(5)-GAIN_sched_vels(1)) )...
        *(GAIN_sched_vels(k)-GAIN_sched_vels(1));
        eval( [ TestMatrix{i,k},'(j)=val' ] )
        end
```

```
    end                                                                    51
end                                                                        52
%-------------------------------------------------------------------------  53
%uncomment below to print out values for a1 to a25                         54
% clc                                                                      55
% for i=1:5                                                                56
%     for j=1:5                                                            57
%         eval(TestMatrix{i,j})                                            58
%     end                                                                  59
% end                                                                      60
%-------------------------------------------------------------------------  61
% function return variable                                                 62
                                                                           63
MatrixOut =  {a1  a2  a3  a4 a5;                                           64
              a6  a7  a8 a9 a10;                                           65
              a11 a12 a13 a14 a15;                                         66
              a16 a17 a18 a19 a20;                                         67
              a21 a22 a23 a24 a25};                                        68
%-------------------------------------------------------------------------  69
```

C. Engine key parameters and corresponding model variables

Table C.1.

model parameter	simulation variable
LP shaft speed	N1_rpm
HP shaft speed	N2_rpm
Fan inlet pressure	P0
Fan inlet temperature	T0
Total thrust	TOTAL_THRUST
Duct thrust	DUCT_THRUST
Duct airflow velocity	BYPASS_Vel
Core thrust	CORE_THRUST
Core airflow velocity	BYPASS_Vel1
Fan duct efficiency	FanDuctEfficiency
Fan duct discharge temperature	see Bypass temperature
Fan duct massflow (at fan exit)	FanDuctMassflow
Fan duct power	FanDuctPower
Fan duct pressure ratio	FAND_Y_PRs

Continued on next page

– continued from previous page

model parameter	simulation variable
Bypass massflow (after bleeds injection)	BYPASS_W
Bypass temperature	BYPASS_T
Bypass pressure	BYPASS_P
Fan power (total)	FAN_POWER
Fan core efficiency	FanCoreEfficiency
Fan core discharge temperature	T2
Fan core massflow	W2
Fan core power	FanCorePower
Fan core pressure ratio	FANC_Y_PRs
LPC efficiency	BoosterEta
LPC inlet pressure	P2
LPC inlet temperature	T2
LPC inlet massflow	W2
LPC discharge pressure	P23
LPC discharge temperature	T23
LPC discharge massflow	W23
HPC efficiency	CompEta
HPC inlet pressure	P25
HPC inlet temperature	T25
HPC inlet massflow	W25
HPC discharge pressure	P3
HPC discharge temperature	T3
HPC discharge massflow	W3

Continued on next page

– continued from previous page

model parameter	simulation variable
Combustor exit pressure	P4
Combustor exit temperature	T4
Combustor exit massflow	W4
HPT efficiency	HPT_Eta
HPT inlet	P41
HPT inlet	T41
HPT inlet	W41
HPT exit	P42
HPT exit	T42
HPT exit	W42
LPT efficiency	LPT_Eta
LPT inlet	P48
LPT inlet	T48
LPT inlet	W48
LPT exit	P5
LPT exit	T5
LPT exit	W5
Fuel flow	Fuel_Flow
Fan power (total)	FAN_POWER
Fan duct power	FanDuctPower
Fan core power	FanCorePower
LPC power	LPC_POWER
HPC power	HPC_POWER

Continued on next page

– continued from previous page

model parameter	simulation variable
HPT power	HPT_POWER
LPT power	LPT_POWER
VSV position degrees	VSV_DEG
VBV position degrees	VBV_DEG
VBV percentage extracted	VBV_W
HPC cooling flow	HPC_Cooling_W
LP shaft speed	Reg1
Max HP shaft speed	Reg2
LP spool accel rate	Reg3
HP spool accel rate	Reg4
Max EGT	Reg5
Max PR over HPC	Reg6
Max fuel	Reg7
Max fuel rate of increase	Reg8
LP shaft min speed	Reg9
HP shaft min speed	Reg10
LP shaft decel rate	Reg11
HP shaft decel rate	Reg12
Min fuel	Reg13
Min PR over HPC	Reg14
Max fuel rate of decrease	Reg15

Bibliography

[1] Qusai Z. Al-Hamdan and Munzer S. Y. Ebaid. Modeling and simulation of a gas turbine engine for power generation. *Journal of Engineering for Gas Turbines and Power*, 128:302–311, April 2006.

