Doctoral thesis for the degree of Doctor of Philosophy

Development of methods for analysis and optimization of complex jet engine systems

By

Tomas Grönstedt

Department of Thermo- and Fluid Dynamics
Chalmers University of Technology
SE-412 96 Göteborg, Sweden

Göteborg, 2000

ISBN: 91-7197-910-7 ISSN: 0346-718X
Avhandlingsnummer:1595
Tryck: Vasastadens bokbinderi

Göteborg, 2000

It is a miracle that curiosity survives formal education

Albert Einstein

This thesis is based on the following work:

Paper 1:

Grönstedt U. T. J., Johansson T. and Håll U., **"The Optimization of a Seven Stage Compressor"** , International Symposium on Fluid Machinery and Fluid Engineering, Sept. 9-12, 1996, Beijing, China.

Paper 2:

Grönstedt U. T. J., Håll U., **"Mission Dependent Optimization of Advanced Fighter Engines"**, XIII International Symposium on Air Breathing Engine, Sept. 7-12, 1997, Chattanooga, Tennessee, U.S.A.

Paper 3:

Grönstedt U. T. J. **"Advanced Solvers for General High Performance Transient Gas Turbine Simulation Tools"** , 14th International Symposium on Air Breathing Engines, Sept. 5-12, 1999, Florence, Italy.

Paper 4:

Grönstedt U. T. J., Pilidis P., **"Control Optimization of the Transient Performance of the Selective Bleed Variable Cycle Engine During Mode Transition"** , Recommended for publication in the Journal of Engineering for Gas Turbines and Power.

The development of GESTPAN:

A new simulation tool, GESTPAN (GEneral Stationary and Transient Propulsion ANalysis) has been developed as part of this project. Some of the progress relevant for this work has been included in the thesis.

Development of methods for analysis and optimization of complex jet engine systems

by

Tomas Grönstedt

thgr@tfd.chalmers.se
http://www.tfd.chalmers.se/~thgr
Department of Thermo- and Fluid Dynamics
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden

Abstract

This thesis describes the development of GESTPAN (GEneral Stationary and Transient Propulsion ANalysis), a generalized system for the design, steady-state and transient simulation of gas turbine systems. Some of the main achievements in the thesis are related to the development of new algorithms or integration of existing numerics tailored to simplify the structure and use of generalized gas turbine simulation systems. In particular, a method for performing system design utilizing the analysis equations, i.e. an inverse design method, has been developed. Furthermore, attention is drawn to a number of advantages of using an implicit high order differential algebraic system solver for transient gas turbine system analysis.

The simulation studies carried out with the GESTPAN system have focused on the performance optimization of the Selective Bleed variable cycle engine. In particular, a method for controlling the engine during mode transition was developed. Work with the implementation of a hybridized optimization method suitable for mission optimization of variable cycle engines is also described. The method couples the cycle selection and the control optimization of the engine variable geometry. Simulations performed with the method indicate that previously published designs of the Selective Bleed Variable cycle engine can be downsized considerably.

Early work carried out in the research project concentrated on developing a method for optimizing the performance of variable geometry compressors integrated in gas turbine systems. Although the method was limited to subsonic operation of compressors, it was successfully used to simulate the core driven fan stage of the double bypass variable cycle engine.

Acknowledgments

This work was carried out at the Department of Thermo- and Fluid Dynamics at Chalmers University of Technology. The project is funded by NFFP (the National Flight Research Program). It is part of a cooperative project between Chalmers University of Technology, Volvo Aero Corporation and the Royal Institute of Technology.

I would like to express my gratitude to Professor Ulf Håll for being a constant source of inspiration and for many fruitful discussions of scientific matters, and to Thomas Johansson for his contributions to the first paper in this thesis. Furthermore, I would like to thank Anders Lundbladh for initiating and continuously contributing to the development of GESTPAN and in particular for inspiring me to learn more about numerical analysis. I would also like to thank Björn Ryberg and Mattias Petterson for struggling with me on the coding and validation of GESTPAN.

Finally, I would like to thank my colleagues at the Department of Thermo- and Fluid Dynamics and at Volvo Aero Corporation for creating a very stimulating working atmosphere.

Nomenclature

A	Cross sectional area
ASFC	time Averaged Specific Fuel Consumption
ATF	Advanced Tactical Fighter
bpr	Bypass Ratio
CDFS	Core Driven Fan Stage
C_d	Flow coefficient
C_v	Thrust coefficient
EGV	Exit Guide Vane
F	Thrust
fa	Fuel air ratio
FPR	Fan Pressure Ratio
G	Torque
GE	General Electric
GESTPAN	GEneral Stationary and Transient Propulsion ANalysis
h	Stagnation enthalpy
H	Altitude
HPC	High Pressure Compressor
HPT	High Pressure Turbine
I	Moment of Inertia
ICV	Intercomponent Volume Method
IGV	Inlet Guide Vane
IPC	Intermediate Pressure Compressor
JSF	Joint Strike Fighter
K	Auxiliary parameter
LPC	Low Pressure Compressor
LPT	Low Pressure Turbine
m	Mass flow
m_0	Jet engine air mass flow
m_f	Jet engine fuel mass flow
M	Mach number
n	Rotational speed
OPR	Overall Pressure Ratio
P	Stagnation pressure
P_s	Static pressure
PLF	Pressure Loss Factor
P&W	Pratt and Whitney
Q_f	Caloric heat value of fuel
R	Gas constant
SCAR	Supersonic Cruise Aircraft Research program
SFC	Specific Fuel Consumption
SLS	Sea Level Static
SST	Supersonic Transport program
SCAT	Status Cycle Analysis of Test
T	Stagnation temperature
T_s	Static temperature
TIT	Turbine Inlet Temperature

VCE	Variable Cycle Engine
VABI	Variable Area Bypass Injector
VATN	Variable Area Turbine
v	Intercomponent volume
V_0	Flight velocity
V_j	Jet velocity
W	Power
W_{out}	Engine power output

Greek symbols

ΔT_{isa}	Deviation from ISA standard
β	Fuel schedule factor
η_p	Propulsive efficiency
η_{th}	Thermal efficiency
η_o	Overall efficiency
η_m	Mechanical efficiency of shaft
η	Isentropic efficiency
η_∞	Polytropic efficiency
γ	Specific heat ratio
π	Pressure ratio
τ	Temperature ratio
Π_{rec}	Pressure recovery factor
σ	reaction rate parameter
ω	Pressure loss coefficient
X	Mass flow parameter

Subscripts

1	Inlet to component
2	Exit from component
dp	Design point
bb	Backbone point
rp	Reference point
a	Ambient Conditions
∞	Ambient Conditions

Superscript

$\sim$	Corrected entity

Wiring diagram nomenclature

pa	Ambient Pressure

Contents

Chapter 1

Engine system simulations

Traditional performance analysis of gas turbine systems has come to involve three elementary tasks:

1. Design
2. Off-design analysis
3. Transient analysis

on which a variety of more complex studies can be based. Engine **design** can be used for "paper engine" studies or to predict the performance of derivatives of existing engines (this is a mix of design and off-design analysis). **Off-design** studies can be used for supplementing performance data for existing engines for which only a limited amount of information is available or to assess the nature and magnitude of effects of changes in engine component characteristics on system performance. Furthermore, off-design models can be used for integration into tools for engine diagnostics, engine degradation studies, analysis tools for engine performance rig testing, analysis of data collected during flight, customer decks (models provided to airframers and end users) etc. **Transient** studies can be used to ensure safe and stable operation early in the design phase and to carry out the design of the control system.

Engine performance modeling studies play a central role in almost all parts of the engine life cycle, which is illustrated in Figure 1.1. Thus, a simulation tool that is able to perform all three elementary tasks can be used for a multitude of purposes, in particular to reduce turn around times of the engine development process. Such a tool would also be ideally suited to act as a catalyst for a transfer of engineering competence between the different processes in the engine life cycle. A key factor for its success is the extent to which the system complexity can be kept at a manageable level. Ultimately, the tool must begin to supplant the specialised systems in order to remain economically viable.

This thesis describes the development of a new simulation system, GESTPAN (GEneral Stationary Transient Propulsion ANalysis), that allows the user to perform the three elementary tasks stated above. The numerics and the algorithms of the tool have been tailored to keep the complexity of its use and maintenance at a minimum.

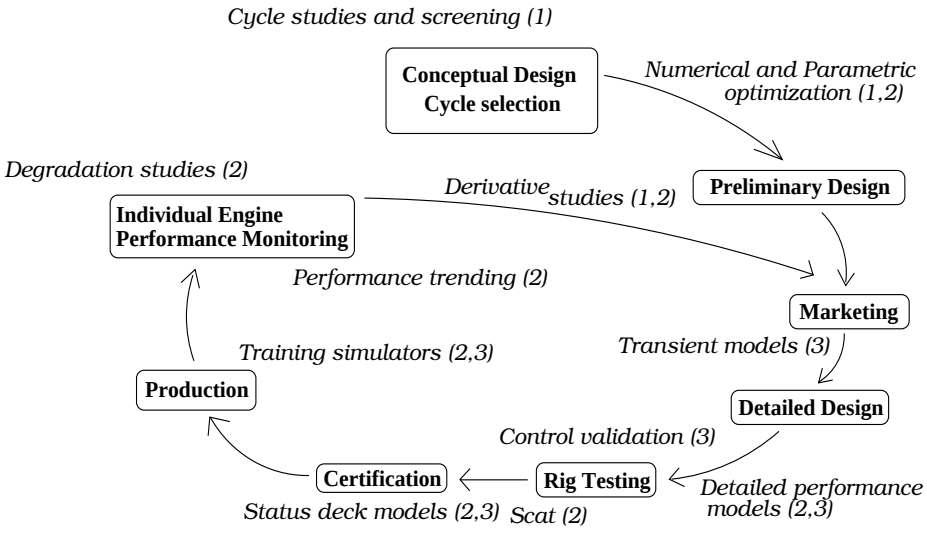


Figure 1.1: Performance modeling during the engine life cycle (based on [1]). The figures correspond to the three elementary tasks defined above.

1.1 Generalized performance simulation tools

Even if a simulation system is well documented and structured in a way that makes it easy to modify, reprogramming of the system will almost always require more training and experience of the user than modifying input files or interacting with some kind of user interface. It is therefore considered a design requirement for a powerful engine system simulation tool that the user is provided the functionality of assembling arbitrary engine configurations through input files or through a user interface. In this thesis, a system with this capability and the ability to perform all three of the elementary tasks above is called a **generalized** performance tool. Without the transient feature, such a system is called a **static generalized** performance tool. The term **semi-generalized** performance tool refers to a system for which new engine configurations can be modeled with limited reprogramming. Codes only intended to model a specific engine are called **engine specific** performance tools.

1.2 Generalized tools - a brief review

Numerous performance tools have been developed over the years, and simulation results as well as descriptions of the implementations are available in public literature. Here, an attempt is made to review only the most notable and powerful semi-generalized and generalized performance tools. It should be pointed out that a large amount of the work carried out in this field has been kept confidential by engine companies.

At the beginning of the 1970s, research groups at NASA Lewis experienced an increasing need for a generalized gas turbine simulation tool [2], mainly because the cost for developing and validating a new program for every engine

studied was rapidly increasing. It was also becoming computationally possible to develop and use such a system. The first static semi-generalized code tested on several engine cycles, GENENG/GENENG 2, was developed by Fishbach and Koening [3, 4]. The GENENG 2 program provided the user the ability to simulate nine different engine cycles. Sub-derivatives of these configurations could be obtained by the use of logical flags, and up to six modes of operation could be defined for an engine model. The latter feature made the system very suitable for simulations of variable cycle engines.

The GENENG codes were developed further into a static generalized code, NNEP (the Navy NASA Engine Program), by Fishbach and Caddy [5], allowing the user to freely assemble virtually any engine configuration by means of input files. A number of additional features have been added to the NNEP code over the years, such as engine weight analysis, estimations of installation losses and features for simulating chemical equilibrium compositions for calculating effects of dissociation or exotic fuels. Another derivative of the GENENG codes, DYNGEN [6], was given transient capabilities, thus being a semi-generalized tool. DYNGEN was generally considered difficult to modify and it was difficult to scale the model equations to reflect real engines [2]. A hybrid computer version of the code, HYDES [7], was developed to overcome these deficiencies. Another fully digital transient code, DIGTEM, was made even simpler to modify by improving the modularisation of the software design [2].

Work to obtain a fully generalized engine model has recently been initiated at NASA Lewis Research Center. The National Cycle Program (NCP) is currently being developed to provide the architectural framework for the Numerical Propulsion System Simulation (NPSS) project [1, 8]. The NCP program will provide all the functionality available in traditional performance tools, thus inheriting the features available in NNEP as well as the framework needed for novel NPSS features, such as coupling of multidisciplinary tools at various levels of detail. The NCP is coded in C++, providing an object-oriented platform for coupling to other codes through the use of CORBA (Common Object Request Broker Architecture). CORBA was designed to make it possible for applications to communicate with each other, regardless of where they are located (local or network), the language in which they were developed or the operating system in which they are run. For example, an engine system analysis code written in C++ for a PC environment could communicate with a CFD model developed in Fortran 77 running on a multiprocessor workstation. The industry savings through increased engineering productivity by the use of NCP is estimated at $50 millon per year [9]. A visionary operation of the NCP system is shown in Figure 1.2.

Another organisation with a strong tradition in performance modeling is Cranfield University of Technology. In 1974, the work of MacMillan [10] resulted in a static generalized performance tool called TURBOMATCH. Although it did not provide all the features included in the NNEP code, such as the multimode feature, the functionalities for assembling almost any kind of gas turbine system were similar. A derivative of the TURBOMATCH code, TURBOTRANS [11], was in fact a fully generalized code completed as early as 1982. Both codes will be discussed further in this thesis.

Recently, some interesting work has been done in developing user-friendly simulation tools with Graphical User Interfaces (GUI:s). A very thorough semi-generalized system offering the user a multitude of features was developed by

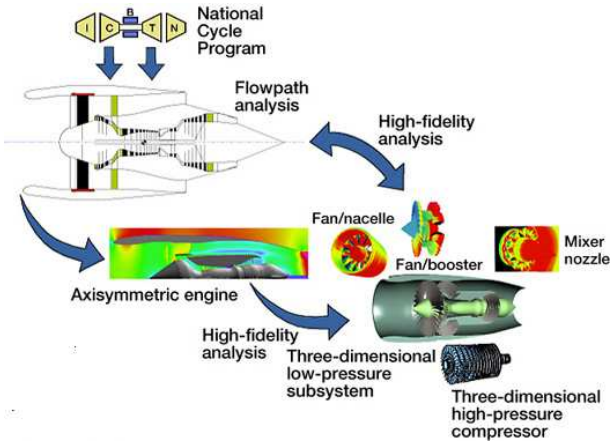


Figure 1.2: Visionary operation of NCP [9]

Kurzke [12]. Another GUI based fully generalized gas turbine tool more interesting from a software design perspective originated from the work of Reed and Abdollah [13, 14]. They developed an all Java based object-oriented system providing the user with the unique advantages of the language: platform independency and support for interactive web versions. Recent work on a more advanced successor to the system, Onyx, is described in [15, 16]. The Onyx system implements a number of interesting functionalities such the ability to plug new engine component models into the framework and have them interoperate without modification to the system.

Around the same time as NNEP and TURBOMATCH were being completed, a static generalized simulation tool, ta45 [17], was developed at Volvo Aero Corporation. This code was the starting point for the design of the GESTPAN system.

1.3 Generalized tools - lessons learned

Since semi-generalized systems need reprogramming to be able to model new configurations, these additions are usually retained in the code as they are added and then made accessible to future users. Thus, the functionality of semi-generalized codes approaches the functionality of generalized codes during their life cycle. It may seem that the effort of developing a fully generalized program would not be worthwhile, but one can actually make a very strong argument for this effort. Even if a semi-generalized system supports a majority of the engine configurations that have ever been in use or have been suggested for use, a detailed engine model still tends to require reprogramming of the system. For instance, some rare component interactions might be too important to neglect for a particular engine. Also, detailed models of control systems tend to have unique features. It is the belief of the author that the difficulty and the frequent need to modify this class of codes is an underlying reason for the fact that the semi-generalized performance systems have found their use mainly

in preliminary studies of new engines or engine derivatives. For the level of accuracy necessary for these kinds of studies, it is often sufficient to modify input files of existing configurations.

On the other hand, generalized tools are likely to become even more complex and difficult to modify when this is eventually needed. The work carried out at the beginning of this project to introduce multimode operation [18, 19, 5] into ta45 illustrates this. It soon became clear to the author that the number of persons capable of carrying out this modification, without a major effort involved, was very limited (one [the same person that originally coded most of the system]). Another interesting observation indicating the difficulty of modifying generalized codes can be made by studying the modifications described in open reports of the NNEP simulation code. In a majority of the cases changes were introduced in cooperation or solely by L.H. Fishbach (one of the original developers of the system). A third example in this vein is that the use of the TURBOTRANS code, developed at Cranfield University of Technology, was discontinued after the retirement of Dr. J. R. Palmer.

The proper interpretation of these three examples is of course not that the first generation of generalized codes was impossible to modify unless the original programmer was available, but that the effort needed to perform the work normally limited the number of capable/interested engineers to a small group of specialists. This group had often been part of the initial development of the system, and new players infrequently entered the scene. The author himself spent approximately a year penetrating the ta45 code before he could modify it freely. Even after this initial period, the modifications of the system necessary to provide transient capabilities were considered very extensive and it was decided that it would be more meaningful to integrate the old functionalities of ta45 into an entirely new system. This would then allow the full use of a number of novel features included in the Fortran 90 standard.

A final remark about the first generation of generalized gas turbine codes is that those which were kept in use were done so for a remarkably long time. TURBOMATCH, NNEP and ta45 were all used for more than 20 years, surviving a great number of platforms and operating systems and, although the number of engineers capable of modifying the codes was very limited indeed, the number and variety of the users was most often not.

1.4 The next generation of generalized tools

It seems clear that the design of the first generalized software systems neglected maintainability and the ease by which the system could be modified. This is of course at least partially due to the facts that computing resources at that time were very modest and that the generalized codes required absolutely state of the art computers for execution. Today, transient simulations can be executed faster than real time. The author has demonstrated transient simulations of a nonlinear model of a mixed-stream turbofan engine using 14 states and six coupled equations executing five times faster than real time. The simulation was run on a Digital Alpha Workstation 433 au. Steady state simulations can be carried out in fractions of a second. Thus, when developing a new generalized performance system, the programmer can now concern himself less about performance and focus on obtaining a clear and modifiable structure of the code.

Traditional performance tools formulate the design problem in such a way that the equations are eliminated. The price for this simplification is that all engine component models have two different algorithms, one for design and one for off-design problems. Consequently, any change in modeling must be implemented in both versions of the module equations to maintain a consistent set of equations. One of the major achievements in this thesis is the development of a new methodology for inverse design, that allows the use of the *same algorithms* for design, off-design and transient simulations. The inverse design functionality is obtained by simultaneously solving the design and off-design equations, producing equation systems with roughly twice as many unknowns as those used for off-design problems. Thus, the new design methodology makes full use of the present computer resources to reduce the complexity of the generalized performance tool.

Another aspect that allows for obtaining superior systems, compared with the first generation of generalized codes, is that the programming languages have evolved quite considerably over the last 20 years. Most notably the revolution of software development based on the Object-Oriented Programming paradigm [20, 21], allows the software developers to produce systems much less complex to maintain and modify.

GESTPAN was developed using a pseudo object-oriented approach. The Fortran 90 module concept allows construction of static classes, which makes "object-oriented like" encapsulation of data and routines possible [22]. The beauty of multiple instantiation of objects and association of pertinent data is unfortunately not possible in Fortran 90. For instance, if a Fortran 90 module is used to contain code for a compressor and two compressor modules are used for the engine model (e.g. fan and hpc), either the module itself or external code has to keep track of which module is currently being executed. An object-oriented language could have "manufactured" two instances of the compressor class dynamically, and data specific to the fan and the hpc could be kept separated by the modules in a very intuitive and straightforward fashion. Such a feature would make the implementation of the routines connecting the engine components considerably less complex.

1.5 Gas turbine simulation in commercial codes

A number of commercial simulation tools are available for developing system models, such as [23, 24, 25]. These systems usually offer the user a very powerful GUI for model development and analysis. However, evaluations of the MatrixX system [24] carried out in an early stage of this project showed that, although the system had advanced well in the field of dynamic modeling and control design, it was not very suitable for carrying out engine system design.

Chapter 4 explains why the methodology used for solving differential algebraic systems in the latest version of the SIMULINK system is not very efficient. For the gas turbine test cases studied in Chapter 4, a loss of about an order of magnitude in computational speed is to be expected. The superior performance of the GESTPAN solution procedure is achieved by solving the differential and algebraic equations simultaneously [26, 27].

Other shortcomings were experienced by Reed and Abdollah when they worked with the AVS system [25, 28]. For example, the transfer of data be-

tween the separate UNIX processes used to simulate different components in the engine model was found to be very slow [28].

Although weaknesses can be found in any commercial tool for system simulations, one should note that these systems are continuously being improved, e.g. the data transfer performance problem of the AVS system was solved in later a version. Similarly, the option of rewriting SIMULINK to implement the direct approach for solving differential algebraic systems is presently being discussed [29]. When making these observations, one should also keep in mind, that as costs for developing new systems to the same level of sophistication as the commercial systems increase, licence fees for the commercial systems tend to increase as well. Furthermore, limited access to the source code can often be a decisive obstacle in the development process, especially when non-standard tasks are being performed.

Chapter 2

Generalized performance tools - inverse design

The first section of this chapter briefly explains how design and off-design simulations are carried out in traditional performance tools. The material is intentionally kept short and the interested reader is advised to study some of the many detailed and excellent texts on this topic [30, 31, 32]. In the following sections, attention is drawn to a number of negative aspects with the traditional approach and a new algorithm for inverse design of nonlinear gas turbine systems is developed. The methodology is tested and demonstrated on a conventional mixed-stream turbofan engine and two variable cycle engines.

2.1 Conventional design/off-design methodology

The traditional approach for the implementation of gas turbine simulation systems has been to use two different modes in the simulation code:

1. Design mode
2. Off-design mode

For the design mode, the governing equations are put in a form that allows the user to specify parameters that can easily be estimated at the preliminary stage of engine design. These parameters are usually obtained from estimates on technology levels or projected technology levels. Typical parameters are compressor and turbine pressure ratios and efficiencies, pressure losses in burners and nozzles, C_d:s and C_v:s of nozzles, rotational speeds, turbine inlet temperatures, Mach numbers in the mixer etc. These parameters are then used to compute other, more suitable parameters for off-design iteration, such as pressure loss coefficients, design power requirements, burner reaction rate parameters, turbine areas, mixer areas, nozzle areas etc.

The governing equations relating the interaction between the engine components are derived from a number of compatibility requirements [33]:

- Compatibility of work
- Compatibility of flow

- Compatibility of rotational speed

For a general engine configuration, these assumptions can be formulated as a system of n nonlinear equations in n unknowns, i.e.:

$$\begin{aligned} f_1(x_1, x_2, \cdots, x_n) &= 0 \\ f_2(x_1, x_2, \cdots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, x_2, \cdots, x_n) &= 0 \end{aligned} \tag{2.1}$$

where $x_1, x_2, ..., x_n$ are iteration variables and $f_1, f_2, ..., f_n$ are corresponding residuals to be iterated to zero. In this thesis, the system represented by equation (2.1) is referred to as the compatibility or off-design equations.

2.1.1 Traditional design mode methodology

The standard design mode technique used in traditional performance tools starts with the inlet and proceeds in the flow direction through the engine, using data from the already designed components as they become available. This usage of upstream data makes it possible to automatically satisfy equation (2.1). For instance, the compressor in a turbojet engine will be designed with a certain power requirement, and the turbine placed on the same shaft will then automatically be designed for this requirement. The mass flow can be treated in the same "hyperbolic manner" (sequentially proceeding from upstream components to downstream). All engine components will be sized, that is design areas will be computed to accommodate the mass flow exiting the upstream component. By proceeding in this manner, the compatibility equations (2.1) can be satisfied automatically and no iteration will be required.

2.1.2 Traditional off-design mode methodology

In the off-design mode, a number of residuals are formed using the compatibility equations (2.1), as well as some parameters computed during the design process. The 1-D compressible continuity equation, i.e. equation 2.2, is frequently used to form some of the residuals.

$$\frac{m\sqrt{RT}}{c_d A P} = \sqrt{\gamma} M \left(1 + \frac{\gamma - 1}{2} M^2\right)^{-\frac{\gamma+1}{2(\gamma-1)}} \tag{2.2}$$

An equal number of suitable iteration variables is selected, and equation (2.1) is then solved.

For a specified set of values on the selected iteration variables, all other thermodynamic variables can readily be obtained. Thus values for m, T, P, M, c_d, γ and R will be available and the cross-sectional area can be computed using equation 2.2. This area will then be compared to the design area, one of the parameters computed during the design process, and a residual can then be formed. The solution is normally found by employing a Newton or quasi-Newton algorithm. If the solver converges, continuously improved values for the iteration variables are selected, proceeding towards the solution of the system. See Appendix B for a general description on Newton and quasi-Newton methods.

The size of the equation system (2.1) is dependent on how the algorithms of the engine component models are arranged. For instance, to model a turbojet engine, the ta45 program iterates in the compressor pressure ratio and the engine mass flow, i.e. x_1, x_2 in equation (2.1), and two corresponding residuals (f_1, f_2) are formed by comparing computed turbine and nozzle "flow areas" with design areas. The GESTPAN code, on the other hand, iterates in compressor and turbine exit pressures together with rotational speed, balancing compressor and turbine power, burner and turbine mass flow and matching nozzle areas with design areas. This formulation is more suitable for modules to be used both in a stationary and a transient environment.

2.2 The traditional methodology - Downsides

A consequence of the "twin-mode" approach is that the engine component implementations have to incorporate separate algorithms for the design and the off-design processes. This makes the system considerably more complex. Furthermore, modifications in the physical modeling must affect both the design and the off-design equations in a consistent manner. If the system will be used for more refined engine modeling, e.g. detailed models of the control system, secondary air systems, heat transfer effects and so on, the maintenance of two sets of equations becomes increasingly complex and cumbersome.

Another important issue in the design of a generalized tool is to decide on a strategy for keeping track of iteration variables and residuals. The method used in ta45 was to associate iteration and error variables to the engine components in a pre-determined way. Components normally located in the forward parts of jet engines, such as inlets and compressors, would be designated iteration variables, and components typically located in the rear end would be given residuals. A rule based system would then guide the user to assemble engines with the same number of variables and residuals. Although a seemingly elegant method, this approach makes the system more difficult to modify.

The ta45 program used a preprocessing program called ta44 to analyse an engine configuration file. One of the tasks was to compute the number of iteration variables and error variables in the model specified by the user and to make sure that the number of equations and residuals matched. If a user wanted to add a new engine component model to the system, it would then be necessary to also make changes to the ta44 code. This required a deeper understanding of the implementation of the system.

In some component models, this pre-determined methodology for handling the iteration variables would actually give rise to "three-mode" implementations. If two compressors were placed on the same shaft, they could not both be given an iteration variable since the pressure ratio iteration of the first component would determine a rotational speed, and this rotational speed would then set the pressure ratio over the next compressor. Thus, the off-design internal algorithm in the compressor implementation had to be separated into two different cases with either pressure ratio or rotational speed as input.

The next section describes an algorithm that allows routines external to the component models to keep track of iteration and error variables. This feature can be programmed into the system once and for all, and the user would have to make no extra provision for managing error and iteration variables when

new engine components are added. The subsequent section describes an inverse design methodology which markedly reduces the complexity of the component implementations. The method solves the design equations and the compatibility equations simultaneously, still allowing the user to specify typical gas turbine design parameters.

2.3 An overview of the structure of GESTPAN

2.3.1 User interfaces

The overall structure of GESTPAN is presented in Figure 2.1. Depending on the task to be performed, GESTPAN offers the user two versions of the program with different interfaces. The Graphical User Interface (GUI [top left in Figure 2.1]) version of GESTPAN is intended to be used by engineers interested mainly in a particular engine using an already developed model. The GUI, which is implemented in JAVA 2 [34], communicates with the rest of the simulation system via a socket protocoll [35]. All the other parts of the GESTPAN program are coded in ANSI Fortran 90, except for a few routines responsible for the Fortran part of the socket communication. These routines were implemented using the Microsoft Foundation Classes available in the Digital Visual Fortran development environment [35]. Thus, when the GUI version of the program is run, two separate processes are executed simultaneously, i.e. the JAVA GUI process and the Fortran simulation process. Inter-process communication is carried out solely through the socket, as indicated in Figure 2.1.

The User Interface (UI [top right in Figure 2.1]) version of GESTPAN offers the user an interactive command line based environment for developing and analysing engine models. The UI is implemented entirely in ANSI Fortran 90. This version of GESTPAN has been compiled and run on Digital Alpha workstations, Silicon Graphics workstations and PC:s (NT 4 operating system). Although not as straightforward to work with as the GUI, the UI offers a much wider range of functionalities. Except for off-design and transient simulation capabilities, the UI also allows the user to carry out design (on three different levels of detail), control schedule optimization, cycle optimization, mission optimization and trajectory optimization. These features will be described in subsequent chapters.

2.3.2 Machine Interface and Engine Procedures

The Machine Interface is a Fortran 90 module [22] not directly accessible to the user. It contains a number of routines responsible for administrating computations, e.g. reading input files, preparing and invoking calls to numerical routines, calling routines in the connector module (described below) etc. These routines can be called from both the GUI and the UI. The Engine Procedures module is also used as an administrative layer between the user interfaces and the more elementary building blocks of GESTPAN. However, the Engine Procedures module is used only in the UI version of GESTPAN.

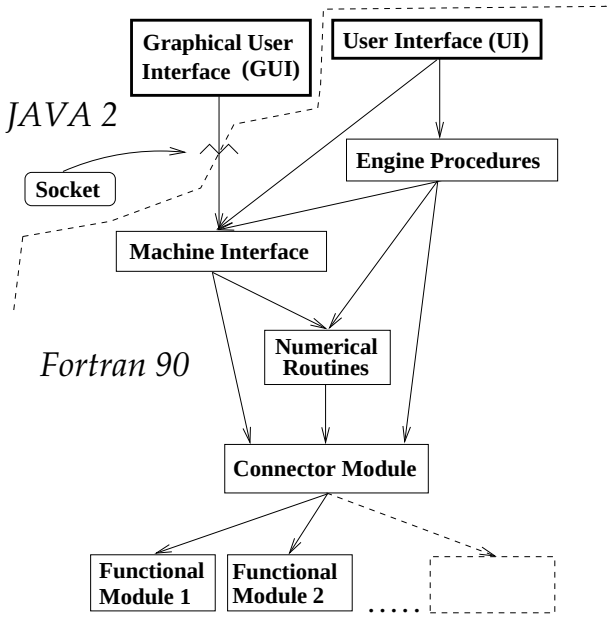


Figure 2.1: The overall structure of GESTPAN

2.3.3 Numerical Routines

The numerical routines integrated in GESTPAN offer functionalities for solving systems of nonlinear equations, Ordinary Differential Equations (ODE) as well as Ordinary Differential Algebraic Equations (ODAE). The quasi-Newton algorithm used to solve systems of nonlinear equations is described in this chapter and in Appendix B. The selection and characteristics of the routines implemented in GESTPAN for the solution of ODE and ODAE problems are described in Chapter 4. A number of routines for optimization have also been implemented in the GESTPAN system. These are described more closely in Chapter 5.

