AIRCRAFT JET ENGINE CONDITION MONITORING THROUGH SYSTEM IDENTIFICATION BY USING GENETIC PROGRAMMING

Seyed Hosein Nayyeri

A thesis
in
The Department
of
Electrical and computer Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Electrical Engineering
Concordia University
Montréal, Québec, Canada

April 2013

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: **Seyed Hosein Nayyeri**

Entitled: **Aircraft Jet Engine Condition Monitoring through System Identification by using Genetic Programming**

and submitted in partial fulfillment of the requirements for the degree of

Master of Electrical Engineering

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining commitee:

_____ Dr. Raut, Chair

_____ Dr. Hashtrudi Zad, Examiner

_____ Dr. Dolatabadi, Examiner

_____ Dr. Khorasani, Supervisor

Approved by _____

Dr. W. E. Lynch

Department of Electrical and Computer Engineering

_____ 2013 _____ _____

Dr. Robin A. L. Drew, Dean

Faculty of Engineering and Computer Science

Abstract

Aircraft Jet Engine Condition Monitoring through System Identification by using Genetic Programming

Seyed Hosein Nayyeri

In this thesis a new approach for aircraft jet engine condition monitoring is proposed based on system identification and by using Genetic Programming (GP). This approach consists of two fault detection and isolation parts. In the detection part, the relationship between the engine Exhaust Gas Temperature (EGT), as a major indicator of the engine health condition, and other engine parameters and operating conditions corresponding to different phases of the flight is modelled using the GP technique. Towards this end, flight characteristics are divided into several phases such as the take-off and the cruise. The GP scheme is then used to discover the structure of the interrelations among engine variables. The constructed model is then used to detect abrupt faults in the engine performance.

For the isolation purpose, a hierarchical approach is proposed which narrows down the number of possible faults toward the target fault. The GP algorithm is then exploited to extract a series of nonlinear functions of the engine variables called fault indices. These indices attempt to magnify the signature of a fault in the engine by combining the effects of a fault on the engine parameters. These indices subsequently provide the necessary residuals for classifying the faults.

The approaches developed in this thesis provide an effective strategy for inspecting the aircraft jet engine health condition without requiring any specific information on the engine internal characteristics. The main advantage of the proposed approaches over other data driven methods such as neural networks is that our approaches provide a simple and tangible mathematical model of the engine rather than a black box model. The performance of the proposed algorithms are demonstrated and illustrated by implementing them on a double spool jet engine data that is generated by using the Gas turbine Simulation Program (GSP) software.

Acknowledgments

I would like to dedicate this thesis to my dear wife for all the supports and love she gave to me. I would like to sincerely thank my supervisor, Dr K. Khorasani, for his constant support, encouragement and inspiring advices through the course of this thesis. I would also like to thank all my friends and colleagues at Concordia University.

Contents

List of Figures

List of Tables

List of Abbreviations and Symbols

am	Ambient
C	Compressor
CC	Combustion chamber
η	Efficiency
f	Fuel
H_u	Fuel specific heat,$\frac{J}{Kg}$
R	Gas constant, $\frac{J}{Kg.K}$
HC	High pressure compressor
HT	High pressure turbine
γ	Heat capacity ratio
d	Intake
LC	Low pressure compressor
LT	Low pressure turbine
$\dot{m}$	Mass flow rate, $\frac{Kg}{s}$
M	Mach number
n	Nozzle
P	Pressure, *pascal*
P_0	Pressure at sea level at standard day
W_C	Power consumed by compressor, W
W_T	Power generated by turbine, W
J	Rotor moment of inertia, $\frac{Kg}{m^2}$
N	Rotational speed, *rpm*
N_1	Rotational speed of spool connecting the low pressure compressor to the low pressure turbine, *rpm*
N_2	Rotational speed of spool connecting the high pressure compressor to the low pressure turbine, *rpm*
c_p	Specific heat at constant pressure, $\frac{J}{Kg.K}$
c_v	Specific heat at constant volume, $\frac{J}{Kg.K}$
T	Turbine
T	Temperature, K
T_0	Temperature at sea level at standard day

Chapter 1

Introduction

With the development of air travel industry and transportation, aircraft have become an inevitable part of the everyday life. Growing demand and manufacturing costs and constraints necessitate aircraft to be able to stay in service as long as possible and fly more frequently with the lowest possible costs. On the other hand, safety considerations and reliability issues are the key factors in the success of any aerial business [1]. This trade-off between safety requirements and maintenance and operational costs have resulted in significant efforts to develop efficient health monitoring systems that are able to reduce maintenance costs as well as increasing the flight reliability.

Among all parts of an aircraft, engines are probably the most complex, expensive and critical component. Any major fault in an aircraft engine can lead to a tragedy. The traditional approach in maintaining aircraft engines is a time-based schedule. However, due to human safety and uncertainties embedded in this approach, extremely conservative and high safety factors are considered in time-based scheduling, resulting in unnecessary or sometimes late maintenance actions. Consequently, finding an optimal time in which the engine has used most of its useful life without violating safety thresholds are of much interest. A lot of efforts have been made to develop flexible maintenance schedules and health monitoring systems that

enable one to monitor the aircraft engine performance and even predict the critical conditions.

Engine health management solutions are used in performing fault detection, identification and isolation and consequently, in prediction of the system health parameters. Engine degradation or ageing can be estimated by using such algorithms. Engine performance deviation is linked to the change in the monitored health parameters of the engine. Not necessarily any change in the system states can be attributed to a fault, as such changes can be caused due to the system operating conditions changes. By properly detecting engine degradation, maintenance actions can be taken before a fault actually occurs. In most cases unscheduled maintenance actions that are triggered by faults are more expensive to resolve than scheduled and condition-based maintenance. Furthermore, by early anomaly detection it becomes possible to prevent damages to the engine that are caused by the initial fault propagation [2].

The importance of aircraft jet engine health monitoring and management has been well recognized in the literature. Research approaches in this area can be mainly categorized into model-based approaches and intelligent-based approaches. The main advantage of the model-based approaches is their analytical properties while most of intelligent-based approaches are basically a black box which provide no or low intuition about their operation. In addition to this in many practical applications there is a possibility of having limited recorded snapshot data from the engine variables (sensors) that are collected over a flight. In this case it is difficult to train data driven models such as neural networks. On the other hand, model-based approaches usually contain more simplifying assumptions than intelligent-based approaches. In addition, model-based approaches require knowledge about the engine physics and operation. Recently hybrid approaches have become popular as they can provide advantages of both approaches in one scheme while minimizing the corresponding disadvantages at

the same time [3].

1.1 Problem Statement

The main objective of this research is to combine two model-based and intelligent-based approaches in the context of fault detection and isolation (FDI) technology in jet engines. Towards this end, in this thesis the main goal is to develop a scheme based on the genetic programming (GP) algorithm to capture degradations and abrupt faults in a dual spool aircraft jet engine. The goal is to simultaneously take advantage of benefits of both data driven techniques and model-based approaches for developing a fault detection and isolation methodology.

In comparison with model-based approaches, the main advantage of using GP algorithm is that no information on the engine components characteristics such as the turbine and the compressors maps are required. On the other hand, in comparison with data driven approaches such as neural networks, the GP approach provides simple, practical, and explicit models (and not black box models) for the engine at different operating points.

1.2 Literature Review

1.2.1 Fault Detection, Isolation and Identification

The need for ensuring reliable and safe systems has attracted a significant research towards the health monitoring and fault detection and isolation (FDI) problem. Health monitoring refers to techniques and processes used to monitor the condition of the system through the so-called indicating parameters. Changes in these parameters

correlate to deviation of the system from normal operation. Health monitoring enables one to predict the possible failure in the system due to a fault or as a result of scheduled and required maintenance actions to prevent more damage to the system. It is more cost effective to predict the condition of the engine and take preventive actions before it suffers from suboptimal and unsafe performance which may decrease the efficiency and eventually turn into a total failure. Traditional approach towards FDI is to use hardware redundancy, such as multiple sensors to measure specific system parameters and compare the results with other sensor measurements and seek for incompatibilities between their outputs (Figure 1.1). Although this approach is very reliable but the main problem with it is the high cost of the required equipment and extra weight and space necessary for them in some applications such as aircraft. These limitations have motivated researchers toward developing software health monitoring or analytical redundancy techniques that have led to the development of a large body of FDI schemes in the literature.

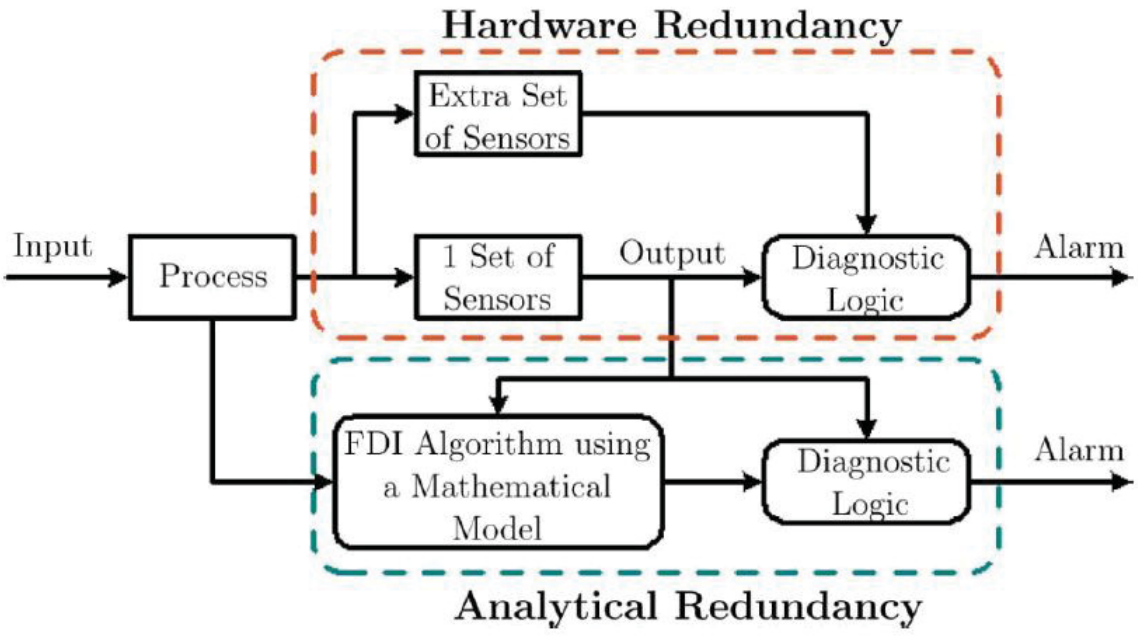

Figure 1.1: Hardware redundancy methodology [4].

Recent surveys on different FDI techniques can be found in [4] and [5]. Figure 1.2 shows a summary of the common FDI approaches that are available in the literature.

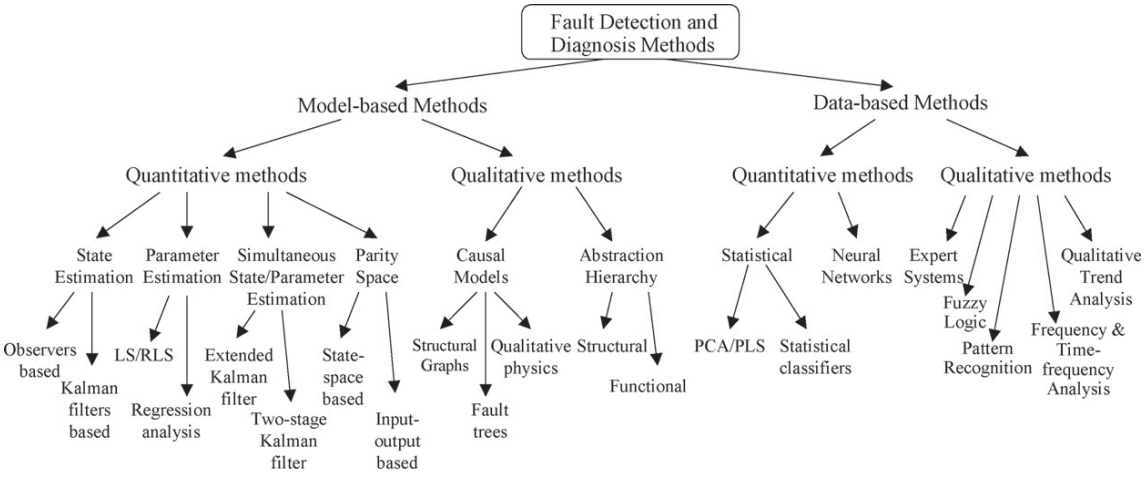

Figure 1.2: FDI approaches classification [5].

Software redundancy methods are divided into quantitative and qualitative approaches. Quantitative approaches use explicit mathematical model of the system while qualitative approaches use implicit models developed using artificial intelligence methods [4]. A key element in any FDI approach is finding a set of robust residuals that have low sensitivity to noise and disturbances. Available techniques for generating residuals can be classified into several classes. Such as (a) observer-based and Kalman filter-based approaches [6, 7, 8], (b) unknown input observers [9, 10, 11], and (c) parity relations approaches [12, 13, 14]. Another approach for generating robust residuals is to convert the residual generation problem into an optimization problem in which minimizing the cost function leads to the development of the residuals with maximum sensitivity to the fault and minimum sensitivity to the disturbances. A survey of different approaches in this area can be found in [15].

System identification that is also the topic of this thesis is another technique to

determine the required FDI residuals. FDI can be accomplished through system identification or parameter estimation. In this approach a model of the healthy operating system is developed by using online or offline measured data. With the assumption that a fault in the system shows itself through changes in the model parameters one can perform the FDI by comparing the output of this model and the actual system output. A survey on current practices in this area is provided in [16].

1.2.2 FDI in the Jet Engines

The sensitivity and importance of the aircraft engine has made it a point of interest for many researches in the field of FDI. Many researchers have selected jet engine systems to validate and examine their novel methods. The references [17] and [18] represent some examples. Research approaches in this area can be mainly categorized into model-based approaches and intelligent-based approaches. In model-based approaches, the main objective is to construct an analytical relationship among the engine variables, and develop an explicit criterion for health monitoring. In intelligent-based approaches, it is common to use neural networks that is trained to implicitly learn the relationship among the engine critical variables and the engine health status, and generate reliable health status reports. Model-based methods are applicable when an accurate mathematical model of the engine and its components characteristics are available [19]. Various approaches are applied to model the changes in the engine parameters. A common approach to calculate engine unknown variables is using linearized engine model and estimation algorithms [20]. Nonlinear approaches can also be found in the litrature [21]. Intelligent-based methods need experimental aircraft data such as the flight recorded data. Data driven fault diagnosis methods cover a vast range of approaches, namely machine learning, statistical methods, and competitive learning techniques [22, 23, 24, 25, 26].

One of the main techniques in the jet engine fault diagnosis is the Gas Path Analysis (GPA) approach. Physical faults and degradations in the jet engine components result in changes in the thermodynamic performance of the engine, and consequently changes in the engine states such as temperature, pressure and rotational speeds. The most common type of fault in engine components are corrosion, compressor fouling, external objects, etc. [27]. These faults are characterized by components efficiencies and flow capacities. In the GPA approach one tries to analyze the engine health condition by comparing measurements such as pressure and temperature from different stages of the engine with estimated data from the engine model. Reference [27] has done a comprehensive survey on the available techniques in the GPA. This approach was extensively developed by Urban [28] and Volponi [29]. In this regard, using Kalman filter as an estimator is very common. References [30, 31] implemented different types of Kalman filters in engine analysis. Reference [32] provided a comparison survey on different filtering approaches for aircraft engine health estimation. In [1] authors use a differential analysis scheme to identify significant deviations in the performance of two engines on a single aircraft.

Artificial intelligence approaches such as neural networks [33, 34], and Bayesian networks [35], have also been widely used to estimate the health parameters in the gas path analysis. Variety of neural networks have been implemented for the FDI in gas turbines such as modular neural network system [36] and the feed forward back propagation neural networks [37]. Reference [38] provided a study on the effective feature extraction using neural networks for novelty detection in highly dynamic systems such as the gas turbines. A comprehensive review of neural network-based FDI methods can be found in [39]. Support vector machines and fuzzy logic networks have also been introduced for fault diagnosis of jet engines in the literature [40].

Hybrid approaches have also been used in FDI of gas turbines. A hybrid automata

is proposed in [41] as a tool to perform FDI in the gas turbine engine. References [42] proposed a hybrid neural-network by using influence coefficients to model part of the system.

Among all engine parameters the Exhaust Gas Temperature (EGT) is known as the major indicator of the health condition of the engine [43]. Many researchers have investigated the engine performance by studying the EGT as a time series data. There are a number of methods that are available for change detection in one-dimensional time series data. These include statistical approaches [2, 44, 45], signal processing techniques [46] and computational intelligence techniques such as neural networks [47], and fuzzy logic [48]. Reference [49] applied univariate change detection techniques and multivariate change detections in aircraft engine fault diagnosis. Genetic algorithm (GA) has also been implemented in the field of gas turbine fault diagnosis. In [50] the authors introduced a multiple operating point fault diagnosis method using genetic algorithm analysis for the gas turbine fault diagnosis. Sensors fault diagnosis system with measurement noise and sensor biases are investigated in [51] through optimization of a cost function using a genetic algorithm.

1.2.3 Genetic Programming

In the context of system identification problem the main interest is on finding a mathematical model that can describe a real system with sufficient accuracy. System identification reduces to parameter estimation in case of physical systems with phenomenological models whose structures are built from physical considerations [52]. However, most of the practical systems have a complex nonlinear structure which makes it difficult to represent all the physical considerations underlying the phenomenon structures [52].

A common approach to deal with nonlinear complex systems is to linearize the

model. Although linear system identification methods provide simple and efficient tools for vast range of applications, there are applications in which this approach is not accurate enough. Model-based fault diagnosis is an example of these types of analysis [53]. The linearization modelling error in this case may lead to failure in detecting the fault or wrong fault alarms. As a result finding nonlinear models that can describe the system with high precision is of great importance.

Many system identification algorithms are introduced in the literature [54]. The most common approach in artificial intelligence is system modelling using neural networks. Neural networks have been successfully applied to many applications. In spite of the capabilities of neural networks to capture the input-output map of the system there is no systematic way to extract the structure of the model making the model a black box representation. Therefore many experiments are necessary to find an appropriate model. On the other hand in most cases any change in the system parameters requires new training of the model.

Genetic programming (GP) is an alternative approach for system identification. The GP is a powerful tool for modeling and identification of nonlinear systems. It allows one to develop nonlinear model structures that best fit the experimental data [55]. The GP is a relatively new field. It was first introduced by Koza in 1992 [56]. The GP is a generalization of the better understood genetic algorithm. The main difference between GP and GA is that in GP individuals are parse trees instead of fixed-length binary strings in GA. It is successfully used to develop nonlinear models in several applications such as finding appropriate Lyaponov functions in systems control [57, 58], identification of chemical processes [59] and [60], and the development of signal processing algorithms [61]. Reference [52] proposed an observer-based fault detection approach using genetic programming. It is shown that this observer is convergent under specific conditions and GP can be used to increase the convergence

of the proposed observer.

In [55] a twin water tank system is modelled by applying the GP algorithm. In addition, it is shown that GP is able to extract a meaningful model for an helicopter system using flight data. A hybrid approach composed of the GP and Simulated Annealing is proposed in [61] to automate the adaptive filter design procedure in digital signal processing applications. Reference [62] is an example of GP application in image processing. In that work GP is used to enhance the interest point design scope by automatically producing optimal interest point detectors to improve the human-machine innovation viewpoint. Both single and multi-objective optimizations are analyzed in that paper.