[2] *Alstom GTX100 dynamic simulation program.* Alstom.

[3] R. Douglas Archer and Maido Saarlas. *An Introduction to Aerospace Propulsion.* Prentice Hall, New Jersey, 1996.

[4] S. Borguet, V. Kelner, and O. Leonard. Cycle optimisation of a turbine engine: an approach based on genetic algorithms. Technical report, University of Liege, Aerospace and Mechanical Engineering Department, 2006.

[5] S. M. Camporeale, B. Fortunato, and A. Dumas. Dynamic modeling and control of regenerative gas turbines. *98-GT-172*, 1998.

[6] S. M. Camporeale, M. Agresti, and B. Fortunato. An object-oriented program for the dynamic simulation of gas turbines. *ASME 2000-GT-42*, 2000.

[7] S. M. Camporeale, L. Dambrosio, and B. Fortunato. One-step ahead adaptive control for gas turbine power plants. *Journal of Dynamic Systems, Measurement, and Control*, 124:341 – 348, 2002.

[8] S. M. Camporeale, B. Fortunato, and M. Mastrovito. A modular code for real time dynamic simulation of gas turbines in simulink. *Transactions of the ASME*, 128:506 – 514, 2006.

[9] Vittorio Cortellessa, Bojan Cukic, Diego Del Gobbo, Ali Mili, and Marcello Napolitano. Certifying adaptive flight control software. In *Proceedings, ISACC 2000: The Software Risk Management Conference*, 2000.

[10] N. A. Cumpsty. *Compressor aerodynamics*. Longman Scientific, 1989.

[11] S. Garg. A simplified scheme for scheduling multivariable controllers and its application to a turbofan engine. In *Paper 96-GT- 104, presented at the ASME International Gas Turbine and Aeroengine Congress & Exhibition, Birmingham, UK*, June 1996.

[12] Zane D. Gastineau and Gemunu Happawana. Robust model-based control for jet engines. In *34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*. American Institute of Aeronautics and Astronautics, 28-30 June 1998 1998.

[13] L. N. Hannett and Afzal Khan. Combustion turbine dynamic model validation from tests. *IEEE transactions on Power Systems*, 8(1):152 – 158, 1993.

[14] D. Henrion, L. Reberga, J. Bernussou, and F. Vary. Linearisation and identification of aircraft turbofan engine models. In *Proceedings of the IFAC Symposium on Automatic Control in Aerospace, June 14-18, 2004, St Petersburg, Russia.* LAAS-CNRS Research Report No. 03432, October 2003.

[15] F.P. Incropera and D.P. Dewitt. *Fundamentals of Heat and Mass Transfer*. John Wiley and Sons, 1990.

[16] J. H. Kim, T.W. Song, T.S. Kim, and S.T. Ro. Model development and simulation of transient behavior of heavy duty gas turbines. *Journal of Engineering for Gas Turbines and Power*, 123:589–594, 2001.

[17] A. Kreiner and K. Lietzau. The use of onboard real-time models for jet engine control. Technical report, MTU Aero Engines, Germany, 2001.

[18] Rainer Kurz. Gas turbine performance. In *Proceedings of the 34th turbomachinery symposium*, 2005.

[19] Joachim Kurzke. *GasTurb Software*.

[20] Joachim Kurzke. Model based gas turbine corrections. In *Proceedings of ASME Turbo Expo 2003*, Atlanta, Georgia, USA, 16 - 19 June 2003.

[21] Joachim Kurzke and Claus Riegler. A new compressor map scaling procedure for preliminary conceptual design of gas turbines. In *Proceedings of ASME IGTI Turbo Expo 2000*, Munich, 8 - 11 May 2000.

[22] L. J. Larkin and J. Philpott. Implementation of a multivariable control. *Transactions of the ASME*, 126:472 – 474, 2004.