2.3.4 Connector Module

The connector module contains the most elementary level of administrative routines implemented in GESTPAN. Some examples of such functionalities are:

- Routines for determining the number of residuals and iteration variables of an engine model (see paragraph directly below).
- Routines for assembling information arrays of engine models through inquiry calls to the functional modules (see section 2.3.6 below).
- Routines for rebuilding engine model information arrays after mode switches in variable cycle engines (see section 6.4)

New engine configurations are assembled by means of input files. All connections between module inputs and outputs must be listed in the input files.

Functionalities available in the UI allow the configuration files to be built interactively, and many module connections can be made automatically by the system.

2.3.5 Detection of iteration variables and residuals

If a module input is neither connected nor given a value (fixed or defined through table interpolation), the connector module processing the input file will detect this undefined input and automatically make this variable an iteration variable. Another situation that will generate an iteration variable is that in which two module inputs are connected to each other. Likewise, if the connector module fails to sort the execution order of the modules in such a way that all inputs are available when executing a particular module, iteration variables will be introduced (there are of course cases when no such order exists). Residuals are generated when two module outputs are connected to each other or when residuals are defined within the modules.

This methodology implies that the iteration and error variables are defined by the connections of the model and possibly also by module internal residuals. Thus, when the user adds new modules to the system, no extra provision needs to be made for keeping track of error and iteration variables.

2.3.6 Functional modules

The approach for implementing the governing equations of the performance model has been to use one Fortran 90 module for each engine component. To reduce the development time and system complexity, all engine components were developed using a template file. Modules developed using this file are referred to as functional modules (see bottom of Figure 2.1).

One of the key design criteria for developing GESTPAN was to make the addition of new functional modules very simple. The use of a template file allowed a standardized calling interface between the module connector and the functional modules to be used. This made the addition of new modules particularly simple. To add a new functional module to the system, the user has to:

- Add a use statement in the connector module referencing the new module.
- Add a standardized calling interface in the connector module referencing the new module.
- Increase the integer parameter defining the number of modules in the connector module by one.

A graphical description of a subset of the variables present in the standardized interface is shown in Figure 2.2. The module interface contains a number of additional variables enabling transfer of variable names, module internal residuals, logical control flags, an error tolerance for module internal iterations, variables used for transient simulations, etc. The input and output signals, see Figure 2.2, are used to connect the modules when engine models are assembled. The real and integer parameters are used to specify the design of the components. During transient and off-design simulations, the real and integer

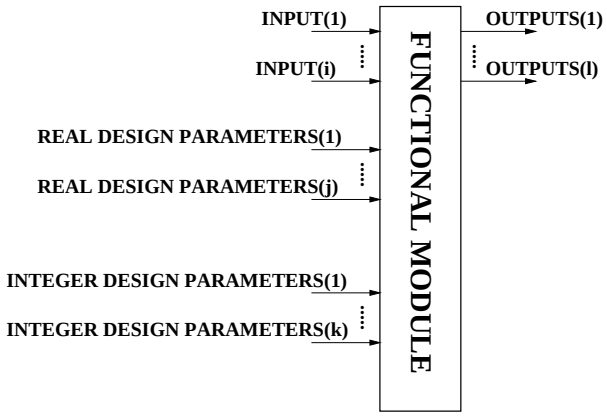


Figure 2.2: A subset of variables used in the standardized functional module interface.

parameters are fixed, but some of the real parameters are used as iteration variables during design.

2.4 Inverse design

2.4.1 Basic idea

The basic idea for carrying out inverse design is to use the off-design algorithm of the engine component equations for the design process. This means that some of the parameters used to define the design point in traditional gas turbine design codes may not be available as input to the model. However, these parameters can be computed as component outputs and be specified by forming module external residuals. For instance, although the fuel flow is used as an input to the burner module in the off-design formulation, the burner exit temperature is still available as an output from the module. Thus, it can be compared with an output specification given in an input file to form a design residual. To keep the number of iteration variables and residuals equal, the same number of real design parameters must be introduced as iteration variables. In the example above, the fuel flow would then be used as an iteration variable.

2.4.2 Requirements and approach

The principal requirements for the implementation of the design method were:

1. The design should be specified using typical design parameters readily obtainable at the design stage.

2. Three levels of design fidelity will be included: preliminary design, detailed preliminary design and fine-tuned design.

3. The user should only have to supply values for the parameters specifying the design.

To fulfill the requirements above, two different algorithms for system design were implemented:

- Direct design
- Robust design

Direct Design

The direct design method implements the basic idea for inverse design in a straightforward fashion. The iteration methodology is the same as that used for solving off-design systems, with the off-design equations completed with the design equations. The number of design equations are usually about the same or up to two times the number of the off-design equations, depending on whether features for "specifying off-design iteration variables" and "optimizing the design formulation" are used (see sections below). Keep in mind that traditional gas turbine design tools normally carry out design without any equations whatsoever.

To succeed, the algorithm generally requires starting estimates for the design and off-design iteration variables that are fairly close to the solution. It has been found to be quite difficult to obtain such estimates, requiring expert knowledge on the models included in GESTPAN. Even with these skills, the work is tedious and error prone. The direct design method would probably be very useful for design problems described by linear systems, with component maps not restricted to certain variable ranges.

The highest level of design fidelity is used for the direct method. This means that all real and integer design parameters must be specified by the user to be able to run the procedure. To model a mixed-stream turbofan, about 100 real parameters and 15 integer parameters would have to be set.

Robust design

The robust design method is used to automatically find good starting estimates for the simultaneous design and off-design equations. Thus, the method can be perceived as a preprocessing algorithm to the direct design method described above.

The basic idea in the robust design algorithm is to do one sweep through the engine, during which the off-design and design iteration variables are adjusted to minimize the residuals in the equation system. For the first modules in the execution order there is not a great deal of constraining information available, because most of the off-design residuals are formed by comparing two outputs from different modules. For instance, if a compressor module is being executed, no power balance error can be introduced since the turbine has not yet been executed. On the other hand, when the turbine is executed, the power balance is set, as are other off-design residuals. The normal situation is thus that modules executed early in the sweep will give rise to under-determined systems and modules executed late in the sweep will give rise to over-determined systems. This situation is addressed using an optimizer minimizing the residual. This can be done in both cases, for which the first leads to finding one out of infinitely many sets of iteration variables satisfying the equations and the latter to finding

a set that satisfies the equations as far as possible. The Nelder and Mead Simplex method [36] is used for the module optimizations.

A very favorable special case of the design problem can be obtained when many of the design residuals are specified for modules executed early in the execution order. Then, systems with an equal number of iteration variables and residuals, i.e. square systems, can be obtained. Similarly, this can make the typically over-determined modules occurring late in the execution order square as well. Thus, the design problem is then subdivided into small systems that can be solved in order, making the design process very fast indeed.

Figure 2.3 illustrates the robust design algorithm. The labels in the boxes correspond to individual routines in the connector module or the engine procedures module (see Figure 2.1). Short descriptions of the individual routines are given below:

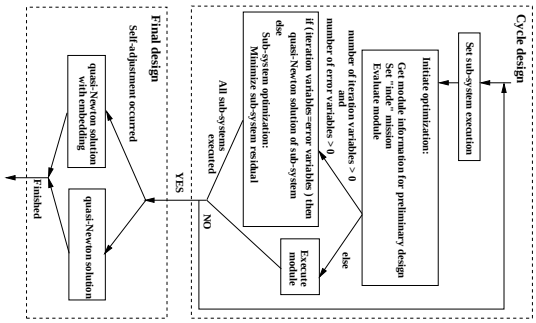


Figure 2.3: The algorithm for robust design process

Set sub-system execution: informs the module connector that a sub-system will be executed. During standard off-design simulations, one routine in the connector module, the connector_comp routine, is used to execute all the modules in the model in a specific order. The set_sub_system_execution routine prepares the connector_comp routine to execute a sub-system of the entire model. For the robust design method, this always means executing a single module.

Get module information for preliminary design: diagnostic routine determining what off-design and design errors to use. This is one of the more sophisticated routines of the robust design method. For instance, if an off-design iteration variable is connected to several inputs, this variable is used only as an iteration variable the first time it is encountered by the get_module_information routine. The reverse situation occurs when external off-design errors are diagnosed. Then, errors are introduced only if the other end of the connection has already been computed.

Set inde mission: informs the module connector to initiate design during the next evaluation call.

Evaluate module: inputs and design iteration variables are initiated. If inputs have been computed as outputs from previous modules, these values are used; otherwise, default values are applied. Initial values for design iteration variables are set using inputs and other design variables.

The module may also "self-adjust", i.e. one or more of the design point specification variables are changed temporarily. Such a feature can be useful if the module has a limited range of definition for some tables or if the equations are very nonlinear. Since the cycle design process (dashed upper box in Figure 2.3) is carried out by doing one sweep through the engine, it might not be possible to evaluate some modules with the user-specified real design parameters and the computed inputs. To deal with this problem, a self-adjusting functionality can be added in the module, i.e. the module changes one or more of its real design parameters originally specified by the user to a new set computed from the inputs. This temporary deviation from the user-defined design point allows the cycle design method to find one set of iteration variables close enough to a working design in order for the final design method (dashed lower box in Figure 2.3) to obtain convergence in the new point. A later step is then to find a way back to the original user specification of the design point by employing an embedding method. None of the three test cases below employed the self-adjustment feature.

Execute module: the module is executed.

Sub-system optimization: the sub-system is executed repeatedly through an optimizer, minimizing the root mean square of the errors. If the system is square an attempt to solve the problem directly is made.

quasi-Newton solutions: The equation system is solved using a quasi-Newton method with a Broyden update (see Appendix B). Embedding simply refers to solving the equations, progressing repeatedly from the real design parameters computed by the self-adjustment procedure towards the real parameters specified by the user.

2.4.3 Application tailored numerics

The price for the reduction in module complexity achieved by using the inverse design methodology is paid by having to make a considerable effort in order to develop suitable numerical methods for solving the problem. In particular, it was necessary to introduce a number of application specific solutions in the original routines [37] to make the robustness of the methodology satisfactory. Since engine modules are constantly being modified and added by users of the system, while the design methodology can be integrated in the system once and for all, this shift in system complexity is highly desirable.

Sub-system optimization

Whenever a different number of iteration and residual variables (both greater than zero) has been detected, a multidimensional nonlinear optimizer attempts to bring the root mean square of the module errors as close to zero as possible. The method used for the sub-system optimization is the Nelder and Mead

downhill simplex method [36]. The algorithm relies upon carrying out a sequence of transformations of a geometric object, a so called simplex, consisting of n+1 points (vertices) in n dimensions. By comparing the function values at the n+1 vertices and replacing the highest value by another point, a decreasing sequence of function values is created. The method is described in greater detail in Chapter 5.

The method is not particularly efficient with respect to the number of necessary function evaluations needed to obtain an optimal solution. Still, for small and medium sized problems, the performance of the method is usually acceptable. Since the largest sub-systems being solved in the three test cases, which represent standard and more advanced gas turbine systems, have four independent variables, speed is not a problem. Another aspect of much greater importance is that the simplex method does not have to estimate derivatives of the goal function and is thus fairly insensitive to non-smooth modules. It is not uncommon that users define modules based on tables resulting in discontinuous functions or discontinuous function derivatives. It is also quite possible that the discontinuities are part of the physics of the problem. One example would be the opening and closing of a compressor bleed valve at a particular rotational speed. Such discontinuities often causes great difficulties for gradient methods.

Another advantage of the simplex method is its reluctance to wander off to very exotic solutions when the system is under-determined. The solver will find a point close to the initial estimate which minimizes (zeroes) the root mean square of the available off-design and design equations.

Furthermore, module diagnostic information can be taken into account by the optimizing algorithm in a very straightforward manner. A great number of error calls report problems with inherent module execution in GESTPAN. For example, a pressure ratio much larger than the design point pressure ratio is detected as an anomaly by the compressor module, which invokes a call to the error handling module. The simplex method solver detects that an error call has occurred during the module execution, and a function value higher than the highest of the present simplex vertices is ascribed to the point. This effectively limits the search to suitable combinations of input parameters. Similar penalty schemes can of course also be introduced in gradient-based optimization algorithms [38], but greater care must be taken to introduce the penalty in a continuous fashion.

Solution of the simultaneous design and off-design equations

In [37], Vetterling et al. writes:

> There are *no* good, general methods for solving systems of more than one nonlinear equation. Furthermore, it is not hard to see why (very likely) there never will be any good, general methods.

They also state that:

> ...root finding becomes virtually impossible without insight! You will almost always have to use additional information, specific to your particular problem, to answer such basic questions as, "Do I expect a unique solution?" and "Approximately where?"

The first question is certainly not trivial to answer for gas turbine systems. It is not very hard to come up with parameter combinations for which no design solution exists. Standard examples are when the equal static pressure balance of the gas stream exiting the bypass channel and the low pressure turbine of a two spool mixed-stream turbofan can not be found or when the power extraction obtainable from the turbines is insufficient for driving the compressors. The latter situation occurs for instance when fan pressure ratios are increased on unmixed high bypass ratio engines. Multiple solutions for one set of inputs are possible, for instance for the compressor module. For a specific rotational speed and pressure ratio, two output mass flows are possible. One of these mass flows normally corresponds to an unstable operating point (rotating stall or surge).

No attempt is made to assist the user in finding a set of parameters which ensures the existence and uniqueness of the solution. This is a deliberate choice based on the philosophy that the sole task of the design process is to try always to find the solution when it exists. Finding design parameter combinations that produce feasible or optimal solutions can be done by numerical optimization. Such a technique is described in Chapter 7.

The entire cycle design step of the robust design methodology tries to answer the second question. By matching the component designs to the previously computed ones (using the sub-system optimization or the quasi-Newton solution procedures), the design and off-design errors can usually be reduced to modest levels. The final design step can then be started at a point relatively close to the solution (assuming it exists), which increases the chance of obtaining a converged solution. Equally important for the final design step is that it can be started at a point where all modules can be executed within their operating range.

The final design step commences by forming the Jacobian of the full system (design *and* off-design equations). The resulting linear system is then solved and a new iteration vector is obtained (see Appendix B for a description of Newton and quasi-Newton solvers). Updating the iteration variables with the solution of the linear system is often referred to as making a full Newton step. Two very relevant measures for making the solution procedure more robust are taken at this stage. The first measure is to **make sure that the residual actually decreased** as the full Newton step was taken. The second measure is to **check whether module errors were reported**. If the residual increased or a module error occurred when evaluating the full Newton step, the direction of the step is retained but a fraction of the full step is tried instead (backtrack). The backtracking process is repeated until an error-free decrease in the residual is obtained.

The backtracking algorithm (part of the *lnsrch* routine in [37]) was modified slightly, since the original routine had no provisions for error checks. A simple inquiry statement was added after every module execution call to force a backtrack upon error detection. If an error-free step results in an increase in the residual, the end point of the step is used to make an improved estimate of a new step length (by fitting a second or third degree interpolating polynomial to the available data). Thus, the algorithm also had to be modified to make sure that all the old function values on which the step fraction was computed corresponded to error-free execution. Another modification to the original algorithms (the *fdjac* routine in [37]), which can be of great value when problems with discontinuities or rapid changes are modeled, was to replace the forward

differencing scheme with a central differencing scheme.

2.4.4 Design point specification

The performance characteristics of the engine components are determined by their design parameters. All functional modules have a vector that stores their type. Four categories exist:

1. Design parameters used to specify preliminary design
2. Design parameters used to specify detailed preliminary design
3. Design iteration parameters
4. Parameters used to fine-tune a design

This information is made available to the connector module by module inquiry calls, and the connector module can thus determine which design parameters must be read from input files during preliminary design. Table 2.1 gives the design variables used for specifying preliminary and detailed preliminary design (variables for the lit afterburner case are not included). On the preliminary

	Preliminary Design		Detailed Preliminary Design		
Compressor:	π_{rp}	η_{rp}		ν	Ω
Burner:		η_{dp}			
Turbine:	η_{dp}	π^*	ψ_1	ψ_2	ψ_3
Nozzle:			$C_{d,4}$	$C_{v,0}$	A_{ratio}
Afterburner:			$C_{d,4}$	$C_{v,0}$	A_{ratio}

Table 2.1: Design specification variables

design level, only the most essential parameters are used to define the design point. Such parameters are typically bypass ratios, compressor efficiencies and pressure ratios, turbine inlet temperature, turbine efficiencies and rotational speeds. For the detailed preliminary design cooling flow distributions, values for C_d:s and C_v:s, compressor map shape parameters etc. are specified. Since some of these are not available as real design parameters, the design implementation also allows module outputs, as well as some inputs, to be set to fixed values.

Module outputs are set by forming design residuals. To keep the design system square, the default settings of the functional modules then have to specify a greater number of design iteration variables, than design residuals. The present selection of design iteration variables is given in Table 2.2.

Except for the iteration variables which are strongly coupled to the module internal design residuals (see section below) or module pressure losses, the iteration variables presently selected specify the sizing of the components and the fuel flow. This selection could easily be changed by modifying the vector storing the type of the real design parameter. For instance, all the pressure loss parameters can be changed from type 3 to type 4 and be given zero default values. To keep the design system square, the corresponding design point pressure loss set by fixing module outputs would then have to be dropped. In the future,

these default overrides will probably be done from input files instead of making changes directly in the source code.

The type 4 parameters are given default values during the preliminary design and the detailed preliminary design. A converged design obtained through the preliminary or detailed preliminary design process can be fine-tuned by re-specifying these parameters using the direct design method described above.

Module internal design residuals

Some design specifications are made more in terms of equations than in absolute values. For instance, at the point of convergence, the turbine design power (parameter used in the efficiency correlation [see Appendix A]) must equal the power computed from the off-design equations. Similarly, the value of the design point Reynolds number parameter, also defined in Appendix A, must equal the value computed from the off-design equations. To determine the value of these parameters during the design process, they are made iteration variables and module internal residuals are introduced to equate them. Several design parameters that have to be computed in the design point can be treated in this fashion. Table 2.2 summarizes the internal errors and the iteration variables used during the design process. Detailed definitions of the variables stated in the table are given in Appendix A.

	Design iteration variable	Design equation
Compressor	Corrected mass flow	$\frac{\Omega\pi_{bb}-\pi}{\pi_{rp}}$
Duct	Pressure loss coefficient (ω)	
Nozzle	Nozzle area	
	Pressure loss coefficient (ω)	
Burner	Design fuel flow	
	Pressure loss coefficient (ω)	
	σ_{dp}	$\frac{\sigma-\sigma_{dp}}{\sigma_{dp}}$
Turbine	Δh_{dp}	$\frac{\Delta h-\Delta h_{dp}}{\Delta h}$
	κ_{dp}	$\frac{\kappa-\kappa_{dp}}{\kappa}$
	Turbine area	
Afterburner	Nozzle area	
	Pressure loss coefficient (ω)	

Table 2.2: Module design iteration variables and internal errors

2.4.5 Specifying off-design iteration variables

Some of the variables used for off-design iteration are frequently part of the design point specification in traditional gas turbine models. To take advantage of this, routines were developed to allow the iteration variables to be "turned off" and kept constant during the design process. Of course, if an off-design iteration variable is specified, one of the residuals must be excluded from the system as well. For instance, if the bypass ratio of the splitter module is specified, a suitable measure for maintaining the equality of unknowns and iteration variables would be to remove the specification of one of the unifier Mach numbers.

2.4.6 Optimizing the design formulation

It is very simple to invert the equations corresponding to the $\sigma_{dp}, \Delta h_{dp}, \kappa_{dp}$ iteration parameters. This has been done in order to reduce the number of equations. Thus, instead of iterating in the turbine design power drop, the value of the power drop computed during the execution is fed directly into the design iteration vector, thereby eliminating the equation as well as the iteration variable. This was done for all the internal design errors in the test cases below.

2.5 Inverse design - Test cases

To demonstrate the operation of the design method, three test cases have been assembled. These are a conventional mixed-stream turbofan engine and two variable cycle engines: the Selective Bleed variable cycle engine and the Double Bypass variable cycle engine. Both variable cycle engines will be described more closely in Chapter 3. In all three cases, the design and off-design equation systems can be arranged in such a way that square sub-systems are solved. To demonstrate the functionality of the sub-system optimization feature, an additional simulation case was assembled for the Selective Bleed engine.

2.5.1 Case 1 - The mixed-stream turbofan

Figure 2.4 gives the wiring diagram for the mixed stream turbofan. The design point specification is given in Table 2.3. As seen from the wiring diagram, only

Module	Type	Design variable	Value
INLET	input	altitude	0 m
INLET	input	flight Mach number	0
COMPRESSOR 1	specified input	rotational speed	150 r/s
COMPRESSOR 1	output	mass flow	100 kg/s
COMPRESSOR 1	real design parameter	π_{rp}	5.0
COMPRESSOR 1	real design parameter	η_{rp}	0.88
SPLITTER 1	specified input	bypass ratio	0.5
DUCT	output	$\frac{p_2}{p_1}$	0.95
COMPRESSOR 2	specified input	rotational speed	300 r/s
COMPRESSOR 2	real design parameter	π_{rp}	3.0
COMPRESSOR 2	real design parameter	η_{rp}	0.88
SPLITTER 2	input	bypass ratio	0.1
COMPRESSOR 3	real design parameter	π_{rp}	3.0
COMPRESSOR 3	real design parameter	η_{rp}	0.88
SPLITTER 3	input	bypass ratio	0.1
BURNER	output	η_{dp}	0.99
BURNER	output	burner exit temperature	1800 K
BURNER	output	$\frac{p_2}{p_1}$	0.95
TURBINE 1	real design parameter	π^*	2.00
TURBINE 1	real design parameter	η_{dp}	0.90
TURBINE 2	real design parameter	π^*	2.50
TURBINE 2	real design parameter	η_{dp}	0.90
UNIFIER	output	M_{bypass}	0.30
AFTERBURNER	output	$\frac{p_2}{p_1}$	0.95

Table 2.3: Turbofan design point specification

the bypass ratio of SPLITTER 1 is used as an iteration variable. The bypass ratio for SPLITTER 2 and SPLITTER 3 are used to specify the amount of air used for cooling. For completeness of the design point specification both the

cooling bypass ratios as well as the Mach number and altitude for the inlet are given in Table 2.3. The full design problem is represented by a system with 17×17 iteration variables and residuals. The square sub-systems are given in Table 2.4. Finally, the converged values of the design and off-design iteration

Module	Type	Variable
COMPRESSOR 1 (2×2)	off-design iteration variable	corrected mass flow
	off-design iteration variable	exit pressure
	internal design error	pressure ratio specification
	external design error	mass flow specification
DUCT 1 (1×1)	design iteration variable	pressure loss coefficient
	external design error	pressure ratio specification
COMPRESSOR 2 (2×2)	off-design iteration variable	corrected mass flow
	off-design iteration variable	exit pressure
	internal design error	pressure ratio specification
	off-design error	mass flow balance (with SPLITTER 1)
COMPRESSOR 3 (2×2)	off-design iteration variable	corrected mass flow
	off-design iteration variable	exit pressure
	internal design error	pressure ratio specification
	off-design error	mass flow balance (with SPLITTER 2)
BURNER (2×2)	design iteration variable	pressure loss coefficient
	design iteration variable	fuel flow
	external design error	pressure ratio specification
	external design error	exit temperature specification
TURBINE 1 (2×2)	off-design iteration variable	exit pressure
	design iteration variable	area
	off-design error	torque balance (with ADD module)
	off-design error	mass flow balance (with ADD module)
TURBINE 2 (2×2)	off-design iteration variable	exit pressure
	design iteration variable	area
	off-design error	torque balance (with ADD module)
	off-design error	mass flow balance (with ADD module)
UNIFIER (2×2)	design iteration variable	bypass area
	design iteration variable	core area
	off-design internal error	static pressure match
	external design error	core Mach number specification
AFTERBURNER (2×2)	design iteration variable	pressure loss coefficient
	design iteration variable	area
	off-design internal error	flow and design area match (equation 2.2)
	external design error	pressure ratio specification

Table 2.4: Turbofan design sub-systems

variables are given in Table 2.5.

2.5.2 Case 2 - The Selective Bleed Variable Cycle Engine

Square sub-systems

A wiring diagram of the Selective Bleed variable cycle engine is shown in Figure 2.5. The engine is designed with both nozzles partially open. The design point specification is given in Table 2.6. The full design problem is represented by a system with 20×20 iteration variables and residuals. The square sub-systems are given in Table 2.7. Finally, the converged values of the design and off-design iteration variables are given in Table 2.8.

Sub-system optimization case

To illustrate the sub-system optimization functionality, the specification of the burner exit temperature was instead moved to the low pressure turbine exit, which was set to 1300 K. This means that the burner sub-system becomes a

Module	Variable	Value
COMPRESSOR 1 (2×2)	corrected mass flow	100.0000 kg/s
	exit pressure	506625.0 Pa
DUCT 1 (1×1)	pressure loss coefficient	50514.86
COMPRESSOR 2 (2×2)	corrected mass flow	19.79527
	exit pressure	1519875.0 Pa
COMPRESSOR 3 (2×2)	corrected mass flow	7.09780
	exit pressure	4559625.0 Pa
BURNER (2×2)	pressure loss coefficient	279224.6
	fuel flow	1.76897 kg/s
TURBINE 1 (2×2)	exit pressure	1193258.8 Pa
	area	0.0163278 m^2
TURBINE 2 (2×2)	exit pressure	520102.6 Pa
	area	0.0561587 m^2
UNIFIER (2×2)	core area	0.1837859 m^2
	bypass area	0.0540059 m^2
AFTERBURNER (2×2)	pressure loss coefficient	1301.8974
	area	0.1686279 m^2

Table 2.5: Converged iteration variables for the turbofan

2X1 system. The fuel flow giving a low pressure turbine exhaust temperature of 1300 K can not be determined when executing the burner module. Instead, the fuel flow is initialized using the input mass flow and a default setting of the fuel air ratio of 0.02. This gives a burner exit temperature of 1459 K after the cycle design procedure (the converged value satisfying the turbine exit requirement of 1300 K is found for a burner exit temperature of 1786 K). The pressure loss residual is minimized to zero by using the two iteration variables.

The turbine now represents an over-determined system, a 2X3 system. The optimizer manages only to reduce the module residual from 0.23 to 0.19, and the Simplex optimizer exits after 40 executions of the turbine module. Still, after completion of the cycle design step, only three of the 20 residuals differ from zero, and these with less than 30% of the error scales. The subsequent final design step is completed successfully, with a low pressure turbine exhaust temperature of 1300 K.

2.5.3 Case 3 - The Double Bypass Variable Cycle Engine

The Double Bypass variable cycle engine, see Figure 2.6, is a low bypass ratio, two-spool turbofan engine. The fan is split into a front and a rear block. The forward block is driven by a variable area low pressure turbine. The rear fan block, the core-driven fan stage (CDFS), is fixed to the high pressure shaft together with the high pressure compressor and is driven by the high pressure turbine. The engine type is described more closely in the variable cycle engine chapter. Descriptions of the advanced features of the Double Bypass VCE have been presented by several authors [39], [40], [41], [42] and [43].

The design point specification is given in Table 2.9. The full design problem is represented by a system with 19×19 iteration variables and residuals. Once again, the use of a design specification typical for traditional gas turbine systems tools leads to a design problem formulation that can be broken down into square sub-systems. The sub-systems are given in Table 2.10. Finally, the converged values for the design and off-design iteration variables are given in Table 2.11.