Reference [63] suggested augmenting GP with Orthogonal Least Square (OLS) algorithm to increase the speed and efficiency of the GP in finding order and structure of nonlinear models. Basically OLS is used to evaluate the contribution of each branch of the individual's structure tree. An interesting fully automated algorithm based on GP is presented in [64] to debug and repair computer codes. It is shown that this algorithm successfully repaired ten C programs with a total of 63000 lines. In [65] GP algorithm is used as a feature extraction tool in the power transformer Dissolved Gas Analysis (DGA) field. It is shown that three artificial neural networks (ANN), support vector machine (SVM) and K-nearest neighbour (KNN) classifiers augmented with GP have better classification performances. A fault diagnosis scheme for establishing the fault type of the power transformers insulation based on fuzzy model and genetic programming (GPFM) is proposed in [66]. In this method the structural fuzzy relationships among fuzzy variables are built by using GP.

1.3 Thesis Contributions

To the best of the author knowledge, the GP algorithm has not been used in the field of fault diagnosis and system identification of jet engines. The major contributions of this work are as following:

- A fault detection scheme has been introduced based on system identification of the jet engine using genetic programming technique for offline health monitoring of the aircraft engine.
- Four mathematical models are presented for estimating the engine exhaust gas temperature of a dual spool jet engine model using the GSP software [67] for the take-off and cruise phases of the aircraft flight. These models can predict the EGT with an error less than 0.2%.
- An enhanced fault isolation methodology is introduced to isolate eight types of faults in the dual spool jet engine. Seven analytical expressions are obtained as isolation residuals to isolate the faults.
- Mathematical models obtained for two take-off and cruise phases of flight are statistically validated by comparing the results with the industrial jet engine modeling software GSP.

1.4 Thesis Outline

This thesis consists of five chapters as follows: Chapter 2 includes an overview of the fault detection and isolation terminology. It also provides necessary notions and ideas in the field of evolutionary algorithms with focus on genetic programming. This chapter also includes a brief introduction to the jet engine mathematical models and thermal equations. An introduction to the GSP software is provided at the end of this

chapter. The proposed fault detection strategy and simulation results and discussions are presented in Chapter 3. Chapter 4 describes the methodology and simulation results for isolating eight types of faults in the dual spool jet engine. Future work and conclusions are summarized in Chapter 5.

1.5 Conclusions

This chapter provided an introduction to the problem of FDI in jet engines. The problem under the study was described and literature review on different approaches in the field of fault detection and isolation with focus on jet engines and state of the art researches using the GP algorithm are briefly reviewed.

Chapter 2

Background Information

2.1 Fault Detection, Isolation and Identification

The term fault in dynamical systems corresponds to any unusual deviation of the system from operating point or system parameters from nominal values [68]. Fault is different from failure. Failure is defined as permanent breakdown in the system in which system cannot continue to perform its tasks. Usually a growth of a fault leads to failure in the system. Generally faults are classified into three categories, actuator faults, sensor faults and component faults [26]. Fault in sensors corresponds to the cases when the output of a sensor is different from the actual value of the measured quantity. Some examples of sensor faults are zero offset, change of gain and change of hysteresis. In actuator faults, actuators are not working properly. In this situation system properties are not affected but the controlling capability is modified or disabled. Component faults are changes in system elements that modify the dynamical input-output characteristics of the system.

Faults are also categorized based on time, namely abrupt, incipient (degradation) and intermittent. Abrupt faults are sudden permanent changes in a system characteristic while for the incipient fault a system feature gradually deviates from its normal

value. Intermittent faults are short time lasting abrupt faults, as shown in Figure 2.1.

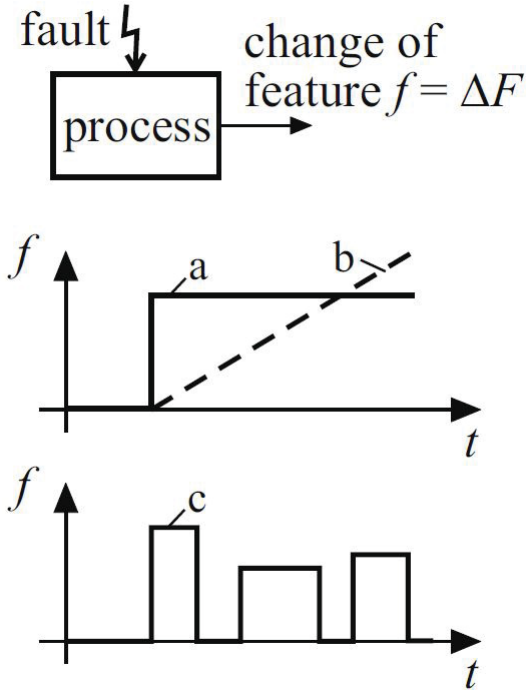

Figure 2.1: Time dependency of faults: (a) abrupt, and (b) incipient, (c) intermittent [68].

The task of health monitoring a system is to continuously monitor the system operation in order to find possible faults and prevent them to become a failure. A health monitoring and fault diagnosis system consists of the following subsystems:

Fault detection: Determining if a fault has occurred in the system or not,

Fault isolation: Finding the location of the fault in the system e.g. which actuator or sensor is faulty,

Fault identification: Determining the severity and the type of the fault.

The process of fault detection consists of the following steps:

Residual generation: residuals are signals that reflect the presence of a fault. Residuals are generally considered as the difference between the real system outputs and outputs from the system model.

Residual evaluation: The residuals are analyzed to find the time and location of the fault occurrence. An ideal residual reacts only to the fault to be detected but because of unknown inputs and noise in the measurements and also modelling errors, there are some variations in the residual even when there is no fault in the system. As a result to prevent false alarms wider thresholds must be considered. However, large threshold levels causes low severity faults not be detectable. Consequently, there is a trade off between the smallest possible detectable faults and the robustness of the detection residual. This situation can be improved by maximizing the sensitivity of the residual to the fault, filtering high frequency noise, by using adaptive thresholds and increasing the model precision [68]. Figure 2.2 shows a simple residual generation scheme.

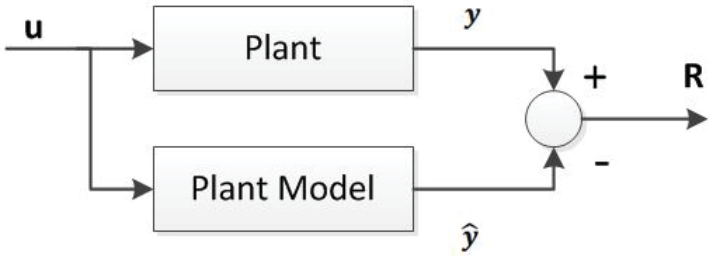

Figure 2.2: Simple residual generation process.

All model-based approaches more or less use a mathematical model of the system to generate residuals. Consequently, the most important part of this approach would be to obtain an accurate system identification [69]. Figure 2.3 shows the general configuration of a model-based fault detection system. Process model provides a mathematical representation of the relation between the input signal U and the output measurement Y. This model extracts required features of the process for different fault diagnosis techniques. The feature can be a physical system parameter or system state. Fault detection can be accomplished by comparing the estimated feature with expected or nominal value of the feature [68].

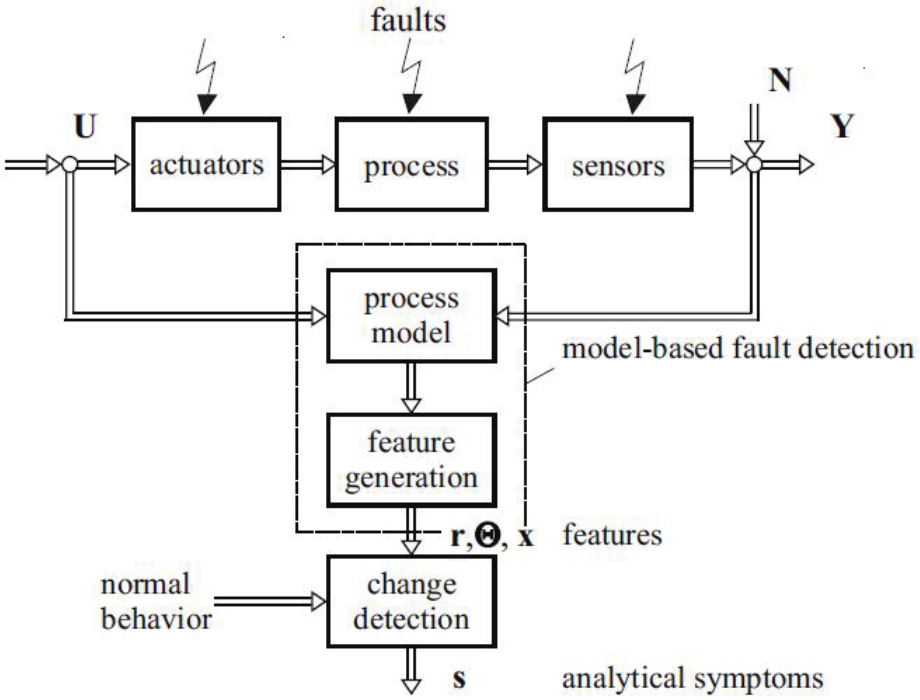

Figure 2.3: Model-based fault detection process [68].

Model-based FDI approaches are mainly classified into observer-based approaches, parity space methods and parameter identification approaches [68]. In many practical applications the precise model of the system is not available. Consequently, system identification must be applied to find the appropriate system model. Figure 2.4 shows different identification methods.

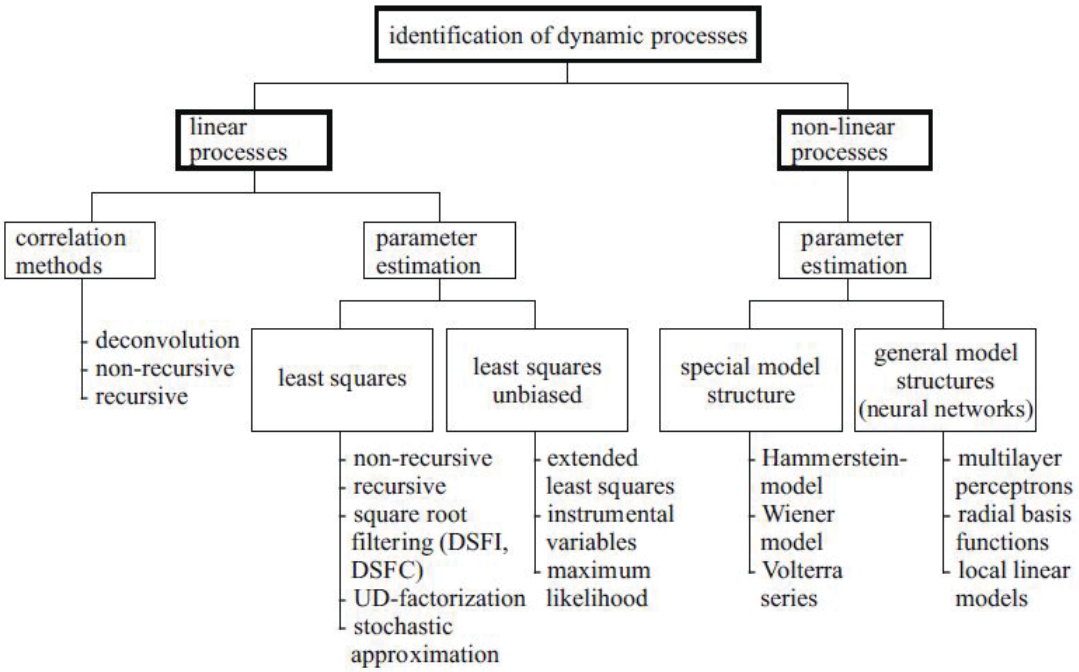

Figure 2.4: System identification approaches [68].

2.2 Jet Engine Overview

Figure 2.5 shows a schematic of the principal components of a jet engine. Briefly stated in one cycle of a single spool jet engine air flows inside the engine, the compressor compresses the input air and increases its pressure. This high pressure stream then enters the combustion chamber in which the fuel is injected into it and is burned. The combustion increases the pressure and temperature of the process gas. This high pressure and temperature flow enters the turbine and its energy converts to mechanical energy. As the flow passes through the turbine its temperature and pressure drops and finally expands to the ambient pressure from the end nozzle. This high speed gas produces required thrust for the aircraft to move forward. Although in a broad overview jet engines seem to operate in a same manner but closer look at it reveals a complex interaction among its components [70].

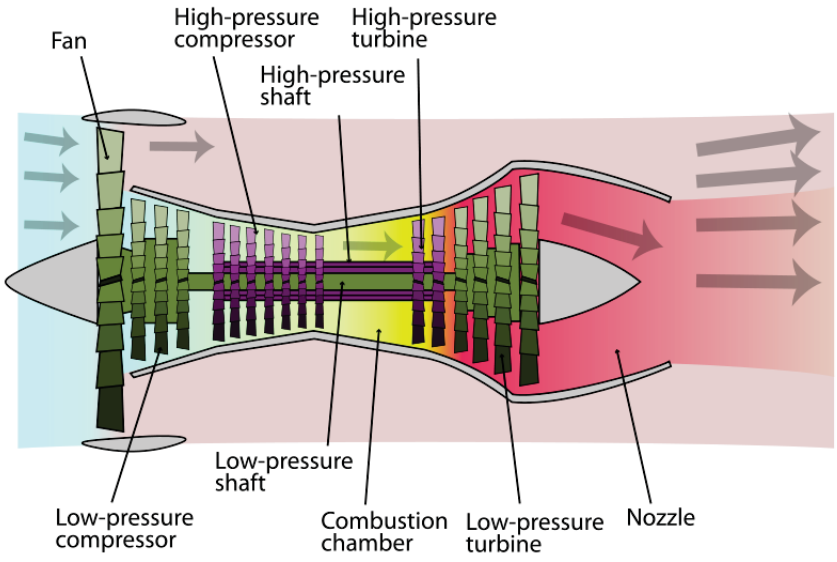

Figure 2.5: Turbofan engine components [71].

Varieties of engine designs are available in the industry. One classification of the jet engine is based on the number of compressors and turbines namely single, twin and three spools engines. Single spool engines are rarely used these days [72]. In a twin shaft engine two shafts connect the high temperature and the low temperature turbines to the high pressure and the low pressure compressors, respectively. In this thesis a twin spool turbofan engine model in the GSP software [67] is used to generate the required engine measurements. In the turbofan engine, a jet engine is used to rotate the fan at the front of the engine. In this engine, part of the incoming air passes through the fan directly producing the thrust and part of it enters the core jet engine which provide the required power to rotate the fan.

2.2.1 Jet Engine Modelling

Jet engines are built in many shapes and configurations. In general, different components in a jet engine are strictly coupled resulting in a complex nonlinear dynamic system. Figure 2.6 shows the information flow of the engine variables in a dual spool

engine module. In this figure dash lines show the flow of information and solid lines represent material flow among different components of the engine. The variable $\dot{m}_f$ represents the fuel flow to the combustion chamber. The air flow, temperature and pressure at different stages of the engine are shown by $\dot{m}_{**}$, T_{**} and P_{**}, respectively. The variable N_1 is the rotational speed of the low pressure turbine and compressor and N_2 is the rotational speed of the high pressure turbine and compressor.

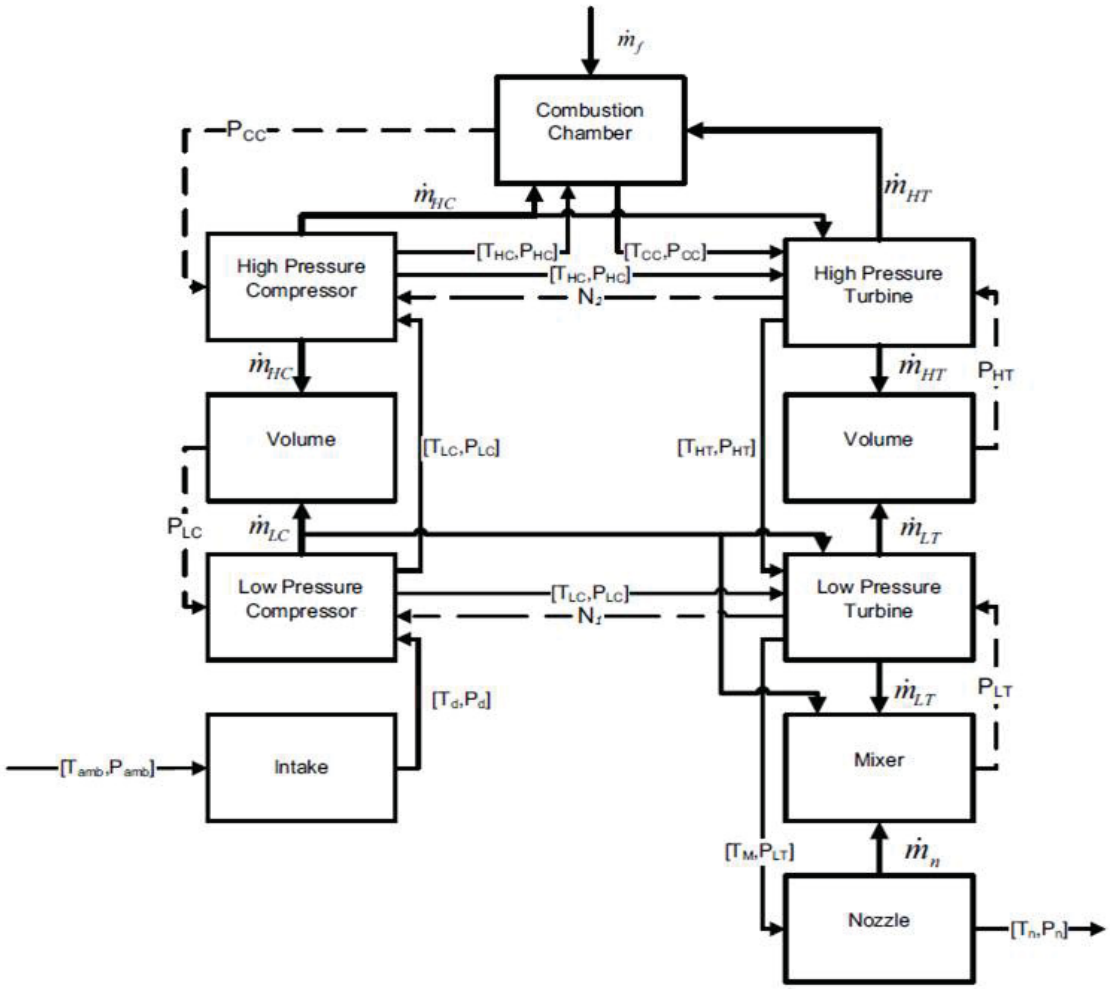

Figure 2.6: Engine parameters and components interactions [73].

Gas turbine engine in the simplest form can be modeled as a compressor, combustion chamber and turbine (Figure 2.7). Gas turbine operation at each stage in the engine can be identified by finding a collection of unknowns as summarized in Table 2.1.