[23] J. S. Litt, Khary I. Parker, and Santanu Chatterjee. Adaptive gas turbine engine control for deterioration compensation due to aging. In *16th International Symposium on Airbreathing Engines*, Cleveland, Ohio, 2003.

[24] Sonny Martin, Iain Wallace, and Declan G. Bates. Development and validation of an aero-engine simulation model for advanced controller design. In *American Control Conference*, pages 2334 – 2339, 11-13 June, 2008.

[25] Sonny Martin, Iain Wallace, and Declan G. Bates. Development and valida-

tion of an aero-engine simulation model for advanced controller design. *J. Eng. Gas Turbines Power*, Volume 130, Issue 5:051601(1–15), September 2008.

[26] Mathworks. *Simulink Manual*, 2005.

[27] W. F. McGreehan and M. J. Schotsch. Flow characteristics of long orifices with rotation and corner radiusing. *Journal of Turbomachinery*, 110:213–217, 1998.

[28] Netherlands National Aerospace Laboratory. *NLR GSP 10*, 10th edition, 2005.

[29] A. W. Pike. Development of the gtx100 gas turbine simulink dynamic model. Technical report, Alstom, November 2000.

[30] Ian Postlethwaite, Raza Samar, Byung-Wook Choi, and Da-Wei Gu. A digital multi-mode h-infinity controller for the spey turbofan engine. In *European Control Conference*, 1995.

[31] L. Reberga, D. Henrion, J. Bernussou, and F. Vary. Lpv modeling of a turbofan engine. In *Proceedings of the IFAC World Congress on Automatic Control, July 4-8, 2005, Prague, Czech Republic*. LAAS-CNRS Research Report No. 04506, October 2004.

[32] B. E. Ricketts. Modelling of a gas turbine: a precursor to adaptive control. *IEEE: Adaptive controllers in Practice (Colloquium)*, 1997.

[33] Claus Riegler, Michael Bauer, and Joachim Kurzke. Some aspects of modelling compressor behavior in gas turbine performance calculations. In *Proceedings of ASME IGTI Turbo Expo 2000*, Munich, 8 - 11 May 2000.

[34] W. I. Rowen. Simplified mathematical representations of heavy-duty gas turbines. *Transactions of the ASME Journal of Engineering for Power*, 105:865 – 869, 1983.

[35] Raza Samar, Ghassan Murad, Ian Postlethwaite, and Da-Wei Gu. A discrete time h-infinity observer-based controller and its application to an aero-engine. In *American Control Conference*, 1995.

[36] H.I.H. Saravanamuttoo, H. Cohen, and G.F.C. Rogers. *Gas Turbine Theory*. Longman, 4th edition, 1996.

[37] Q. Song and M. J. Grimble. Design of a multivariable neural controller and its application to gas turbines. In *American Control Conference*, Seattle, 1995.

[38] A. H. Spang and H. Brown. Control of jet engines. *Control Engineering Practice*, 7:1043–1059, 1999.

[39] Philip Thomas. *Simulation of Industrial Processes for Control Engineers*. Butter-Heinemann, 1999.

[40] W. P. J. Visser, O. Kogenhop, and M. Oostveen. A generic approach for gas turbine adaptive modeling. *Journal of Engineering for Gas Turbines and Power*, 128:13 – 19, 2006.

[41] I. Wallace. Private comunication., 2009.

[42] P.P Walsh and P. Fletcher. *Gas Turbine Performance*. Blackwell Science, 1st edition, 1998.

[43] J. W. Watts, T. E. Dwan, and C. G. Brockus. Optimal state-space control of a gas turbine engine. *Journal of Engineering for Gas Turbines and Power*, 114:763 – 767, 1992.

[44] S.R. Watts and S. Garg. A comparison of multivariable control design techniques for a turbofan engine control. In *ASME 40th Gas Turbine and Aeroengine Congress and Exposition, Houston, TX*, June 1995.

[45] J.R. Welty, E.C. Wicks, R.E. Wilson, and G. Rorrer. *Fundamentals of momentum, heat, and mass transfer*. John Wiley and Sons, 2001.