Module	Type	Design variable	Value
INLET	input	altitude	0 m
INLET	input	flight Mach number	0
COMPRESSOR 1	specified input	rotational speed	150 r/s
COMPRESSOR 1	output	mass flow	100 kg/s
COMPRESSOR 1	real design parameter	π_{rp}	3.0
COMPRESSOR 1	real design parameter	η_{rp}	0.88
SPLITTER 1	specified input	bypass ratio	0.3
DUCT 1	output	$\frac{p_2}{p_1}$	0.98
NOZZLE 1	output	$\frac{p_2}{p_1}$	0.98
COMPRESSOR 2	real design parameter	π_{rp}	2.4
COMPRESSOR 2	real design parameter	η_{rp}	0.88
SPLITTER 2	input	bypass ratio	0.3
DUCT 2	output	$\frac{p_2}{p_1}$	0.98
NOZZLE 2	output	$\frac{p_2}{p_1}$	0.98
COMPRESSOR 3	specified input	rotational speed	300 r/s
COMPRESSOR 3	real design parameter	π_{rp}	3.6
COMPRESSOR 3	real design parameter	η_{rp}	0.88
BURNER	output	η_{dp}	0.99
BURNER	output	burner exit temperature	1800 K
BURNER	output	$\frac{p_2}{p_1}$	0.95
TURBINE 1	real design parameter	π^*	2.00
TURBINE 1	real design parameter	ηdp	0.90
TURBINE 2	real design parameter	π^*	2.50
TURBINE 2	real design parameter	ηdp	0.90
AFTERBURNER	output	$\frac{p_2}{p_1}$	0.95

Table 2.6: Selective Bleed engine design point specification

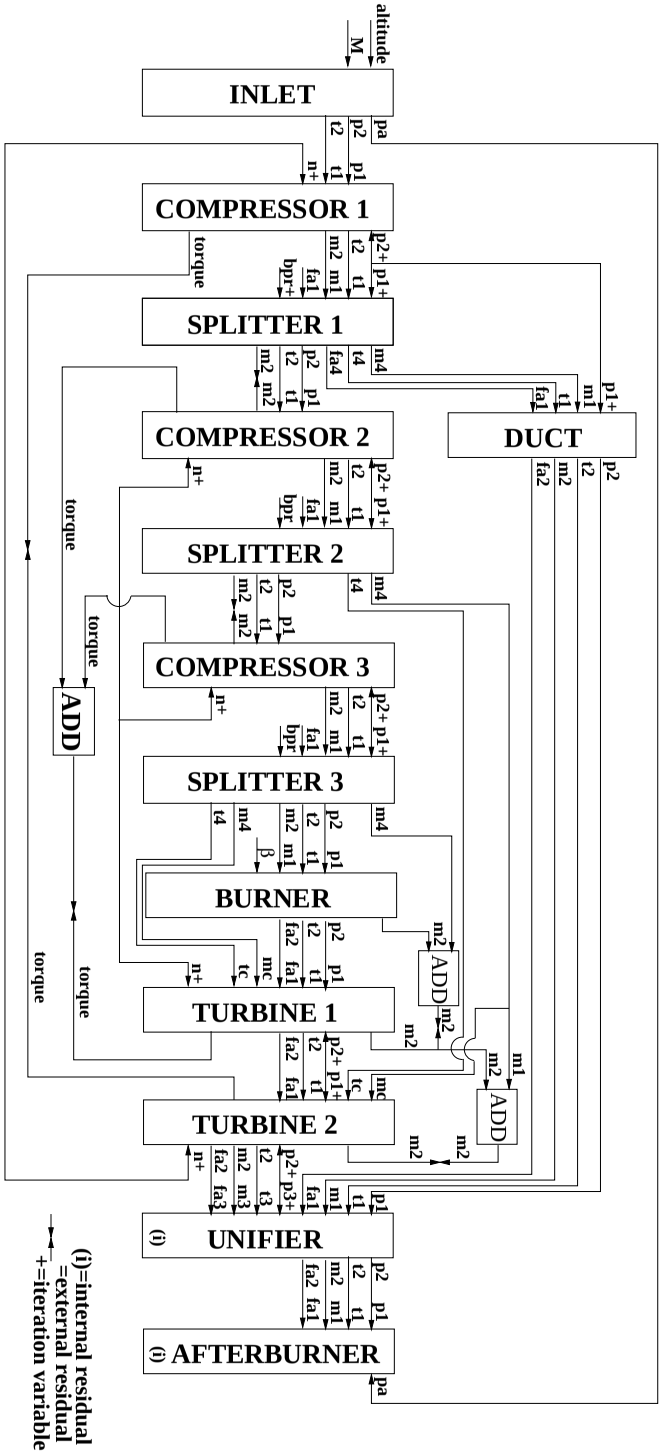


Figure 2.4: The wiring diagram of the mixed-stream turbofan

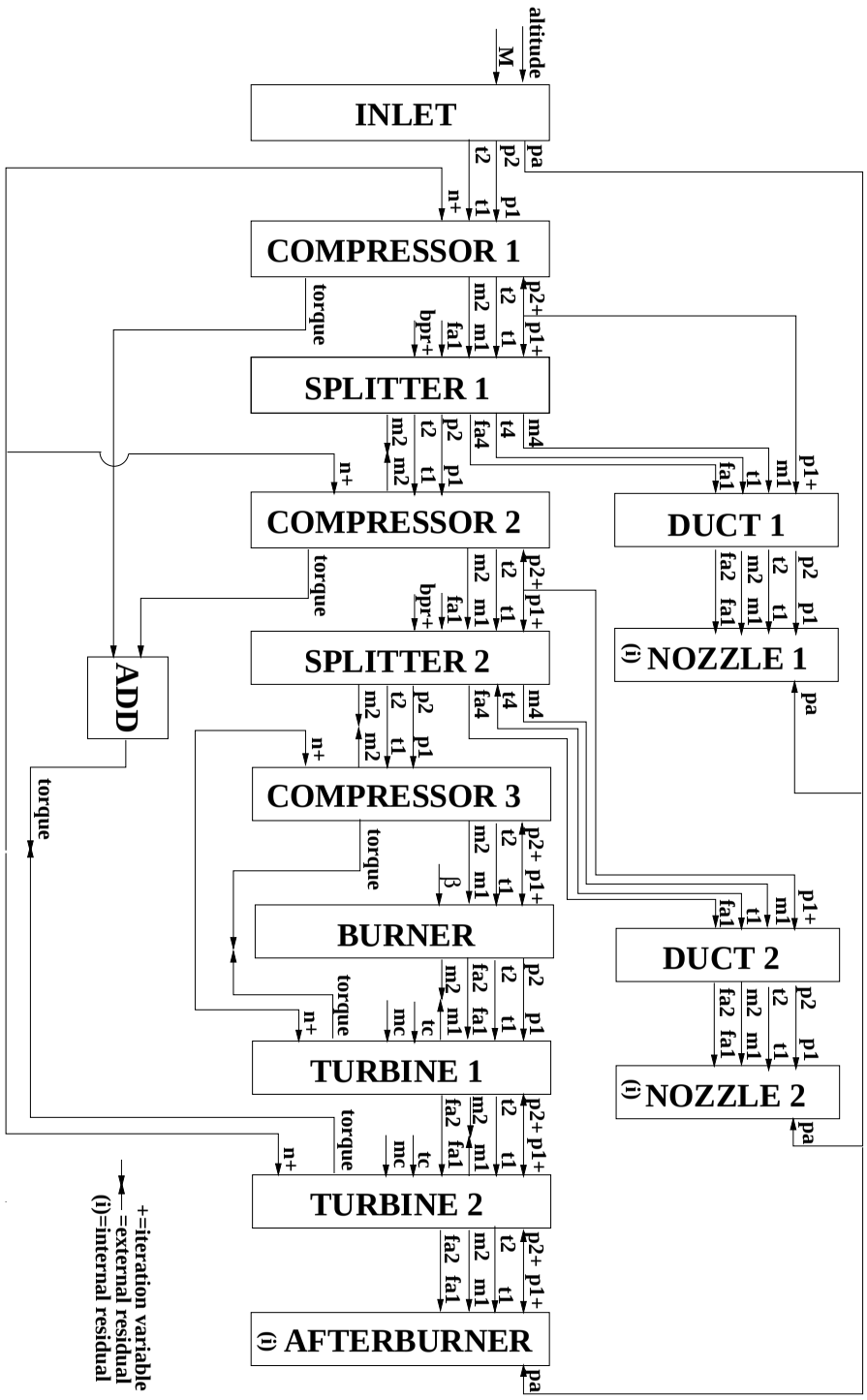


Figure 2.5: The wiring diagram of the Selective Bleed engine

Module	Type	Variable
COMPRESSOR 1 (2×2)	off-design iteration variable	corrected mass flow
	off-design iteration variable	exit pressure
	internal design error	pressure ratio specification
	external design error	mass flow specification
DUCT 1 (1×1)	design iteration variable	pressure loss coefficient
	external design error	pressure ratio specification
NOZZLE 1 (2×2)	design iteration variable	pressure loss coefficient
	design iteration variable	area
	off-design internal error	flow and design area match (equation 2.2)
	external design error	pressure ratio specification
COMPRESSOR 2 (2×2)	off-design iteration variable	corrected mass flow
	off-design iteration variable	exit pressure
	internal design error	pressure ratio specification
	off-design error	mass flow balance (with SPLITTER 1)
DUCT 2 (1×1)	design iteration variable	pressure loss coefficient
	external design error	pressure ratio specification
NOZZLE 2 (2×2)	design iteration variable	pressure loss coefficient
	design iteration variable	area
	off-design internal error	flow and design area match (equation 2.2)
	external design error	pressure ratio specification
COMPRESSOR 3 (2×2)	off-design iteration variable	corrected mass flow
	off-design iteration variable	exit pressure
	internal design error	pressure ratio specification
	off-design error	mass flow balance (with SPLITTER 2)
BURNER (2×2)	design iteration variable	pressure loss coefficient
	design iteration variable	fuel flow
	external design error	pressure ratio specification
	external design error	exit temperature specification
TURBINE 1 (2×2)	off-design iteration variable	exit pressure
	design iteration variable	area
	off-design error	torque balance (with ADD module)
	off-design error	mass flow balance (with ADD module)
TURBINE 2 (2×2)	off-design iteration variable	exit pressure
	design iteration variable	area
	off-design error	torque balance (with ADD module)
	off-design error	mass flow balance (with ADD module)
AFTERBURNER (2×2)	design iteration variable	pressure loss coefficient
	design iteration variable	area
	off-design internal error	flow and design area match (equation 2.2)
	external design error	pressure ratio specification

Table 2.7: Selective Bleed engine design sub-systems

Module	Variable	Value
COMPRESSOR 1 (2×2)	corrected mass flow	100.0000 kg/s
	exit pressure	303975.0 Pa
DUCT 1 (1×1)	pressure loss coefficient	8499.117
NOZZLE 1 (2×2)	pressure loss coefficient	8162.552
	area	0.040371 m^2
COMPRESSOR 2 (2×2)	corrected mass flow	30.52207
	exit pressure	729540.0 Pa
DUCT 2 (1×1)	pressure loss coefficient	62884.84
NOZZLE 2 (2×2)	pressure loss coefficient	60394.60
	area	0.014888 m^2
COMPRESSOR 3 (2×2)	corrected mass flow	11.220908
	exit pressure	2626344.0 Pa
BURNER (2×2)	pressure loss coefficient	124875.2
	fuel flow	1.87823 kg/s
TURBINE 1 (2×2)	exit pressure	1374004.6 Pa
	area	0.0268491 m^2
TURBINE 2 (2×2)	exit pressure	934616.27 Pa
	area	0.0249544 m^2
AFTERBURNER (2×2)	pressure loss coefficient	8910.3520
	area	0.0648283 m^2

Table 2.8: Converged iteration variables for the Selective Bleed engine

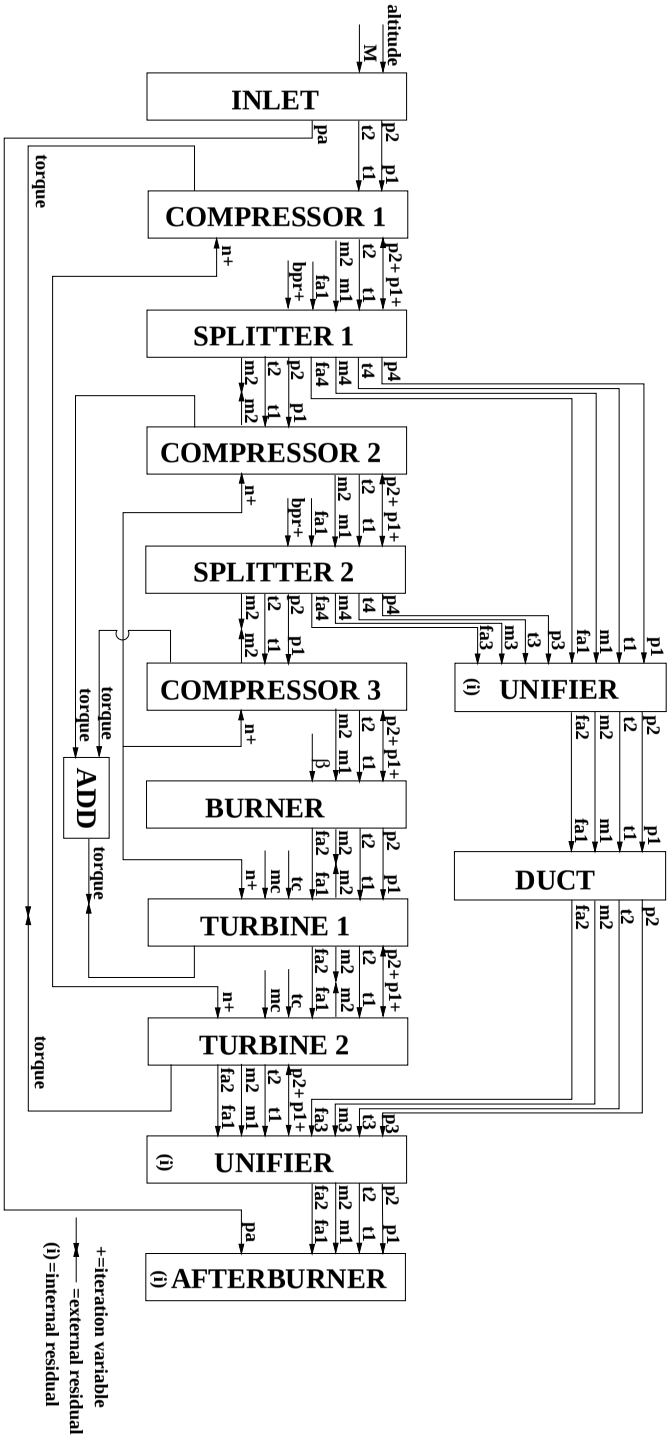


Figure 2.6: The wiring diagram of the Double Bypass variable cycle engine

Module	Type	Design variable	Value
INLET	input	altitude	0 m
INLET	input	flight Mach number	0
COMPRESSOR 1	specified input	rotational speed	150 r/s
COMPRESSOR 1	output	mass flow	100 kg/s
COMPRESSOR 1	real design parameter	π_{rp}	5.0
COMPRESSOR 1	real design parameter	η_{rp}	0.88
SPLITTER 1	specified input	bypass ratio	0.3
COMPRESSOR 2	real design parameter	π_{rp}	1.37
COMPRESSOR 2	real design parameter	η_{rp}	0.88
SPLITTER 2	input	bypass ratio	0.3
UNIFIER 1	output	$M_{gas\ stream\ 3}$	0.70
DUCT	output	$\frac{p_2}{p_1}$	0.98
COMPRESSOR 3	specified input	rotational speed	300 r/s
COMPRESSOR 3	real design parameter	π_{rp}	4.0
COMPRESSOR 3	real design parameter	η_{rp}	0.88
BURNER	output	η_{dp}	0.99
BURNER	output	burner exit temperature	1800 K
BURNER	output	$\frac{p_2}{p_1}$	0.95
TURBINE 1	real design parameter	π^*	2.00
TURBINE 1	real design parameter	η_{dp}	0.90
TURBINE 2	real design parameter	π^*	2.50
TURBINE 2	real design parameter	η_{dp}	0.90
UNIFIER 2	output	$M_{gas\ stream\ 1}$	0.30
AFTERBURNER	output	$\frac{p_2}{p_1}$	0.95

Table 2.9: Double Bypass engine design point specification

Module	Type	Variable
COMPRESSOR 1 (2×2)	off-design iteration variable	corrected mass flow
	off-design iteration variable	exit pressure
	internal design error	pressure ratio specification
	external design error	mass flow specification
COMPRESSOR 2 (2×2)	off-design iteration variable	corrected mass flow
	off-design iteration variable	exit pressure
	internal design error	pressure ratio specification
	off-design error	mass flow balance (with SPLITTER 1)
UNIFIER 1 (2×2)	design iteration variable	bypass area
	design iteration variable	core area
	off-design internal error	static pressure match
	external design error	gas stream 3 Mach number specification
DUCT (1×1)	design iteration variable	pressure loss coefficient
	external design error	pressure ratio specification
COMPRESSOR 3 (2×2)	off-design iteration variable	corrected mass flow
	off-design iteration variable	exit pressure
	internal design error	pressure ratio specification
	off-design error	mass flow balance (with SPLITTER 2)
BURNER (2×2)	design iteration variable	pressure loss coefficient
	design iteration variable	fuel flow
	external design error	pressure ratio specification
	external design error	exit temperature specification
TURBINE 1 (2×2)	off-design iteration variable	exit pressure
	design iteration variable	area
	off-design error	torque balance (with ADD module)
	off-design error	mass flow balance (with ADD module)
TURBINE 2 (2×2)	off-design iteration variable	exit pressure
	design iteration variable	area
	off-design error	torque balance (with ADD module)
	off-design error	mass flow balance (with ADD module)
UNIFIER 2 (2×2)	design iteration variable	bypass area
	design iteration variable	core area
	off-design internal error	static pressure match
	external design error	gas stream 1 Mach number specification
AFTERBURNER (2×2)	design iteration variable	pressure loss coefficient
	design iteration variable	area
	off-design internal error	flow and design area match (equation 2.2)
	external design error	pressure ratio specification

Table 2.10: Double Bypass engine design sub-systems

Module	Variable	Value
COMPRESSOR 1 (2×2)	corrected mass flow	100.0000 kg/s
	exit pressure	506625.0 Pa
COMPRESSOR 2 (2×2)	corrected mass flow	11.67954
	exit pressure	694076.2 Pa
UNIFIER 2 (2×2)	core area	0.127477 m^2
	bypass area	0.016290 m^2
DUCT 1 (1×1)	pressure loss coefficient	17369.72
COMPRESSOR 3 (2×2)	corrected mass flow	11.679546
	exit pressure	2776305.0 Pa
BURNER (2×2)	pressure loss coefficient	137943.7
	fuel flow	1.86261 kg/s
TURBINE 1 (2×2)	exit pressure	1174171.0 Pa
	area	0.0250641 m^2
TURBINE 2 (2×2)	exit pressure	492281.96 Pa
	area	0.0519061 m^2
UNIFIER 2 (2×2)	core area	0.2370055 m^2
	bypass area	0.0765053 m^2
AFTERBURNER (2×2)	pressure loss coefficient	1185.3818
	area	0.1769239 m^2

Table 2.11: Converged iteration variables for the Double Bypass engine

Chapter 3

Variable cycle engines

The first section in this chapter described the underlying mechamism for the potential performance benefits of Variable Cycle Engines (VCE:s). Subsequently, two variable cycle engine concepts: the selective bleed engine and the double bypass engine, are described. Further studies of both these cycles are described later in this thesis.

3.1 Variable cycle engine characteristics

Jet engine performance is primarily expressed in two numbers:

- SFC (Specific Fuel Consumption)
- F (Thrust)

To achieve high propulsive efficiencies and thereby reduce the SFC, exhaust jet velocities should not exceed flight velocities more than necessary for the required thrust levels. This can be shown by relating the SFC to a set of efficiencies applicable to jet engines.

As a first step, the propulsive efficiency, η_p, is introduced as the ratio of the aircraft power, FV_0, to the power out of the engine, W_{out}, i.e.

$$\eta_p = \frac{FV_0}{W_{out}} = \frac{FV_0}{\frac{(m_0+m_f){V_j}^2}{2} - \frac{m_0{V_0}^2}{2}} = 2\frac{[(m_0+m_f)\,V_j - m_0V_0]\,V_0}{(m_0+m_f)\,{V_j}^2 - m_0{V_0}^2} \tag{3.1}$$

by assuming a single exhaust and exit and ambient pressure to be equal. If the fuel flow is neglected in equation 3.1, then the relation can be simplified to

$$\eta_p = \frac{2}{1+\frac{V_j}{V_0}} \tag{3.2}$$

Secondly, the thermal efficiency is defined as the ratio of the power out of the engine, $\frac{(m_0+m_f){V_j}^2}{2} - \frac{m_0{V_0}^2}{2}$, to the rate of energy supplied by the fuel, m_fQ_f, i.e.

$$\eta_{th} = \frac{\frac{(m_0+m_f){V_j}^2}{2} - \frac{m_0{V_0}^2}{2}}{m_fQ_f} \tag{3.3}$$

Finally, the overall efficiency of the power plant is formed as the ratio of the aircraft power, FV_0, to the rate of energy supplied by the fuel, $m_f Q_f$, i.e.

$$\eta_o = \frac{FV_0}{m_f Q_f} \tag{3.4}$$

Also note, from the definition of η_o, η_{th} and η_p, that

$$\eta_o = \eta_p \eta_{th} \tag{3.5}$$

Since the SFC is

$$\text{SFC} = \frac{m_f}{F} = \frac{V_0}{Q_f \eta_0} = \frac{V_0}{Q_f \eta_p \eta_{th}} = \frac{V_0 \left(1 + \frac{V_j}{V_0}\right)}{2 Q_f \eta_{th}} \tag{3.6}$$

It is clear from equation 3.6 that the SFC will benefit from a reduction of V_j. Low exhaust jet velocities are attained by selecting a large value for the design bypass ratio. The resulting low specific thrust, $(\frac{F}{m})$, is then compensated for by increasing the design mass flow. Increasing the flight Mach number strongly reduces the specific thrust of these high bypass ratio cycles, and heavy use of afterburning is then needed to provide the required thrust.

If the mission is dominated by supersonic operation, high specific thrust designs are needed and the optimal value of the design bypass ratio is then reduced, approaching the turbojet cycle. These cycles show poor performance operating in subsonic cruise flight.

Selecting optimal cycle parameters for a mixed mission for which substantial amounts of fuel are consumed during both supersonic and subsonic flight normally implies great compromises in engine performance. A cycle purely designed for one part of the mission would most likely perform poorly elsewhere in the flight envelope. Variable cycle engines show great potential for for achieving *superior mixed mission performance* in comparison with conventional turbofan and turbojet engines. To some extent, the VCE combines the benefits of the turbojet and turbofan cycles by switching its mode of operation depending on flight conditions.

3.2 Development of the double bypass engine

The history of the double bypass variable cycle engine started with the Supersonic Transport program (SST) initiated in the USA in the mid 1960s. This program was intended to define an engine for civil passenger service able to cruise supersonically. General Electric and Pratt & Whitney were contracted by NASA to find suitable engines for these vehicles. The program was terminated in 1971. During this period, GE worked on two different engines, the GE4/J5 and the GE4/J6. Both had substantial noise problems, although the redesign, the GE4/J6, was fitted with an annular plug nozzle with a jet noise suppressor, which alleviated the noise problem somewhat. Basically, the GE4/J6 was an upscaled version of the GE4/J5. The larger air flow provided the necessary thrust for take-off, dry power climb, acceleration and supercruise. On the other hand, the engine was heavier than the GE4/J5, which was already a very bulky design.

In 1972 NASA initiated the Supersonic Cruise Aircraft Research (SCAR) program. A number of variable cycle engines were tested in this program. The most promising technologies that emerged in this program were incorporated into the GE21 (based on the double bypass variable cycle engine concept). The following sections are intended to describe some of the development phases and the principles of the double bypass variable cycle engine.

3.2.1 The single bypass variable cycle engine

A predecessor to the double bypass variable cycle engine was the single bypass VCE. This engine is basically a low bypass ratio, two-spool turbofan engine with a variable area mixing device (VABI = variable area bypass injector) mounted in the end of the bypass channel. Figure 3.1 shows the schematic layout of the engine.

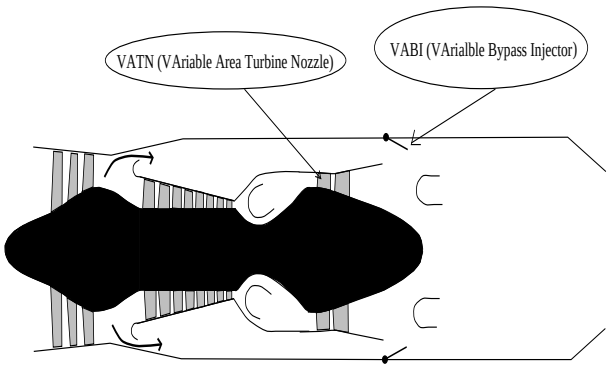


Figure 3.1: The single bypass variable cycle engine

The VABI enables a balancing of the exhaust system static pressures over a range of operating conditions. As a result, the bypass ratio of the engine can be modulated. This variable bypass ratio capacity can be further increased by using a variable area turbine nozzle (VATN). Increasing the VABI area and decreasing the turbine nozzle area offer the potential to modulate the bypass ratio quite substantially.

3.2.2 The double bypass variable cycle engine

The double bypass variable cycle engine was designed as a follow on to the single bypass engine. Figure 3.2 shows the schematic layout of the engine. The double bypass variable cycle engine is a low bypass ratio, two spool turbofan engine. The fan is split into a front and a rear block. The forward block is driven by a variable area low pressure turbine. The rear fan block, the core-driven fan stage (CDFS), is fixed to the high pressure shaft together with the high pressure compressor and is driven by the high pressure turbine.

Other advanced features of the double bypass VCE are the forward variable area bypass injector (forward VABI) and the rear variable area bypass injector (rear VABI). As shown in Figure 3.2, the DBE is designed with two bypass

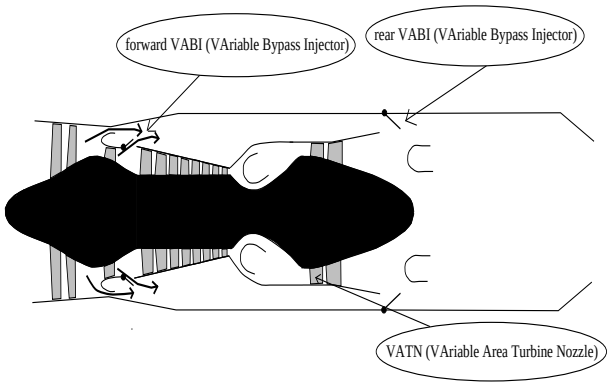


Figure 3.2: The double bypass variable cycle engine

streams. An outer bypass stream is positioned directly behind the fan and an inner bypass stream immediately behind the CDFS.

The VCE operates in two different modes, double or single bypass, depending on the flight conditions, see Figure 3.3. Descriptions of the advanced features of the double bypass VCE have been presented by several authors [39], [40], [41], [42] and [43].

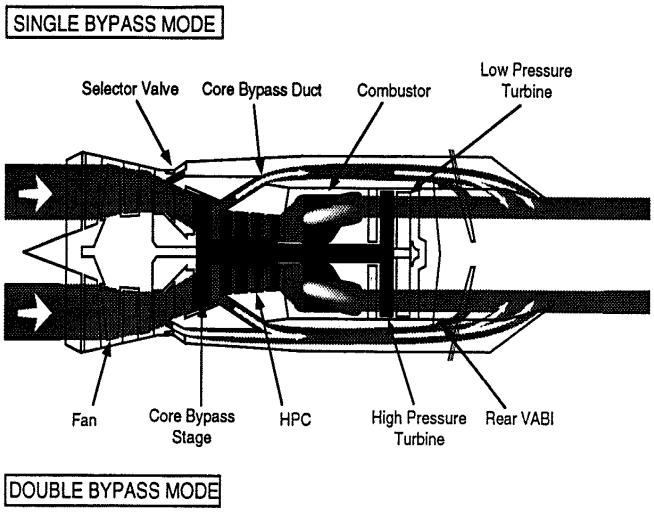


Figure 3.3: The Double Bypass Variable Cycle Engine

Single bypass mode is used during acceleration and supersonic cruise (operation with unlit afterburner) by closing the selector valve (Figure 3.3). All the air is then forced through the CDFS, achieving the specific thrust required for these flight conditions.

Double bypass mode operation, achieved by opening the selector valve, is used at take-off and subsonic cruise. The control system is set to maximize flow by loading the low pressure turbine (closing the stator vanes of the variable area turbine) and opening the forward and rear VABI:s. Due to the mismatch in spool

speeds, the core cannot swallow all the flow and the excess flow is dumped into the bypass stream. Double bypass mode is used for subsonic cruise and take-off.

3.3 Installation effects - spillage/afterbody drag

For conventional turbofan engines, the decreasing mass flow at part power causes a substantial inlet and afterbody drag [40]. The variable cycle features of the double bypass VCE allows the engine to match the inlet air supply by maintaining a high air flow at part power, whereas the conventional cycle has to accept a certain air spillage. The full performance potential of the double bypass VCE can thus only be estimated if inlet/engine air flow matching is considered.

3.4 Outlook for the double bypass VCE

In 1983 GE and P&W were each awarded a Demonstration/Validation phase program by the US Air Force for the future power plant of the ATF (Advanced Tactical Fighter). The GE engine was a double bypass variable cycle engine concept, the XF120/YF120 engine, selected on the basis of the double bypass VCE's ability to meat the diverse mission requirements of the ATF.

In 1991, the US Air Force selected the P&W engine concept to power its F22 air superiority fighters. A crucial factor in this decision was the assessment of a higher technological risk involved in the double bypass variable cycle engine. This decision was made despite the substantially better flight test performance of the double bypass VCE. See [44] for an excellent discussion of the F119/F120 program development phases. Recent GE variable cycle efforts have focused on the the IHPTET program. The XTE76 Joint Technology Demonstrator Engine (JTDE), implements the double bypass engine cycle [45].

3.5 The selective bleed engine

The Selective Bleed VCE concept has been developed with a Short Take-Off Vertical Landing (STOVL) aircraft in mind. The engine operates in two different modes depending on flight conditions: the subsonic and the supersonic mode. The two operating modes of the Selective Bleed Variable Cycle Engine are illustrated in Figure 3.4. During subsonic operation, air is bled at the back of the low pressure compressor powering a continuously vectorable convergent nozzle. The nozzle is vertically positioned for take-off and horizontally positioned for subsonic cruise. The intermediate compressor operates with closed stator vanes. During supersonic operation, the front nozzle is closed and the air is discharged through a convergent-divergent nozzle. The intermediate compressor is high flowed by opening its stator vanes. This gives the rear nozzle the specific thrust suitable for dry supersonic cruise at Mach 1.6.

A number of studies assessing the performance of the Selective Bleed engine have been carried out at Cranfield University [47, 48, 46, 49]. These studies have focused on selecting a suitable engine design point for the specified aircraft mission and optimizing the steady state control system of the engine. Work in this thesis addresses the key issue of safe and efficient handling of the transient

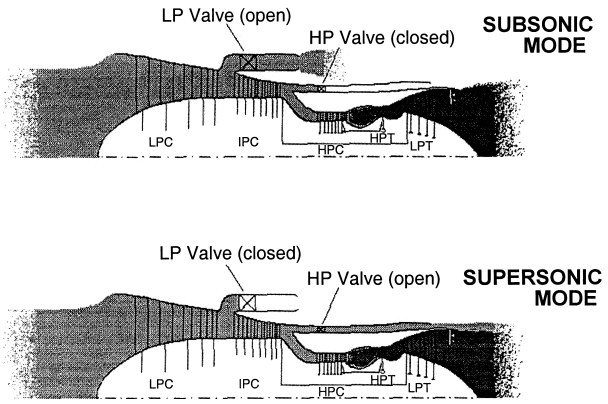


Figure 3.4: The operating modes of the selective bleed engine [46]

engine performance during the switch from subsonic to supersonic mode. Furthermore, a mission optimization study described in Chapter 7 indicate that the design suggested in [48] can be downsized considerably.

Chapter 4

Transient simulations

The first section in this chapter describes how transient capabilities were introduced in GESTPAN. The following sections describes work associated with selecting a suitable solver for generalized gas turbine systems. In particular, attention is drawn to the advantages of using Ordinary Differential Algebraic Equation (ODAE) problem formulations, the use of high order implicit methods and interpolation methods tailored for these solvers. All numerical tests have been conducted with solvers freely available for down-loading from the NETLIB archive [50].

4.1 Transient capabilities in GESTPAN

To simulate transient engine behavior, dynamic components are introduced into the engine model. Rotor components are used to model shaft inertia and to integrate rotational speed changes as a function of torque imbalances. Volume components are used between the static engine components to simulate storage of thermal energy and gas mass (this modeling technique is frequently referred to as the Inter-Component Volume method [51]). This approach is perfectly viable, since the typical time scales for the internal dynamics of the components are much shorter than the time scales for the phenomena relevant for the gas turbine operation. The governing equations for the dynamic modules are given in Appendix A.

Refined dynamic modeling features could be included in the GESTPAN system in the same way as the volume and rotor modules have been introduced. Such refinements could include compressor stage models with volumes integrated between the stages, allowing gas turbine post stall behavior to be simulated [52, 53, 54]. Another effect, important also for stable gas turbine performance, is heat transfer. Detailed heat transfer models may be used both for improving the accuracy of engine performance models and for improving predictions of stability margins [55, 56].

4.2 Gas turbine transient modeling

The engine manufacturer has to be able to predict engine transient behavior early in the design program, see Figure 1.1, in order to ensure that adequate

response rates can be attained without compromising engine safety. Rapid thrust response is important both for maneuverability and safety. For civil aircraft the dominating engine response design requirement is to be able to cope with an aborted landing. For fighter aircraft, designed to operate in a number of flight situations and to experience great variations in throttle settings, the engine response rate is a key performance parameter.

4.2.1 Problem formulation

Dynamic gas turbine modeling has been in use for almost five decades [57] and digital dynamic engine simulation tools for more than 20 years [51, 6]. Early on it was concluded that jet engines constitute stiff systems and that the speed of integration can profit greatly from the implementation of implicit solvers [2]. Although several authors have reported the successful use of low order (first/second) implicit ODE (Ordinary Differential Equations) solvers [58, 53, 59], very little attention has been paid to the direct use of differential algebraic system solvers and quantifying the improvements of higher order integration techniques.