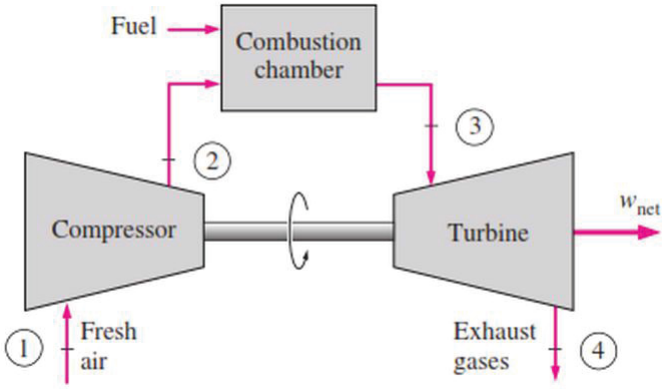

Figure 2.7: A simple gas turbine unit. 1-Stagnation conditions at compressor inlet. 2-Stagnation conditions at compressor outlet, and 3-Stagnation conditions at turbine inlet. 4-Stagnation conditions at turbine outlet [74].

Parameter	Description	Parameter	Description
$\frac{P_{02}}{P_{01}}$	Compressor pressure ratio	$\frac{N}{\sqrt{T_{03}}}$	Turbine non dimensional rotational speed
$\frac{N}{\sqrt{T_{01}}}$	Compressor non dimensional rotational speed	$\frac{\dot{m}\sqrt{T_{03}}}{P_{03}}$	Turbine mass parameter
$\frac{\dot{m}\sqrt{T_{01}}}{P_{01}}$	Compressor mass parameter	η_{turb}	Turbine efficiency
η_{comp}	Compressor efficiency	$\frac{\triangle T_{034}}{T_{03}}$	Temperature decrease in turbine
$\frac{\triangle T_{021}}{T_01}$	Temperature increase in compressor	$\frac{T_{03}}{T_{01}}$	Turbine inlet temperature to compressor inlet temperature ratio
$\frac{P_{03}}{P_{02}}$	Combustion chamber pressure ratio	W_{net}	Net power output
$\frac{P_{03}}{P_{04}}$	Turbine pressure ratio		

Table 2.1: Uknown engine parameters that need to be determined.

The above unknowns can be found by solving thermodynamical equations and component compatibility equations. The number of unknowns is more than the available equations and the process of finding the unknowns is an iterative process. For the compressor and turbine it is common to write the equations based on non-dimensional parameters as given below

$$\frac{\dot{m}\sqrt{T_1}}{P_1} = f_1(\frac{N}{\sqrt{T_1}}, \frac{P_2}{P_1}) \tag{2.2.1}$$

$$\eta_{comp} = f_2(\frac{N}{\sqrt{T_1}}, \frac{P_2}{P_1}) \tag{2.2.2}$$

$$\frac{\dot{m}\sqrt{T_3}}{P_3} = f_3(\frac{P_3}{P_4}) \tag{2.2.3}$$

$$\eta_{turb} = f_4(\frac{N}{\sqrt{T_3}}, \frac{P_3}{P_4}) \tag{2.2.4}$$

The explicit expressions for the non-dimentional functions f_1, f_2, f_3 and f_4, are provided by manufacturers in chart format called compressor and turbine operating maps. The temperature change equations are given by

$$\frac{\Delta T_{21}}{T_1} = \frac{1}{\eta_{comp}} \left[\left(\frac{P_2}{P_1} \right)^{\frac{\gamma-1}{\gamma}} - 1 \right] \tag{2.2.5}$$

$$\frac{\Delta T_{34}}{T_3} = \frac{1}{\eta_{turb}} \left[1 - \left(\frac{P_2}{P_1} \right)^{\frac{\gamma-1}{\gamma}} \right] \tag{2.2.6}$$

The engine components input and output are matched together by using compatibility equations for the rotating speed and the mass flow rate and the pressure ratios as follows

$$\frac{N}{\sqrt{T_3}} = \frac{N}{\sqrt{T_1}}\sqrt{\frac{T_1}{T_3}} \tag{2.2.7}$$

$$\frac{\dot{m}\sqrt{T_1}}{P_1} = \frac{\dot{m}\sqrt{T_3}}{P_3}\frac{P_3}{P_2}\frac{P_2}{P_1}\sqrt{\frac{T_1}{T_3}} \tag{2.2.8}$$

$$\frac{P_3}{P_4} = \frac{P_3}{P_2}\frac{P_2}{P_1}\frac{P_1}{P_4} \quad P_4 = P1 \tag{2.2.9}$$

2.2.2 Faults in the Jet Engine

By an abrupt fault one implies rapid reduction in the engine components performances such as reduction in the compressor or the turbine efficiencies. Engine degradation is a gradual reduction in the engine performance during its operation. An example of this kind of fault is the engine degradation that results from fouling or erosion [1].

Faults in a jet engine can occur in sensors, components or actuators. Sensor fault happens when the output of the sensor is different from the actual value of the measured parameters while actuator fault is the reduction in the actuating capability of the actuators. Examples of sensor fault and actuator fault in the engine are the wrong temperature reading of the thermocouple and the fuel valves failure to open or close correctly. The focus in this thesis is on component faults. Examples of common component faults in jet engines are fouling and erosion. Fouling results from accumulation of small particles on the turbine or the compressor blades resulting in reduction of the blades cross sections and overall reduction in the flow capacity in that component. Erosion results from collision of small particles with the compressor and the turbine blades. Both of these faults reduce the performance of the corresponding

engine components.

In mathematical modeling of the jet engine it is common to model physical faults in different components by considering some percent reduction in the component flow capacity or efficiency. In a typical dual spool jet engine 8 types of faults can be defined as shown in Table 2.2.

Component Fault	Description
HTflow	Decrease in the High pressure Turbine Mass flow capacity
HTeff	Decrease in the High pressure Turbine Efficiency
HCflow	Decrease in the High pressure Compressor Mass flow capacity
HCeff	Decrease in the High pressure Compressor Efficiency
LTflow	Decrease in the Low pressure Turbine Mass flow capacity
LTeff	Decrease in the Low pressure Turbine Efficiency
LCflow	Decrease in the Low pressure Compressor Mass flow capacity
LCeff	Decrease in the Low pressure Compressor Efficiency

Table 2.2: The description of the considered component faults.

2.2.3 Engine Performance Parameters and Condition Monitoring

Condition monitoring is used to capture deviations in the engine that occur over many flights. Data used in the trend analysis is a set of engine status parameters that are sampled at the same time. These parameters include gas properties such as temperature T and pressure P at different stages of the engine and control parameters such as fuel value, bleeding valve, etc. and also operational conditions such as ambient temperature T_{amb} and altitude H. Depending on the application, methodology and the flight phase, the data are either gathered continuously as a time series or as individual snapshots in each flight. Continuous sampling is useful for studying transient behaviour and short time phases of the engine operation such as the engine start up. Snapshots are used to monitor engine health status and performance deviation in the

long term. Each data point is collected when the engine has reached its steady state condition. It is common to take one snapshot in each flight and for each of the cruise or the take-off phases. In this research individual snapshots are used to analyse engine health status. GSP software is used to simulate the aircraft flights. Because of change in the weather conditions, flight distances and mission requirements a typical aircraft has different flight profiles during each mission. As a result the collected snapshots belong to an n-dimensional space of the engine variables, where n is the number of measured variables in each data snapshot. Table 2.3 summarizes the engine variables and their definition that are used in obtaining the results in this thesis. Part or all of these variables are considered as engine snapshot in different simulation scenarios in the next chapters.

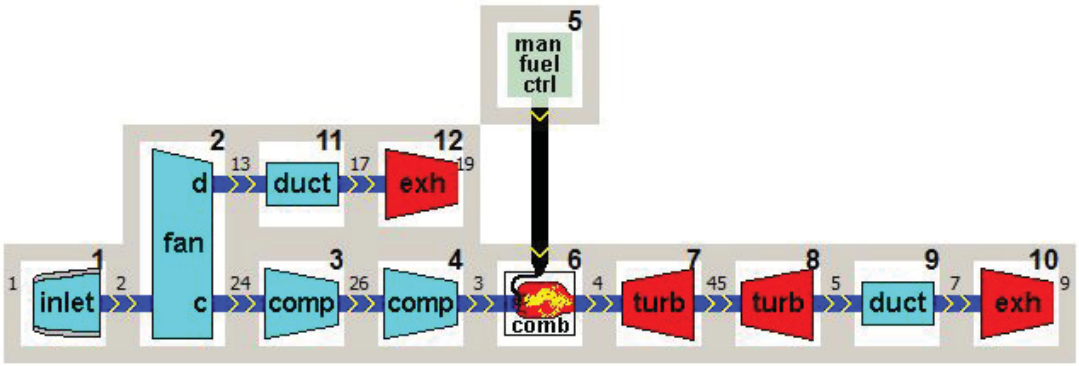

Figure 2.8: Engine variables and their location in the engine [75].

Engine variables	
W_f	Fuel flow
$T_{lpt(EGT)}$	Temperature after low pressure turbine
P_{lpt}	Pressure after low pressure turbine
T_{hpt}	Temperature after high pressure turbine
P_{hpt}	Pressure after high pressure turbine
T_{lpc}	Temperature after low pressure compressor
P_{lpc}	Pressure after low pressure compressor
T_{hpc}	Temperature after high pressure compressor
P_{hpc}	Pressure after high pressure compressor
N_1	High speed spool
N_2	Low speed spool
Operating conditions parameters	
H	Altitude
M	Mach
Ta	Ambient temperature

Table 2.3: Engine variables and the operating condition parameters.

2.3 Evolutionary Algorithms

Evolutionary algorithms (EAs) are a class of optimization algorithms inspired by the natural biological evolution and Dariwinian theory of evolution in order to find the optimum solution of a predefined problem. Artificial evolutionary algorithms generally consist of the following steps:

1. A coding system that enables the algorithm to represent a population of problem solutions that are referred to as individuals. In the GP a tree-based structure is a commonly used approach.

2. A fitness function is selected that evaluates the performance of each individual and ranks the individuals according to their assigned fitness value. The definition of this function depends on the problem objective and formulation. This fitness function value determines the probability of survival of each individual for the next generation in each evolution.

3. A set of operations that produce new individuals by manipulation of an individual or subset of individuals. Two commonly used operations are mutation and crossover. Mutation operation consists of creating small changes in an individual to obtain a new one. Crossover operation deals with recombination of two individuals in order to create new possible solutions.

4. A stochastic selection mechanism that decides which individual is to be passed on to the next generation and be used to produce the new populations thereafter [76].

Fitness-proportional selection, combined with mutation and crossover operators produce generation after generation of solutions. Since solutions with higher fitnesses values are given higher probabilities to reproduce, one can expect the solutions to be improved as generations continue.

2.4 Basic Concepts of Genetic Programming (GP)

The GP is arguably the most advanced and complex technique that is used in evolutionary computation, a generalization of the better understood and more widely used genetic algorithm (GA) [77], [78].

It is a symbolic-based optimization technique, that was developed by John Koza. Conventional optimization techniques usually consider a fixed structure for obtaining the possible solution and try to tune the parameters of the model. In contrast GP methods not only manipulate the parameters of the model but also attempt and use different model structures. The GP algorithm is summarized in Figure 2.9.

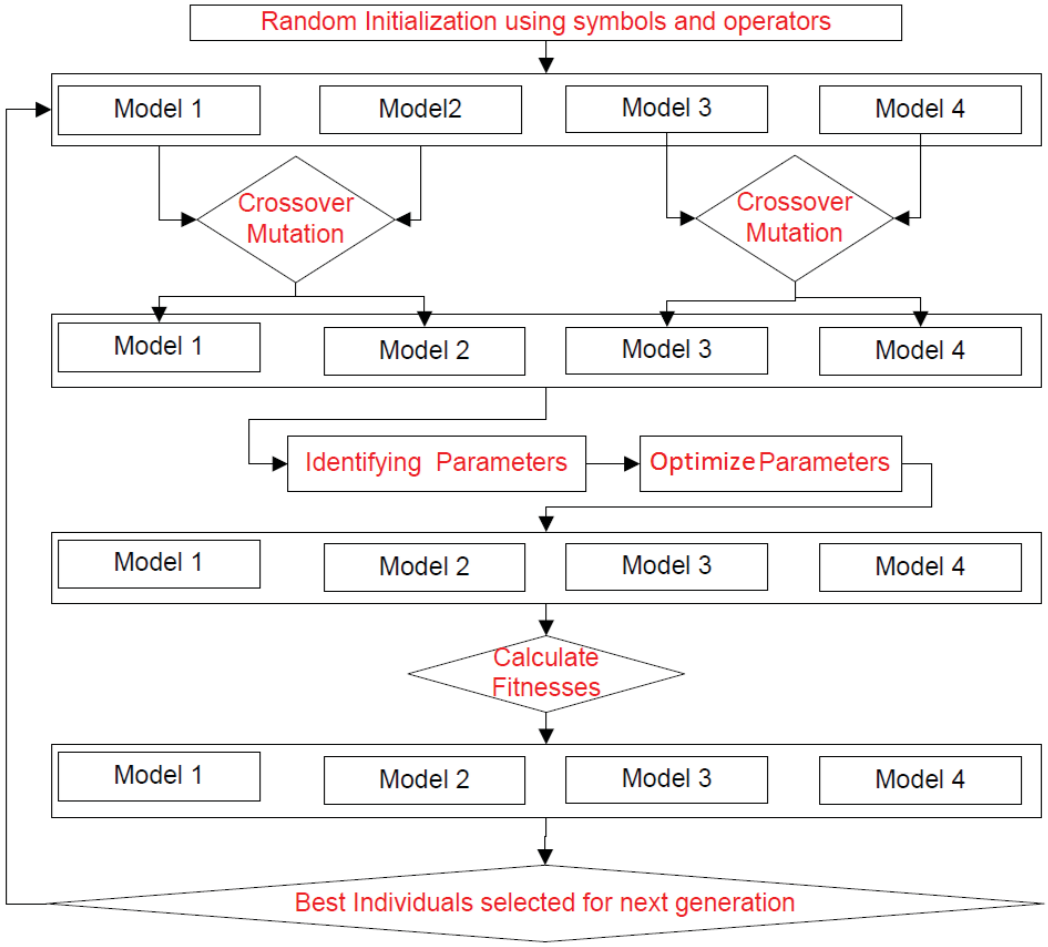

Figure 2.9: The flow chart depicting the GP modelling process.

2.4.1 Representation of Models

Individual models in GP are commonly constructed in a tree-like architecture using basic functions (non-terminal nodes) linking nodes of states and parameters (terminal nodes) [63]. In commonly used GP algorithms, individual solutions are represented through parse trees because this structure can be implemented by using simple computer programs, functions and mathematical operators [76].

The set of operators (non-terminal nodes) can contain the basic arithmetic operations, mathematical functions, conditional operators, Boolean operators or any other specifically defined function. For the sake of computational simplicity in this work

we just consider the following set of operators $O = \{+, -, *, /, \hat{}\}$.

The set of terminal nodes S contains the arguments and parameters of the functions. For example, we use $S = \{x_1, x_2, a, b\}$ with x_1 and x_2 denoting two independent variables and a and b representing the parameters.

With the above two sets a candidate model can be represented by a tree with ordered branches, using operators from the operations set and arguments or parameters from the terminal set. Figure 2.10 depicts an example tree.

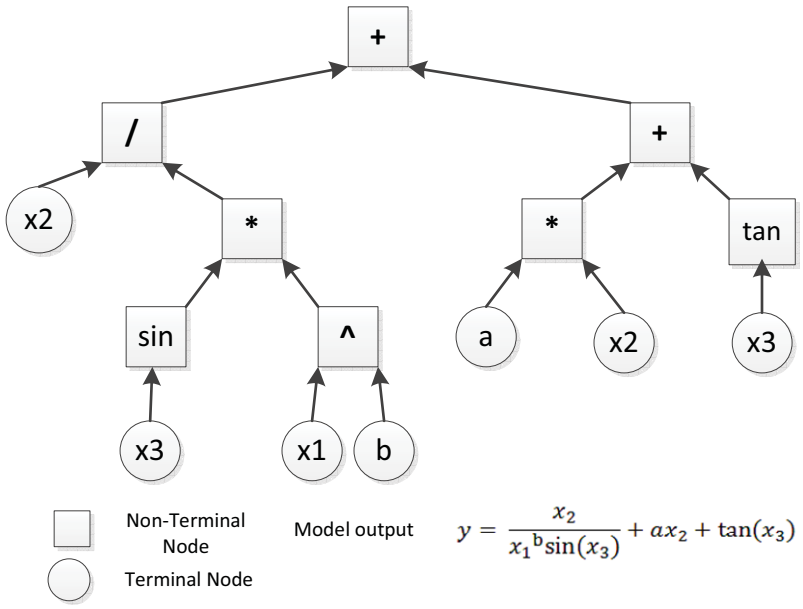

Figure 2.10: An example tree of an individual structure.

2.4.2 Mutation Operator

Mutation creates new features in the solution space by generating random perturbations in the model. Mutation operator randomly selects a node in an individual that is selected for mutation operation and replaces it with a new expression in order to obtain a new individual. Figure 2.11 shows an example of the mutation operation.

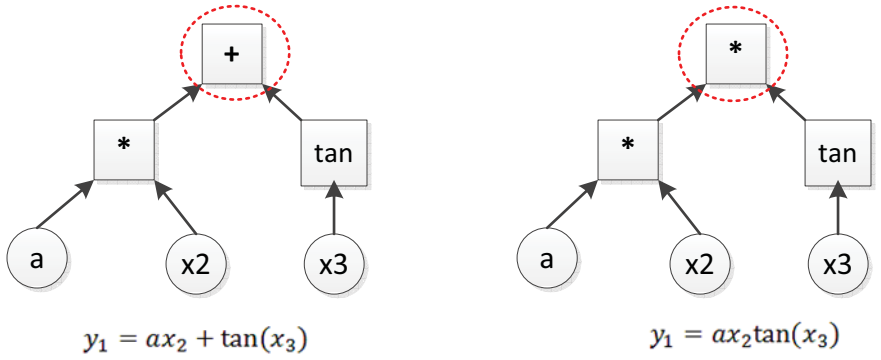

Figure 2.11: Mutation operation.

2.4.3 Crossover Operator

Crossover operator randomly selects two nodes of the two selected individuals and then exchanges their related subtrees to create new individuals. Figure 2.12 shows an example of the crossover operator.

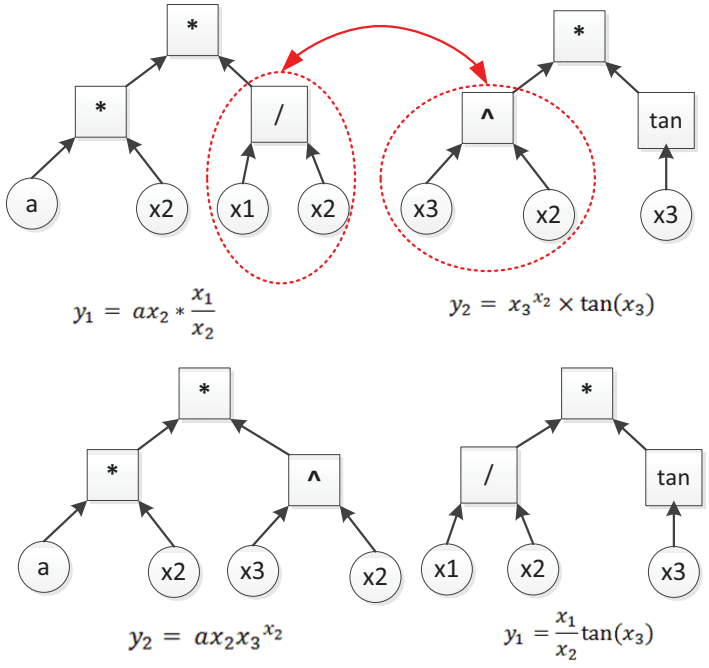

Figure 2.12: Crossover operation.