A critical matter concerning the modeling of a gas turbine system is to decide whether an ODE or an ODAE (ODE with coupled algebraic equations) model is going to be applied. This choice will have quite a substantial impact not only on the required computation time but also on the complexity and effort necessary for assembling the model and getting it running.

The performance of high order ordinary differential equation solvers is strongly dependent on the smoothness of the engine model. Most performance codes use component map data from which the required variables are interpolated. The selection of a poor interpolation method can have a devastating effect on the solver. Here, an attempt is made to measure the extent of which the smoothness of the interpolating function influences the performance of high order solvers on a typical gas turbine transient. By introducing first, third and fifth order splines [60] to approximate all map data in the performance model the smoothness can be increased in a step wise manner, and consequently the effect of the interpolation method on the solver performance can be studied.

In order to be able to test the solvers without the influence of interpolation schemes a special set of analytic nonlinear test equations were developed. These are given in the appendix to Paper 3. The main guide-line for deriving these engine models, was to obtain a nonlinear gas turbine system model with as realistic dynamic performance as possible, at a minimum of complexity. E.g. using constant specific heat ratios $\gamma = \gamma_h$=1.333 for engine components with a fuel air ratio $\neq 0$ and $\gamma = \gamma_c$=1.400 otherwise, was considered acceptable since it was anticipated to have little effect on how the solvers would perform. Effects of mechanical losses were also neglected. Although metal air heat transfer effects are very important for transient modeling in general no such effects were included here, since these phenomena do not add very much to the burden for the solvers. This is because the characteristic time scales for the thermal inertia are much larger than those for the rotor and especially mass and thermal gas dynamics. Thus the eigenvalues relating to the thermal inertia have small negative real parts causing little trouble for ODE/ODAE solvers.

4.3 Gas turbine dynamics and ODAE:s

If an attempt to model an arbitrary gas turbine system is made the most probable system that will emerge from this effort is a semi-explicit ordinary differential algebraic system, i.e.:

$$\begin{aligned} x' &= f(t, x, z) \\ 0 &= g(t, x, z) \end{aligned} \tag{4.1}$$

This means that unless special care is taken during the modeling process some algebraic equations (algebraic loops) will arise coupled to the differential equation system. Note that local equations completely contained within a component do not constitute algebraic equations. In the rest of this chapter the x variables in Eq. 4.1 will be referred to as differential variables and the z variables as algebraic variables.

The semi-explicit ODAE represented by Eq. 4.1 is very general indeed. Shampine et al. [27] shows that a system of equations can be represented in a SIMULINK block diagram if and only if the system can be written as an ODAE in the form represented by Eq. 4.1.

4.3.1 Methods for solving the ODAE problem

Basically three main strategies for solving Eq. 4.1 exist:

1. The "direct approach"
2. The "ODE approach" = "the indirect approach"
3. Transforming the ODAE model to an ODE model by algebraic manipulations.

The direct approach is studied in this work by the use of the DASSL solver [61]. This code uses a kth order Backward Differentiation Formula (BDF), where k varies from one to five, to approximate the derivatives of a more general expression than Eq. 4.1. The DASSL code then solves the resulting equation system directly, i.e. it solves for the differential and algebraic variables simultaneously.

The ODE approach, or the indirect approach, is based on solving the algebraic equations in Eq. 4.1 for every function evaluation required by the ODE solver. Here, the algebraic equations are solved with a globally convergent Broyden method [37]. Three different ODE solvers have been tested. One implicit method (also a BDF method) [62], and two explicit methods; a variable order Adams Bashforth method [62] and the forward Euler method.

4.3.2 Transforming the ODAE to ODE

Using the Inter-Component Volume Method for transient gas turbine modeling sometimes allows the algebraic equations of Eq. 4.1 to be eliminated through manipulations of the component formulas. This has been done in the "Engine 3" test case studied here (see burner, nozzle and mixer component sections in the appendix of Paper 3).

4.3.3 System advantages with ODAE formulations

When working with a general gas turbine simulation tool the use of such algebraic manipulations as those described above are unfortunate from the system complexity perspective. In order to make a specific engine model free from algebraic equations, component physics has to be duplicated in more than one algorithm making the simulation system more complex without any direct benefits. It is highly desirable to be able to use the most straightforward and robust way of formulating the component physics for all engine models, including both transient and steady state formulations.

4.4 Engine models

The nonlinear analytic engine component models have been used to generate three turbofan models:

1. Turbofan - rotor dynamics - ODAE
2. Turbofan - rotor/volume dynamics - ODAE
3. Turbofan - rotor/volume dynamics - ODE

The wiring diagrams in Fig. 4.1 and Fig. 4.2 as well as in Fig. 4.3 illustrate how the engine components are interconnected. The differential and algebraic variables used during the integration, as well as their initial value for the test transient, are given in Table 4.1. The index numbering and the referencing to algebraic variables given in Table 4.1 are clear from the wiring diagrams and the abbreviations given in the table text.

4.5 Solver comparisons - Methodology

4.5.1 Test transients and accuracy requirements

The engine test transient selected for the measurement of the solver performance is an engine acceleration trajectory from 67% to 95% of engine maximum rotational speed.

4.5.2 Error control

To establish a converged solution integration was carried out for a decreasing series of tolerances. The three codes having a proper error control (not the forward Euler implementation) were tested in this way and produced the same solutions for the for all three engines. Note that the converged solution of Engine 2 and Engine 3 will be the same but Engine 1 will differ due to the absence of volumes.

A meaningful solver performance measurement has to be carried out at a specific error tolerance. For gas turbine engines a solution which is more accurate than the maximum attainable accuracy of a tuned system model would be wasteful. Another relevant aspect is that the way errors are transported along the solution trajectory depend on the dynamic system itself. A numerical error made at one point might be attenuated along the trajectory. With all this in

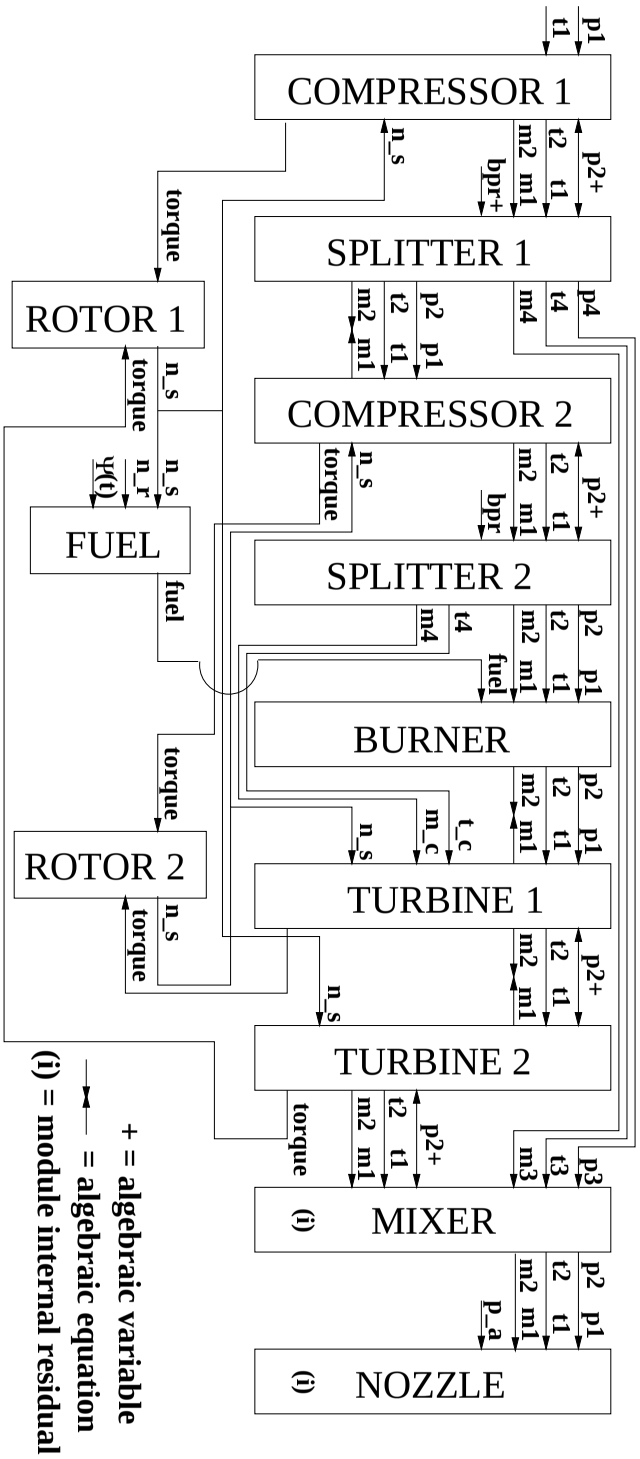


Figure 4.1: Engine 1 - rotor dynamics - ODAE model

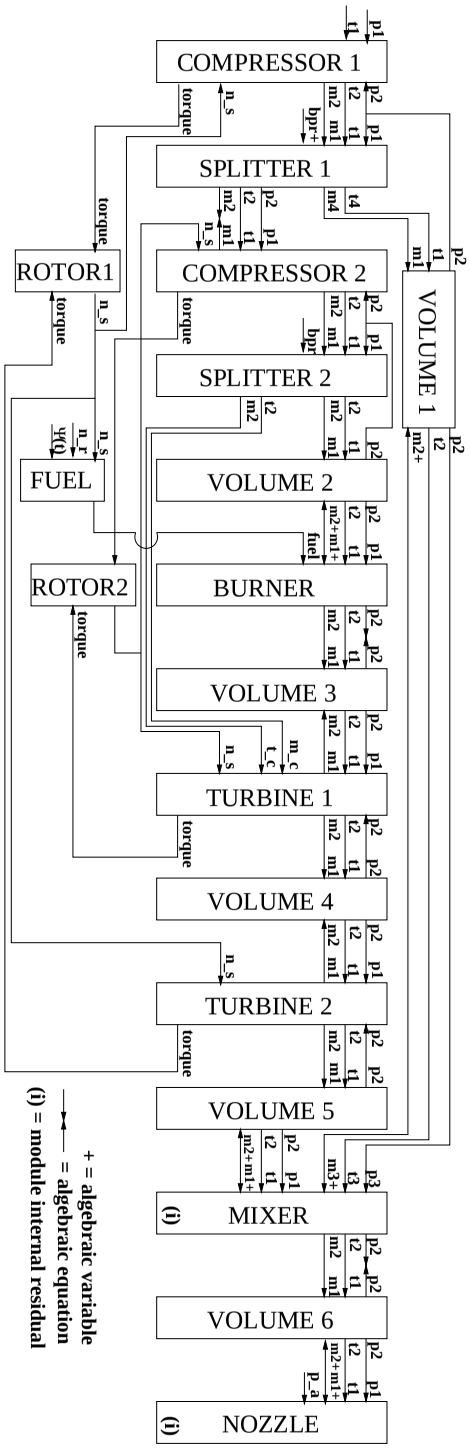


Figure 4.2: Rotor and volyme dynamics - ODAE model

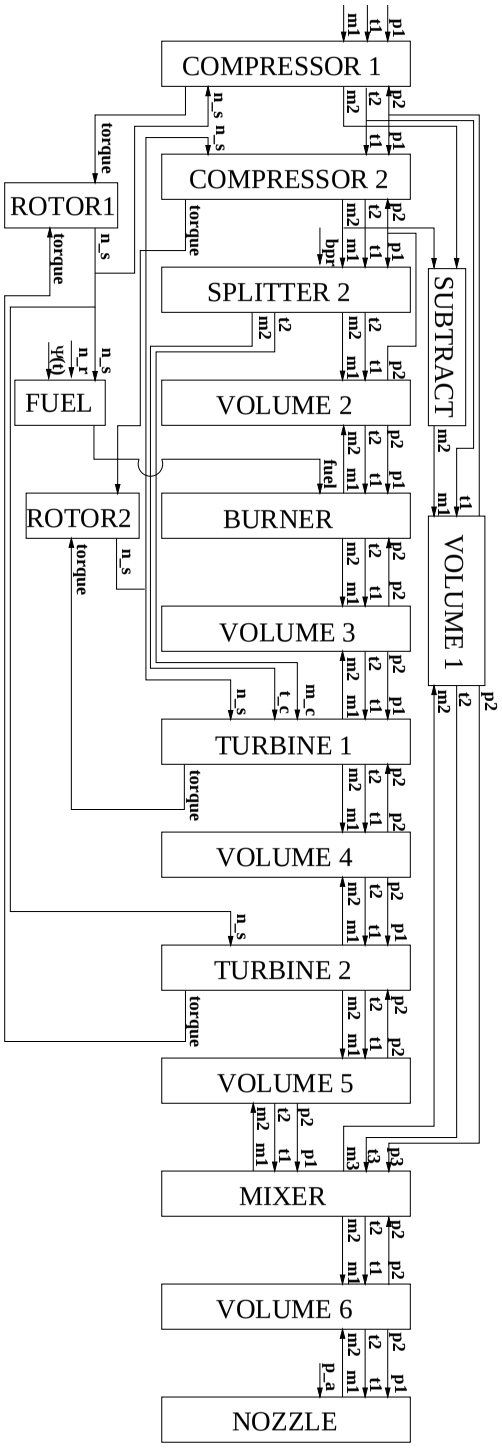


Figure 4.3: Rotor and volyme dynamics - ODE model

	Engine 1,2,3	
x_1	$n_{s,R1} = 124.29$	
x_2	$n_{s,R2} = 223.79$	
	Engine 2,3	
x_3	$T_{V1} = 411.62$	
x_4	$m_{V1} = 0.4124$	
x_5	$T_{V2} = 727.07$	
x_6	$m_{V2} = 1.3426$	
x_7	$T_{V3} = 1373.4$	
x_8	$m_{V3} = 0.6743$	
x_9	$T_{V4} = 1061.7$	
x_{10}	$m_{V4} = 0.3099$	
x_{11}	$T_{V5} = 911.78$	
x_{12}	$m_{V5} = 0.4401$	
x_{13}	$T_{V6} = 767.61$	
x_{14}	$m_{V6} = 0.5270$	
	Engine 1	**Engine 2**
z_1	$P_{2,LPC} = 2.43E5$	$bpr_{S1} = 0.4122$
z_2	$bpr_{S1} = 0.4122$	$m_{3,M} = 11.712$
z_3	$P_{2,HPC} = 1.40E6$	$m_{2,V2} = 26.601$
z_4	$P_{2,HPT} = 4.72E5$	$m_{2,V5} = 28.922$
z_5	$P_{2,LPT} = 2.30E5$	$m_{2,V6} = 40.634$

Table 4.1: Differential and algebraic variables used during integration with their initial conditions (V=Volume, R=Rotor, M=Mixer)

mind a global test was formed based on all the points along the integration compared to the converged solution, i.e. it was required of a solution to pass the following test:

$$\sqrt{\frac{\sum\left(\frac{y-y_{conv}}{y_{conv}}\right)^2}{nps}} < 0.005 \tag{4.2}$$

where nps is the number of points on the trajectory. All cases were sampled at 100 Hz, i.e. 500 test points were used. The error was checked on the fuel scheduled by the feedback controller, i.e. the fuel component. y_{conv} was obtained using the DASSL code with a pure relative error control using a tolerance of 10^{-10}.

The transient was started from the initial conditions given in Table 4.1 and a required rotational speed was set to n_r=180.0. The fuel scheduling trajectory relevant for Engine 2 and Engine 3 is displayed in Fig. 4.4.

4.6 Solver comparisons - Results

The number of function evaluations, i.e. engine evaluations, given in Table 4.2 below are the minimum number necessary for the solution to pass the error test defined by Eq. 4.2. A number of observations can be made based on Table 4.2.

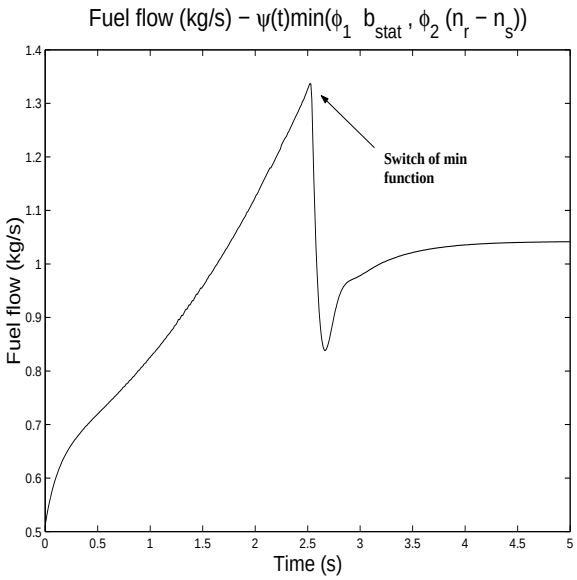


Figure 4.4: Fuel scheduled by the control unit

Engine	F. Eul.	LSODE	DASSL	Adams
1	11240	3362	761	3898
2	625143	11879	1585	2393325
3	62502	1313	1147	286078

Table 4.2: Performance of the solvers

Elimination of algebraic equations: In Engine 3 the formulas of the mixer, the burner and the nozzle were rearranged in order to eliminate the algebraic equations present in Engine 2. This operation resulted in an increase in computational speed with about an order of magnitude, for all solvers except for the DASSL code. The reason for this is that the Forward Euler, the LSODE and the Adams Bashforth implementations use an "ODE approach" to solve the differential algebraic problem, i.e. the solution of a nonlinear equation system has to be performed for every function evaluation required by the ODE-solver. The error tolerance for the equation solver was set to a fraction 10^{-3} of the local error requirement of the ODE-solver. This seemed, after some trade-off studies, to be close to the optimal choice. A larger value increased the number of function evaluations, since the inaccurately solved equation system would disturb the ODE-solver.

The DASSL code, on the other hand solves the problem by the "direct approach" treating the differential and algebraic variables simultaneously. Note also that the number of function evaluations required by DASSL is reduced only with about 30% as the the system is transformed from the ODAE system represented by Engine 2 to the ODE system represented by Engine 3.

Engine 1 - non-stiff case: In the Engine 1 case all the volumes have been eliminated. Since these are the main source of stiffness (large negative real parts of the eigenvalues on the system Jacobian) the explicit solvers perform fairly well.

Engine 2 and 3: Judging from the performance measurements of the variable order Adams method and the forward Euler method the use of explicit methods for inter-component volume models seems completely out of the question. However, there is one less appealing remedy to this problem. By increasing the size of the volume giving rise to the pole with the largest negative real part the stiffness of the model can be reduced. Of course, such a trick could also introduce some unwanted physical effects [51].

A comment about the Forward Euler method: The reason for the fact that the forward Euler method is actually faster than the Adams method for Engine 2 and Engine 3 is that this implementation has no automatic stability control. The engine transients were recalculated repeatedly for the Euler method in order to obtain the largest possible step size without making the solution unstable. As a matter of fact the order used by the Adams method throughout the entire integration was one (this is the forward Euler method) and since the Adams code monitors the stability automatically it is reasonable for the method to be slower. However, in the Engine case 1, the problem is no longer stiff and the stability requirements will no longer limit the selection of order for the Adams method. Up to order 6 was then observed for the Adams code during integration.

4.7 "Real" performance code effects

The major difference between a real performance code and the nonlinear test equations, is that the performance code will most likely use a number of empirical maps for estimating the component performance. When interpolation methods are used on these maps discontinuities in higher derivatives will be introduced in the table break points. More refined codes will then use smaller step sizes around these points to get an accurate solution. This can reduce the performance of high order solvers quite drastically, as is shown below.

By the use of higher order approximating spline routines this problem can be alleviated, and in some cases measurement noise can be filtered away at the same time. The methodology for obtaining the spline approximations are described in great detail by Dierckx [60].

The GESTPAN simulation system also has a number of standard component performance models based on measurement data or more accurate models. A system with components corresponding to Engine 3 was set up (the same wiring diagram). In this case the model used 26 empirical tables. Three sets of spline coefficient approximations were defined for every table: linear, cubic and quintic approximating splines. The performance model was tuned to give roughly the same off-design performance as the analytical model and the code was started from close to the same initial conditions.

Since it is part of the conclusions of this thesis that the direct approach implemented in the DASSL solver is the most suitable approach for generalized

gas turbine simulation systems, it was selected for this study. Table 4.3 gives the results for the runs.

Smoothness	DASSL
Linear interpolation	10504
Linear spline approximations	9902
Cubic spline approximations	2387
Quintic spline approximations	2323

Table 4.3: Effect of map data smoothness on solver performance

It is seen that a poor interpolation method can increase the number of function evaluations between four and five times.

The DASSL solver can be set up to limit the order of the BDF method. This was done reducing the variable order BDF-method to the backward Euler method. This increased the number of necessary function evaluations to 17256 in the linear interpolation case and 15246 in the quintic spline case. This demonstrates that the benefits of high order derivative smoothness is small for low order BDF methods. Likewise, the benefits of high order BDF methods applied to models with poor smoothness is limited. In fact, if linear interpolation was used the optimal BDF for this problem was of order two.

4.8 Using Matlab and SIMULINK

Until recently, the methods for solving ODAE equations in SIMULINK were very crude indeed. As Shampine observes [27]: "These versions had a limited capability for solving models with algebraic loops, so users had to resort to ad hoc changes to models in order to solve DAE:s beyond the capabilities of the language". However, Shampine et al. [27] have now introduced improved methods for the solution of ODAE:s both in Matlab and SIMULINK [23], including a method using the direct approach for the Matlab environment. The SIMULINK environment still uses the "indirect approach".

Although the LSODE and DASSL solvers show comparable performance for the Engine 3 case, the DASSL code is about 4.4 times more efficient than LSODE in the Engine 1 case and 7.5 times in the Engine 2 case. This indicates that the indirect approach used by SIMULINK for dealing with ODAE:s corresponds to a considerable loss in performance compared to the direct method.

4.9 Chapter summary

The direct approach for solving the ODAE system arising from the acceleration test transient, has been shown to work very efficiently. The DASSL solver, which implements this approach, was about 4.4 times more efficient in the Engine 1 case and 7.5 times in the Engine 2 case, compared to the most efficient solver using the indirect method. Furthermore, the DASSL code solved the full inter-component volume model represented by Engine 2, using less than half the number of function evaluations required by the Adams method to solve the rotor dynamic model represented by Engine 1. This indicates that the justifiable use

of models with only rotor dynamics is limited to cases when a minimum of complexity is required.

It has been demonstrated that high order BDF techniques can give increased performance with as much as a factor of 6.6 compared to the BDF method of order one, i.e. the backward Euler method, if suitable methods for data interpolation are used.

As has been concluded by other authors, [51, 6], the performance penalties associated with using explicit methods for simulation of stiff gas turbine systems (Engine 2 and Engine 3) can be excessive. A reduction in speed with about a factor of 1500 is obtained for Engine 2.

The application of the direct method allows the complexity of the simulation system to be kept at a minimum, by using the same engine component formulations for all engine models maintained by the system, including both steady state and transient formulations.

Chapter 5

Optimization techniques

This chapter gives some background information on the optimization methods selected to solve the various optimization tasks described in this thesis. The first section discusses the relative benefits of classical and evolutionary algorithms. The following sections go on to describe the three optimization methods selected for implementation in GESTPAN. The description of the algorithms is intentionally kept short, and the interested reader is advised to study the referenced material.

5.1 Classical versus evolutionary algorithms

Basically two classes of optimization methodologies have been used for gas turbine optimization:

- Classical methods/hill-climbing methods
- Genetic Algorithms (GA:s)

The use of GA:s has its stronghold in "hard and small" problems, i.e. hard in the sense that the objective function may be discontinuous and multi-modal (many local optima), and small in the sense that a great number of function evaluations can be afforded within an acceptable time frame. Another way of expressing the usefulness of GA:s is to say that they work well on a broader class of problems but are generally much less efficient than classical algorithms.

Classical methods or hill-climbing methods use local information of the solution space to improve on a current solution. This makes them particularly suitable for problems with a few or, ideally, one local optimum. Another property of the problem indicating the suitability of classical algorithms is that the local behavior of the goal function is "nice", i.e. the function is smooth enough to allow the determination of gradients. Most traditional performance simulation tools do not, by default, fulfill the latter criteria. Frequently, polynomial interpolation is used to represent the empirical tables on which the models are based. The very powerful SQP algorithm, described below, will generally fail on such goal functions (the method requires C^2 continuity). To overcome this difficulty, all internal and external tables in GESTPAN have been represented using approximating splines [60].

If proper interpolation algorithms are used to represent empirical data, the strengths of the classical and the evolutionary algorithms can be combined (hybrid algorithms) to obtain very powerful and robust optimization methods.

5.2 Genetic Algorithms

The Genetic Algorithm implementation used for this thesis is based on a real encoding, in contrast to the traditional binary encodings. In the early days of genetic algorithms, the binary coded algorithms dominated the scene, owing especially to the pioneering work by Holland [63] and De Jong [64]. In 1990, however, Goldberg [65] made the following observation about real coded GA:s:

> The use of real-coded or floating point genes has a long, if controversial, history in artificial genetic and evolutionary search schemes, and their use as of late seem to be on the rise. This rising usage has been somewhat surprising to researchers familiar with fundamental genetic algorithm theory ([66], [63]), because simple analyses seems to suggest that enhanced schema processing is obtained by using alphabets of low cardinality, a seemingly direct contradiction of empirical findings that real codings have worked well in a number of practical problems.

More recent work by Houck et al. [67] and Michalewicz [68] confirm this trend. Extensive performance measurements indicate that real coded GA:s used for function optimization often give an order of magnitude in speed up as compared with binary coded GA:s [68]. Some recently published work on gas turbine optimization by Nadon et al. [69] used a Matlab implementation of a real coded GA [67]. The GA used for the simulations carried out in this thesis uses a modified Fortran 90 implementation of this code. The data structure was completely rewritten, since the original design was found to be unsuitable for this particular task. Also, a simple algorithm for handling failed design attempts, by a penalty method was integrated in the algorithm.

5.2.1 A Description of the algorithm

The basic algorithm for any evolution-based system [68] is given in Figure 5.1. For the particular GA implementation used here, normalized geometric ranking was used for the selection method. The alter step in Figure 5.1 is subdivided into one mutation and one crossover part. The mutation part uses uniform and non-uniform mutation and the recombination part uses simple and arithmetic crossover [68]. The detailed settings of the optimization parameters are described in Chapter 7.

5.3 The Nelder and Mead downhill simplex method

The downhill simplex method was introduced by Nelder and Mead [36]. The method relies upon carrying out a sequence of transformations of a geometric object, a so called simplex. The simplex consists of n+1 points (vertices) in n dimensions. By comparing the function values at the n+1 vertices and replacing

```
begin
    initialize population P(t)
    evaluate P(t)
    while (not termination-condition) do
    begin
        t <- t + 1
        select P(t) from P(t-1)
        alter P(t)
        evaluate P(t)
    end
end
```

Figure 5.1: Generic structure of a genetic algorithm [68]

the highest value by another point, a decreasing sequence of function values is created.

5.3.1 A description of the algorithm

Let $P_0, P_1, ..., P_n$ be the n+1 points in the n-dimensional space at the vertices of the simplex. Denote the function value at P_i with y_i and introduce:

$$y_h = \max(y_i) \tag{5.1}$$

$$y_l = \min(y_i) \tag{5.2}$$

Let $\bar{P}$ be the centroid of the points with $i \neq h$ and let $[P_i P_h]$ denote the distance from P_i to P_j. At each stage in the process, P_h is replaced by a new point. Three elementary operations are used to generate the downhill sequence:

- Reflection
- Contraction
- Expansion

The reflection of P_h is denoted as P^*, and its coordinate values are obtained through the following relationship:

$$P^* = (1 + \alpha)\bar{P} - \alpha P_h \tag{5.3}$$

Let, for instance, $n = 3$ and let the simplex be a regular tetrahedron; $\alpha = 1$ then corresponds to a mirroring of P_h through the plane formed by the other three base points (an $\alpha = 0$ will get you exactly to the centroid). If the resulting y^* lies between y_h and y_l, then P_h is replaced with P^* and the procedure is restarted. If y^* is less than y_l, an expansion procedure is carried out. If y^* is greater than y_h, a contraction procedure is carried out.

If y^* is less than y_l, we expand P^* to P^{**} by the relation:

$$P^{**} = \gamma P^* - (1 - \gamma)\bar{P} \tag{5.4}$$

The expansion coefficient, γ, is set greater than unity. It is defined as the ratio of the distance $[P^{**}\bar{P}]$ to $[P^{*}\bar{P}]$ (i.e. distances from the centroid to the expansion points). If the resulting y^{**} is less than y_l, then P_h is replaced with y_l and the procedure is restarted. If, on the other hand, y^{**} is greater than y_l the expansion process has failed and P^* must suffice as a replacement for P_h.

If reflection point P^* yields a y^* such that $y^* > y_i$ for all i not equal to h_m, then the new P_h will either be the old P_h or P^*, depending on which has the lower value. The contraction point, P^{**}, is thus defined as:

$$P^{**} = \beta P_h - (1-\beta)\,\bar{P} \tag{5.5}$$

The contraction coefficient, β, lies between 0 and 1 and is the ratio of the distance $[P^{**}\bar{P}]$ to $[P_h\bar{P}]$ (i.e. distances from the point to the centroid and from the point to the contraction point). Here we might encounter the worst case scenario for which the contraction point has a higher value than both points y^* and y_h. For such complete failure, the P_i:s are replaced by $\frac{P_i+P_l}{2}$ and the process is restarted.

The form used to test for convergence is to compare the "standard error" of the y:s in the form $\sqrt{\frac{\sum(y_i-\bar{y})^2}{n}}$ with a preset value. The success of the criterion depends on the simplex not becoming too small in relation to the curvature of the surface until the final minimum is reached.

5.4 Sequential Quadratic Programming - SQP

Sequential quadratic programming is a recently developed method for optimization, and it is generally a more direct and efficient method than the penalty function methods used (which have enjoyed a widespread use for constrained problems). Simply stated, the technique applies Newton's method to find the stationary points of the Lagrangian function by solving a sequence of quadratic optimization problems. Here, the NAG implementation was utilized, which is essentially identical to the SOL/NPSOL routines coded by [70].