2.4.4 Fitness

In every iteration, the GP algorithm needs to evaluate the individuals. The fitness function quantifies and evaluates the goodness of a potential model. The probability of the selection of an individual is proportional to this value. Usually, the fitness function is defined based on the mean square error (MSE) between the calculated outputs from an individual model and the measured outputs from the actual system, as given by equation (2.4.1), namely

$$Fit = \frac{1}{\frac{1}{N}\sum_{i=1}^{N}\left(y(i) - y'(i)\right)^2} \tag{2.4.1}$$

where N is the number of the data-points that are used for the identification of the model, y is the measured data points and $y^{'}$ is the output from the actual model.

2.4.5 Parameter Estimation

Each model structure in the GP algorithm needs to be augmented with adequate tunable numerical parameters. A model structure alone may not give good results but the parameterized model does. To prevent loosing suitable model structures because of poor parameterization it is necessary to optimize and select these parameters for each nonlinear model before evaluating its fitness. The models are randomly generated and can therefore contain linearly dependent parameters or also contain parameters that have no effect on the output. Consequently, gradient based methods cannot be used. For this reason, other optimization methods such as the Nedler-Simplex and the simulated annealing methods can be applied to determine these parameters [79, 55]. In this thesis , the Nedler-Mead optimization technique has been implemented to optimize these parameters in each model [80].

Nedler-Mead algorithm is a simplex-based method which belongs to the more

general familiy of optimizations called direct search algorithms. It is designed to perform unconstrained optimization of a scalar nonlinear function $f : R^n \rightarrow R$ by just using the function value and without using any gradient. More information about direct methods are available in [81]. The simplex is the convex hull that is composed of $n + 1$ nodes $x_0, x_1, ..., x_n$ in the R^n space. A simplex in R^2 is a triangle and a simplex in R^3 is a tetrahedron [82].

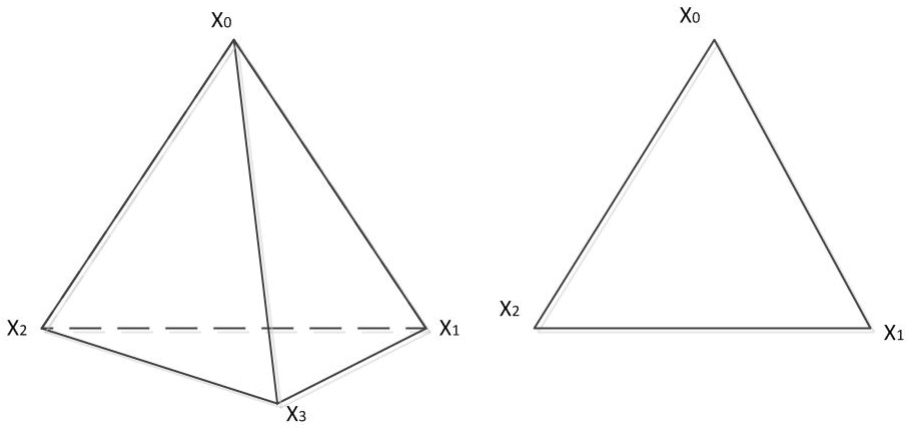

Figure 2.13: Simplex in R^3 and R^2.

In each step of the Nedler-Mead algorithm the current working simplex evolves to another simplex by computing several test points in a way that the value of the target function decreases. This process continues until the termination criteria is satisfied. To construct the initial simplex an initial guess is required. Assuming that X_0 is the initial guess to build a simplex around it a 5% is added to each element of the vector X_0 to X_0. These points together with X_0 compose an $n + 1$ points simplex. The simplex is modified by using the following steps.

Let S_i be the set of points in the current simplex. The value of the target function $f(x)$ is calculated for each of points in the simplex and sorted from the lowest $f(x_1)$ to the highest $f(x_n + 1)$.

-Find the reflected point x_r with respect to the point c and corresponding $f(x_r)$

by using the following equation

$$x_r = 2c - x_{n+1} \tag{2.4.2}$$

where c is the centroid of the best side $\sum_{i=1}^{n} \frac{x_i}{n}$.

- If $f(x_1) \leq f(x_r) < f(x_n)$ stop the iterations and accept the reflection point.

- If $f(x_r) < f(x_1)$ find the expansion point x_e and $f(x_e)$ by using the following equation

$$x_e = c + 2(c - x_{n+1}) \tag{2.4.3}$$

- If $f(x_e) < f(x_r)$ stop the iteration and accept the expansion point otherwise use x_r and accept the reflection point.

- If $f(x_r) \geq f(x_n)$ find the contraction point x_c by using the better of the two points x_{n+1} and x_r.

- If x_{n+1} is better than x_r ($f(x_r) < f(x_{n+1})$) then $x_c = c+(x_r c)/2$. If $f(x_c) < f(x_r)$ stop iteration and accept x_c, namely contract outside.

- If x_r is better than x_{n+1} ($f(x_{n+1}) < f(x_r)$) then $x_c = c + (x_{n+1} c)/2$. If $f(x_c) < f(x_{n+1})$ stop the iteration and accept the x_c namely contract inside.

- Find n new vertices using the following equation and the corresponding function values for $j = 2$ to $n + 1$

$$x_j = x_1 + (x_i - x_1)/2 \tag{2.4.4}$$

Construct the next working simplex using x_1, x_j, $j = 2$ to $n + 1$. Figure 2.14 shows possible points that are calculated during the simplex transform step.

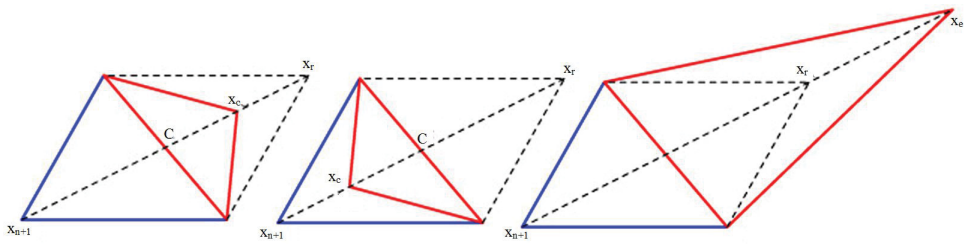

Figure 2.14: Simplex transform calculated points [82].

2.5 GSP Software

Gas turbine Simulation Program (GSP) [67] is a powerful modeling and simulation environment developed by the National Aerospace Laboratory (NLR) for offline analysis of gas turbine engines. It has been widely used in industrial gas turbines as well as aerospace applications such as off-design performance analysis, control system design and engine degradation and fault diagnostics. GSP is a graphical component based simulation environment. Different types of gas turbine engines can be constructed in this environment by adding and rearranging engine components. Both transient and steady state behaviour in the engine can be modelled using GSP [83]. Figure 2.15 shows the GSP software modeling environment.

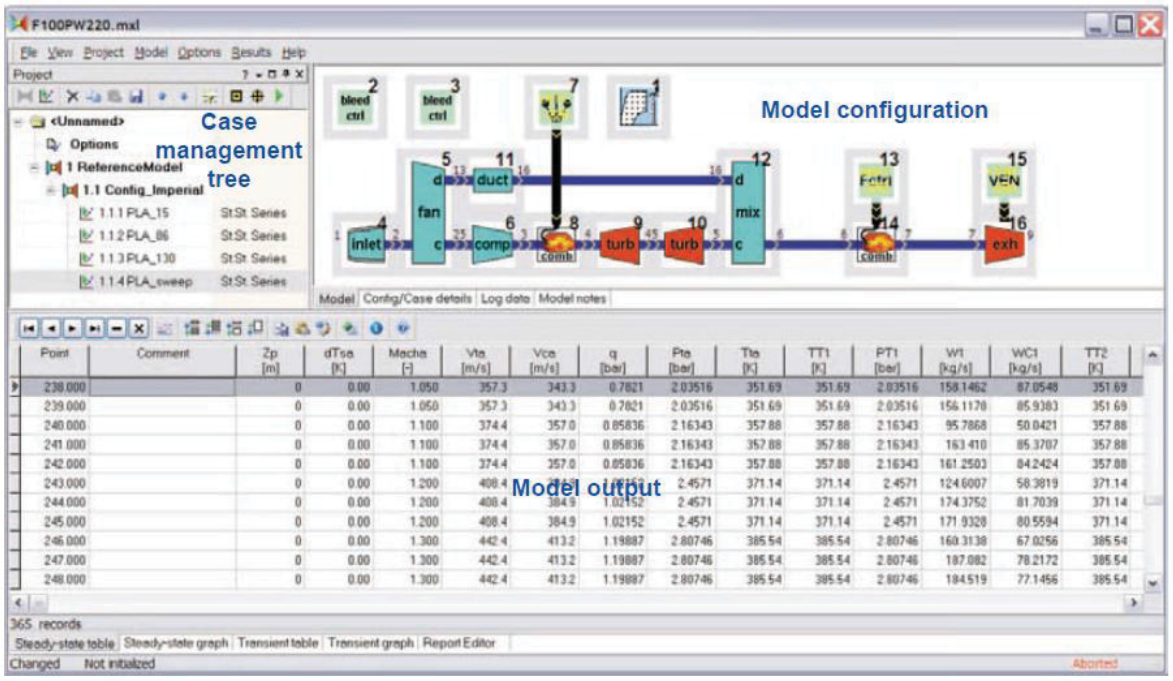

Figure 2.15: GSP software modeling environment [83].

Each of engine components such as the compressor and the turbine has a series of characteristics that either can be set by the user interests or extracted from the software data base. GSP uses a zero dimensional (non-dimentional) analysis to model the engine operation. Zero dimensional implies that at each cross section area between two engine components the average of the working flow properties is considered. In this method the thermodynamical equations of the working fluid in each component is solved by using inlet and outlet boundary conditions. Starting from the first component which has the flight conditions as its boundary, for example inlet, GSP finds the components characteristics step by step by using the output of the previous component in the model as the inlet conditions for the next component. Input conditions and characteristic maps of each component are used to solve a set of nonlinear thermal and dynamical equations of the process gas to find the component outputs [67], [84]. A typical map of the compressor in the GSP database is shown in Figure 2.16. Operational maps provide relations between component parameters such as efficiency,

flow rate and pressure ratio at different operating points.

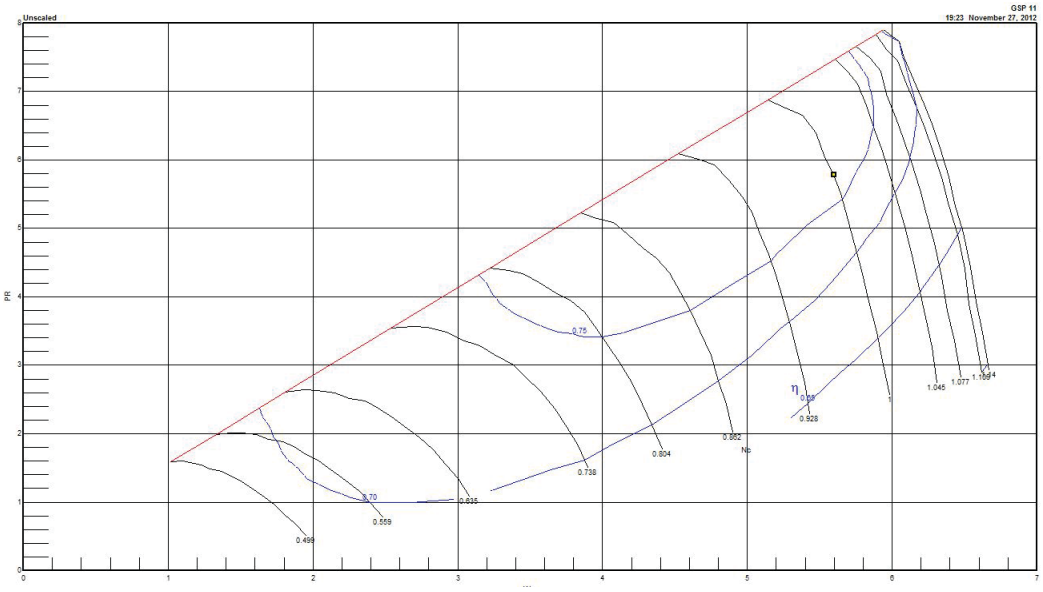

Figure 2.16: Compressor map in the GSP database [75].

2.6 Conclusions

In this chapter a review on the concepts and methods that are applied in this research were presented. First general notions in fault diagnosis were explained. Jet engine operation and components and different types of faults in the jet engine were discussed in the second part. Basic concepts in the GP algorithm and parameter estimation techniques were described in the third part. The GP algorithm starts from a generation of randomly generated models known as parents. These models are basically a series of nonlinear functions built from a set of basic functions and states. Basic functions set can contain simple mathematical operators, boolean operators or any defined function. Through the GP algorithm this first generation evolves and the best models defined by the highest fitnesses survive to the next generation. As a result the final generation is a collection of system models that best describe the

system operation. A brief review on the GSP software were also provided at the end of this chapter.

Chapter 3

GP Algorithm for Jet Engine Fault Detection

As mentioned in the previous chapter, we are interested in detecting the faults and performance deviations in the aircraft engine in order to reduce the maintenance and operating costs as well as risk of human catastrophes. Towards this end, in this thesis the focus is on offline analysis of the collected data during the aircraft flight. In the previous chapter the GP algorithm as a powerful optimization and system identification tool was introduced. In this chapter we are going to use the capabilities of the GP algorithm in the context of fault detection in a jet engine.

3.1 Methodology

In the offline health monitoring of the jet engine, the recorded data from the aircraft engine are usually a set of snapshots of the engine states and operational conditions during each flight and for different phases of the flight such as take-off and cruise. Since most data is recorded in the steady state conditions, one can expect that the interrelationship of the engine parameters can be approximated by a static

nonlinear function corresponding to a range of flight conditions. By invoking this assumption, our goal is to construct the structure of such interrelationships by using the GP scheme. The main idea here is to use the GP technique to find a nonlinear function that relates the engine gas temperature (EGT) as a major engine degradation indicator to the other engine parameters and operational conditions as $EGT = f(T_i, P_i, W_f, N_1, ...)$. The resulting model is then used to detect abrupt faults in the engine performance.

Table 3.1 summarizes the obtained models for the two take-off and cruise phases of the flight. These models are explained subsequently in the simulation results section. Faults or degradations in the aircraft jet engine usually manifest themselves through increases in the engine EGT. By comparing the estimated EGT from the GP model and the measured data from the engine flights one can identify the deviations and faults in the engine performance.

Take-off models	$EGT = \frac{aN_1}{3P_3+6N_1+dW_f}$
	$EGT = \frac{1}{2}(\frac{a}{D}W_f + \frac{b}{D}W_f^{\frac{cM}{2*T_{am}+D+2}} + 1)$
Cruise models	$EGT = \frac{aM^3-bH+C}{dN_1^{3N_2}}$
	$EGT = \frac{aW_f+B}{cH^2-dHT_{am}^2+E}$

Table 3.1: Models obtained for estimating EGT in different flight phases.

3.1.1 Health Monitoring Procedure

The key element in this approach is finding an accurate model to estimate the engine EGT and compare it with the measured EGT from the engine. The fault detection residual can be defined as the difference between these two values. Once the model is found in short term we can expect that small changes in the engine do not alter the entire structure of the model and these changes can be captured by optimizing

the parameters of the related model structure at each time period. From the long-term perspective due to ageing, the engine dynamics may deviate significantly that in addition to the numerical parameters the structure of the model also needs to be modified. In this case the new structure can be found by applying the GP approach again. In summary, the following steps can be identified in our proposed fault detection procedure.

Use the GP technique and collected engine snapshots to find the best model fitted to the selected data in each category. This is done by minimizing the difference between the model EGT and the measured EGT. The number of required data points depends mainly on the complexity of the selected engine, range and diversity of the data points and the number of engine parameters used in constructing the models structures. The fault detection residual is then constructed as the difference between the estimated EGT using this model and the measured EGT values. If the error is more than a predefined threshold, a fault has occurred in the engine or engine is degraded more than an acceptable threshold. Appropriate maintenance decision can then be made based on the type and severity of the fault.

In time and with engine ageing the accuracy of the model may reduce and the obtained structure may not be able to estimate the engine accurately enough. In this case the GP algorithm needs to be run again to find the new models for the older engine. Since it is not possible to obtain a general expression for the engine in all flight phases and operating conditions, if the collected data in flight are quite diverse then in order to get better estimation, it is appropriate to divide the data points into several categories based on the engine operating conditions and extract a model for each category separately. For example, the flight data in the cruise phase can be categorized based on the flight altitude range and the Mach number.

3.1.2 GP Implementation and Computational Limitations

As mentioned in the previous chapter in the GP scheme each individual is structured in a tree-like fashion, with basis functions linking nodes of inputs and parameters. In evaluating the fitness value of individuals in the GP scheme, it is common to add parameters to each structure and then tune them in order to obtain an optimal structure. The problem with this classical approach is that by increasing the depth of the tree the number of numerical coefficients increases drastically. On the other hand since the models are generated randomly simple models may find a very but not necessarily complicated form that is difficult to understand and investigate.

In order to reduce the number of required numerical coefficients and simplify the candidates form, in this thesis models are represented in a symbolic format. Although this approach makes the algorithm implementation more complicated but this technique enables us to use Matlab symbolic power to simplify the models before performing parameter optimization or applying the GP operator (refer to Figure 3.1).

In addition, this approach enables one to directly consider some of the numerical coefficients in the generation evolution instead of adding them to the structure afterwards. Consequently, the numbers and the form of these coefficients are also optimized during the training process. It was observed experimentally that this approach leads indeed to better results.

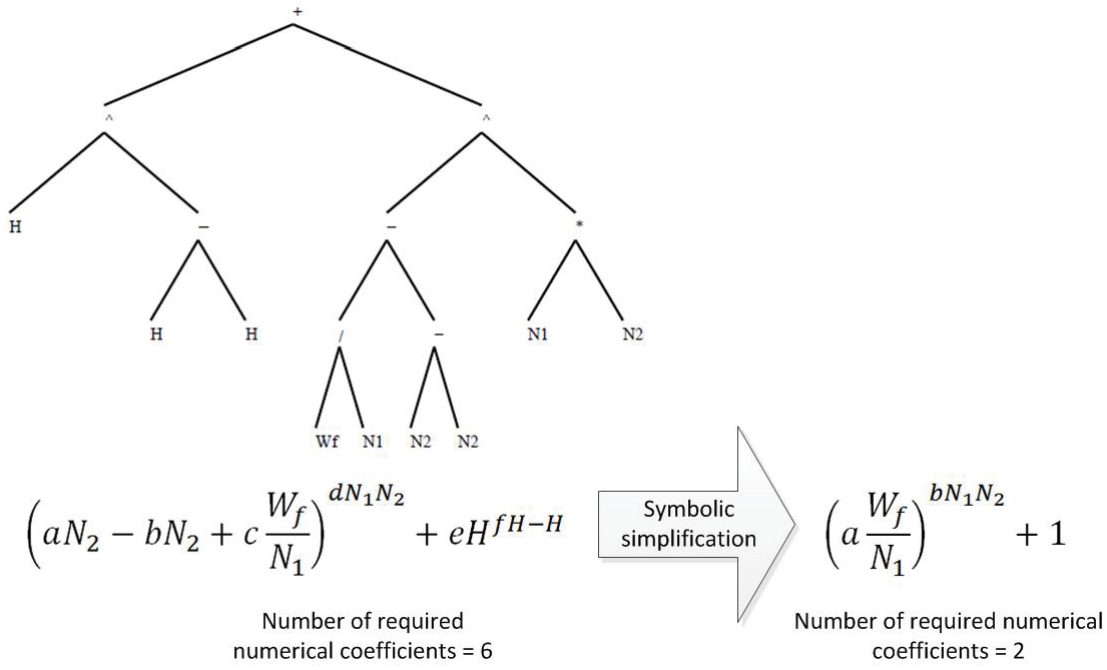

Figure 3.1: Symbolic simplification of the individual models.

As mentioned in the previous chapter, in the course of the GP run and in finding each individual's fitness, the corresponding numerical coefficients needs to be determined using the simplex optimization method. Simplex method success in finding the global minimum depends on the initial conditions (initial simplex) and also the number of iterations. As a result to increase the chance of the simplex algorithm in finding the best parameters of a model structure corresponding to its minimum error one can use several initial guesses instead of a fixed initial condition. However, due to the fact that the individual models are generated randomly one does not have any *a priori* knowledge about the range of these numerical coefficients. Consequently, it is necessary to define some intervals for the initial guesses and select the initial simplex randomly or by partitioning these intervals.

Another issue is that depending on the number of numerical coefficients n of a model the search space is an n-dimensional space. Although the simplex algorithm is very efficient in practice, however its performance reduces extremely by increasing

the number of its parameters. It is shown in [85] that the worst-case complexity of the simplex method is exponential in time. To overcome this problem, we have to limit the number of numerical parameters of each model structure. Since the algorithm calls the fitness function frequently due to these limitations it is not possible to consider a large number of initial points and iterations during the GP algorithm run. As a result, during the GP algorithm run we use a fixed number of iterations and a maximum number of parameters that are added to each model. Although it may lead to omitting some appropriate structures but it reduces the computational load and enables the algorithm to search more model structures. Finally, the output model structure of the GP algorithm will be fine tuned by adding more number of numerical coefficients and higher number of initial conditions.

3.1.2.1 Fitness Definition

For the fault detection model generation the fitness function is defined based on the mean square error (MSE) between the calculated EGT from an individual model and the measured EGT from the GSP engine model at the same operating conditions for the set of training data points, as given by equation (3.1.1)

$$Fit = \frac{1}{\frac{1}{N}\sum_{i=1}^{N}\left(EGT_m(i) - EGT_{GSP}(i)\right)^2} \tag{3.1.1}$$

where N is the number of the data snapshots from the GSP engine model that are used for the characterization of the model, EGT_m is the output from an individual model in the GP algorithm and EGT_{GSP} is the gas temperature from the GSP software.

3.1.2.2 Data Normalization

The collected parameters from the engine have different units and diverse scales. For example, the rotational speed numbers are in the order of 50000 rpm while the Mach number vary between 0 and 1. Directly using these values in the GP algorithm causes the smaller numbers to be ignored. It also reduces the efficiency of the optimizer in finding the correct numerical coefficients. To overcome these problems, different quantities are adjusted to a common scale where the min-max normalization technique is used which normalizes data by mapping each set of parameters to the range of 0 to 1. This is done by using equation (3.1.2), where X_{max} and X_{min} are the maximum and the minimum of an engine parameter in all the data set and X is the parameter that is going to be normalized, that is

$$X_n = \frac{X_{max} - X}{X_{max} - X_{min}} \tag{3.1.2}$$

3.2 Simulation Results

In this section, the method that was described in the previous sections is validated through simulations by using the GSP and the Matlab software. We have used the GSP software to simulate the aircraft engine and obtain the measured data points.

Engine faults is modelled by abrupt reduction in the efficiencies and flow capacities of the engine components. To model different flight situations, the operating conditions of each data point is selected different from the others that are compatible to the considered take-off or the cruise flight modes. A dual spool jet engine model is selected from the GSP databases for which the maximum flow rate of the combustion chamber is 2.4912kg/s. The engine is simulated for the take-off and the cruise modes seperately.

3.2.1 Take-off mode

The take-off mode is the time that an aircraft is accelerating in the run way to take-off from the ground. In this phase of flight, aircraft has its maximum weight and the engines are working at the maximum power to provide the required trust. As a result changes due to degradation and fault in the engine health parameters are more dominant in the take-off mode [43]. In this section, the results that were summarized in Table 3.1 are explained in details.

3.2.1.1 GP Algorithm Implementation

To obtain models shown in equations (3.2.1) and (3.2.2), initial parents models were produced randomly by combining the set of engine states and mathematical operators $\{+, -, *, /, \hat{}\}$ building a set of nonlinear expressions. Each of these structures was augmented with the appropriate number of parameters (Figure 3.2).

The added parameters to these expressions were optimized by using the simplex optimization algorithm and 20 snapshots of the engine data (corresponding to 20 flights) to best match the corresponding EGT from the GSP software by using the fitness function as described in equation (3.1.1). It is shown in Section 3.2.1.4 that by using more data points to optimize the parameters does not affect the results significantly. The best model structure is the model with the smallest error or the highest fitness. These initial models evolved in the GP algorithm to converge to the best models are shown in equations (3.2.1) and (3.2.2).

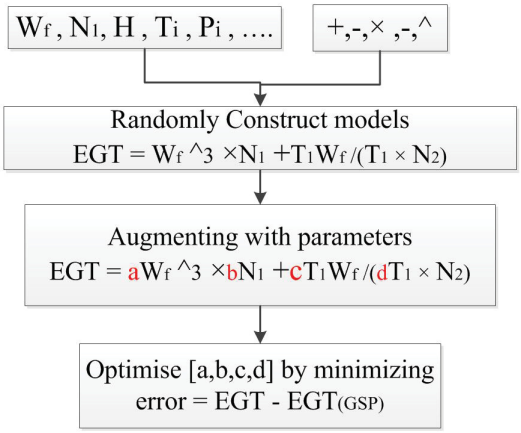

Figure 3.2: Parents initialization of the models.

The settings of the GP algorithm and the simplex optimizer algorithm are summarised in Table 3.2

GP Settings	**Value**	**Description**
N_P	30	Number of population
N_{TB}	60	Total number of new born individuals in each generation
P_{mut}	0.7	Mutation probability
P_{Cross}	0.3	Cross over probability
N_{gen}	50	Number of generations
N_{past}	3	Number of best individuals from current generation to be added to the new born children to keep good individuals
Optimization Settings		
N_{param}	5	Maximum number of parameters inserted in an individual model
Min_{err}	0.02	Minimum error value in simplex optimizer to stop the optimization process
It_{max}	150	Maximum number of iterations in the optimizer algorithm to stop the optimization
R_p	-1000,1000	R_p is the range of initial guesses of the numerical coefficients in the simplex algorithm

Table 3.2: GP algorithm settings and the parameter optimization settings.

3.2.1.2 Model Generation and Validation

It was shown that two different sets of inputs were used to obtain the two models to estimate the engine EGT. In the first model the only inputs to the GP algorithm are the external states and the operating conditions of the engine. These inputs consist of the altitude H, ambient temperature T_a, fuel flow W_f and the Mach number M. In the approach some internal sates of the engine namely the rotational speed of the low speed and the high speed spools N_1, N_2 and pressures after high pressure compressors P_3 are also fed into the GP algorithm in addition to the previous parameters.

The two obtained models for only the external states inputs and both internal and external states are shown in equations (3.2.1) and (3.2.2), respectively

$$EGT = \frac{1}{2}(\frac{a}{D}W_f + \frac{b}{D}W_f^{\frac{cM}{2*T_{am}+D+2}} + 1) \tag{3.2.1}$$

a	257.13
b	76.408
c	150.42
D	-645.845

$$EGT = \frac{aN_1}{3P_3 + 6N_1 + dW_f} \tag{3.2.2}$$

a	2.93
b	-3.967

These models performances are represented by comparing the EGT output of the obtained models and a set of test data from the GSP software. Typical results are plotted in Figure 3.3 and Figure 3.4. In these figures the 20 first data are the data points that were used to find the model and optimize the model parameters. The remaining data are the test data. The maximum error of the estimated EGT from these models and GSP model is less than 0.2 %.

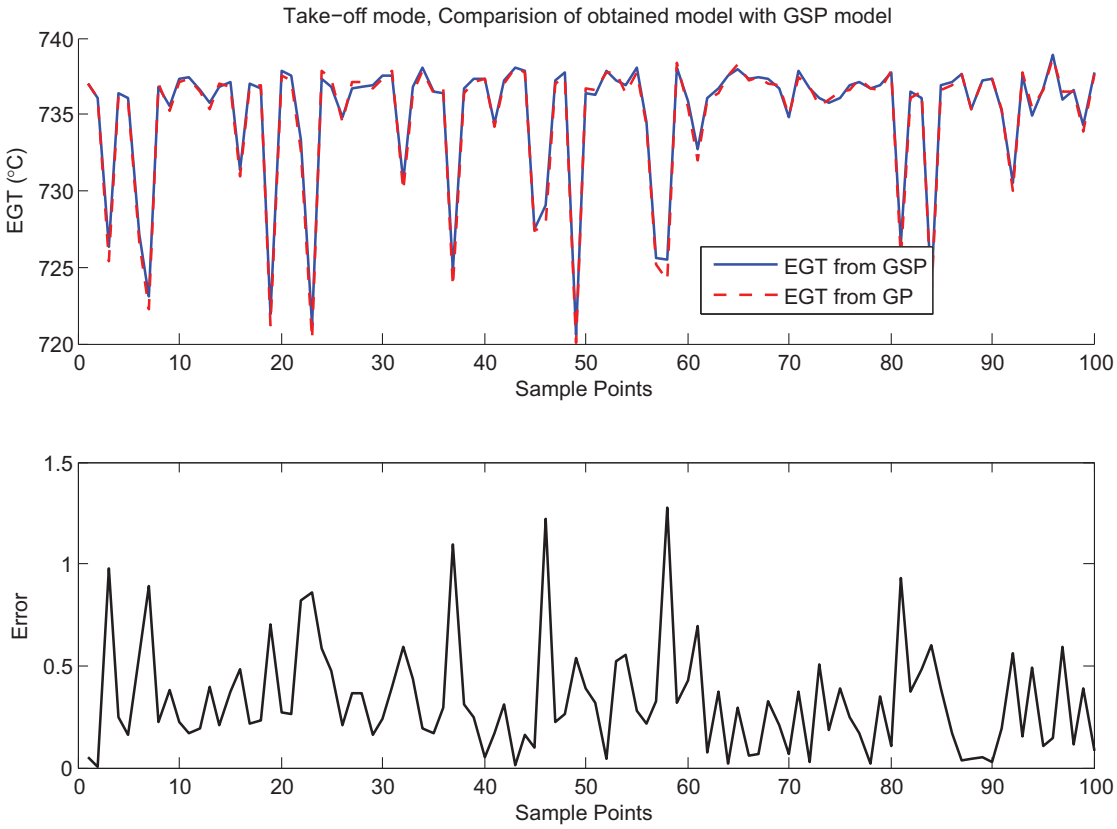

Figure 3.3: Model EGT output vs GSP software EGT output in the take-off mode, equation (3.2.1).

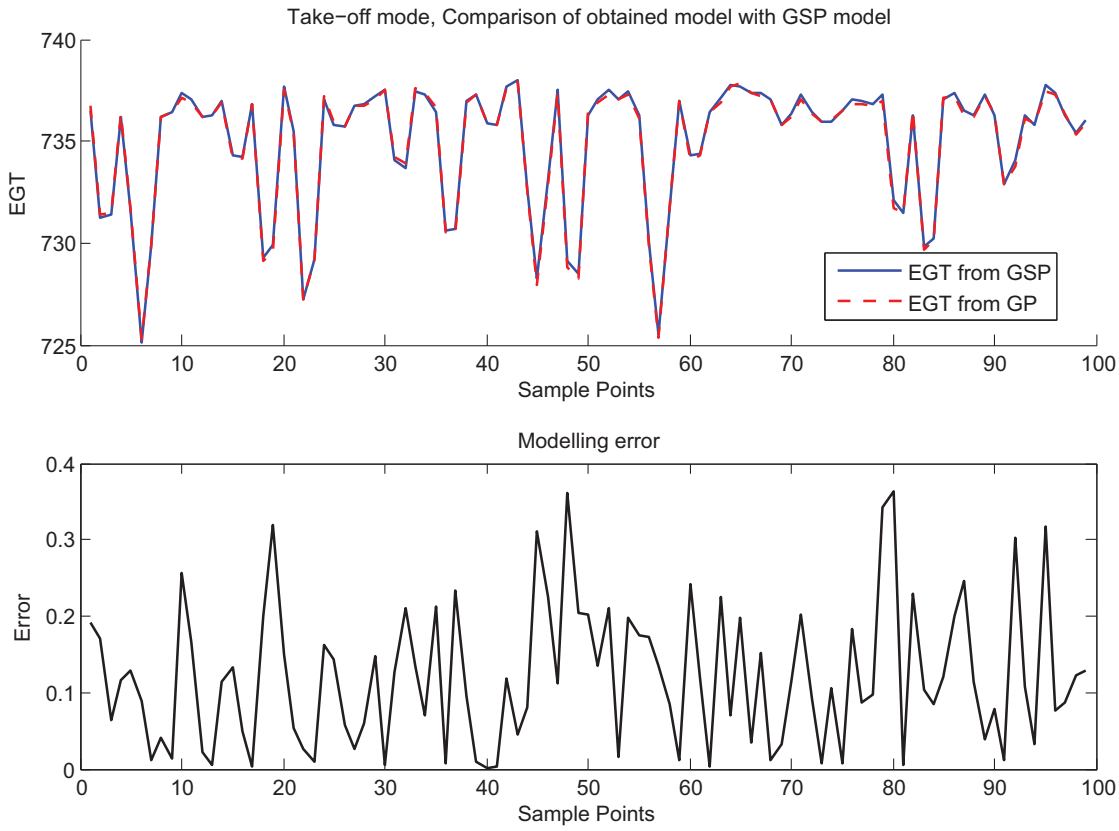

Figure 3.4: Model EGT output vs GSP software EGT output in the take-off mode, equation (3.2.2).

The main advantage of the first approach over the second one is that the resulting model estimates the healthy engine EGT at all time since it only depends on the engine operating conditions and not on the internal states of the engine. In addition, this model can even be used to predict the EGT in future flights by knowing the aircraft flight schedule and operating conditions. This property is useful when one is interested in prognosis and estimating the remaining useful life of the engine. However, this model's precision is less than the second model. Also, it may not be possible to find such a model in case of complicated engine model or real experimental data. On the other hand, the second model uses internal states of the engine to estimate the engine EGT. It implies that when the engine is faulty the data provided to the model is faulty and although the model depict the deviation of the engine behaviour but the estimated EGT does not necessarily represent the EGT from the healthy engine at the same operating condition.

In obtaining these models and their parameters, the required data snapshots of the engine parameters were obtained by using the aforementioned dual spool engine model in the GSP software for a range of different operating conditions and throttle settings. We have tried to select these operating conditions and throttle settings close to the flight profile of an actual aircraft during a period of few months. To account the changes in airports elevations the aircraft take-off altitude is considered to vary between 0 to 1000 m.

The models shown in equations (3.2.1) and (3.2.2) were statistically validated by testing them for 15 different sets of data. Table 3.3 summarizes the range of the operational conditions used to generated these data sets. Each data set consists of 100 flights samples. Figure 3.5 shows the distribution of W_f, altitude and $Mach$ number parameters within these 15 data sets.

Parameter	min	max
$W_f(kg/s)$	1.6	2
$Altitude(m)$	0	1000
$Mach$	0.42	0.56
$Tam(C)$	23	28

Table 3.3: Engine operational conditions ranges.

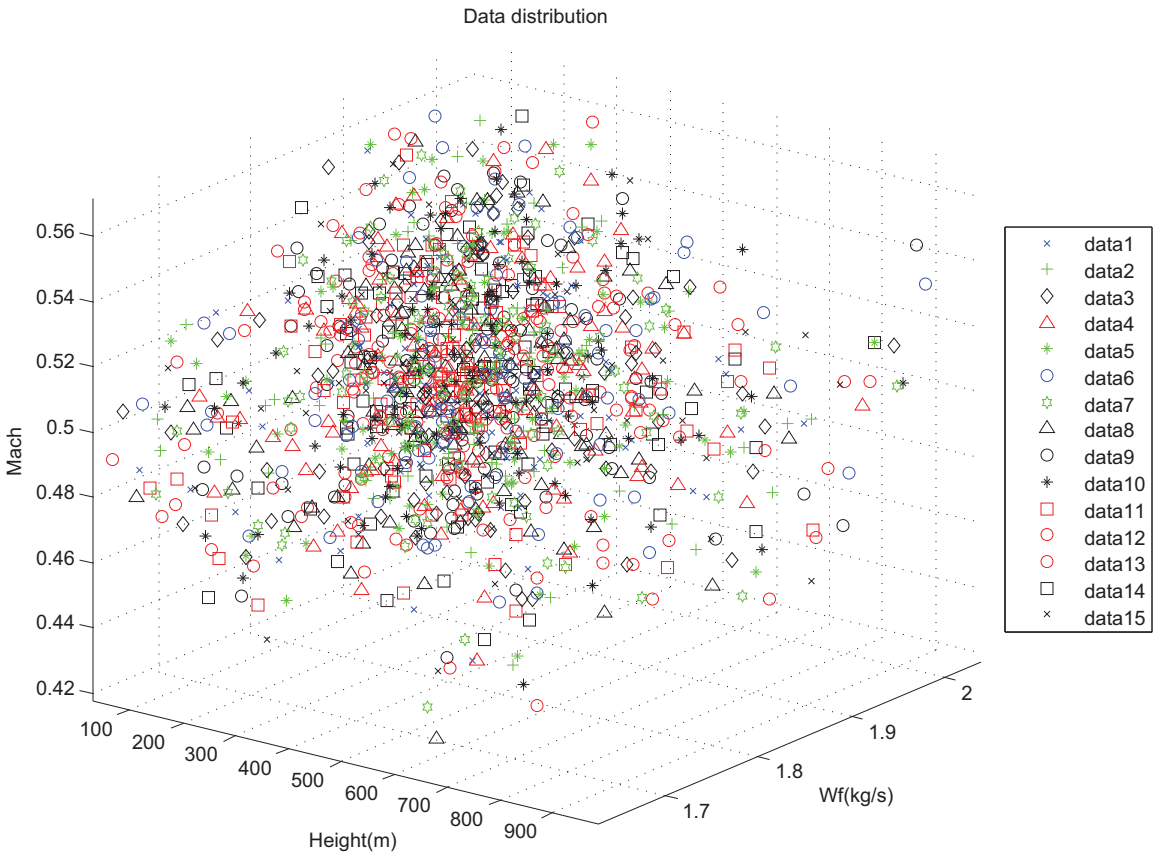

Figure 3.5: Distribution of the W_f, altitude and $Mach$ number.

Figures 3.6 to 3.9 show the model output for typical responses of the models to two different sets of data.