5.4.1 A description of the algorithm

For a nonlinear optimization problem with only equality constraints, i.e. for a problem of the type:

$$\text{minimize } f(x)$$

$$\text{subject to } h_i(\mathbf{x}) = 0, \qquad i = 1, 2, ..., m$$

The Lagrangian to the system is given by $L = f(\mathbf{x}) + \sum_{k=1}^{m} \lambda_k h_k(\mathbf{x})$. The fourth Kuhn-Tucker requirement states [71]:

$$\nabla L = 0 \tag{5.6}$$

Together with the constraints, this represents n+m equations in n+m unknowns. If the Newton method is applied in solving this system of nonlinear equations, one obtains:

$$\left[\begin{array}{cc} \overbrace{\left[\begin{array}{ccc} \frac{\partial^2 f}{\partial x_1{}^2} + \cdots + \frac{\partial^2 f}{\partial x_1 \partial x_n} + \sum_{k=1}^{m} \lambda_k \frac{\partial^2 h_k}{\partial x_1{}^2} + \cdots + \sum_{k=1}^{m} \lambda_k \frac{\partial^2 h_k}{\partial x_1 \partial x_n} \\ \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} + \cdots + \frac{\partial^2 f}{\partial x_n{}^2} + \sum_{k=1}^{m} \lambda_k \frac{\partial^2 h_k}{\partial x_1 \partial x_n} + \cdots + \sum_{k=1}^{m} \lambda_k \frac{\partial^2 h_k}{\partial x_n{}^2} \end{array} \right]}^{\nabla^2 L} & [A]^T \\ [A] & [0] \end{array} \right]_j \left\{ \begin{array}{c} \Delta X \\ \Delta \lambda \end{array} \right\}_j = - \left[\begin{array}{c} \nabla L \\ h \end{array} \right]_j$$

where $A^T = (\nabla h_1, ..., \nabla h_m)$. If $\nabla^2 L$ is introduced as indicated above, we get:

$$\left[\begin{array}{cc} [\nabla^2 L] & [A]^T \\ [A] & [0] \end{array} \right]_j \left\{ \begin{array}{c} \Delta X \\ \Delta \lambda \end{array} \right\}_j = - \left[\begin{array}{c} \nabla L \\ h \end{array} \right]_j$$

If we introduce:

$$\begin{aligned} \Delta X_j &= X_{j+1} - X_j \\ \Delta \lambda_j &= \lambda_{j+1} - \lambda_j \end{aligned}$$

The above relations can be combined to yield:

$$\left[\begin{array}{cc} [\nabla^2 L] & [A]^T \\ [A] & [0] \end{array} \right]_j \left\{ \begin{array}{c} \Delta X_j \\ \Delta \lambda_{j+1} \end{array} \right\} = - \left[\begin{array}{c} \nabla f_j \\ h_j \end{array} \right]_j$$

This equation can be solved to find the change in the design vector, ΔX_j, and the new values of the Lagrange multipliers, λ_{j+1}. The iterative process can be continued until convergence is achieved. Also, the above equation can be seen to be equivalent to solving a quadratic problem with linear constraints. Very efficient methods have been developed to deal with these problems, see [72]. The extension to dealing with inequality constraints is also described in [72]. Further information on how to solve the system of nonlinear equations arising from the formulation above can be found in [72], [38] and [73].

Chapter 6

Trajectory optimization

This chapter combines some of the optimization methods described in Chapter 5 with the transient modeling technique outlined in Chapter 4. More specifically, a methodology for obtaining optimal control schedules of the Selective Bleed Variable cycle engine (SBVCE) during the mode switch transient was developed. The work is a continuation of previous studies of the performance of the engine [49, 47]. The chapter also describes the implementation of a multimode functionality suitable for variable cycle engine simulation. This functionality eliminates the need for a matching procedure, which is necessary when two or more engine designs are used to describe the operating modes of the variable cycle engine [47].

6.1 Selection of the Design Point

A detailed study of the design point optimization and cycle selection of the Selective Bleed VCE (see section 3.5 for a description of the basic operation of the engine) was done by Nascimento and Pilidis [47]. In that study, two separate engine models representing the subsonic and the supersonic modes were used in combination with a matching procedure that ensured that the two cycles corresponded to the same engine design.

Since the transient operation of the mode switch must use both nozzles actively, the separate engine approach was not suitable for this work. Instead, the engine design was carried out in the intermediate mode and a feature for dynamic connections to switch between the intermediate mode and two single nozzle modes was developed, see section 6.4 below. The design point, see Table 6.1, was selected in such a way that it would match as closely as possible the mission optimized design obtained by Nascimento and Pilidis [47].

6.2 Engine variable geometry and controls

The engine model has six variable geometry control signals: variable geometry in all three compressors as well as in the three nozzles. In addition, the fuel flow must be controlled in a suitable manner, giving a total of seven degrees of freedom for the control optimization. The variable geometry compressor model

Design parameter	Value	Design parameter	Value
h	0.0 m	π_{HPC}	3.60
M	0.0	η_{HPC}	0.88
π_{LPC}	3.0	m_{core}	170 kg/s
η_{LPC}	0.88	TIT	1500.0 K
bpr_1	0.62	η_{HPT}	0.90
π_{IPC}	2.40	η_{LPT}	0.90
η_{IPC}	0.88	$\pi_{nozzles}$	0.98
bpr_2	0.24		

Table 6.1: Design point of the Selective Bleed VCE

is identical to the one used in [46] to optimize the steady state control of the Selective Bleed VCE.

6.3 Selection of the transition point

The selection of the transition point could be made in a straightforward way. The specific fuel consumption (SFC) of the engine was minimized for a number of flight cases in the subsonic and in the supersonic modes. Subsequently, a suitable point for transition could be selected among these optimal cruise points, defining both the initial and the end point of the trajectory as well as the flight case at which the transition will occur.

To ensure safe and stable operation in the optimal cruise points, a number of constraints had to be imposed on the control optimization. These constraints are given in Table 6.2. The same definition of surge margin as the one used for optimizing the steady state controls of the Selective Bleed VCE, see [46], was used here:

$$\Psi = \frac{\pi - \pi_{choke}}{\pi_{surge} - \pi_{choke}} \tag{6.1}$$

By this definition, surge would occur for $\Psi = 1.0$. Thrust requirements for the entire flight mission of the STOVL aircraft have been given in Nascimento and Pilidis [47].

Ψ_{LPC}	$\leq$	0.8
Ψ_{IPC}	$\leq$	0.8
Ψ_{HPC}	$\leq$	0.8
TIT	$\leq$	1650.0 K
Net Thrust	$\geq$	134.0kN
Mass flow	$\leq$	Design mass flow

Table 6.2: Constraints

The selection of the transition point is of crucial importance to the successful control of the transient. For instance, if the rotational speeds of the subsonic mode cruise point and the supersonic mode cruise point differ considerably,

inertia of the rotors will make it very difficult to perform the transient within a reasonable time. Also, if the steady state cruise points selected are too close to the limits of safe and stable operation, the margin is likely to be insufficient for carrying out a successful mode transition. For this reason, a more conservative selection of constraints was made for the steady state cruise optimization, than for the mode switch transient.

The original design point optimization of the supersonic mode of the Selective Bleed Variable cycle engine was carried out for a flight Mach number of 1.2 and an altitude of 6000m [47]. This altitude was selected for the mode transition for comparative reasons. Several optimizations of the control settings were carried out to determine a suitable transition Mach number. The optimal cruise SFC curves for the two modes are shown in Fig. 6.1. Although

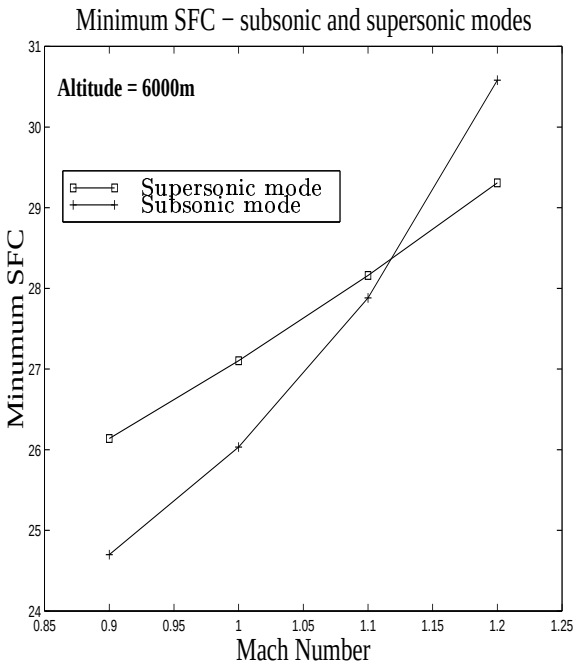


Figure 6.1: Minimum SFC for the two modes

Fig. 6.1 indicates that the most optimal point for transition is around M=1.12, the (h=6000m, M=1.2) flight case was the final selection of transition point since this flight case was the supersonic mode design optimization point and also part of the STOVL aircraft mission specification.

The optimization problem was solved using Sequential Quadratic Programming, see section 5.4. The NAG implementation of the algorithm, based on the work of Gill et al. (1986), was found to work very efficiently.

The variable geometry and fuel flow schedules for the two modes were selected by optimizing the cruise SFC for this flight condition. The optimal control settings are given in Table 6.3.

	Subsonic mode	Supersonic mode
LPC restagger	$-2.03696°$	$4.44104°$
IPC restagger	$12.2441°$	$-9.66672°$
HPC restagger	$-0.772156°$	$-0.531882°$
Front nozzle	0.331422 (m^2)	0.000000 (m^2)
Mid nozzle	0.000000 (m^2)	0.116615 (m^2)
Exhaust nozzle	0.456220 (m^2)	0.741889 (m^2)
Fuel flow	4.105108 (kg/s)	3.927331 (kg/s)

Table 6.3: Cruise optimized control settings for the transition point (M=1.2, h=6000m).

6.3.1 Control optimization of all flight cases in mission

Variable geometry optimization calculations were performed for all points in the original mission. It was noted that the highest turbine inlet temperature required for any of the flight cases was 1577 K. This indicates that the original engine cycle has been somewhat oversized.

6.4 The GESTPAN multimode functionality

The GESTPAN multimode feature is based on defining new modes of operation from one "super mode". In the super mode, all flow paths are defined simultaneously. Derivative modes of this super mode are then obtained by specifying the deletion of appropriate connections in the model. If all input connections of a module have been specified for deletion, the algorithm will automatically delete the entire module. There is no corresponding feature for adding new modules, which explains why all modes of operation must be defined from a super mode.

6.4.1 Implementation of the feature

After the super mode has been designed, mode switches to the sub-modes are defined as off-design points. Internally, all the connections of a GESTPAN engine model are represented by a connection array. Another array stores information about which engine components are part of the model. The process of performing a mode switch starts by a check of whether any of the engine components have had all their input connections deleted. If so, the component is deleted from the component array. Subsequently, all the elements in the connection array corresponding to the deleted connections are removed. The same kind of operation must be performed on the arrays storing the design point definition as well as model inputs. If connections are added to the system, these must be added to the appropriate arrays.

All the mode switch operations described above correspond to modification of the data that is normally created during the processing of the engine model input files. When this process has been completed, the routines normally run to establish derivative arrays of from the input data files have to be re-run. This re-running of the new configuration may, for instance, lead to a system with a

different number of iteration variables and residuals, than present in the super mode.

6.4.2 Definition of modes of the SBVCE

The multimode functionality made it very convenient to specify the subsonic and supersonic modes of the Selective Bleed engine with Table 6.1 defining the super mode. The subsonic mode was obtained by deleting all connections to the DUCT 1 and NOZZLE 1 modules, see Fig. 6.2 (note that the steady state model used for finding optimal cruise points has no rotor or volume components). The supersonic mode was obtained by deleting all connections to the DUCT 2 and NOZZLE 2 modules. To make the definition of the two sub modes complete a connection specifying constant and zero bypass ratio of the two splitter modules (SPLITTER 1 and SPLITTER 2) had to be given as well.

Another approach toward modelling the two engine modes would have been to use the super mode model directly, with very small areas to simulate closed nozzles. However, schedule optimizations using such a model were found to be virtually impossible to perform, owing to the restricted mass flow operating range of such nozzle designs. Small variations in the iteration variables used by the quasi-Newton solver, see Appendix B and Chapter 2, could result in an attempt to balance an engine with a negative flow through the engine nozzles.

6.5 Optimization of the mode transition

To model the transition trajectory, an inter-component volume, see section 4.1, model was assembled. The wiring diagram of the engine model is illustrated in Fig. 6.2. Numerically, the engine was represented by a differential algebraic equation system with 14 states and six equations. The procedure used to solve the equations was outlined in detail in Chapter 4.

The start and end points of the trajectory were approximated as intermediate mode points with nozzle areas 0.0001 times the areas given in Table 6.3. The optimal control schedules obtained for the subsonic and supersonic mode cruise points, were not corrected for this very small deviation in initial and end points.

6.5.1 Optimality criteria during mode transition

To ensure safe operation of the engine during the mode transition, all three compressors must operate well away from the surge line. Furthermore, the engine thrust should not drop below the aircraft thrust requirement. A small increase in thrust during the transient is probably in agreement with most aircraft system requirements. The surge margin requirement was relaxed for the transient mode switch and was limited to 0.95. Also, the turbine inlet temperature was allowed to reach 1700 K during the transient. Shaft inertias were set to $I_{LP} = 30\ kgm^2$ and $I_{HP} = 20\ kgm^2$, and volume sizes were all set to 0.0001 m^3. The selection of the rotor inertias and volume sizes is motivated below.

6.5.2 Linear interpolation of the schedules

The first attempt to control the engine during transition was made by a simple linear interpolation of the optimal schedules determined for the subsonic and

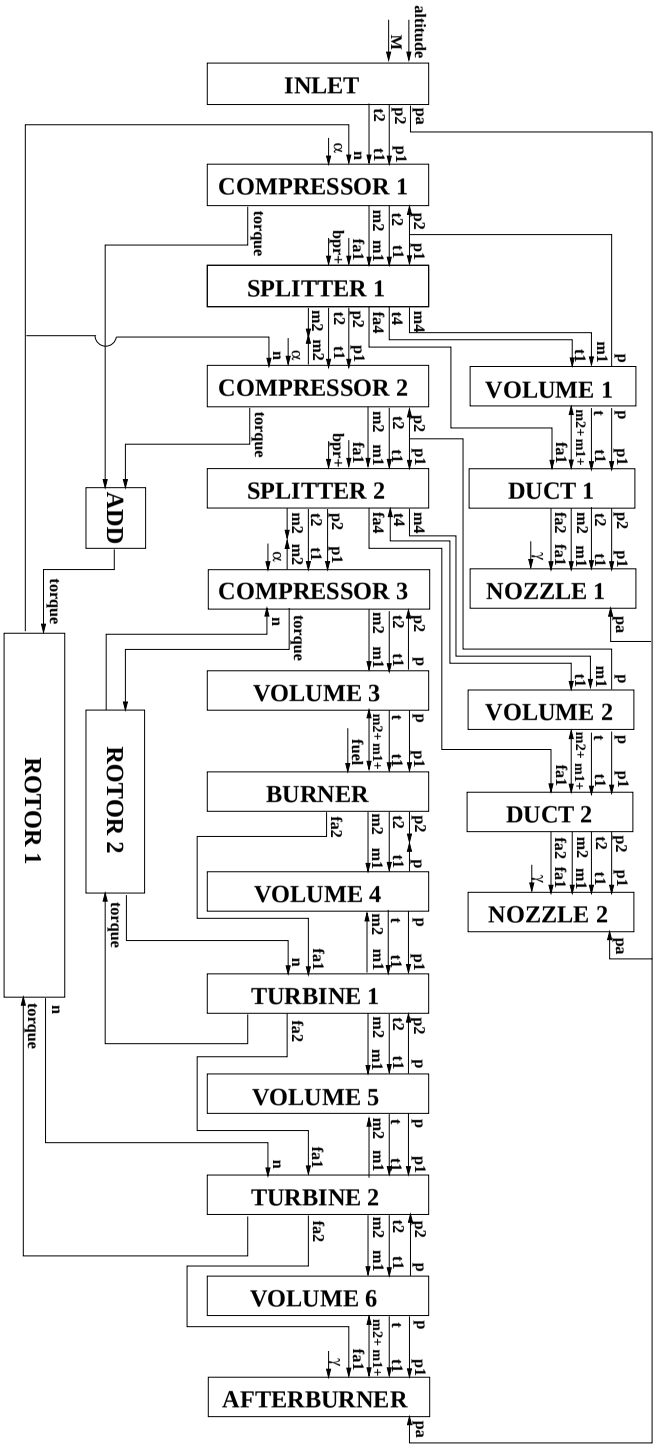


Figure 6.2: Wiring diagram of the transient Selective Bleed VCE model

supersonic mode cruise points. The engine was scheduled to switch within 0.5 seconds. The resulting surge margins, thrust and turbine inlet temperature trajectories are shown in Fig. 6.3. A drop in thrust can be observed during the first fraction of the transient, but the compressors operate well away from surge.

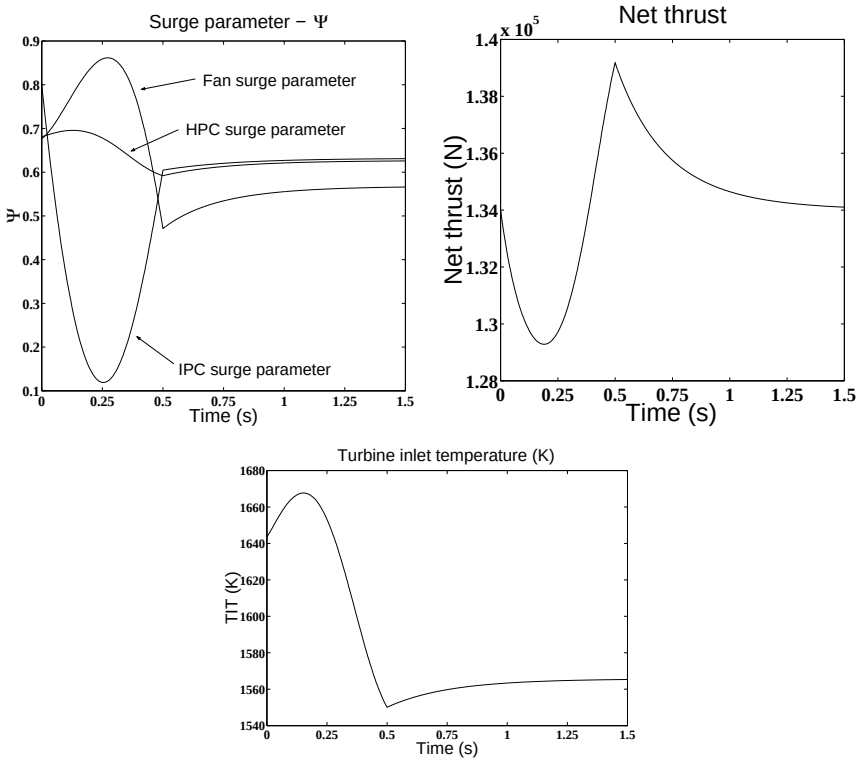


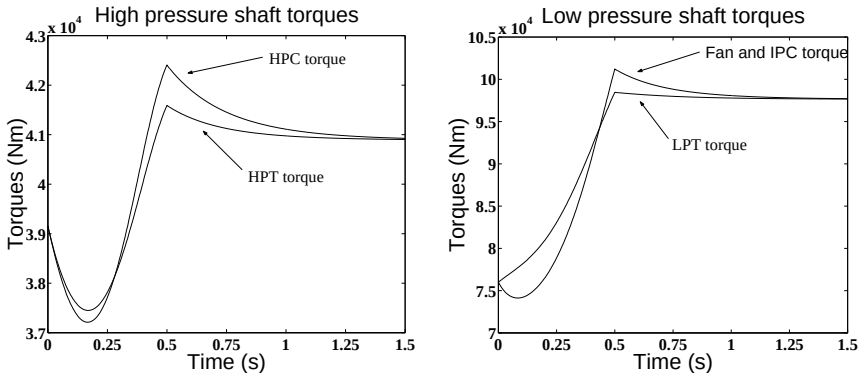
Figure 6.3: Constraint variables - linear scheduling of controls

6.5.3 Component modeling assumptions

The component off-design behavior is based on the models described in Appendix A. These correlations do not include any methods for estimating the moment of inertia of the shafts or the inter-component volume sizes. However, the dependency of the trajectory on the values selected for the shaft inertias and the volume sizes was observed to be very limited.

Dependency on shaft inertia

Since the high pressure shaft only changed its rotational speed from 153.7 rps to 158.1 rps and the low pressure shaft from 207.4 rps to 205.0 rps, the effect of uncertainty in estimating shaft inertias on the trajectory was very limited. The shaft torques for two simulations with different values on shaft inertia are shown in Fig 6.4. The proximity of the turbine and compressor torques also



(a) I_{LP}=15 kgm^2, I_{HP}=10 kgm^2 (b) I_{LP}=15 kgm^2, I_{HP}=10 kgm^2

(c) I_{LP}=30 kgm^2, I_{HP}=20 kgm^2 (d) I_{LP}=30 kgm^2, I_{HP}=20 kgm^2

Figure 6.4: Effect of doubling shaft inertias

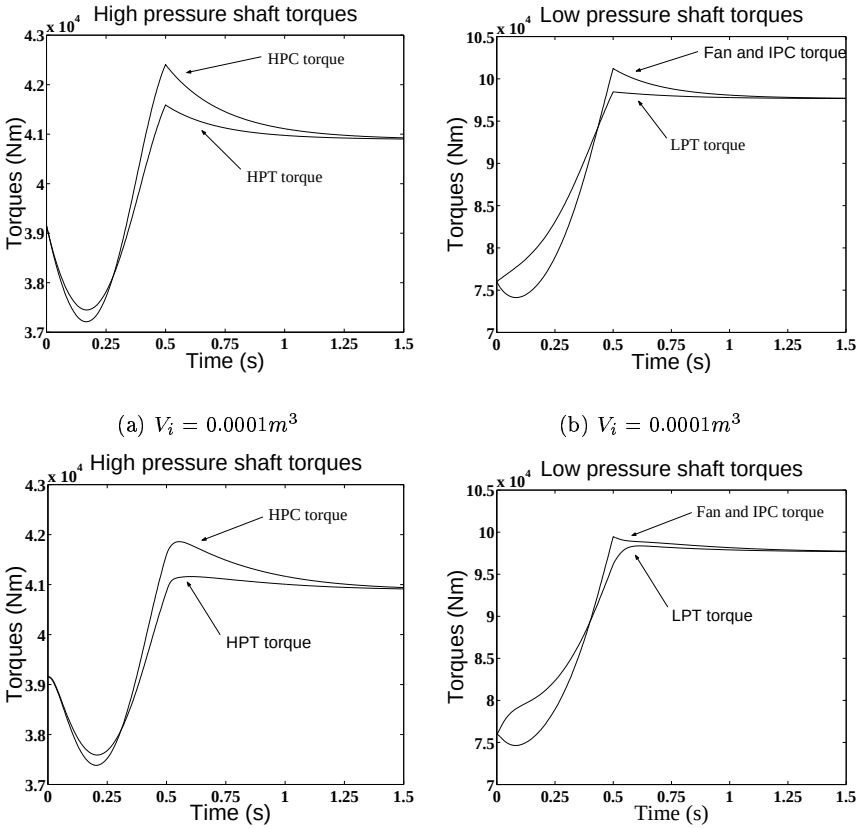
demonstrates that the intermediate points are relatively close to equilibrium points.

Dependency on inter-component volume sizes

Since the dynamics of the volumes is much faster than the time required for the mode switch, the effect of uncertainties in estimating volume sizes is also limited. Two simulations with volume sizes of $0.0001m^3$ and $0.5m^3$ are shown if Fig 6.5. Note that the implicit solver technique makes possible the use of very small volume sizes without any notable change in computational time.

6.5.4 Optimization of the trajectory

Since the linear interpolation of the control schedules produced an initial drop and a fairly large variation in thrust, some further development of the control methodology was undertaken. The initial 0.5 s during which the control scheduling occurs was discretized into three time intervals: (0,0.167), (0.167.0.333) and



(a) $V_i = 0.0001m^3$ (b) $V_i = 0.0001m^3$

(c) $V_i = 0.5m^3$ (d) $V_i = 0.5m^3$

Figure 6.5: Effect of change in inter-component volume sizes

(0.333,0.5). At t=(0.167,0.333), the values of the nozzle area schedules, the burner fuel flow and the IPC variable geometry parameter were allowed to vary by 20% around the schedules produced by the linear interpolation. The control settings at all the intermediate points could then be obtained using interpolating cubic splines (Dierckx, 1993).

The resulting control optimization problem thus had eight degrees of freedom. The constraints on the compressor surge margins, the turbine inlet temperature and the thrust trajectories form five nonlinear constraints. The goal function was formed by minimizing the maximum thrust during the mode switch. This selection of goal function would minimize the total thrust variation.

6.5.5 Final control settings

The surge margins, turbine inlet temperature and thrust trajectories resulting from the optimization are shown in Fig. 6.6, and the corresponding control parameters are shown in Fig 6.7. Both the fan surge constraint and the turbine inlet constraint are active, i.e. the maximum allowed value is obtained at some point along the trajectory. The optimization eliminated the initial drop in thrust and the total variation in thrust was reduced from 7.4% to 2.9%.

6.5.6 Further improvements

This methodology can be further refined by introducing a greater number of discretization points of the control variables and by using the fan, the HPC and the afterburner area variable geometry parameters as optimization variables. Too many discretization points might result in control schedules requiring variations in control signals to occur faster than the response times of the control system. Furthermore, such refined controls might be misleading if the accuracy of transient model is not sufficient. Great variations in the schedules can also cause convergence problems for the differential algebraic solver, e.g. some combination of control variables evaluated during optimization may cause some of the nonlinear component models to operate outside their region of definition. However, the present results are sufficient for demonstrating the usefulness of the method and showing that the transient can be controlled both safely and efficiently.

6.5.7 Conclusions

It has been shown that the mode switch of the Selective Bleed VCE can be carried out safely without violating surge and thrust constraints and that the time for the mode switch is of the same order as the actuator times of a typical hydraulic system. The modeling uncertainties introduced by the estimation of the inter-component volume sizes and the shaft inertias were also very limited on the trajectory selected for the mode switch. Furthermore, it was observed that the thrust requirements set by the aircraft mission can be fulfilled at a considerably lower turbine inlet temperature than has been reported by previous authors. This indicates that, by an optimal use of the control system, the present design can be downsized. This matter is addressed in the next chapter.

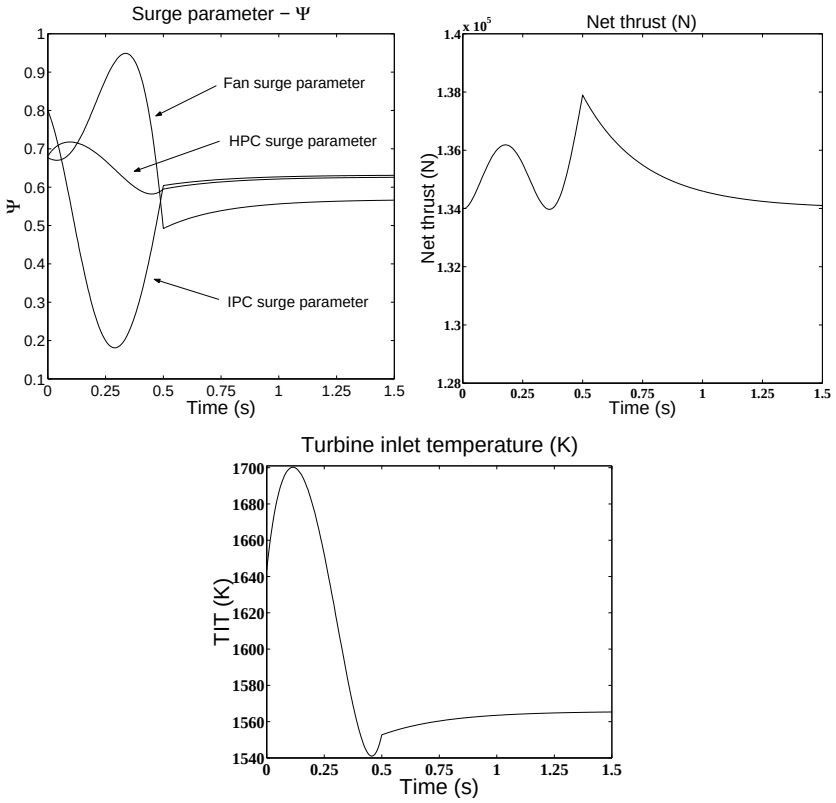


Figure 6.6: Constraint variables - optimized scheduling of controls

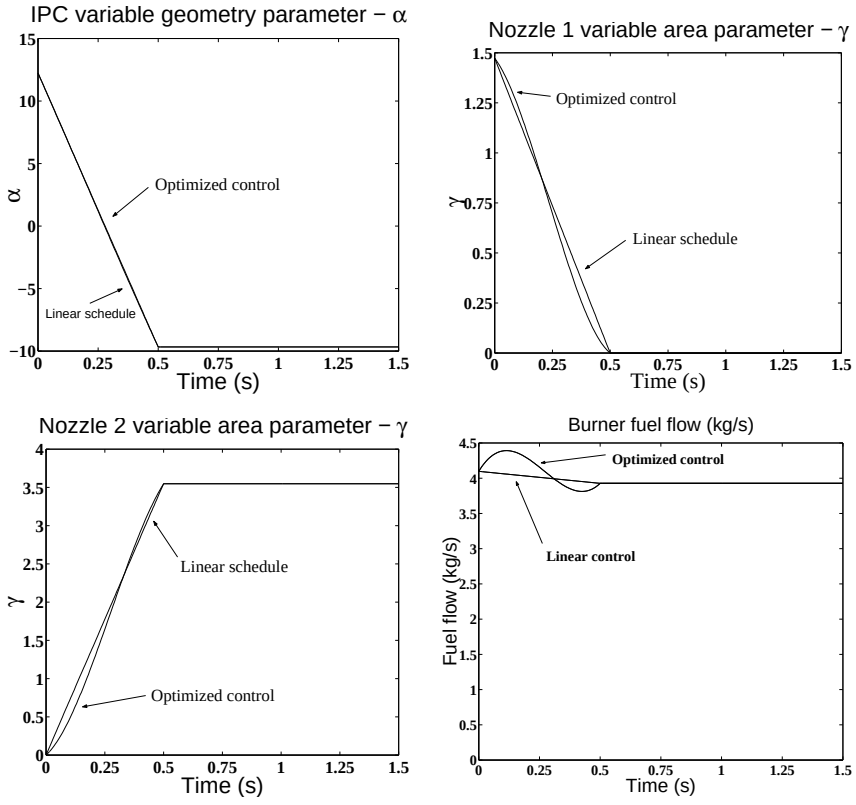


Figure 6.7: Control schedules

Chapter 7

Mission Optimization

This chapter describes some recent work on mission optimization of the Selective Bleed variable cycle engine. The coupled problem of selecting an optimal cycle design point and controlling the variable area schedules of the engine is studied. The simulations indicate that the engine cycle originally proposed in [47, 48, 46] can be downsized considerably and still fulfill the mission requirements.

7.1 The mission optimization problem

The mathematical formulation of the problem is:

$$\text{minimize } f(\mathbf{x})$$

subject to:

$$\begin{cases} h_i(\mathbf{x}) = 0, & i = 1, 2, ..., p \\ g_j(\mathbf{x}) \leq 0, & j = 1, 2, ..., m \end{cases}$$

where f is the goal function defining the function to be minimized and h_i and g_j are equality and side constraints respectively. The vector $\mathbf{x} = x_1, x_2, \ldots, x_n$ contains the independent variables which form the search space.

7.1.1 Goal function definition

A crucial step for the formulation of any optimization problem is how to define the goal function. For a jet engine a great number of parameters could be argued to be relevant for inclusion in the goal function, such as the weight and cost of the engine, life of the engine components and specific fuel consumption. Weight factors could then be set to balance the influence of the different parameters. Here, the most obvious and simple choice was made, a time weighted SFC average, $\overline{\text{SFC}}$, according to

$$\overline{\text{SFC}} = \frac{\sum_{i=1}^{n} t_i \cdot \text{SFC}_i}{\sum_{i=1}^{n} t_i} \tag{7.1}$$

where t_i represents the time spent operating in a specific flight leg of the mission.

7.1.2 Constraint definitions

Constraints are introduced to ensure that the engine operates with an adequate surge margin, and that temperature limits such as the turbine inlet temperature or the high pressure compressor exit temperature are maintained. Additionally, the thrust requirements in the different flight legs of the mission are represented as constraints.

7.2 A hybrid method for mission optimization

Selecting methods for optimization of nonlinear problems is generally a very interesting matter. As stated in [74]:

> It's unrealistic to expect to find one general nonlinear programming code that's going to work for every kind of nonlinear model. Instead, you should try to select a code that fits the problem you are solving. If your problem doesn't fit in any category except 'general', or if you insist on a globally optimal solution (except when there is no chance of encountering multiple local optima), you should be prepared to have to use a method that boils down to exhaustive search, i.e., you have an intractable problem

Exhaustive search is never a very attractive solution procedure, especially in a multidimensional search space. This section discusses some of the characteristic properties of the gas turbine system and how this influences the choice of suitable optimization methods.

7.2.1 Properties of the gas turbine solution space

One characteristic difficulty associated with gas turbine design optimization is that the nonlinear equations governing the design process frequently have no solution. To illustrate this, the variation in SFC with fan pressure ratio and turbine inlet temperature for the turbofan cycle described in Chapter 2, has been plotted in Figure 7.1. For a particular fan pressure ratio only a limited range on the turbine inlet temperature will allow successful mixing of the flow in the unifier (see governing equations in Appendix A.7). Combinations of TIT and FPR for which no solution exist are indicated in the figure as having zero SFC. Thus, an optimization method used to explore the gas turbine design space must have the ability to deal with failed solutions.

7.2.2 Classical methods and smooth functions

One reason for the increasing use of GA:s (see section 5.2), in the engineering community relates to their ability to optimize multi-modal functions (functions with many local optima). Equally important is probably the inability of the more powerful classical methods to deal with "noisy" functions [5]. A traditional gas turbine simulation code uses tabulated data to assess the system performance (see Appendix A). Unless special care is taken to represent these tables properly, discontinuities in the derivatives of the interpolating functions might be introduced in the table knots.

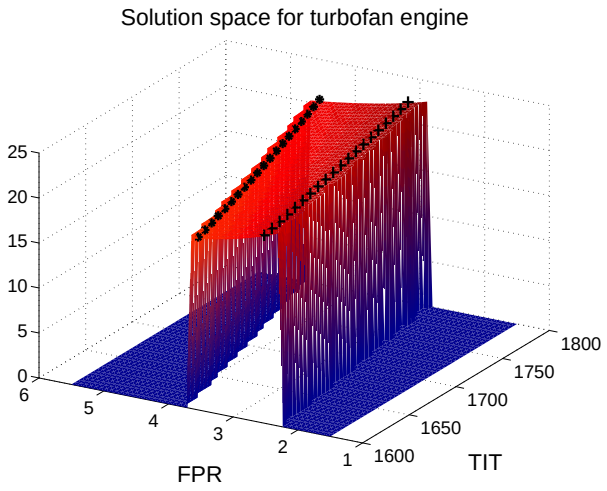


Figure 7.1: Solution space for turbofan engine

If a powerful classical optimizer, such as the SQP method (see section 5.4), is applied to an existing performance tool, one should expect failure rather than success. Unless it is considered worthwhile to go through the entire code and, wherever necessary, replace existing interpolation methods with methods giving the code C^2 continuity (continuity in the second derivatives [required by the SQP method]), it is probably better to attempt to use a method that does not rely on numerical estimation of derivatives. Such methods are called zero order methods. A classical zero order method is the Nelder and Mead Simplex method [36]. Genetic algorithms are also zero order methods.

7.2.3 The method

To combine the advantages of genetic algorithms and classical optimization techniques, a hybrid method for gas turbine optimization is suggested. The method uses an SQP implementation, see [76], to control the engine in the mission legs, and a real coded GA for the cycle selection. The method is illustrated in Figure 7.2.

As shown in Figure 7.2, the first step in the evaluation of an engine individual is to attempt to design the engine for the cycle parameters suggested by the GA. If the design process fails a logical "failure-flag" is set true and the control is subsequently returned to the GA optimizer. After every completed generation the failed engine designs of the population are ascribed a fitness value equal to a fraction of the fitness for the worst successful design. Since the GA optimizer seeks to find the global maximum of the goal function, the $\overline{\text{SFC}}$ value is inverted before control is returned to the GA.

If the design process is successful, an attempt to complete the mission is made. All the flight legs of the mission are executed in sequence. Since the engine components of the engine system have limited regions of operation, it is usually necessary to make the change from the design point to the user specified mission legs in several steps. Thus, optimal controls are found for a number of intermediate points, in order to reach the final mission leg.

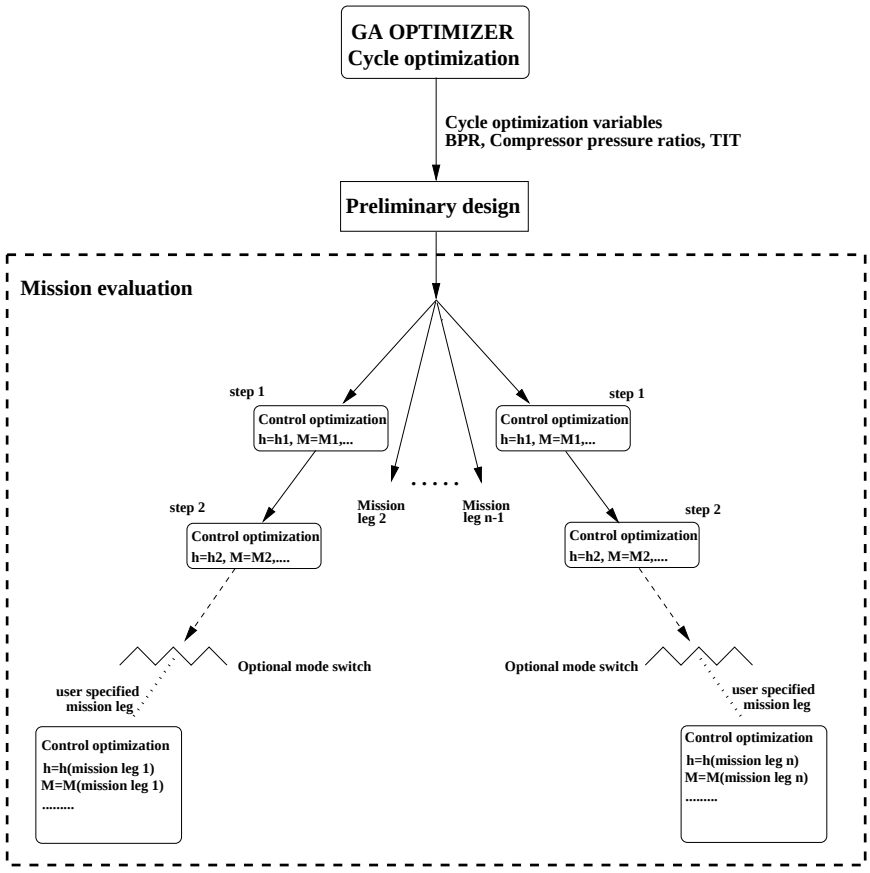


Figure 7.2: Schematic of the hybrid mission optimization methodology

	Alt.(km)	Mach No.	Thrust	Time	Flight leg no.
Take-off	0.0	0.00	140	1 min	4
Subsonic cruise	9.0	0.78	14	25 min	2
Loiter	9.0	0.65	13	60 min	3
Supersonic cruise	9.0	1.50	100	10 min	1
Subsonic cruise	9.0	0.78	14	38 min	2
Vertical landing	0.0	0.00	140	1 min	4

Table 7.1: STOVL aircraft mission [48].

7.3 Mission specification

The operation of the Selective Bleed VCE has been described in Section 3.5 and in Chapter 6. The original mission for which the power plant was optimized is given in Table 7.1. The flight leg numbers are used to group mission legs with the same specification together.

7.4 Formulation of the optimization problem

The number of parameters involved in cycle optimization studies are generally rather limited [30]. For the Selective Bleed VCE, six parameters have been included in the optimization process. These are given in Table 7.2. Experiments with larger populations and larger variable domains served to isolate variable ranges for which successful mission completion was possible. Additional input data necessary to define the design point were taken from Table 2.6, where the given efficiencies were used as polytropic efficiencies.

Parameter	Range
Fan pressure ratio	2.0-6.0
Forward bypass ratio	0.1-1.0
IPC pressure ratio	2.0-4.0
Rear bypass ratio	0.1-0.6
HPC pressure ratio	2.0-6.0
Turbine inlet temperature	1500.0-1800.0 K

Table 7.2: Cycle optimization variables for the Selective Bleed VCE

In order to optimize the engine performance in the four flight legs a number of controls were introduced. The exhaust nozzle area, the fan and IPC variable geometry parameters, the fuel flow and one of the forward and the rear nozzle areas (depending on which mission leg that was being executed), were simultaneously controlled, to optimize mission leg SFC and to satisfy the constraints. The HPC variable geometry control was found to vary very little when it was part of the control optimization process, and was therefore not used as a cycle optimization variable.

The thrust requirements of the mission flight legs are given in Table 7.1. To ensure safe and stable operation of the engine, a number of additional constraints had to be introduced, see Table 7.3. The $\phi_{rel} = 1.0$ parameter guarantees that

Constraint Variables
$\phi_{rel,FAN} = 1.0$
$\phi_{rel,IPC} = 1.0$
$0.4 \leq$ Thrust split ≤ 0.6 (flight leg 4)
Turbine inlet temperature $\leq$ 1800 K

Table 7.3: Constraints for the mission

the fan and the IPC operate in a stable region (see Appendix A). The thrust split constraint is required in the take-off and landing legs, in order to maintain aircraft stability [75].

7.4.1 Mission leg evaluation

The initial values for the optimal control settings were obtained as output from the design process. These are given in Table 7.4.

Variable	Initial value
α_{FAN}	0.0
α_{IPC}	0.0
Fuel flow (kg/s)	Design point fuel flow
$A_{front,nozzle}$ (m^2)	Design point area
$A_{rear,nozzle}$ (m^2)	Design point area
$A_{exhaust,nozzle}$ (m^2)	Design point area
Mode of operation	Intermediate operation

Table 7.4: Initial settings on the control parameters

Since both the subsonic and the supersonic modes of the Selective Bleed engine operate with one of the nozzles closed, a mode switch has to be performed during the evaluation of the mission legs. The switch is carried out immediately before the final step in the leg.

For the problem studied here, steps in the altitude variable of 500-1000 m were found suitable. Mach numbers were changed in the same fashion, with step sizes ranging from 0.0 to 0.2. Furthermore, for every new step the closing nozzle area was reduced to 80% of the previous value. Thus, as the mode switch was performed and the nozzle was closed completely, only a modest change in the iteration variables was necessary to obtain a balanced engine. To limit the change in the control variables between the steps, a maximum variation of 80% to 120% of the value obtained in the previous step was allowed. For the number of steps used here (8-18), the maximum possible variation was quite large. For this reason, these bounds on the control variables were rarely found to be active when the user specified mission leg was reached.

7.4.2 GA and SQP settings

The only non-standard setting for the SQP implementation used here (E04UCF routine in [76]), was to reduce the precision requirement on the optimality tolerance to 1.0E-5. This tolerance is more than sufficient, considering the accuracy of preliminary design models. The settings of the GA are found in Table 7.5.

Parameter	Value
Population size	100
q (normalized ranking parameter)	0.02
b (shape parameter for non-uniform mutation)	2
Arithmetic crossover frequency	10
Simple crossover frequency	10
Uniform mutation frequency	5
Non-uniform mutation frequency	5

Table 7.5: Settings on the control parameters of the GA (see [67, 68] for a definition).

The failure factor was set to 0.75, i.e. all failed designs were given a fitness value equal to 75% of the worst successful design in the population. The initial

100 engine individuals were selected from 500 engines, randomly distributed in the allowed design space (see Table 7.2).

7.5 Simulation results

Since this study focused on reducing the mass flow of the original Selective Bleed engine design given in [48], a series of optimization studies were carried out to determine a suitable design point mass flow that allowed the thrust requirements of the engine to be met. The rate of successful mission completion for the initial population decreased from 85% at 300 kg/s to 15% at 240 kg/s. At 200 kg/s only three out of the initial 500 mission evaluations were successful. The cycle parameters of the design with the lowest $\overline{\text{SFC}}$ of these three engines, are given in Table 7.6. The effect of the supersonic cruise constraint (flight leg no. 1) on the cycle parameters is evident. The relatively low compressor pressure ratios, in particular over the HPC, and the low turbine inlet design temperature give a large margin for energy input at the supercruise condition.

Parameter	Value
Fan pressure ratio	4.4302
Forward bypass ratio	0.8651
IPC pressure ratio	2.2479
Rear bypass ratio	0.1969
HPC pressure ratio	2.2120
Turbine inlet temperature	1524.0 (K)
$\overline{\text{SFC}}$	21.34 (mg/Ns)

Table 7.6: Best successful design of the initial population for a design mass flow of = 200 kg/s

Since the best engine in the initial population of the 240 kg/s case had an $\overline{\text{SFC}}$= 15.30 mg/Ns and in the 300 kg/s population the corresponding value was $\overline{\text{SFC}}$= 14.81 mg/Ns, the 240 kg/s population seemed to offer the most promising compromise between fuel consumption and size. This population was run for additionally ten generations. The resulting optimal cycle point is given in Table 7.7.

Parameter	Value
Fan pressure ratio	3.8764
Forward bypass ratio	0.5423
IPC pressure ratio	3.8930
Rear bypass ratio	0.1579
HPC pressure ratio	5.5184
Turbine inlet temperature	1517.65 (K)
$\overline{\text{SFC}}$	14.497 (mg/Ns)

Table 7.7: Best cycle point after 10 generations

The operation of the engine in the four flight legs are summarized in Ta-

ble 7.8. As seen from the table, the thrust requirement of flight leg no. 1 pushes

Flight leg no.	1	2	3	4
α_{FAN}	-0.4102994	-7.8295991	-8.0380587	-3.2515431
α_{IPC}	-10.0000000	18.1336900	23.6526579	5.5060196
Fuel flow (kg/s)	2.4169096	0.4274185	0.3562629	1.4240298
$A_{front,nozzle}$ (m^2)	—	0.325485	0.336297	0.208032
$A_{rear,nozzle}$ (m^2)	0.031777	—	—	—
$A_{exhaust,nozzle}$ (m^2)	0.186451	0.175372	0.173630	0.113489
Mode of operation	Supersonic	Subsonic	Subsonic	Subsonic
TIT (K)	1798.2556766	1237.4286671	1190.3994594	1483.3560793
SFC (mg/Ns)	24.1690969	14.3759732	13.1559524	10.1716531

Table 7.8: Optimal control settings in the flight legs

the engine close to the temperature limit. Also note how the cruise and loiter points favor high flowing the forward nozzle to obtain better propulsive efficiencies, whereas in the supersonic point the specific thrust is maximized by high flowing the core.

7.6 Discussion

For the original Selective Bleed engine design point, proposed in [48], the total mass flow was set to 370 kg/s. The aim of the mission optimization studies carried out in this chapter have been to show that the design point mass flow of this engine can be reduced considerably, still fulfilling the mission requirements. This mass flow reduction has been achieved by integrating the control of the engine variable geometry into the cycle selection process. It was found possible to satisfy the mission thrust requirements for a design point mass flow of 200 kg/s, but this was achieved at a relatively high $\overline{\text{SFC}}$ value. A design mass flow of 240 kg/s seems to be a more suitable compromise between engine size and performance.

Chapter 8

Summary of papers

The work reported in Paper 3 and Paper 4 was described in Chapter 4 and Chapter 6, respectively. This chapter summarizes work carried out early in the project reported in Paper 1 and Paper 2.

8.1 Refining component modeling - Paper 1

Several physical phenomena occurring in gas turbines must be addressed on a system basis in order to be modeled. For instance, the estimations of engine life, performance and stability of operation all require a system approach. To successfully address questions such as these, the zero-dimensional component-map-based models traditionally in use for system analysis will generally not be sufficient. 2D, 3D or even 3D transient models integrating highly tuned heat transfer simulation capabilities may be necessary. However, simulation of a complex system such as a gas turbine engine using full 3-D viscous models will require massive computational resources. While the physics governing these phenomena may be captured by modeling the system on the highest possible level of complexity, two problems will prevent this from being a feasible approach:

- The amount of detailed input data, such as boundary and initial conditions needed to obtain converged and validated solutions will be enormous.
- The computational time and cost will be excessive.

Thus, the conclusion is that the analyst must tailor the level of approximation to the simulation task, capturing the appropriate physics for each component and integrating it into the system/subsystem simulation. For instance, assessing the effects of adding a fan stage to engine performance may be solved by using a three-dimensional simulation of the fan stage and letting the rest of the engine be modeled at a zero-dimensional level to minimize simulation set-up and computation time.

The following sections will deal with component refinement models developed in this project, and to future work on these. Starting with a discussion on the experience gained porting a streamfunction code to ta44/ta45 will help to define and make apparent the limitations of these techniques. The following sections

will then describe how these methods were applied to the double bypass VCE and indicate how component models should be tailored to serve the needs of preliminary performance estimation of advanced engine concepts.

8.1.1 Streamline codes for variable geometry simulation

Variable compressor geometry in jet engines

Using variable stator vanes is an efficient way to maintain high performance of axial compressors when operating away from the design point. Achieving satisfactory operation often necessitates the combined use of scheduled variable stator vane resetting and interstage bleed. These measures are taken not only to improve internal compressor performance but also to ensure a satisfactory integrated operation, i.e. preventing the compressor from running into surge during start-up.

Extensive rig testing is normally done to find optimal stator vane schedules and bleed levels on new compressor designs. Several papers have been devoted to developing efficient testing algorithms based on multidimensional optimization procedures, such as [77], [78]. The aim has been to minimize the number of test runs in order to find stator vane and bleed settings with satisfactory performance.

The following sections outline a method developed for using the through-flow equations (see [79], [80]) in order to optimize off-design stator vane settings for achieving maximum isentropic efficiency. The method is applied to the optimization of a seven-stage axial compressor from a jet engine perspective. For more extensive treatment concerning this work, see [80].

The through-flow equations have been used extensively in the past for multistage axial compressor design optimizations [81], [82]. The optimization task has in those cases been restricted to a separate compressor component, and the compressor has not been integrated into a jet engine. Egorov [83] also considered the design optimization problem from an engine integration viewpoint. In that particular case, it was shown that consideration of the integrated performance of the compressor unit was more important than assuring the highest efficiency of the component. Egorov did not address the question of stator vane optimization.

Selection of backbones - optimization points

The optimization method developed finds the stator vane settings that give the maximum isentropic efficiency for a certain combination of corrected mass flow

$$m_{corr} = \frac{m\sqrt{\theta}}{\delta}$$

and corrected rotational speed

$$N_{corr} = \frac{N}{\sqrt{\theta}}$$

For every corrected rotational speed line, there is one point with maximum isentropic efficiency. If these points are connected to form a curve, the so called backbone of the compressor map is formed. Thus, the optimization procedure is

able to find the stator vane settings for a certain specified backbone. The level of the isentropic efficiency and the pressure ratio distribution along this backbone are not known initially but are rather a result of the optimization. The curve specifying the backbone can be chosen arbitrarily (within the limits set by the flow field). In the case studied in [80], five parabolas, all running through the compressor design point, were selected. These are shown in Fig. 8.1.

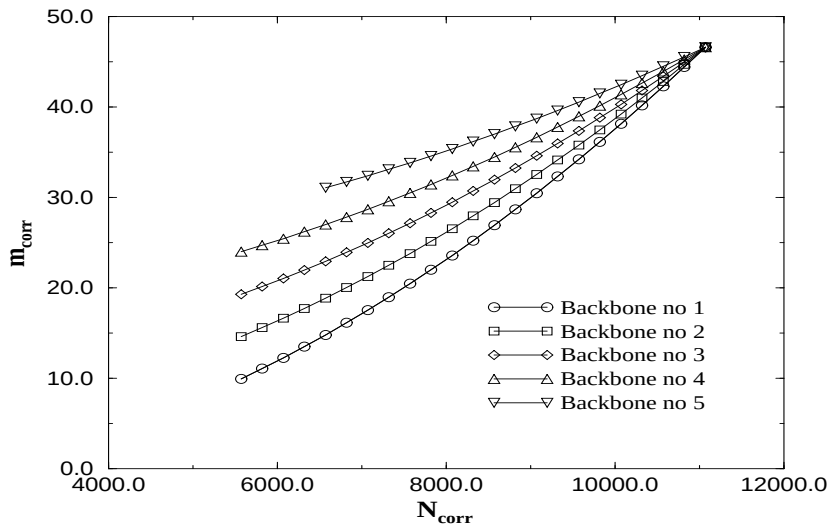


Figure 8.1: Selected backbones

For every corrected rotational speed, the level of the isentropic efficiency achieved by the optimization procedure will depend on the corresponding corrected mass flow that is selected. If the compressor is operated in isolation and not integrated into a jet engine, a straightforward choice of backbone would be the one with the optimal corrected mass flow (yielding the highest level of isentropic efficiency) for every rotational speed. In this case, that curve was found to lie between "backbone no 2" and "backbone no 3", i.e. a backbone selected in this region will produce a compressor map with the highest efficiency maximum.

The optimal choice in a real engine case is governed by the matching to the turbine system. If the turbine stators and the exhaust nozzle operate choked, the position of the running line in the compressor map will depend solely on this system. Thus, the optimal choice will be the compressor map for which the running line lies in the region of highest isentropic efficiency (still offering acceptable stall margins). For the engine studied in [80], "backbone 1" offers the best engine integration. The peak efficiency of this map is lower than that generated by "backbone no 2" and "backbone no 3". Thus, a scheduling producing high efficiencies along the backbone is useless if the engine operating points fall into regions of low efficiencies. While the change in **peak** efficiency between the five backbones was not great, the change in engine SFC, depending on the compressor integration, going from the best to the worst option, was as

large as 3.5%. The crucial factor for engine SFC is thus where the running line is located in the compressor map (the closer to the peak efficiency region, the better).

Evaluation of the streamfunction method

The streamfunction method used for this work offered a compromise between short running times and detail of prediction suitable for optimization calculations and integration into performance tools. However, the streamfunction through-flow method is very sensitive to high relative Mach numbers. This is due to its inherent assumption of a non-unique relation between the streamfunction and the density. The method therefore fails whenever there is a region with a local relative Mach number greater than unity. In many jet engines, the axial inlet Mach number and rotational speed are low enough for the entire high pressure compressor to operate subsonically in large regions of the compressor map, as long as good interstage matching is maintained.

8.1.2 Refinements needed for the double bypass VCE

To fully exploit the performance benefits of the double bypass variable cycle engine, variable geometry of the stator vane in the CDFS must be introduced. When the engine shifts from double to single bypass mode, the whole flow must go through the IPC. To accommodate this flow, the compressor has to have variable stator vanes.

Design of a one-stage IPC

Since the compressor components of ta44/ta45 did not contain any features for predicting of variable geometry effects, and because this was found to be of crucial importance to the operation of the VCE [18], new compressor tables were evaluated using two streamline codes [81], [84]. The design code [81] was used to design a one stage compressor with a design pressure ratio of 1.35. IGV and EGV schedules were then established to attain maximum isentropic efficiencies in the open and closed modes. A multidimensional optimization procedure was used to find optimal stator vane settings [80]. The closed CDFS mode was applied when the VCE operated in double bypass mode, and the opened CDFS mode was used during operation in single bypass mode.

8.1.3 1D models tailored for GESTPAN

Existing turbine and compressor performance models use general trends and empirical databases to predict how a design with a certain performance would behave going off-design. Attempts are currently being made in this project to include 1D design/off-design tools with loss modeling into the code. One goal is to be able to carry out preliminary 1D design in the optimization runs and to use these designs to improve estimations of losses in off-design. In addition, the ability to obtain useful weight estimations of a certain engine will more likely be successful if some of the engine geometry and the number of stages in compressor and turbines are known.

8.1.4 Time marching through-flow codes for GESTPAN

The main setback in the streamline and streamfunction models are their inability to handle choked flows and to accurately predict shock losses. These problems can probably be solved using a time marching Euler solver. To develop such a code and tune it against measurement data is a tedious task beyond the scope of this work, although work on these issues is being carried out at Chalmers University of Technology and at Volvo Aero Corporation [85].

8.2 The double bypass VCE - Paper 2

A comparative study of two jet engine concepts was carried out and reported in the second paper of this thesis. A conventional jet engine and the double bypass VCE (described in the introduction) were compared performing a flight mission, see section 4.2.

Previous studies on the optimization of variable cycle engines [86], [41] reported in the literature have been limited to finding optimal off-design scheduling of variable geometry components for a specific engine design point. This study also addressed the issue of selecting the optimal design point as a function of the flight mission being performed.

The double bypass VCE achieved lower time weighted SFC values, $\overline{\text{SFC}}$, throughout the whole spectrum of missions. This can be seen in Fig. 8.2.

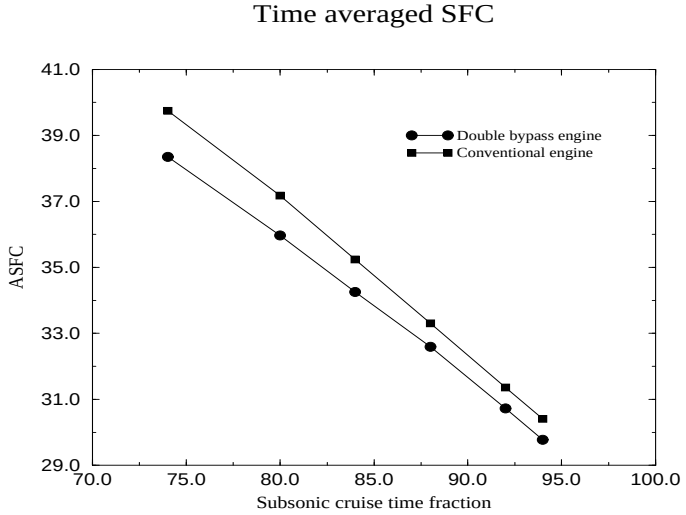


Figure 8.2: Mission optimized ASFC

The improvement ranged from a 3.5% decrease in $\overline{\text{SFC}}$ for the high subsonic cruise fraction missions to a 2.1% improvement in the low subsonic cruise fraction end. The influence of the mission on the selection of optimal design parameters is illustrated in Fig. 8.3. For subsonic cruise time fractions less than about 75%, the optimal design bypass ratio dropped off quickly, indicating a turbojet cycle as an alternative to both engine types studied here.

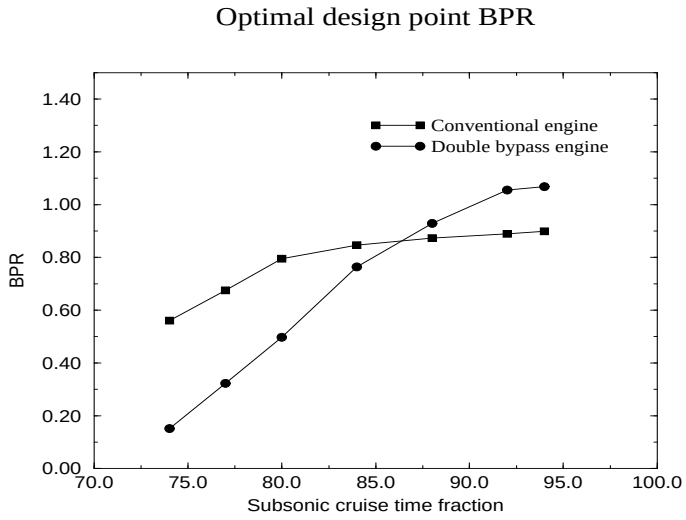


Figure 8.3: Optimal design parameters

8.2.1 Variation in turbine inlet temperature

For the missions with more supersonic intercept time, a drastic drop in turbine inlet temperature was noted in the subsonic cruise mode, see Fig. 8.4, for both engines, although especially for the double bypass VCE. This will have a beneficial impact on turbine life. By increasing the turbine inlet temperature of the VCE, the improvement in $\overline{\text{SFC}}$ may as an alternative have been increased for these missions.

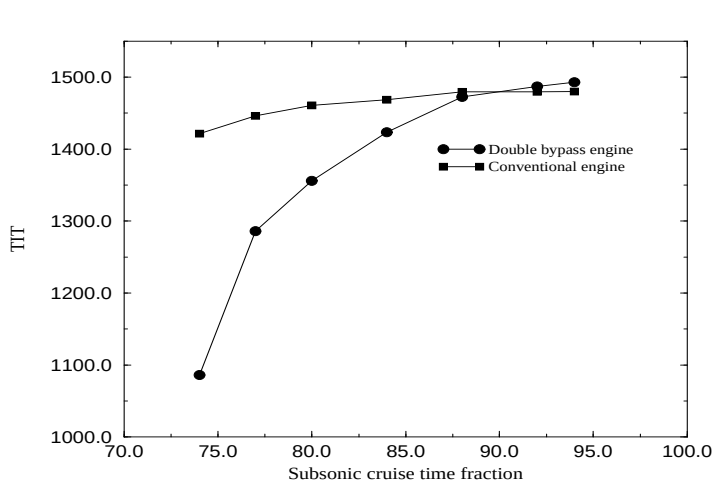


Figure 8.4: TIT variation (subsonic cruise mode)

Fig. 8.4 also reveals the high dry thrust capacity of the double bypass VCE. For supercruise operation (supersonic operation with unlit afterburner) with subsonic cruise time fractions less than 75 %, over 300 K extra turbine inlet

temperature would be available for thrust generation in comparison with the conventional engine.

See [18] for a more detailed description of the simulations carried out in the comparative study of the double bypass VCE and the conventional engine.

Chapter 9

Conclusions

The research effort contained in this thesis focuses on the development a generalized gas turbine simulation tool, capable of design, off-design and transient simulation. The main findings of this work are:

- The inverse design method allows the same engine component equations to be used for design, off-design and transient engine simulation.
- High order BDF methods for the direct solution of ODAE systems have shown to be very efficient for gas turbine transient simulation.
- The importance of suitable interpolation schemes for data representation in gas turbine system simulation tools has been demonstrated for a number of applications.
- Simulations have been performed which indicate that the mode transition of the Selective Bleed variable cycle engine can be performed efficiently within the stable operating regions of the engine.
- A hybrid optimization methodology has been developed for gas turbine cycle selection.
- The coupled problem of cycle selection and stationary control optimization has been studied for a mixed mission for the Selective Bleed variable cycle engine. The selected design point indicate that the engine can be downsized considerably, compared with previously published studies, and still fulfill the mission requirements.
- A comparative study of the Double Bypass VCE and a conventional two-spool low bypass turbofan, indicate that the Double Bypass engine performance is superior to the conventional engine for the complete range of jet engine missions studied.
- The Double Bypass engine fulfilled the specified thrust requirements with turbine inlet temperatures up to 300K less than needed by the conventional engine cycle.