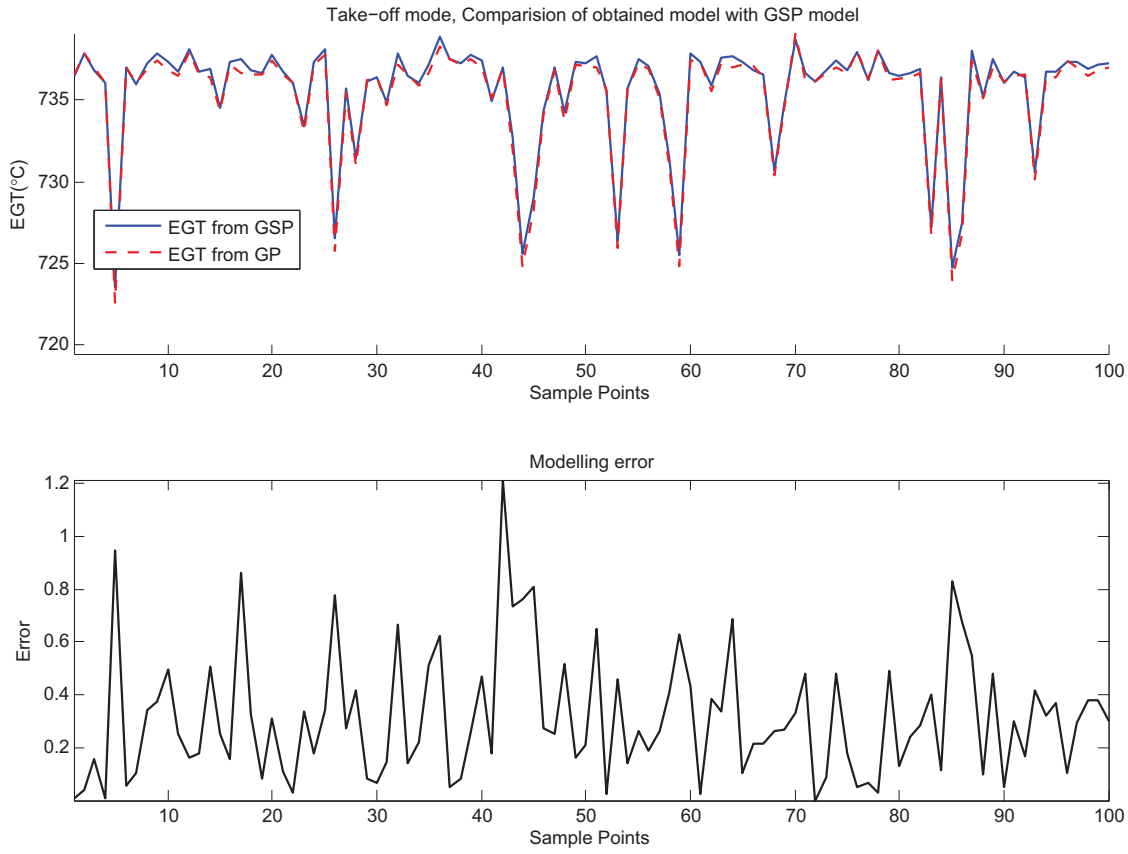

Figure 3.6: Model EGT output vs GSP software EGT output in the take-off mode, equation (3.2.1).

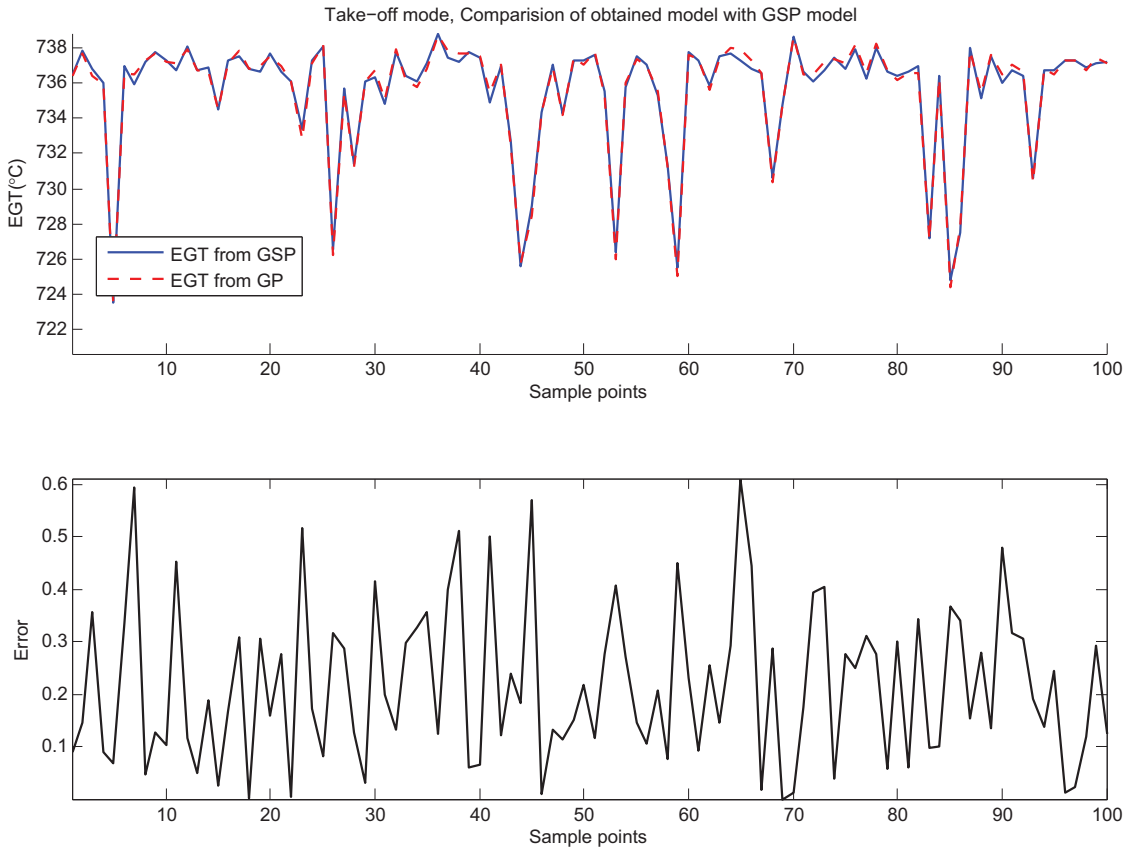

Figure 3.7: Model EGT output vs GSP software EGT output in the take-off mode, equation (3.2.2).

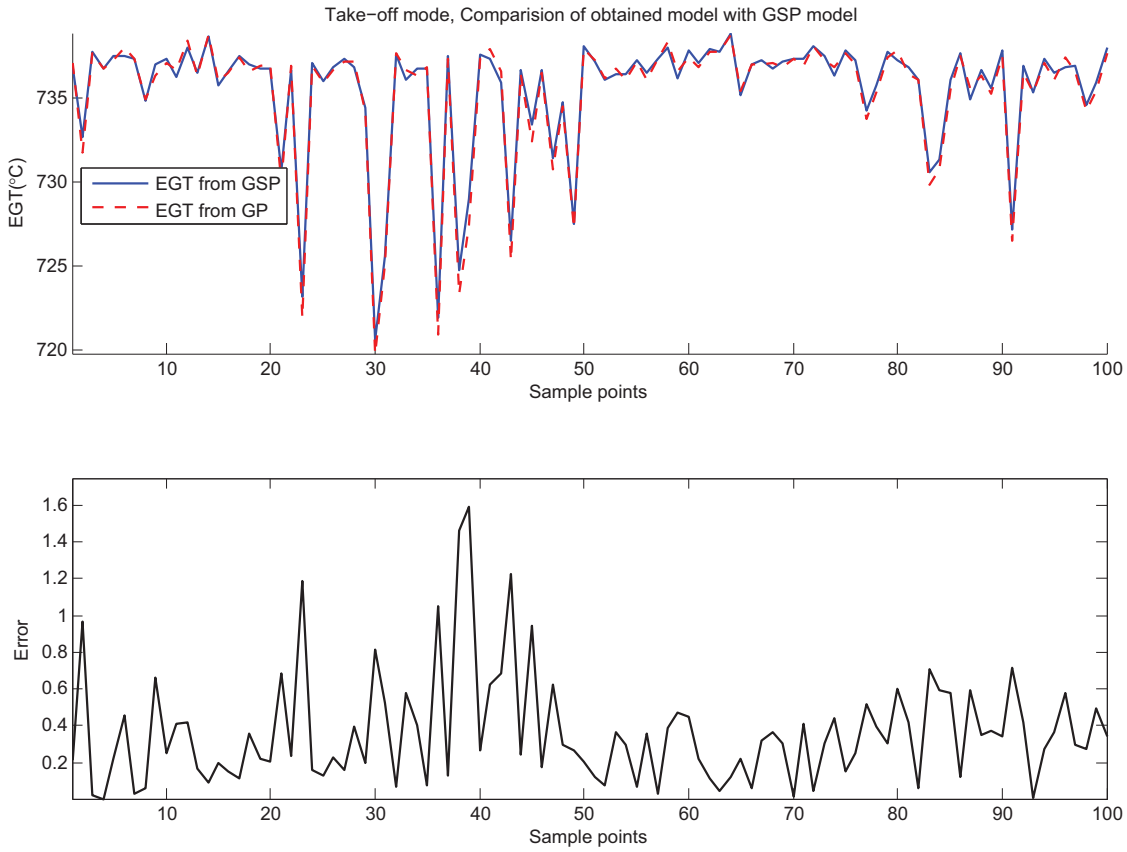

Figure 3.8: Model EGT output vs GSP software EGT output in the take-off mode, equation (3.2.1).

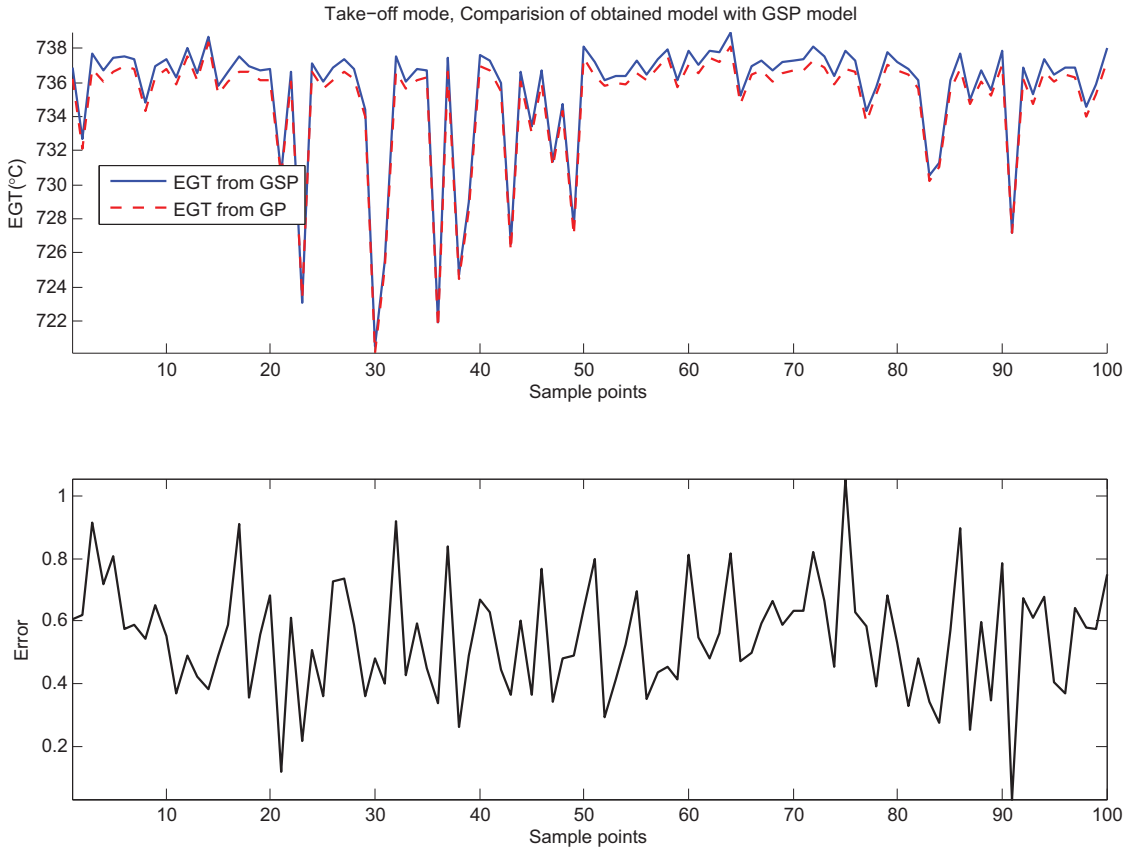

Figure 3.9: Model EGT output vs GSP software EGT output in the take-off mode, equation (3.2.2).

3.2.1.3 Modeling Error

To obtain the overall modelling errors statistically the parameters of the model structures obtained from the GP algorithm (equations (3.2.1) and (3.2.2)) were optimized for 15 different sets of data as mentioned in the previous section. Each data set consists of 100 flights samples in which 20 sample points were used to optimize the parameters and the rest were used as testing data. The mean square error between the two GP and GSP models for each data set and the maximum error for the simulations were selected as thresholds to be applied in the subsequent fault detection simulations.

The mean error and maximum error in each simulation is shown in Tables 3.4 and

Data set	MSE	Max Error
1	0.51	1.47
2	0.44	0.72
3	0.39	1.29
4	0.56	1.01
5	0.64	1.61
6	0.81	1.3
7	0.33	1.31
8	0.47	0.54
9	0.64	1.84
10	0.38	1.55
11	0.59	1.35
12	0.73	1.63
13	0.37	0.63
14	0.61	1.1
15	0.55	0.92

Table 3.4: Mean square error and the maximum error for 15 test data set using equation (3.2.1).

3.5. The maximum error for all simulations is 1.9 C and 1 C (less than 0.2%) for the model in equation (3.2.1) and equation (3.2.2), respectively.

Data set	MSE	Max Error
1	0.21	0.51
2	0.19	0.54
3	0.28	0.98
4	0.22	0.83
5	0.35	0.89
6	0.23	0.46
7	0.36	0.55
8	0.29	0.54
9	0.42	0.84
10	0.31	0.55
11	0.43	0.61
12	0.46	0.57
13	0.29	0.63
14	0.38	0.42
15	0.47	0.96

Table 3.5: Mean square error and the maximum error for 15 test data set using equation (3.2.2).

It can be seen that the maximum error that is obtained by using both the external and internal states is less than the model obtained by only using the external states of the engine.

3.2.1.4 Impact of the Number of Training Points

To investigate the effects of the number of the training data points on the accuracy of the obtained model, the parameters of the model in equations (3.2.2) and (3.2.1) are optimized by using different number of data points. Figure 3.10 shows the absolute mean error of the model output EGT and the EGT from the GSP software for different number of data points that are used to optimize the numerical parameters of the models. As shown, the error reduces with increasing the number of data points until it reaches a level which does not change significantly. Since the optimizer algorithm does not converge to exactly the same parameters because of different initialization there is a fluctuation in the error at this level. One can note that even with small

number of points and as few as 3 points one could have obtained the model parameters. This is one of the advantages of using this approach as compared to neural networks which usually needs a lot more data points for training.

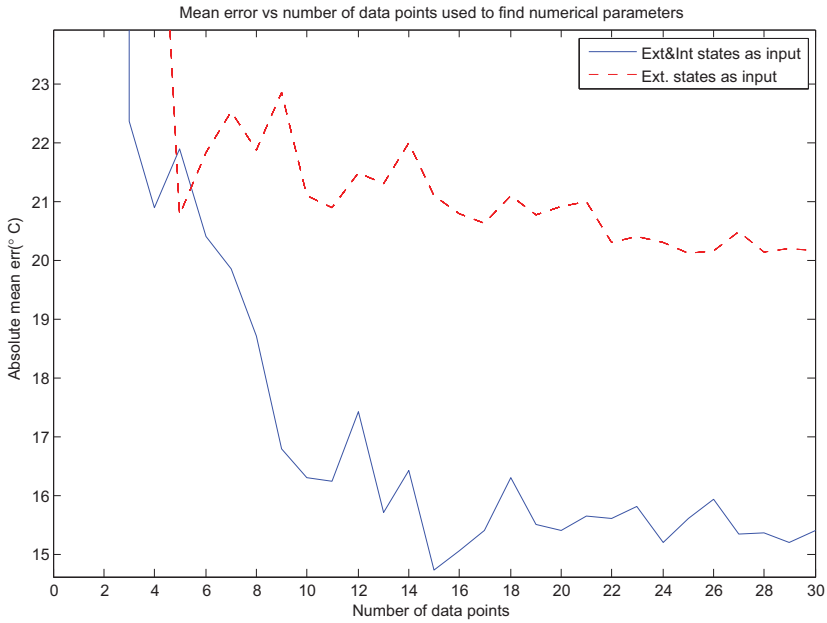

Figure 3.10: Absolute mean error vs the number of data used to obtain the model coefficients.

3.2.1.5 Fault Detection Process

As mentioned earlier for the purpose of fault detection one can build the required residuals from the difference between the EGT output of the developed models and the measurement data from the engine flight or in our case the GSP software. Because of the existence of modelling errors and noise, the residual is not zero even when there is no fault in the engine and the residual fluctuates around its zero mean. To account for this type of uncertainty a threshold has to be defined. In this thesis, the maximum modelling error that is obtained in Section 3.2.1.3 is considered as the residual threshold. If the residual passes this threshold a fault detection alarm is fired. Figures 3.11 and 3.12 show the error between the estimated EGT using the two

aforementioned models in equations (3.2.1) and (3.2.2) and the EGT from the GSP software for different sets of data.

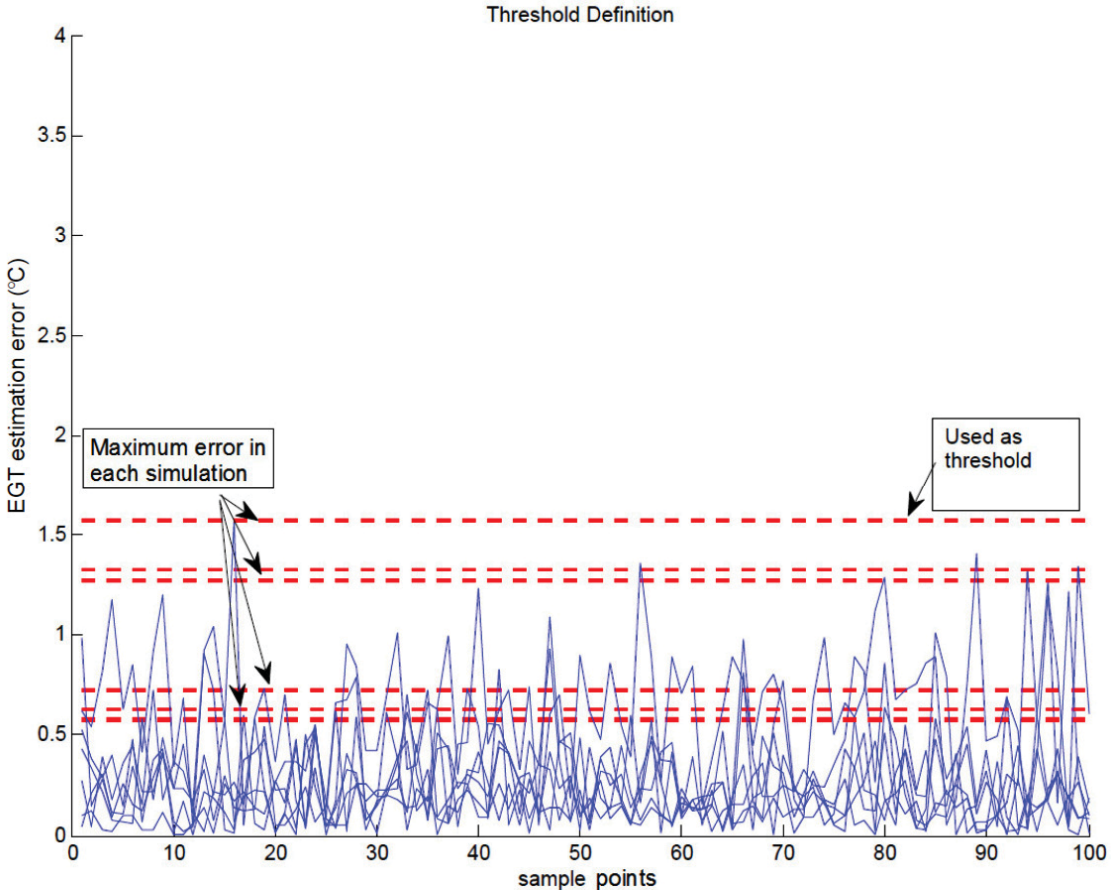

Figure 3.11: Threshold definition in the take-off mode by using equation (3.2.1).

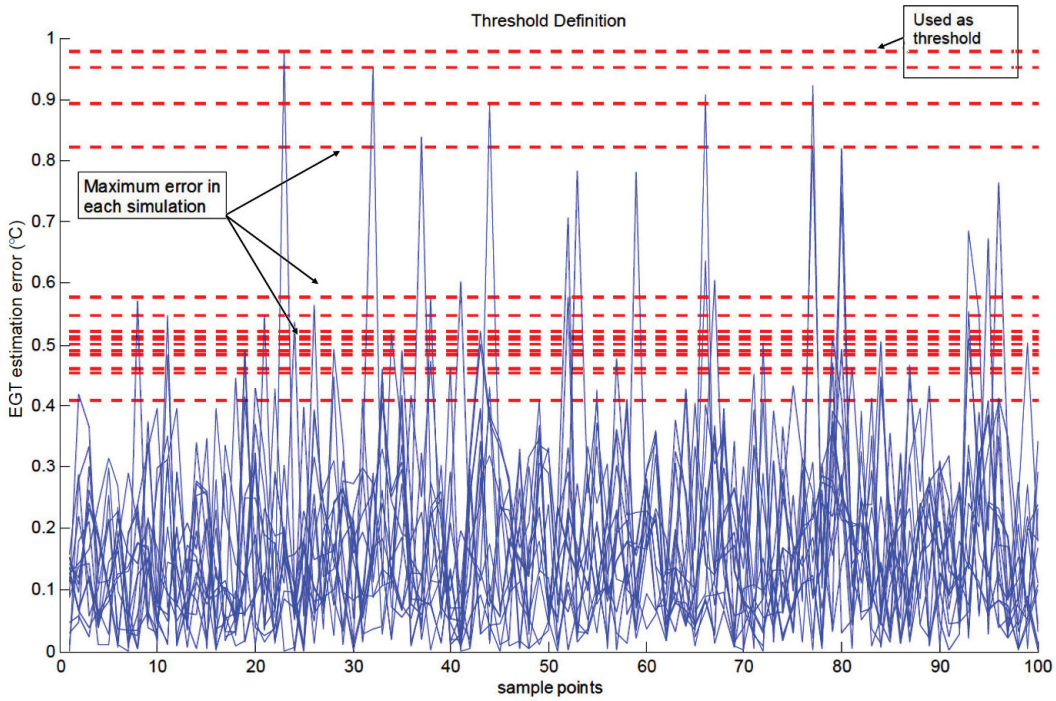

Figure 3.12: Threshold definition in the take-off mode by using equation (3.2.2).

A typical response of the engine model to fault in the high pressure turbine efficiency HT_{eff} is shown in Figures 3.13 and 3.14. In these simulations a 1% fault in HT_{ef} is injected at the 50th sample point. It can be seen that the outputs of the GSP and the GP model deviate from each other after the fault occurrence resulting in increase in the residual (difference between two model output EGT). More results corresponding to different types of faults are presented in Section 3.2.1.6.

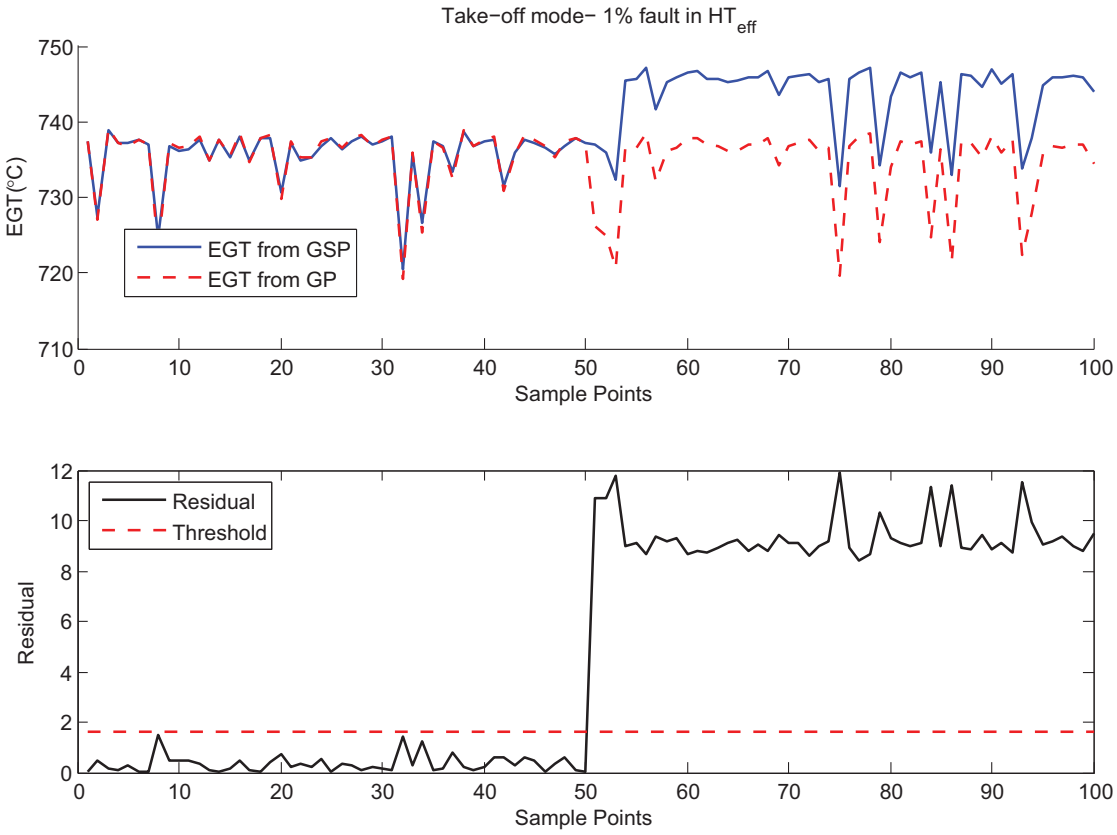

Figure 3.13: Model EGT output vs GSP software EGT output in the take-off mode with fault injected at the 50th sample point corresponding to equation (3.2.1).

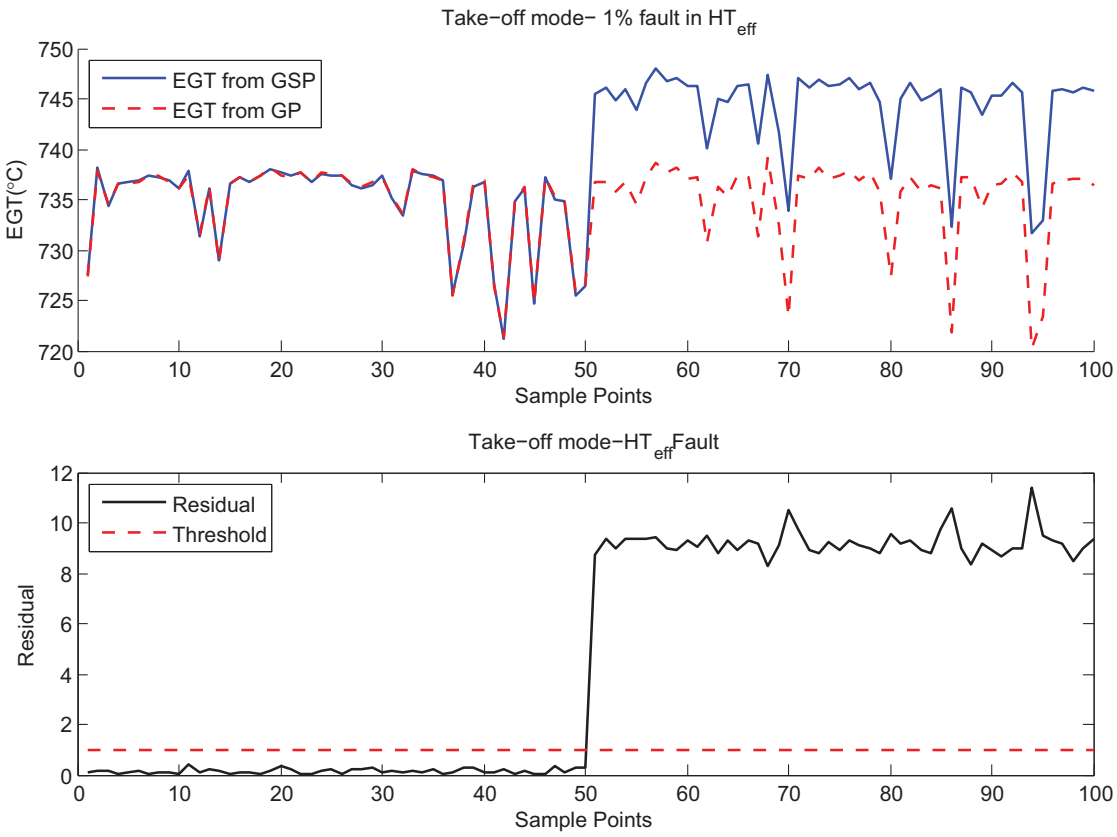

Figure 3.14: Model EGT output vs GSP software EGT output in the take-off mode with fault injected at the 50th sample point corresponding to equation (3.2.2).

3.2.1.6 Smallest Detectable Fault (Confusion Matrix)

Different faults in the engine affect the EGT differently. Some faults such as the HT_{effi} have severe effects on the engine operation while faults such as LC_{flow} have minor effects. Consequently, with the defined thresholds the level of detectability differs for different faults.

In this work confusion matrix is used to illustrate and evaluate the performance of the proposed detection algorithm. Confusion matrix in general is a table with four cells containing the number of true positive, true negative, false positive and false negative classifications. The definition of these expressions in our application are as follows:

- True positive (t.p.): The number of simulations classified as faulty and the engine is also faulty.
- False positive (f.p.): The number of simulations classified as faulty while the engine is healthy.
- True negative (t.n.): The number of simulations classified as healthy and the engine is also healthy.
- False negative (f.n.): The number of simulations classified as healthy while the engine is faulty.

True Positive	False Negative
False Positive	True Negative

Table 3.6: The confusion matrix.

To evaluate the performance of the fault residual in correctly detecting faults a confusion matrix is constructed based on a series of simulations and using different sets of data and for different operating conditions and fault severities. Using these

confusion matrices one can construct the accuracy,precision,true positive rate,true negative rate, false positive rate and false negative rate indicators that are defined as follows:

$$Accuracy = \frac{true\,positive + true\,negative}{true\,positive + true\,negative + false\,positive + false\,negative} \tag{3.2.3}$$

$$Precision = \frac{true\,negative}{true\,negative + false\,negative} \tag{3.2.4}$$

$$TruePositiveRate(TPR) = \frac{true\,positive}{true\,positive + false\,negative} \tag{3.2.5}$$

$$FalsePositiveRate(FPR) = \frac{false\,positive}{false\,positive + true\,negative} \tag{3.2.6}$$

$$TrueNegativeRate(TNR) = \frac{true\,negative}{true\,negative + false\,positive} \tag{3.2.7}$$

$$FalseNegativeRate(FNR) = \frac{false\,negative}{true\,positive + false\,negative} \tag{3.2.8}$$

The accuracy parameter gives a measure of the fault residual performance. For each fault the minimum detectable fault severity can be determined by considering an accuracy level and finding the smallest fault severity such that in the resultant confusion matrix the accuracy indicator is above the considered accuracy level. The minimum detectable faults and the corresponding confusion matrices for the two models in equations (3.2.1) and (3.2.2) are shown in Tables 3.7 and 3.8 for the 70% accuracy level. Other values for accuracy can also be used. The minimum detectable

fault increases by considering higher levels for the accuracy indicator. To obtain the confusion matrix for each fault severity 15 data sets with 100 snapshots are used.

Fault type	Minimum detectable fault (%)	Confusion matrix	Accuracy (%)	Precision (%)	TPR (%)	FPR (%)	TNR (%)	FNR (%)
$\mathbf{LT_{efi}}$	1%	9 1 2 3	80	75	90	40	60	10
$\mathbf{HT_{efi}}$	0.1%	8 2 1 4	80	66	80	20	80	20
$\mathbf{LC_{efi}}$	1%	7 3 1 4	73.3	57	70	20	80	30
$\mathbf{HC_{efi}}$	0.1%	6 4 0 5	73.3	55	60	0	100	40
$\mathbf{LT_{flow}}$	0.5%	8 2 2 3	73.3	60	80	40	60	20
$\mathbf{LC_{flow}}$	1%	8 2 1 4	80	66	80	20	80	20
$\mathbf{HT_{flow}}$	0.5%	8 2 1 4	80	66	80	20	80	20
$\mathbf{HC_{flow}}$	3%	6 4 0 5	73.3	55	60	0	100	40

Table 3.7: Minimum detectable faults and the confusion matrices in the take-off mode by using equation (3.2.1).

Fault type	Minimum detectable fault (%)	Confusion matrix	Accuracy (%)	Precision (%)	TPR (%)	FPR (%)	TNR (%)	FNR (%)
$\mathbf{LT_{efi}}$	1%	9 1 1 4	86.6	80	90	20	80	10
$\mathbf{HT_{efi}}$	0.1%	8 2 1 4	80	66	80	20	80	20
$\mathbf{LC_{efi}}$	0.8%	8 2 1 4	80	66	80	20	80	20
$\mathbf{HC_{efi}}$	0.1%	8 3 0 4	80	57	72	0	100	28
$\mathbf{LT_{flow}}$	0.4%	9 2 1 3	80	60	81	25	75	19
$\mathbf{LC_{flow}}$	1%	8 2 1 4	80	66	80	20	80	20
$\mathbf{HT_{flow}}$	0.4%	8 2 1 4	80	66	80	20	80	20
$\mathbf{HC_{flow}}$	2%	7 2 2 4	73.3	66	77	33	66	22

Table 3.8: Minimum detectable faults and the confusion matrices in the take-off mode by using equation (3.2.2).

For each type of fault, Figures 3.15 to 3.22 show a typical behaviour of the residual to different types of faults that are used to determine the minimum detectable faults in Table 3.8 and the model in equation (3.2.2). One can observe that before a fault occurs the model has a good agreement with the measured data from the GSP software but after the fault occurrence at the 50th data point the residuals start increasing until they pass the defined thresholds. Similar results were obtained for the model in equation (3.2.1).

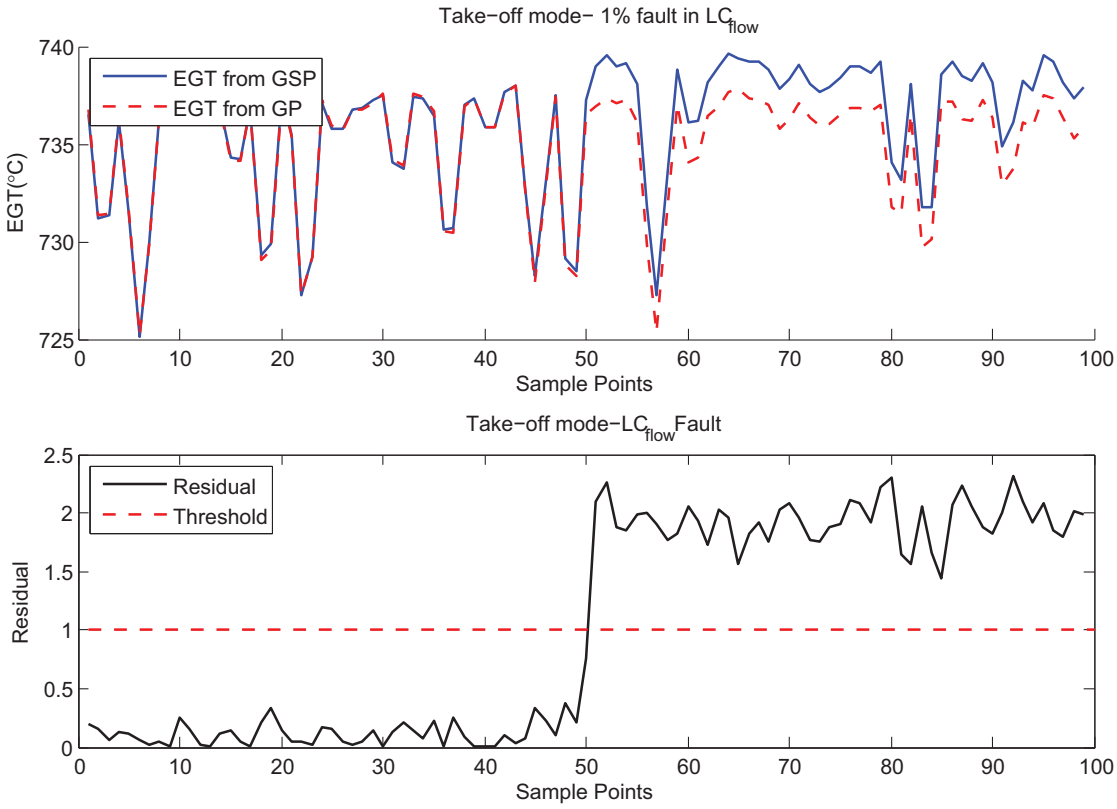

Figure 3.15: Fault in the LC_{flow}.

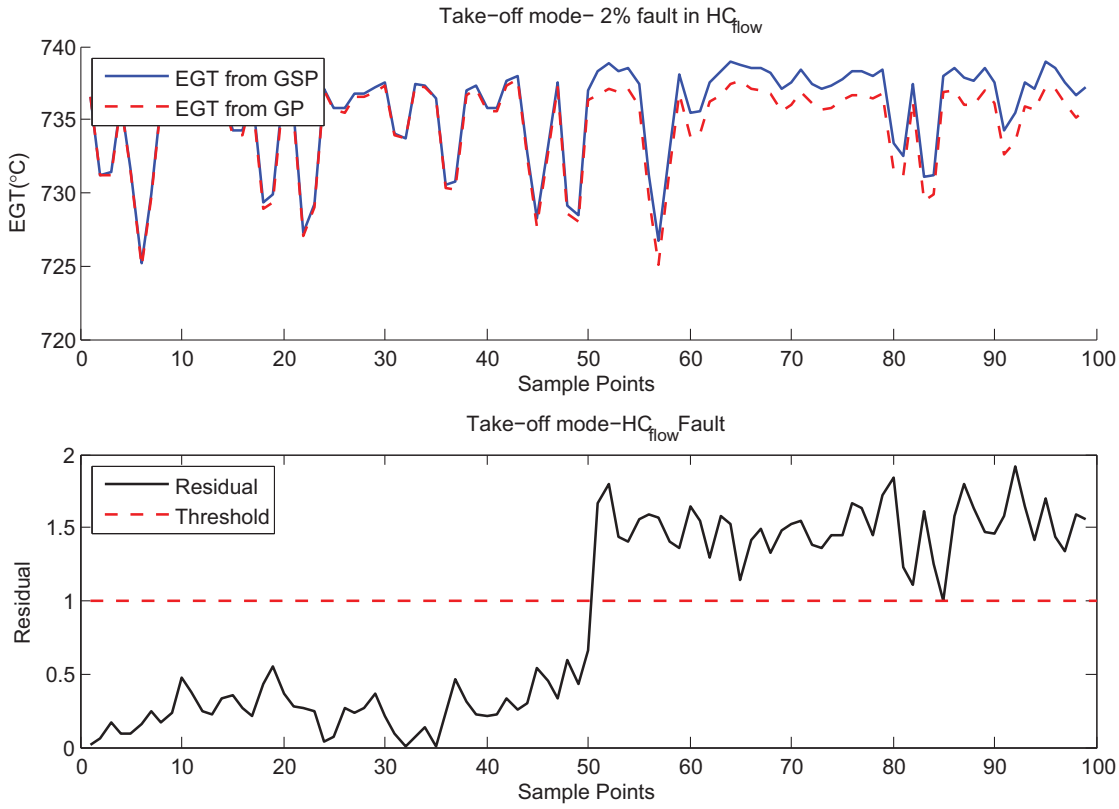

Figure 3.16: Fault in the HC_{flow}.