Bibliography

[1] R. H. Ashleman, T. Lavelle, and F. Parsons. The national cycle program: A flexible system modeling architecture for aircraft engine simulation. *AIAA Paper 98-3114*, 1998.

[2] C. J. Daniele, M. S. Krosel, R. S. John, and E. J. Westerkamp. Digital computer program for generating dynamic turbofan engine models (digtem). Technical Report TM-83446, NASA Lewis Research Center, September 1983.

[3] R. W. Koenig and L. H. Fishbach. Geneng - a program for calculating design and off-design performance for turbojet and turbofan engines. Technical Report TN D-6552, NASA Lewis, Febr. 1972.

[4] L. H. Fishbach and R. W. Koenig. Geneng ii - a program for calculating design and off-design performance of two- and three-spool turbofans with as many as threee nozzles. engines. Technical Report TN D-6553, NASA Lewis, Febr. 1972.

[5] L. H. Fishbach and M. J. Caddy. Nnep - the navy nasa engine program. Technical Report TM-X-71857, NASA Lewis, Febr. 1975.

[6] J. F. Sellers and C. J. Daniele. Dyngen - a program for calculating steady-state and transient performance of turbojet and turbofan engines. Technical Report TN-D-7901, NASA Lewis, April 1975.

[7] J. R. Szuch. Hydes: A generalized hybrid computer program for studying turbojet or turbofan engine dynamics. Technical Report TM-X-3014, NASA Lewis Research Center, April 1974.

[8] A. L. Evans, G. Follen, and C. Naiman. Numerical propulsion system simulation's national cycle program. *AIAA Paper 98-3113*, 1998.

[9] http://www.grc.nasa.gov/WWW/RT1998/2000/2900naiman.html.

[10] W. L. MacMillan. *Development of a Modular Type Computer Program for the Calcuation of Gas Turbine Design Performance.* PhD thesis, Cranfield Institute of Technology, 1974.

[11] J. R. Palmer and Y. Cheng-Zhong. Turbotrans - a programming language for the performance simulation of arbitrary gas turbine engines with arbitrary control systems. ASME 82-GT-200. American Society of Mechanical Engineers, April 1982.

[12] J. Kurzke. Manual gasturb 8.0 for windows - a program to calculate design and off-design performance of gas turbines. Technical report, 1998.

[13] J. A. Reed and A. A. Abdollah. A java-enabled interactive graphical gas turbine propulsion system simulator. *AIAA Paper 97-2333*, 1997.

[14] J. A. Reed and A. A. Abdollah. A java simulator for teaching gas turbine operation. *AIAA Paper 97-0850*, 1997.

[15] J. A. Reed and A. A. Abdollah. Computational simulation of gas turbines: Part i - foundations of component-based models. June 1999.

[16] J. A. Reed and A. A. Abdollah. Computational simulation of gas turbines: Part ii - extensible domain framework. June 1999.

[17] P. L. Hansen, R. Johansson, I. Johansson, and L. Mossberg. Steady state performance of gas turbine engines. Technical report, Internal Volvo Aero Report, 1979.

[18] U. T. J. Grönstedt and U. Håll. Mission dependent optimization of advanced fighter engines. In *13th International Symposium on Air Breathing Engines, Chattanooga, USA.*, 1997.

[19] U. T. J. Grönstedt. Control optimization of the transient performance of the selective bleed variable cycle engine during mode transition. In *ASME TURBO EXPO 2000, Munich, Germany*, 2000.

[20] J. Rumbaugh, Blaha M., and W. Premerlani. *Object-Oriented Modeling and Design.* Prentice-Hall, 1991.

[21] I. Johansson. *Object-Oriented Software Engineering.* Addison-Wesley, 1997.

[22] M Metcalf and J. Reid. *Fortran 90 Explained*, chapter 5. Oxford University Press, 1993.

[23] THE MATHWORKS, INC., 24 Prime Way, Natick MA. *Matlab 5.3 and Simulink 2.3.*

[24] Integrated Systems, INC., http://www.isi.com. *MatrixX Product Family.*

[25] Advanced Visual Systems Inc., Waltham, MA, May 1992. *AVS Developer's Guide.*

[26] U. T. J Grönstedt. Advanced solvers for general high performance transient gas turbine simulation tools. In *14th International Symposium on Air Breathing Engines, Florence, Italy*, 1999.

[27] L. F. Shampine, M. W. Reichelt, and J. A. Kierzenka. Solving index 1 daes in matlab and simulink. *SIAM Review*, 41:538–552, 1999.

[28] J. A. Reed. Internal communication, 2000.

[29] L. F. Shampine. Internal communication, 1999.

[30] H. A. Cohen, G. F. C. Rogers, and H. I. H. Saravanamuttoo. *Gas Turbine Theory.* Longman Scientific and Technical, 1996.

[31] J. D. Mattingly. *Elements of gas turbine propulsion*, chapter 5, page 241. McGraw-Hill, Inc., 1996.

[32] J. L. Kerrebrock. *Aircraft Engines and Gas Turbines*, chapter 2, page p. 21. The MIT Press, 1977.

[33] H. I. H. Saravanamuttoo. Steady and transient performance prediction of gas turbine engines. pages 1.1–1.18. AGARD-LS-183, 1992.

[34] P. Naughton and H. Schildt. *Java 2 : The Complete Reference.* Osborne McGraw-Hill, 1999.

[35] Digital visual fortran reference manual 6.0, 1998.

[36] J. A. Nelder and Mead. A simplex method for function minimization. *Comp. Jour.*, 7:308–313, 1965.

[37] W. H. Press, S. A. Teukolsky, and W. T. Vetterling. *Numerical recipes in FORTRAN: the art of scientific computing.* Cambridge University Press, 1992.

[38] S. Rau. *Engineering optimization - theory and practice.* Belmont, Athena Scientific, 1996.

[39] R. H Brown. Integration of a variable cycle engine concept in a supersonic cruise aircraft. *AIAA Paper 78-1049*, July 1978.

[40] R. D. Allan. General electric company variable cycle engine technology demonstrator programs. *AIAA Paper 79-1311*, 1979.

[41] M. W. French and G. L. Allen. Nasa vce test bed engine aerodynamic performance characteristics and test results. In *AIAA/SAE/ASME Joint Propulsion Conference, Colorado Springs, Colorado*, July 1981.

[42] L. H. Fishbach. Nasa research in supersonic propulsion: A decade of progress. Technical Report TM 82862, NASA Lewis, 1982.

[43] M. E. Brazier and R. Paulsson. Variable cycle engine concept. In *AIAA, Eleventh International Symposium on Air Breathing Engines, Tokyo*, Sept. 1993.

[44] M. J. Hirschberg. The advanced tactical fighter engine development program. In *13th International Symposium on Air Breathing Engines, Chattanooga, USA.*, 1997.

[45] S. Adibhatla, G. J. Collier, and X. Zhao. h_∞ control design for a jet engine. *AIAA Paper 98-3753*, 1998.

[46] L. Oggero and P. Pilidis. A novel optimisation method for variable cycle engines. ASME Paper 98-GT-142. American Society of Mechanical Engineers, 1998.

[47] M.A.R. Nascimento. *The Selective Bleed Variable Cycle Engine.* PhD thesis, Cranfield Institute of Technology, 1992.

[48] M.A.R. Nascimento and P. Pilidis. The selective bleed variable cycle engine. ASME Paper 91-GT-388. American Society of Mechanical Engineers, 1991.

[49] I. Ulizar and P. Pilidis. Transition control and performance of the selective bleed variable cycle turbofan. ASME Paper 95-GT-286. American Society of Mechanical Engineers, 1995.

[50] http://www.netlib.org/.

[51] A. J. Fawke and H. I. H Saravanamuttoo. Digital computer simulation of the dynamic response of a twin-spool turbofan with mixed exhausts. *Aeronautical Journal*, 1973.

[52] E. M. Greitzer. Surge and rotating stall in axial flow compressors - part i: Theoretical compression system model. *ASME Journal of Engineering for Power*, 98:190–198, April 1976.

[53] D. Garrard, M. Jr. Davis, A. Hale, J. Chalk, and S. Savelle. Analysis of gas turbine engine operability with the aerodynamic turbine engine code. In *ISABE97-7034*, pages 223–232. AIAA, September 1997.

[54] M. W. Jr. Davis and W. F. O'Brien. Stage-by-stage poststall compression system modeling technique. *AIAA Journal of Propulsion and Power*, 7, November 1991.

[55] P. Pilidis and H. R. L. MacCalum. The prediction of surge margins during gas turbine transients. ASME Paper 85-GT-208. American Society of Mechanical Engineers, 1985.

[56] R. A. Crawford and A. E. Burwell. Quantitative evaluation of transient heat transfer on axial flow compressor stability. AIAA Paper 85-GT-1352, 1985.

[57] H. Gold and S. Rosenzweig. A method for estimating speed response of gas turbine engines. Technical report, NACA-RM-E51K21, 1952.

[58] M. T. Schobeiri, M. Attia, and C. Lippke. Getran: A generic, modularly structured computer code for simulation of dynamic behavior of aero- and power generation gas turbine engines. *Journal of Engineneering for Gas Turbines and Power*, 116:483–494, July 1994.

[59] M. A. Chappel and P. W. McLaughlin. Approach to modeling continuous turbine engine operation from startup to shutdown. *Journal of Propulsion and Power*, 9, May 1993.

[60] P. Dierckx. *Curve and Surface Fitting with Splines.* Oxford Science Publications, 1993.

[61] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical Solution of Initail-Value Problems in Differential-Algebraic Equations.* Elsevier Science Publishing Co., Inc., 1989.

[62] A. C. Hindmarsch. Lsode and lsodi, two new initial value ordinary differential equation solvers. *ACM Signum Newsletter*, 15:10–11, 1980.

[63] J. Holland. *Adaption in natural and artificial systems.* The University of Michigan Press, Ann Arbor, 1975.

[64] K. A. De Jong. *An analysis of the behaviour of a class of genetic adaptive systems.* PhD thesis, University of Michigan, 1976.

[65] D. Goldberg. Real-coded genetic algorithms, virtual alphabets and blocking. Technical Report Technical Report no 90001, University of Illinois at Urbana-Campaign, September 1990.

[66] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley, 1989.

[67] C. R. Houck, J. A. Joines, and M. G. Kay. Genetic algorith for function optimization: A matlab implementation. North Carolina State University, Releigh, NC, 1994.

[68] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer, 1994.

[69] L. J. J. P. Nadon, S. C. Kramer Cramer, and P. I. King. Multidisciplinary optimization in conceptual design of mixed-stream turbofan engines. *Journal of Propulsion and Power*, 1999.

[70] P.E. Gill, W. Murray, M. A. Saunders, and M.H. Wright. *Users guide for LSSOL.* Department of Operations Research, Stanford University, sol 86-1 edition, 1986.

[71] I. Gustavsson. *Tillämpad optimaringslära.* Institutionen för Datavetenskap, CTH, 1994.

[72] R. Fletcher. *Practical Methods of Optimization.* Wiley, 1987.

[73] D. P. Bertsekas. *Nonlinear programming.* Belmont, Athena Scientific, 1995.

[74] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer, 1994.

[75] R. L. Bucknell. Stovl engine/airframe integration. *Journal of Propulsion*, 5(1):122–125, 1989.

[76] The Numerical Algorithms Group Limited, Wilkinson House, Jordan Hill Road, OXFORD, UK. *The NAG Fortran Library Manual, Mark 15.*

[77] J. E. Garberoglio, J. O. Song, and W. L. Boudreaux. Optimization of compressor vane and bleed settings. In *27th International Gas Turbine Conference and Exhibit*, London, April 1982.

[78] P. T. Kerney. Vane optimization for maximum efficiency using design of experiments. In *29th Joint Propulsion, Conference and Exhibit*, Monterey, CA, June 1993.

[79] C. H. Wu. A general theory of three-dimensional flow in subsonic and supersonic turbomachines of axial-, radial-, and mixed flow type. Technical Report TIN 2604, NACA, 1952.

[80] U. T. J. Grönstedt, T. Johansson, and U. Håll. The optimization of a seven stage compressor. In *International Symposium on Fluid Machinery and Fluid Engineering*, 1996.

[81] R. M. Hearsey. Numerical optimization of axial compressor design. In *Gas Turbine and Aeroengine Congress and Exposition*, Toronto, Canada, June 1989.

[82] A. Massardo and A. Satta. The use of optimization technique and through flow analysis for the design of axial flow compressor stages. In *Conference on Fluid Machinery*, Budapest, Hungary, 1987.

[83] I. N. Egorov. Optimization of a multistage axial compressor in a gas turbine engine system. In *International Gas Turbine and Aeroengine Congress and Exposition*, Budapest, Hungary, 1987.

[84] C. et al. Hirsch. Q3dflo - computer program for turbomachinery flows. Technical report, Numeca vers. 10011, June 1991.

[85] S. Baralon. *On Multistage Analysis of Transonic Compressors: From Axisymmetric Throughflow time-Marching to Unsteady Three-Dimensional Methods.* PhD thesis, Chalmers University of Technology, 2000.

[86] H. Brown. Multi-variable cycle optimization by gradient methods. *AIAA Paper 80-0052*, 1980.

[87] W. H. Robbins and J. F. Dugan. Prediction of off-design performance of multistage compressors. In *Aerodynamic Design of Axial-Flow Compressors*, pages 297–310. NASA SP-36, 1965.

[88] A.B. McKenzie. *Axial Flow Fans and Compressors.* Ashgate Publishing Limited, 1997.

[89] U. T. J. Grönstedt. Gestpan compressor module. Technical report, Volvo Aero Corporation, Dec. 1999.

[90] W. G. Cornell. Experimental quiet engine program - summary report. Technical Report NASA CR-2519, NASA Lewis, March 1975.

[91] P. R. Holloway, G. L. Knight, C. C. Koch, and S.J. Shaffer. Energy efficient engine high pressure compressor detail design report. Technical Report NASA CR-165558, NASA Lewis, May 1982.

[92] Sirinoglou A. Implementation of variable geometry for gas turbine performance. Master's thesis, Cranfield Institute of Technology, September 1992.

[93] D. E. Muir, H. I. H Saravanamuttoo, and D. J. Marshall. Health monitoring of variable geometry gas turbines for the canadian navy. *ASME Journal of Engineering for Gas Turbines and Power*, 111:244–250, April 1989.

[94] J. D. Mattingly. *Elements of gas turbine propulsion*, chapter 10, pages 832–833. McGraw-Hill, Inc., 1996.

[95] A. Stodola. *Steam and Gas Turbines*. Peter Smith, New York, 6th edition edition, 1945.

[96] W. H. Robbins and J. F. Dugan. Prediction of off-design performance of multistage compressors. In *Aerodynamic Design of Axial-Flow Compressors*, page 485. NASA SP-36, 1965.

[97] A. H. Shapiro. *The Dynamics and Thermodynamics of Compressible Fluid Flow*. The Ronald Press Company, 1953.

[98] G. C. Oates. *Aerothermodynamics of Aircraft Engine Components*. AIAA Eductaction Series, 1984.

[99] C. G. Broyden. A class of methods for solving nonlinear simultaneous equations. *Comp. Jour.*, 12, 1969.

[100] C. G. Broyden. A new method for solving nonlinear simultaneous equations. *Comp. Jour.*, 12, 1969.

Component models - Appendix A

A.1 Inlet

The inlet component equations are used to calculate the ambient conditions and ram pressure recovery.

A.1.1 Ambient conditions

The ambient conditions are calculated using a simple ISA model according to:

Flight altitude less than 11000.0 m

$$T_a = 288.15 - 6.5 \cdot 10^{-3}\mathrm{H} \tag{A.1}$$

$$P_a = 101325.0 \cdot (1.0 - 2.2557 \cdot 10^{-5}\mathrm{H})^{5.2561} \tag{A.2}$$

Flight altitude between 11000.0 m and 25000.0

$$T_a = 216.65 \tag{A.3}$$

$$P_a = 22632.0 \cdot e^{-1.5769 \cdot 10^{-4} \cdot (H - 11000.0)} \tag{A.4}$$

A deviation from the ISA standard, ΔT_{isa}, can be specified directly as an input to the module, which modifies the ambient temperature according to:

$$T_a = T_a + \Delta T_{isa} \tag{A.5}$$

A.1.2 Ram pressure recovery

If ploss is set equal to 1.0, ram pressure recovery is computed according to military specification 5008B, i.e.:

Flight Mach numbers less than 1.0 No viscous losses are estimated, i.e. the pressure recovery factor, Π_{rec}, is set to 1.0 in this regime.

Flight Mach numbers greater than 1.0 and less than 5.0

$$\Pi_{\mathrm{rec}} = 1.0 - 0.075 \cdot (M - 1.0)^{1.35} \tag{A.6}$$

If ploss is set to any other value than 1.0, this value is used directly for Π_{rec} instead of the value that the military specification suggests. The ambient conditions and the pressure recovery factor are combined to calculate outlet stagnation properties, T_2 and P_2, from the inlet module according to:

$$T_2 = T_a \cdot (1 + \frac{\gamma - 1}{2} \cdot M^2) \tag{A.7}$$

$$P_2 = P_a \cdot \Pi_{\mathrm{rec}} \cdot \left(1 + \frac{\gamma - 1}{2} \cdot M^2\right)^{\frac{\gamma}{\gamma - 1}} \tag{A.8}$$

Index a was used to denote ambient conditions.

A.2 Splitter

A.2.1 Governing equations

The splitter component is used to split the flow into two streams. The inlet thermodynamic properties are preserved, and the flow is divided into two streams using the bypass ratio, *bpr*, according to:

$$m_2 = \frac{m_1}{\mathrm{bpr} + 1} \tag{A.9}$$

$$m_4 = m_1 - m_2 \tag{A.10}$$

$$\text{where,} \quad bpr = \frac{m_2}{m_4} \tag{A.11}$$

A.3 Compressor

The compressor component modeling technique uses a set of empirical tables to generate compressor maps. In the report on which the procedure is based, NASA SP-36 [87], experimental data on eight multistage compressors were collected and correlated in table form. The method allows performance maps of new multistage-compressor designs to be obtained easily from a knowledge of the design conditions alone. The success of this method is strongly dependent on the extent to which the design methodology applied to the new compressor is a heritage from the previous designs (on which the correlations are based).

A.3.1 Description of the original method

Two concepts must be introduced to describe the methodology: the *reference point* and the compressor *backbone*. The reference point is defined as the point with the highest efficiency in the entire compressor map, and the compressor backbone is defined as the line interconnecting the maximum efficiency points at all rotational speeds.

The dependency of the reference point pressure ratio on the backbone characteristics is collected in three sets of tables. By knowing the reference-point pressure ratio and a given rotational speed, the backbone mass flow, the backbone isentropic efficiency and the backbone pressure ratio is acquired by simple table look-ups. The tables store:

$$\frac{\pi_{bb}}{\pi_{rp}} = f_1(\nu, \pi_{rp}) \tag{A.12}$$

$$\frac{\tilde{m}_{bb}}{\tilde{m}_{rp}} = f_2(\nu, \pi_{rp}) \tag{A.13}$$

$$\frac{\eta_{bb}}{\eta_{rp}} = f_3(\nu, \pi_{rp}) \tag{A.14}$$

$$\text{where } \nu = \frac{\tilde{n}}{\tilde{n}_{rp}} \text{ , } \tilde{m} = \frac{m_1\sqrt{T_1}}{P_1} \text{ , } \tilde{n} = \frac{n}{\sqrt{T_1}}$$

where f_1, f_2, f_3 represent the data in the three tables. Suffixes rp and bb represent the reference and backbone conditions, respectively.

The procedure for determining the variation of pressure ratio, mass flow and efficiency along constant speed lines is based on the formulation of the flow parameter, ϕ, according to:

$$\phi = \frac{m_1\sqrt{\theta_1}}{\delta_1} \cdot \sqrt{\frac{\tau}{\pi}}$$

$$\text{where } \delta_1 = \frac{P_1}{101325.0} \text{ , } \theta_1 = \frac{T_1}{288.15} \text{ , } \tau = \frac{T_2}{T_1} \text{ , } \pi = \frac{P_2}{P_1} \tag{A.15}$$

which is scaled with the backbone value to get the relative value $\phi_{rel} = \frac{\phi}{\phi_{bb}}$. Variation in efficiency and relative temperature rise as a function of the relative flow parameter, ϕ_{rel}, are given in two tables, i.e. the tables store:

$$\frac{(T_2 - T_1)}{(T_2 - T_1)_{bb}} = f_4(\phi_{rel}) \tag{A.16}$$

$$\frac{\eta}{\eta_{bb}} = f_5(\phi_{rel}) \tag{A.17}$$

The empirical observation on which this methodology is based is that by grouping the variables according to A.15,A.16 and A.16, the relative numbers become fairly independent of rotational speed. Thus the two tables giving the off backbone behavior, together with the three tables giving the backbone data, make the map determining procedure complete.

A.3.2 Shortcomings of the original method

A closer inspection of the original method revealed several weaknesses:

- The original correlation presented data only up to 110% of the reference point rotational speed.
- The method used to determine the off-backbone behavior agrees poorly with empirical data.
- The original compressor data were based on compressors designed during the period 1950-1965.

A consequence of the first point is that compressor choking effects are not included in the original NASA SP-36 model. McKenzie [88] notes that the use of a compressor backbone line for prediction of compressor performance is probably a useful strategy but that the off-backbone method described in the original report was not very successful. Evaluations of this part of the method carried out in [89] also point in the same direction.

A.3.3 Modification of the original method

Two major steps were undertaken to improve on the original method:

1. The high speed range of the original method was extrapolated using in-house Volvo Aero Compressor data
2. A new two-parameter model for off-backbone compressor prediction has been developed

This approach retains the good parts of the original method and attempts to improve on the weaknesses of it.

A.3.4 New correlation

Since the re-correlated version of the NASA SP-36 model was developed using in-house Volvo Aero data, the tables can not be given in this report. Only the approach by which the re-correlation was done can be described. A similar modification of the original data may be performed using compressor data available in the public literature, such as [90, 91].

The re-correlations of f_1, f_2 and f_3 were made by adding a constant but different increment to the different ν lines (see Figure 225 a,b and c in [87]) by minimizing the largest deviation from the ν line (its spline interpolation) and the available compressor data. For rotational speed ranges higher than 110% of the reference point rotational speed range, the shape of the 110% curve was preserved. The resulting extrapolated model seemed to agree well with available compressor data. For the lower rotational speed range of the efficiency table,

i.e. f_3, some minor modifications to the shape of the original curves were also made. After these modifications the agreement between data and the model was quite good along the backbone, with discrepancies not greater than 3%.

To obtain a new correlation for the off-backbone compressor behavior, the flow function ϕ defined in the original NASA report as well as above was simplified slightly to the new expression:

$$\phi = \frac{\frac{m_1\sqrt{\theta_1}}{\delta_1}}{\pi}$$

where the dimensionless quantities were defined above. The new model is based on the empirical observation that if $\frac{\pi}{\pi_{bb}}$ ratios and $\frac{\tau}{\tau_{bb}}$ ratios are computed for different ν values and $\frac{\phi}{\phi_{bb}}$ values, the spread between several compressors seems to be fairly modest. Although the spread is not very large between several compressors attempts to fit general functions to the data, such as approximating splines, least square splines or functions fits (3:rd and 4:th degree polynomials) were not successful. Fairly accurate fits could easily be obtained. Unfortunately, the maps generated in this way frequently deviated from typical compressor maps, such as maps with curved choke lines. For this reason a single compressor map with typical behavior was chosen to obtain the new tables for the $\frac{\pi}{\pi_{bb}}$ and $\frac{\tau}{\tau_{bb}}$ ratios.

A.3.5 Variable geometry model

A model suggested by Sirinoglou [92] was implemented to simulate the effect of compressor variable geometry. This method is based on a paper by Muir et al. [93]. The compressor variable geometry affects corrected mass flow, pressure ratio and efficiency according to:

$$\pi_{rp} = \pi_{rp} \cdot g_1(\alpha) \tag{A.18}$$

$$\eta = \eta \cdot g_2(\alpha) \tag{A.19}$$

$$\tilde{g} = \tilde{g} \cdot g_3(\alpha) \tag{A.20}$$

$$-10 \geq \alpha \leq 40$$

where α is the variable geometry angle. The g_1, g_2 and g_3 functions are shown in Figure A.1.

A.3.6 Additional relations

The outlet stagnation temperature, T_2, is calculated using:

$$T_2 = T_1 \left(1 + \frac{1}{\eta}\left(\pi^{\frac{\gamma-1.0}{\gamma}} - 1.0\right)\right) \tag{A.21}$$

where γ is taken at the arithmetic mean temperature. The power is subsequently determined from:

$$W = m_2 \cdot \frac{h_2 - h_1}{\eta_{\mathrm{m}}} \tag{A.22}$$

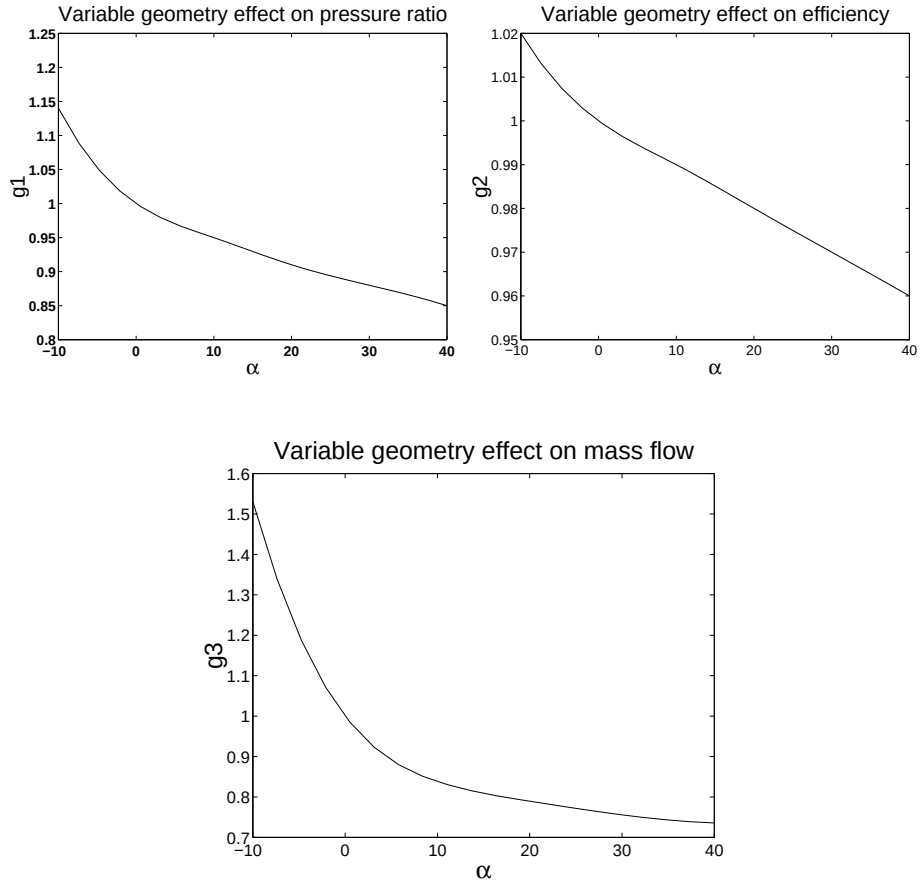


Figure A.1: Compressor variable geometry model [92]

The power is related to the torque, G, according to

$$G = \frac{W}{2\pi n} \tag{A.23}$$

To allow locating the engine design point to a point not on the backbone curve, the Ω parameter is introduced according to:

$$\Omega = \frac{\pi}{\pi_{bb}} \tag{A.24}$$

To move the engine design point along the backbone line, the already defined ν parameter is used.

A.4 Burner

The burner component computes stagnation temperature rise from a specified input fuel flow. A table is used to assess the variation of burner efficiency with thermodynamic conditions. A reaction rate parameter, σ, is defined as:

$$\sigma = \frac{P_1^{1.75} \cdot e^{\frac{300.0}{T_1}}}{m_1} \tag{A.25}$$

The ratio between the σ value and the design σ value, i.e. $\frac{\sigma}{\sigma_{dp}}$, is then used to determine the burner efficiency, η_c, from the relationship:

$$\eta_c = \eta_{dp} \cdot f_1(\frac{\sigma}{\sigma_{dp}}) \tag{A.26}$$

where f_1 represents a table for which it holds that $f_1(1.0) = 1.0$. Such a correlation is described in [94]. An ideal temperature increase is obtained from another table based on the assumption of complete combustion:

$$\Delta t_{\text{ideal}} = f_2(t_1, fa_2) \tag{A.27}$$

where t_1 is the inflow temperature and fa_2 the outlet fuel air ratio. The table is a spline fit of fuel data obtained for a hypothetical liquid hydrocarbon containing 16.0% hydrogen and 84.0% carbon. The fuel air ratio is obtained from the following relation:

$$fa_2 = \frac{\beta m_{f,dp}}{m_1} \tag{A.28}$$

where β is a fuel schedule factor and $m_{f,dp}$ is the fuel flow in the design point. The burner exit temperature can then be computed according to:

$$T_2 = T_1 + \Delta t_{\text{ideal}} \cdot \eta_c \tag{A.29}$$

The pressure loss is computed using:

$$P_2 = P_1 \cdot \left(1.0 - \omega \cdot \left(\frac{m_1\sqrt{T_1}}{P_1}\right)^2\right) \tag{A.30}$$

where the pressure loss coefficient, ω, is determined from the burner pressure drop specified in the design point.

A.5 Duct

The duct component equations are used to calculate pressure losses in the bypass duct as well as other ducts. All thermodynamic properties are preserved through the duct except for the stagnation pressure ratio. The pressure drop is computed according to:

$$P_2 = P_1 \left(1 - \omega \cdot \left(\frac{m_1 \sqrt{T_1}}{P_1} \right)^2 \right) \tag{A.31}$$

where ω is a pressure loss coefficient determined during the design process.

A.6 Turbine component

A.6.1 Turbine mass flow

The pressure ratio is given by:

$$\pi = \frac{p_1}{p_2}$$

where both p_1 and p_2 are input to the module. The turbine inlet mass flow is a function of the turbine pressure ratio according to:

$$\frac{m_1\sqrt{R_1 T_1}}{P_1 A_1} = \begin{cases} \mathrm{X}(\gamma_1)\cdot\sqrt{1-\left(\frac{\frac{\pi^*}{\pi}-1}{\pi^*-1}\right)^2}, & \pi < \pi^* \\ \mathrm{X}(\gamma_1) & ,\pi > \pi^* \\ \text{where,} \quad \mathrm{X}(\gamma_1) = \sqrt{\gamma_1}\frac{\gamma_1+1}{2}^{-\frac{\gamma_1+1}{2(\gamma_1-1)}} \end{cases} \tag{A.32}$$

where π^* is the choking pressure ratio. The choking pressure ratio is an input to the module which depends on the number of stages in the turbine. π^* is set to 2.15, 2.31, 3.05 and 3.50 for one, two, three and four stage turbines, respectively.

Equation A.32 is an extension of the "Stodola Ellipse", see [95]. Stodolas original formula was:

$$\frac{m_1\sqrt{T_1}}{P_1} = \text{const}\cdot\sqrt{1-\left(\frac{P_2}{P_1}\right)^2} = \text{const}\cdot\sqrt{1-\left(\frac{1}{\pi}\right)^2} \tag{A.33}$$

Note that the function actually is an ellipse in $\frac{1}{\pi}$ and $\frac{m_1\sqrt{T_1}}{P_1}$. The Stodola formula has the drawback that the single constant can not be adopted to satisfy a zero mass flow at $\pi = 1$ as well as a continuous derivative and the choking mass flow at $\pi = \pi^*$. This can be achieved by introducing a linear term and a free constant into the Stodola expression according to:

$$\frac{m_1\sqrt{T_1}}{P_1} = c_1\sqrt{c_2 + \frac{c_3}{\pi} + \frac{c_4}{\pi^2}} \tag{A.34}$$

Applying the requirements given above to (A.34) yields (A.32). It can easily be verified that the function gives reasonable physical behavior in the "end points". Evaluating (A.32) in $\pi = 1.0$ yields $\frac{m_1\sqrt{T_1}}{P_1} = 0$. For $\pi = \pi^*$, the corrected mass flow takes the choking value, $\frac{m_1\sqrt{T_1}}{P_1} = \left(\frac{m_1\sqrt{T_1}}{P_1}\right)^*$, which means that the mass flow curve is continuous in the intersection between choked and unchoked conditions. Furthermore, the derivative of the function shows a physical behavior as well:

$$\frac{d\left(\frac{m_1\sqrt{T_1}}{P_1}\right)}{d\pi} = \frac{\pi^*\left(\frac{\pi^*}{\pi}-1\right)\left(\frac{m_1\sqrt{T_1}}{P_1}\right)^*}{\pi^2\left(\pi^*-1\right)^2\sqrt{1-\left(\frac{\frac{\pi^*}{\pi}-1}{\pi^*-1}\right)^2}} \tag{A.35}$$

In the endpoints we get:

$$\frac{d\left(\frac{m_1\sqrt{T_1}}{P_1}\right)}{d\pi}(\pi = \pi^*) = 0$$
$$\frac{d\left(\frac{m_1\sqrt{T_1}}{P_1}\right)}{d\pi}(\pi = 1.0) = +\infty$$

which is reasonable. In a real turbine, the derivative of the mass flow in $\pi = 1$ would of course be less than $+\infty$ but still fairly large.

A.6.2 Turbine efficiency

Off-design turbine efficiency is computed using the expression suggested in [96]:

$$\eta_0 = \eta_{dp}\left(1 - \left(\frac{\frac{n}{\sqrt{\Delta h}}}{\left(\frac{n}{\sqrt{\Delta h}}\right)_{dp}} - 1\right)^2\right) \tag{A.36}$$

A simple Reynolds number correction is subsequently applied to η_0 according to:

$$\eta = 1 - \frac{1 - \eta_0}{\frac{\kappa}{\kappa_{dp}}} \tag{A.37}$$

where:

$$\kappa = \frac{P}{T^{1.2}} \tag{A.38}$$

Efficiency corrections resulting from Equation A.37 that are larger than 1% are very rare.

A.6.3 Cooling scheme

Figure A.2 illustrates the cooling flow arrangement used in the turbine module. The enthalpy before the rotor, $h_{1,r}$, is computed using:

$$h_{1,r} = \frac{h_1 m_1 + h_c\left(m_{c,s} + m_{c,lbr}\right)}{m_1 + m_{c,s} + m_{c,lbr}}$$

where h_c, $m_{c,s}$, $m_{c,r}$ and $m_{c,lbr}$ are the the enthalpy of the cooling flow, the part of the cooling flow cooling the stator, the part cooling the rotor and the part of the leakage cooling flow entering before the rotor, respectively. An iteration is carried out to find the enthalpy drop over the turbine, Δh. For a given Δh, the efficiency is obtained using equation A.36. For this efficiency, the turbine rotor exit temperature, $t_{2,r}$, is obtained according to

$$t_{2,r} - t_{1,r} = t_{1,r}\eta\left(1 - \frac{1}{\pi^{\frac{\gamma-1}{\gamma}}}\right) \tag{A.39}$$

where γ is the arithmetic mean of the γ:s obtained at the two temperatures $t_{2,r}$ and $t_{1,r}$. Δh is then:

$$\Delta h = h_{1,r} - h_{2,r} \tag{A.40}$$

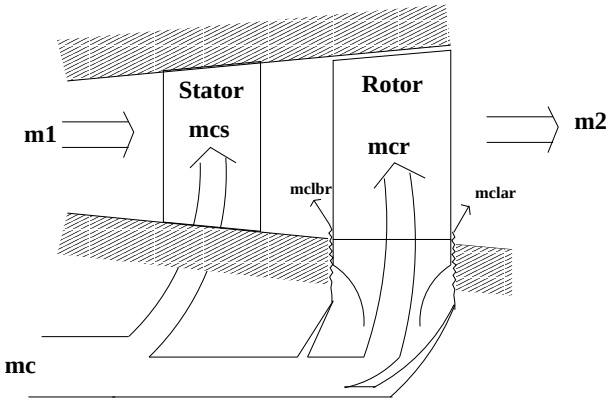


Figure A.2: Cooling flow arrangement

where $h_{2,r}$ is obtained from a gas table using $t_{2,r}$ as input. The iteration process is terminated when

$$abs(\frac{t_{2,r} - t_{2,r,previous}}{t_{2,r,previous}}) \leq \epsilon \tag{A.41}$$

The turbine power, W_t, is obtained from:

$$W_t = \Delta h \left(m_1 + m_{c,s} + m_{c,lbr}\right)$$

and the turbine torque, G_t, from:

$$G_t = \frac{W_t}{2\pi n}$$

The exhaust temperature, t_2, is found from the outlet enthalpy, h_2, given by:

$$h_2 = \frac{h_{2,r}\left(m_1 + m_{c,s} + m_{c,lbr}\right) + h_c\left(m_{c,lar} + m_{c,r}\right)}{m_1 + m_c} \tag{A.42}$$

where $m_{c,lar}$ is the leakage cooling flow entering after the rotor. The input specification of the cooling flow distribution is done by using three non-dimensional numbers, ψ_1, ψ_2 and ψ_3, defined as:

$$\psi_1 = \frac{m_{c,s}}{m_c} \tag{A.43}$$

$$\psi_2 = \frac{m_{c,r}}{m_c} \tag{A.44}$$

$$\psi_3 = \frac{m_{c,lbr}}{m_c - (m_{c,s} + m_{c,r})} \tag{A.45}$$

A.7 Unifier

The unifier component equations are used to model the mixing of two flow streams, which is illustrated in Figure A.3.

Figure A.3: Mixing of two flow streams

A.7.1 Governing equations

The Mach numbers of the flows are related to the mass flow and the thermodynamic properties of the gas through the 1-D compressible continuity equation, i.e.:

$$\frac{m_1\sqrt{R_1T_1}}{C_{d,1}A_1P_1} = \sqrt{\gamma}M_1\left(1+\frac{\gamma_1-1}{2}M_1^2\right)^{-\frac{\gamma_1+1}{2(\gamma_1-1)}} = \mathrm{X}_1(M_1,\gamma_1) \tag{A.46}$$

$$\frac{m_2\sqrt{R_2T_2}}{C_{d,2}A_2P_2} = \sqrt{\gamma}M_2\left(1+\frac{\gamma_2-1}{2}M_2^2\right)^{-\frac{\gamma_2+1}{2(\gamma_3-1)}} = \mathrm{X}_2(M_2,\gamma_2) \tag{A.47}$$

$$\frac{m_3\sqrt{R_3T_3}}{C_{d,3}A_3P_3} = \sqrt{\gamma}M_3\left(1+\frac{\gamma_3-1}{2}M_3^2\right)^{-\frac{\gamma_3+1}{2(\gamma_3-1)}} = \mathrm{X}_3(M_3,\gamma_3) \tag{A.48}$$

The C_d values are used to correct for boundary layer blockage of the flow. C_d values are either given as input data to the module or obtained from tabulated data. To find the Mach number after mixing, the following function is introduced (see [97]):

$$F = \underbrace{P_sA}_{\text{Pressure force}} + \underbrace{\gamma P_sAM^2}_{\text{Flow force}} \tag{A.49}$$

A momentum balance yields:

$$F_2 = F_1 + F_3 \tag{A.50}$$

Introducing A.49 into A.50 yields:

$$P_{2,s}A_2 + \gamma P_{2,s}A_2M_2^2 = P_{1,s}A_1 + \gamma P_{1,s}A_1M_1^2 + P_{3,s}A_3 + \gamma P_{3,s}A_3M_3^2$$

$$\Longleftrightarrow$$

$$P_{2,s}A_2\left(1+\gamma M_2^2\right) = P_{1,s}A_1\left(1+\gamma M_1^2\right) + P_{3,s}A_3\left(1+\gamma M_3^2\right) \tag{A.51}$$

The static pressure is related to the total pressure according to the following relationship:

$$\frac{P}{P_s} = \left(1+\frac{\gamma-1}{2}M^2\right)^{\frac{\gamma}{\gamma-1}} \tag{A.52}$$

Introducing A.46, A.47 and A.48 into A.51 yields:

$$\frac{m_2\sqrt{R_2T_2}}{C_{d,2}\mathrm{X}_2}\frac{(1+\gamma_2 M_2^2)^{\frac{\gamma}{\gamma-1}}}{1+\frac{\gamma-1}{2}M_2^2} = \frac{m_1\sqrt{R_1T_1}}{C_{d,1}\mathrm{X}_1}\frac{(1+\gamma_1 M_1^2)^{\frac{\gamma}{\gamma-1}}}{1+\frac{\gamma-1}{2}M_1^2} + \frac{m_3\sqrt{R_3T_3}}{C_{d,3}\mathrm{X}_3}\frac{(1+\gamma_3 M_3^2)^{\frac{\gamma}{\gamma-1}}}{1+\frac{\gamma-1}{2}M_3^2} \tag{A.53}$$

T_2 can be computed in a straightforward manner since the enthalpy of the mixed stream is a mass weighted average of its two constituents, and the enthalpy can be translated into the corresponding temperature. $C_{d,2}$ is estimated from:

$$C_{d,2} = \frac{A_1C_{d,1} + A_3C_{d,3}}{A_2} \tag{A.54}$$

After computing the Mach numbers, M_1 and M_3, by the use of Equation A.46, A.47 and the inflow properties, T_2 and $C_{d,2}$ can be inserted into Equation A.53 to determine M_2.

A.8 Volume

The volume component is used to simulate storage of energy and mass in transient engine models according to:

$$m'_V = m_1 - m_2 \tag{A.55}$$

$$T'_V = \frac{m_1 T_1 - m_1 T_V}{m_V} \tag{A.56}$$

where m'_V and T'_V are the time derivatives of the integrated temperature, T_V, and the integrated mass, m_V, respectively. The output pressure of the module is computed using the ideal gas law and the states, m_V and T_V, according to:

$$p = \frac{m_V R T_V}{V} \tag{A.57}$$

where V is the volume of the component.

A.9 Rotor

The rotor component is used to integrate rotor speeds in transient modeling. The rotor acceleration, $\dot{n}$, is calculated according to:

$$\dot{n} = \frac{\sum G_t - \sum G_c - G_f}{2\pi I} \tag{A.58}$$

where $\sum G_t$ is the sum of the turbine torques and $\sum G_c$ is the sum of the compressor torques. G_f is used to model additional frictional torque and I is the moment of inertia of the shaft.

A.10 Afterburner and nozzle components

This section is used to describe the equations governing the performance of the afterburner and the nozzle modules.

When estimating the engine performance, the reheat effects are modeled without changing the engine operating point. Thus, for a given inlet condition, a calculation is first made of the cold performance. The afterburner fuel flow required to "fill" the lit exhaust area for the given inlet conditions is then determined. Increases in exhaust area by a factor 1.5-2.0 in the throat area for the lit condition are not uncommon [98].

A.10.1 Governing equations

Four indices are defined for the computational process

- Station 1 = The inlet
- Station 2 = Downstream of the flame holders before heating
- Station 3 = Downstream of the flame holders after heating
- Station 4 = Nozzle exit (nozzle throat for a convergent-divergent nozzle)

Computing cold performance:

The inlet mass flow, m_1, is determined by requiring that the continuity equation in the nozzle exhaust be satisfied. The inlet mass flow is always computed by using the equations that hold for an unlit afterburner. The pressure, P_2, downstream of the flame holders is determined according to:

$$P_2 = \underbrace{\left(1.0 - \omega \cdot \left(\frac{m_1\sqrt{t_1}}{P_1}\right)^2\right)}_{x} P_1 = x \cdot P_1 \tag{A.59}$$

Unlit it holds that $P_4 = P_2$, $m_4 = m_1$ and $T_4 = T_1$. The 1D continuity equation states ($\pi = \frac{P_4}{P_\infty}$):

$$\frac{m_1\sqrt{RT_4}}{C_{d,4}A_4P_4} = \chi(M_4, \gamma_4) = \chi(\pi, \gamma_4) \tag{A.60}$$

where χ is equal to:

$$\chi(M,\gamma) = \begin{cases} \sqrt{\gamma}M\left(1+\frac{\gamma-1}{2}M^2\right)^{-\frac{\gamma+1}{2(\gamma-1)}} = \sqrt{\frac{2\gamma(\pi^{\frac{\gamma-1}{\gamma}}-1)}{(\gamma-1)\pi^{\frac{\gamma+1}{\gamma}}}}, & M < 1.0 \\ \sqrt{\gamma}\frac{\gamma+1}{2}^{-\frac{\gamma+1}{2(\gamma-1)}} = \sqrt{\frac{\gamma}{\frac{\gamma+1}{2}^{\frac{\gamma+1}{\gamma-1}}}}, & M = 1 \end{cases} \tag{A.61}$$

for which the right formula of A.78 was used to obtain the second form of the two cases above. For a guessed value of x, equation A.60 can be evaluated using the equation for χ, which yields a value of m_1. m_1 can then be used to improve the guessed value of x, and the iteration process can then proceed until convergence is achieved.

Modeling of the afterburning process:

A reaction rate parameter, θ, is defined as:

$$\theta = \frac{P_1^{1.4} \cdot e^{\frac{T_1}{1000.0}} \cdot A_{fh} \cdot L}{m_1} \tag{A.62}$$

where A_{fh} is the cross sectional area at the flame holder and L is the afterburner length. The ratio between the θ value and the design θ value, i.e. $\frac{\theta}{\theta_{dp}}$, is then used to determine the afterburner efficiency, η_c, from the relationship:

$$\eta_c = \eta_{dp} \cdot f_2(\frac{\theta}{\theta_{dp}}) \tag{A.63}$$

where f_2 represents a table for which it holds that $f_2(1.0) = 1.0$. The fuel flow is defined indirectly by the exhaust area of the afterburner nozzle, i.e. the temperature increase needed to "fill" the lit nozzle exhaust area is computed using the 1D compressible continuity equation. This method leads to an iteration in the fuel air ratio.

The first step in the process of finding the required fuel air ratio is to determine a hypothetical pre-burn temperature, T_0. This temperature is defined as the initial temperature required to end up with the inlet temperature t_1 after burning fuel corresponding to the inlet fuel air ratio fa_1.

An afterburner fuel air ratio, fa_2, is then guessed and the ideal temperature increase, ΔT_{id}, is computed according to:

$$T_2 = T_1 + \eta_c \left(\Delta T_{id}(T_0, fa_2) - (t_1 - t_0)\right) \tag{A.64}$$

Note that the burner efficiency affects only the temperature increase that actually occurred in the afterburner.

In the case of a lit afterburner the stagnation pressure P_3, will be different from P_2. The Mach number at station 3 will be computed using the equations of momentum and continuity. The 1D continuity equation states:

$$\frac{m_2\sqrt{R_2T_2}}{A_2P_2C_{d,2}} = \sqrt{\gamma}M_2\left(1 + \frac{\gamma_2 - 1}{2}M_2^2\right)^{-\frac{\gamma_2+1}{2(\gamma_2-1)}} = \mathrm{X}_2(M_2, \gamma_2) \tag{A.65}$$

$$\frac{m_3\sqrt{R_3T_3}}{A_3P_3C_{d,3}} = \sqrt{\gamma}M_3\left(1 + \frac{\gamma_3 - 1}{2}M_3^2\right)^{-\frac{\gamma_3+1}{2(\gamma_3-1)}} = \mathrm{X}_3(M_3, \gamma_3) \tag{A.66}$$

To find the Mach number after heating the following function is introduced for the force:

$$F = \underbrace{P_sA}_{\text{Pressure force}} + \underbrace{\gamma P_sAM^2}_{\text{Flow force}} \tag{A.67}$$

A momentum balance yields:

$$F_3 = F_2 \tag{A.68}$$

Introducing A.67 into A.68 yields ($A_3 = A_2$):

$$P_{s3} + \gamma P_{s3}M_3^2 = P_{s2} + \gamma P_{s2}M_2^2 \tag{A.69}$$

$$\Longleftrightarrow$$

$$P_{s3}\left(1+\gamma M_3^2\right) = P_{s2}\left(1+\gamma M_2^2\right) \tag{A.70}$$

$$P_3 \frac{\left(1+\gamma M_3^2\right)}{\left(1+\frac{\gamma-1}{2}M_3^2\right)^{\frac{\gamma}{\gamma-1}}} = P_2 \frac{\left(1+\gamma M_2^2\right)}{\left(1+\frac{\gamma-1}{2}M_2^2\right)^{\frac{\gamma}{\gamma-1}}} \tag{A.71}$$

Introducing A.65 and A.66 into A.71 yields:

$$\frac{m_3\sqrt{R_3T_3}}{C_{d,3}\mathrm{Y}_3} = \frac{m_2\sqrt{R_2T_2}}{C_{d,2}\mathrm{Y}_2} \tag{A.72}$$

where Y was defined as:

$$Y = \frac{X\left(1+\frac{\gamma-1}{2}M^2\right)^{\frac{\gamma}{\gamma-1}}}{(1+\gamma M^2)} = \frac{\gamma M\sqrt{1+\frac{\gamma-1}{2}M^2}}{(1+\gamma M^2)} \tag{A.73}$$

Equation A.72 gives Y_3, which can be used to find M_3. By defining Z according to

$$Z = \frac{\left(1+\gamma M_2^2\right)}{\left(1+\frac{\gamma-1}{2}M_2^2\right)^{\frac{\gamma}{\gamma-1}}} \tag{A.74}$$

equation A.71 can then be rewritten as:

$$P_3Z_3 = P_2Z_2 \tag{A.75}$$

which gives P_3. The assumed fa_2 can then finally be checked using:

$$\frac{m_3\sqrt{RT_3}}{C_{d,3}A_{lit}P_3} = \chi(M_3,\gamma_3) = \chi(\pi,\gamma_3) \tag{A.76}$$

where $m_3 = m_1 + m_{fuel,afterburner}$.

Thrust computations (convergent nozzle):

The main equation for computing thrust is:

$$F = \underbrace{mV_j}_{\text{Flow thrust}} + \underbrace{A\left(P_s - P_\infty\right)}_{\text{Pressure thrust}} \tag{A.77}$$

If $\mathbf{P_s = P_\infty}$: (P_s = static pressure at nozzle exit) the nozzle flow is said to be fully expanded, in which case we have:

$$F = F_{\text{full}} = mV_j = mM\sqrt{\gamma RT_s} = mM\sqrt{\gamma R\frac{T_s}{T_0}T_0} = mM\sqrt{\gamma R\frac{1}{\left(1+\frac{\gamma-1}{2}M^2\right)}T_0}$$

$$\Longleftrightarrow$$

$$\frac{F_{full}}{m\sqrt{RT_0}} = \frac{M\sqrt{\gamma}}{\sqrt{1+\frac{\gamma-1}{2}M^2}}$$

but we also have for the nozzle pressure ratio $\pi = \frac{P_4}{P_\infty}$ = [unlit afterburner] = $\frac{P_2}{P_\infty}$ that (the expansion process is idealized as isentropic):

$$\pi = \left(1+\frac{\gamma-1}{2}M^2\right)^{\frac{\gamma}{\gamma-1}} \Longleftrightarrow M = \sqrt{\frac{2}{\gamma-1}(1-\pi^{\frac{\gamma-1}{\gamma}})} \tag{A.78}$$

which allows for replacing the Mach number in the former relation:

$$\frac{F_{full}}{m\sqrt{RT_0}} = \sqrt{\frac{2\gamma}{\gamma-1}}\sqrt{1-\frac{1}{\pi^{\frac{\gamma-1}{\gamma}}}} \tag{A.79}$$

where $\pi = \pi_{\text{nozzle}}$. It should be noted that for convergent nozzles this level of thrust is only achievable for $\pi = \pi_{\text{nozzle}} < \pi_{\text{critical}}$.

If $\mathbf{P_S > P_\infty}$: the nozzle runs choked, $M = 1$ and it holds that $A(p_s - p_\infty) \neq 0$. We then have:

$$mV_j = m\sqrt{\frac{2\gamma R}{\gamma+1}T_0} \tag{A.80}$$

This can be seen from one of the intermediate stages of the derivation of (A.79) carried out above, if M=1 is used simultaneously. For the pressure thrust, we get:

$$\begin{aligned} A(p_s - p_\infty) = \frac{m}{\rho V_j}(p_s - p_\infty) = [\text{Ideal gas law}] = \\ \frac{mRT_s}{M\sqrt{\gamma RT_s}p_s}(p_s - p_\infty)\frac{m\sqrt{RT_s}}{M\sqrt{\gamma}}\left(1-\frac{p_\infty}{p_s}\right) = \left[\frac{p_\infty}{p_s} = \frac{\frac{p_0}{p_s}}{\frac{p_0}{p_\infty}}\right] = \\ \frac{m\sqrt{RT_0}}{M\sqrt{\gamma}\sqrt{1+\frac{\gamma-1}{2}M^2}}\left(1-\frac{\left(1+\frac{\gamma-1}{2}M^2\right)^{\frac{\gamma}{\gamma-1}}}{\pi}\right) = [M=1] = \\ \frac{m\sqrt{RT_0}}{\sqrt{\gamma}\sqrt{\frac{\gamma+1}{2}}}\left(1-\frac{\left(\frac{\gamma+1}{2}\right)^{\frac{\gamma}{\gamma-1}}}{\pi}\right) = m\sqrt{\frac{2\gamma R}{\gamma+1}T_0}\frac{1}{\gamma}\left(1-\frac{\left(\frac{\gamma+1}{2}\right)^{\frac{\gamma}{\gamma-1}}}{\pi}\right) \end{aligned} \tag{A.81}$$

(A.80) and (A.81) can be combined and introduced into A.77 to yield:

$$\frac{F}{m\sqrt{RT_0}} = \sqrt{\frac{2\gamma}{\gamma+1}}\left(1+\frac{1}{\gamma}\left(1-\frac{\pi_c}{\pi}\right)\right) \tag{A.82}$$

where $\pi_c = \frac{\gamma+1}{2}^{\frac{\gamma}{\gamma-1}}$.

The thrust coefficient for the nozzle, C_v, is defined:

$$C_v = C_{v,0}\frac{F}{F_{\text{full}}} \tag{A.83}$$

where the empirical factor, $C_{v,0}$, is introduced to make some corrections for the 1D isentropic idealizations made here. Note that the theoretical limit of $C_v = 1.0$ only applies for $\pi < \pi_c$. To approach the full thrust level for $\pi \geq \pi_c$ a convergent divergent nozzle would have to be used. The final expression for C_v is:

$$C_v = C_{v0}\cdot\frac{F}{F_{\text{full}}} = C_{v0}\sqrt{\frac{\gamma-1}{(\gamma+1)(1-\frac{1}{\pi^{\frac{\gamma-1}{\gamma}}})}}\left(1+\frac{1}{\gamma}\left(1-\frac{\pi_c}{\pi}\right)\right) \tag{A.84}$$

Thrust computations (convergent-divergent nozzle):

A convergent-divergent nozzle is said to be perfectly expanded when the static pressure in the nozzle exit equals the ambient pressure. The exit Mach number for this case can be computed using:

$$A_{ratio} = \frac{1}{M}\left(\frac{1+\frac{\gamma-1}{2}M^2}{1+\frac{\gamma-1}{2}}\right)^{\frac{\gamma+1}{2(\gamma-1)}} \tag{A.85}$$

and the pressure ratio from:

$$\pi_{perfect} = (1+\frac{\gamma-1}{2}M^2)^{\frac{\gamma}{\gamma-1}} \tag{A.86}$$

For nozzle pressure ratios greater than $\pi_{perfect}$, the exit Mach number can still be computed from Equation A.85, since further expansion of the gas will occur outside the nozzle. For this case, the flow thrust can be computed from:

$$F = F_{\text{full}} = mV_j = mM\sqrt{\gamma RT_s} = mM\sqrt{\gamma R\frac{T_s}{T_0}T_0} = \tag{A.87}$$
$$mM\sqrt{\gamma R\frac{1}{\left(1+\frac{\gamma-1}{2}M^2\right)}T_0}$$

The pressure thrust is obtained from:

$$A\left(p_s-p_\infty\right) = \frac{m}{\rho V_j}\left(p_s-p_\infty\right) = [\text{Ideal gas law}] = \tag{A.88}$$
$$\frac{mRT_s}{M\sqrt{\gamma RT_s}p_s}\left(p_s-p_\infty\right)\frac{m\sqrt{RT_s}}{M\sqrt{\gamma}}\left(1-\frac{p_\infty}{p_s}\right) = \left[\frac{p_\infty}{p_s} = \frac{\frac{p_0}{p_s}}{\frac{p_0}{p_\infty}}\right] =$$
$$\frac{m\sqrt{RT_0}}{M\sqrt{\gamma}\sqrt{1+\frac{\gamma-1}{2}M^2}}\left(1-\frac{\left(1+\frac{\gamma-1}{2}M^2\right)^{\frac{\gamma}{\gamma-1}}}{\pi}\right)$$

The final expression for C_v is obtained by combining Equations A.89, A.88, A.79 and A.83:

$$C_v = C_{v,0}\frac{M\sqrt{\gamma}+\frac{\pi-\left(1+\frac{\gamma-1}{2}M^2\right)^{\frac{\gamma}{\gamma-1}}}{M\pi\sqrt{\gamma}}}{\sqrt{\frac{2\gamma}{\gamma-1}}\sqrt{1+\frac{\gamma-1}{2}M^2}\sqrt{1-\frac{1}{\pi^{\frac{\gamma-1}{\gamma}}}}} \tag{A.89}$$

For pressure ratios less than $\pi_{perfect}$, simple expressions for the thrust coefficient, C_v, can not be obtained due to the presence of compression chocks. However, the area ratio can then be controlled by varying the exhaust area, thereby maintaining high values for C_v. For this case, $C_v = C_{v,0}$ is used.

Solving the compatibility equations - Appendix B

To solve the nonlinear system of equations represented by equation 2.1, GESTPAN applies a quasi-Newton method (secant method). The quasi-Newton method is related to the classical Newton method, which is given by the well known sequence $\{x^k\}$ such that:

$$x^{k+1} = x^k + d^k \tag{B.1}$$

$$J(x^k)d^k = -F(x^k) \tag{B.2}$$

where $J(x^x)$ is the Jacobian (the n-dimensional derivative) of the system.

To proceed from a starting estimate, x^0, the Newton method requires the Jacobian to be known (or at least an approximation obtained from finite differencing). An initial approximation of the Jacobian requires that n+1 function evaluations are performed. One function evaluation is necessary to obtain the error vector at the present point x^k. By knowing $F(x^k)$ and then perturbing one component of the design vector x^k, n elements in the Jacobian (a column) can be found by finite differencing. Thus, determining the whole Jacobian requires n+1 function evaluations and thus n+1 function evaluations are necessary for every iteration step in the Newton method. The quasi-Newton method tries to reduce the computational cost associated with this process by improving (updating) an old estimate of the Jacobian in every iteration step without actually carrying out the full finite differencing procedure.

There is actually a whole class of quasi-Newton methods differing in how the update of the Jacobian is formed. To be precise, it is the inverted Jacobian that is updated in the method used by GESTPAN, as proposed by Broyden in [99] and [100].

The common feature of all quasi-Newton methods is the secant requirement. This is derived by a Taylor expansion of the function F about an arbitrary reference point, say X_0.

$$F(X) = F(X_0) + J(X_0)(X - X_0) \tag{B.3}$$

where $J(X_0)$ is the Jacobian of F in X_0. If we select two points, X_i and X_{i+1}, we can form

$$F(X_{i+1}) = F(X_0) + A(X_0)(X_{i+1} - X_0) \tag{B.4}$$

$$F(X_i) = F(X_0) + A(X_0)(X_i - X_0) \tag{B.5}$$

where A is an approximation of the Jacobian. The secant requirement is formed by subtracting B.4 from B.5, yielding:

$$X_{i+1} - X_i = H(X_i)\left(F(X_{i+1}) - F(X_i)\right) \tag{B.6}$$

where $H(X_0) = A(X_0)^{-1}$, i.e. $H(X_0)$ is formed by inverting the approximated Jacobian. It can immediately be seen that equation B.6 represents a system of n equations in n^2 unknowns. Thus,for for $n > 1$, the choice of $H(X_0)$ is not unique. All secant methods have equation B.6 in common but differ in how the remaining requirements for determining $H(X_0)$ are formulated.

In the quasi-Newton method used by GESTPAN, the Jacobian is only computed for the first step in the iteration (unless it becomes singular, see [37]). H_0, is then formed directly, by inverting the Jacobian, and is subsequently updated by the following formula:

$$H_{i+1} = H_i - \frac{(H_i y_i - t_i p_i)\, {p_i}^T H_i}{{p_i}^T H_i y_i} \qquad \text{(B.7)}$$

$$y_i = F(X_{i+1}) - F(X_i) \quad , \quad x_{i+1} = x_i + t_i \cdot p_i \quad , \quad p_i = H_i F(X_i)$$

Broyden derived equation [B.7] by arguing that new information on how A_i, and thereby also H_i, changes during an iteration step with the secant method, is gained only in that very direction. A reasonable requirement for A_i was then to assume that changes in F predicted by A_i in other directions than the step direction, p_i, should be no different from changes predicted by A_{i+1}, i.e.:

$$A_{i+1} q_i = A_i q_i$$
$${q_i}^T p_i = 0$$

where q_i obviously defines the direction orthogonal to the step direction p_i.

The most important and striking feature of this method is that, although the iteration matrix H_i changes from step to step, no evaluations of $f(x)$ are required beyond those that would have been necessary if H_i remained constant. Another feature of this method is that, as x_i approaches the solution, the assumptions made in deriving it become more valid. Hence, if A_i tends to approach the Jacobian matrix, the rate of convergence is expected to improve, ultimately becoming superlinear.