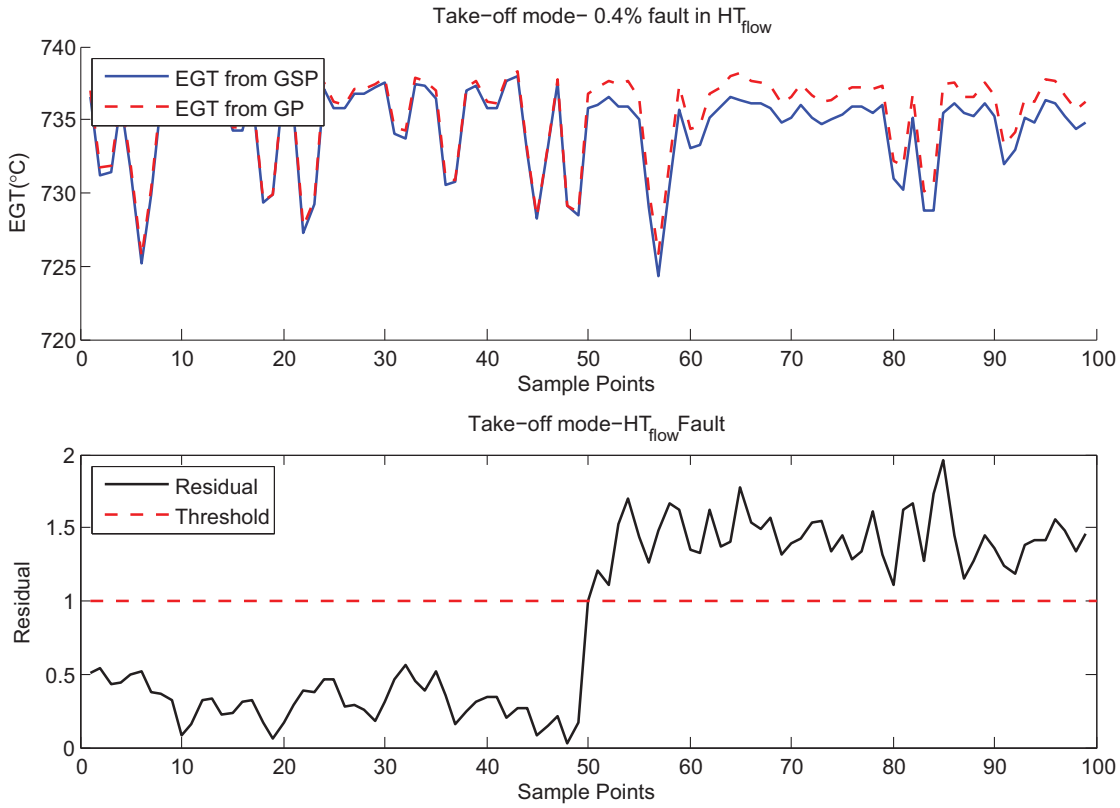

Figure 3.17: Fault in the HT_{flow}.

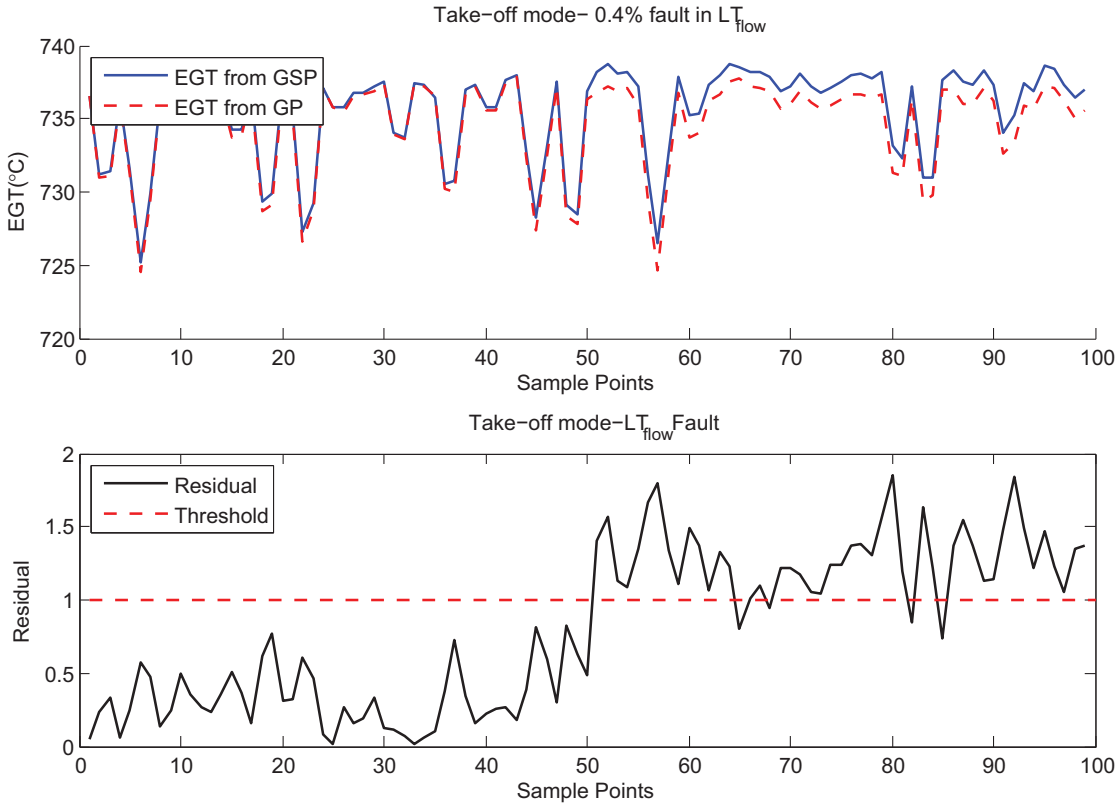

Figure 3.18: Fault in the LT_{flow}.

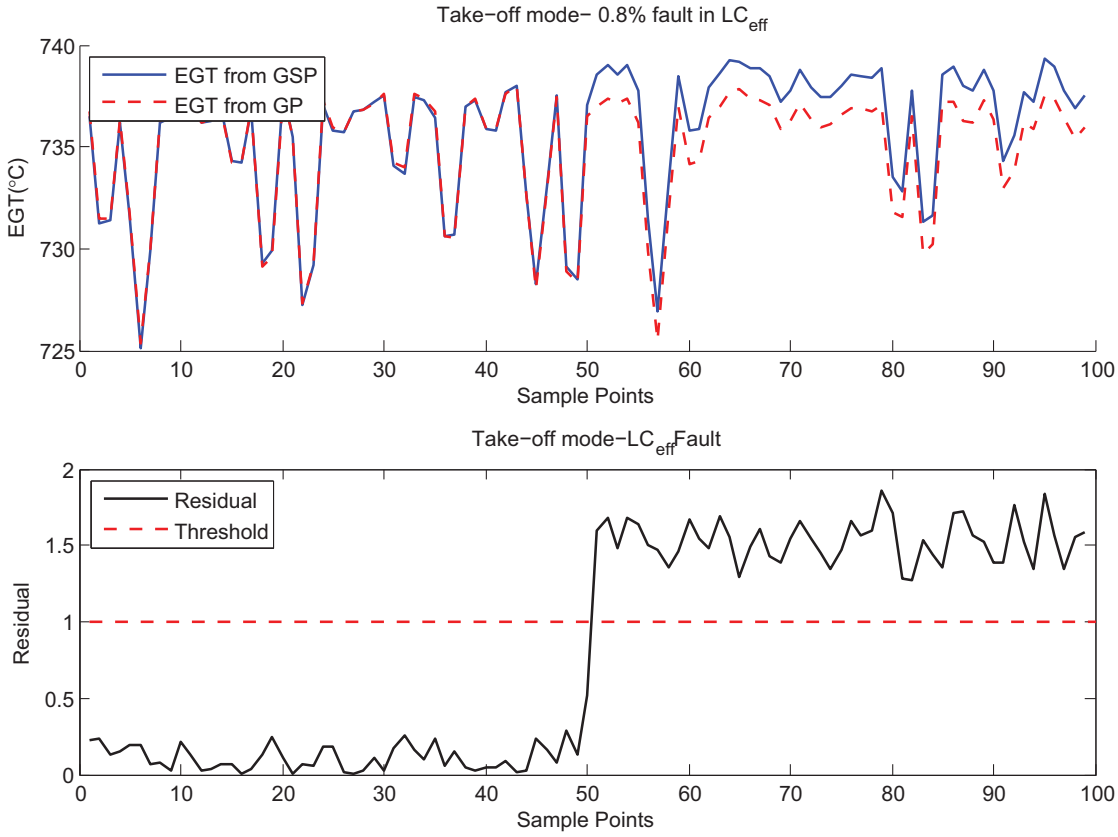

Figure 3.19: Fault in the LC_{eff}.

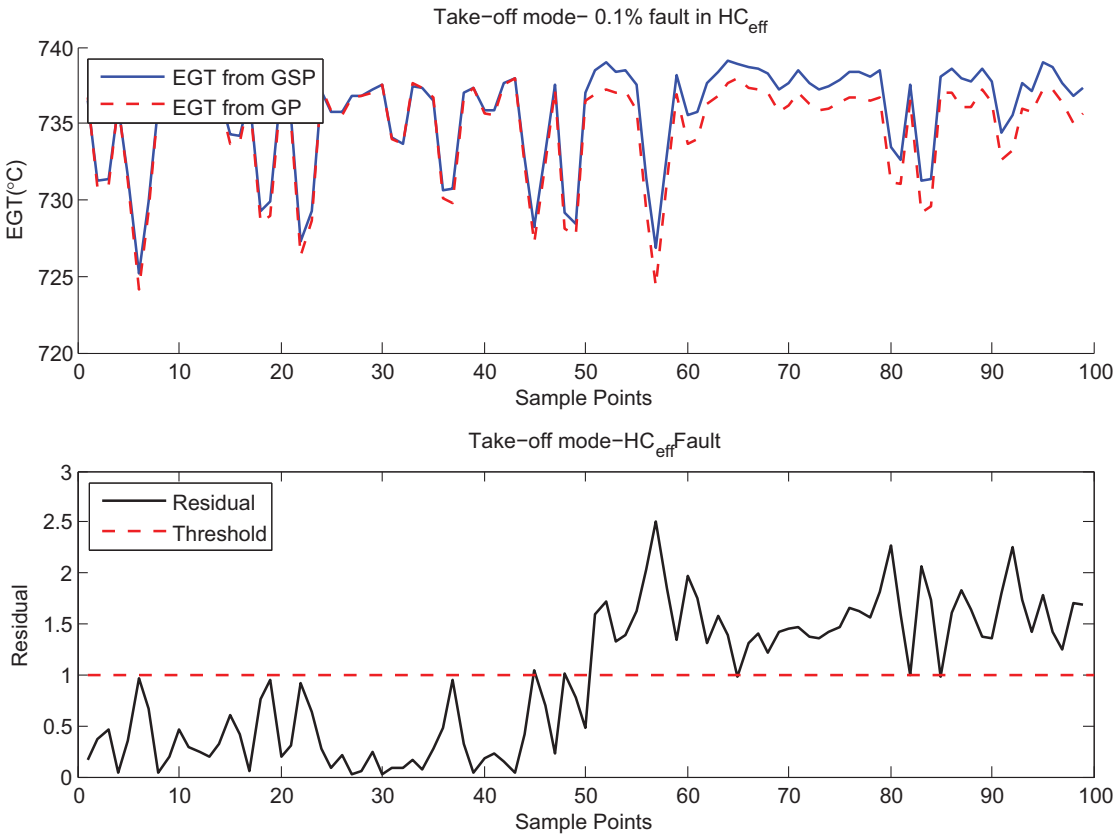

Figure 3.20: Fault in the HC_{eff}.

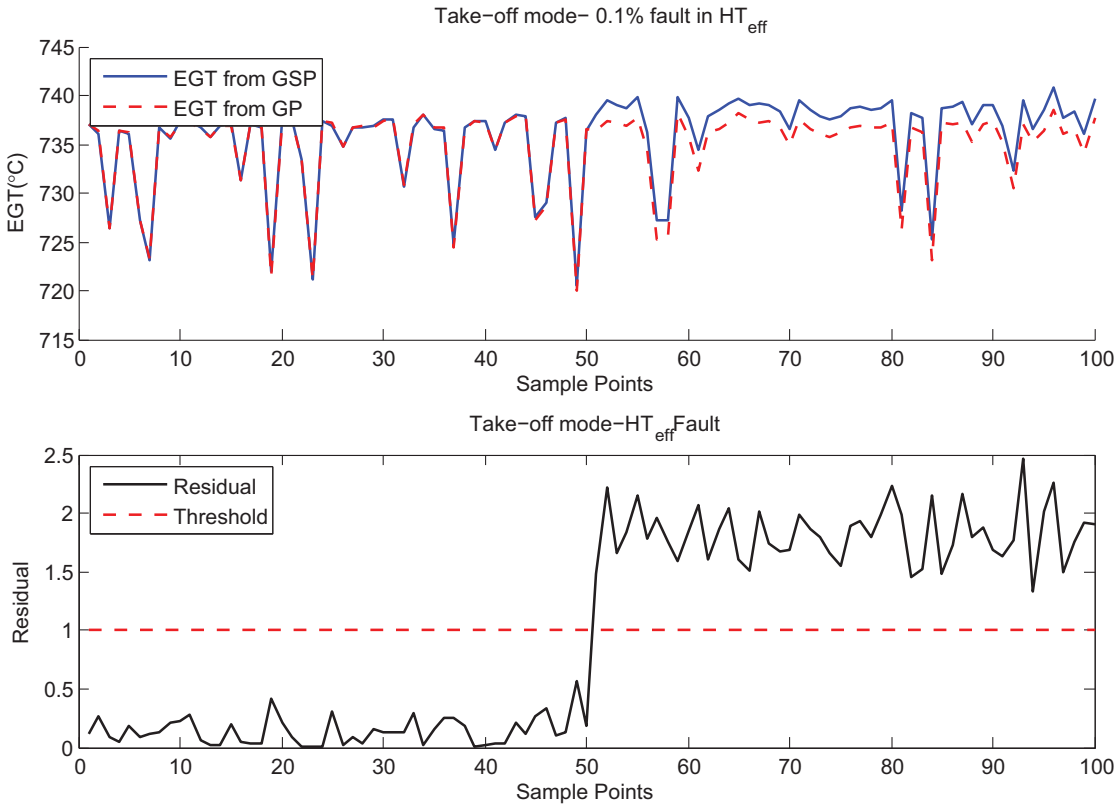

Figure 3.21: Fault in the HT_{eff}.

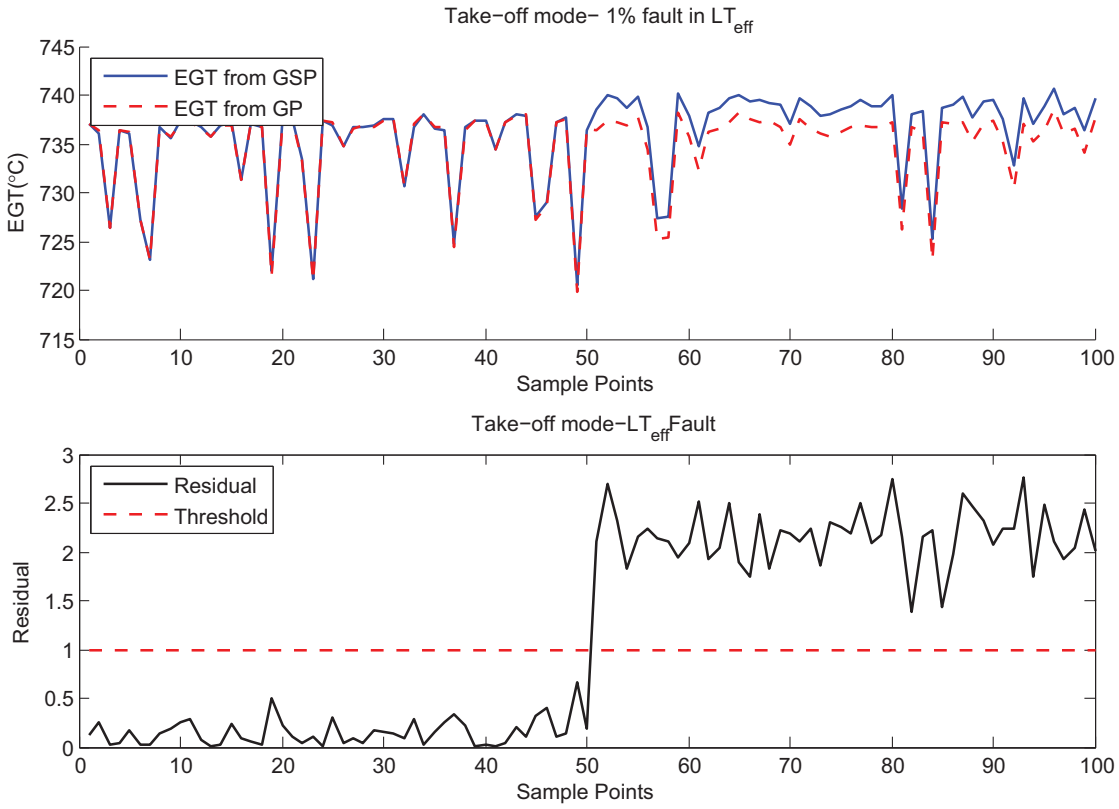

Figure 3.22: Fault in the LT_{eff}.

3.2.2 Cruise Mode

In the previous section, engine EGT was modelled in the take-off mode. As mentioned earlier the engine fault and deviations are more visible in the take-off mode in which the engine is working at its maximum load [43]. In this section, a similar model is obtained but for the cruise mode in which the engine works most of its life. The same mathematical engine model is used to generate a set of data points corresponding to the cruise mode. In the cruise mode the aircraft usually flies at a constant altitude and Mach number.

3.2.2.1 Model Generation and Validation

To make the simulation more realistic flights are supposed to occur between a wide range of altitudes from 3000 m to 14000 m. Using 20 points of the data snapshots the GP algorithm was ran to find the best matching structure. Table 3.9 summarizes the range of the operational conditions used to generated these data sets. Each data set consists of 100 flights samples. Figure 3.23 shows the distribution of W_f, altitude and $Mach$ number parameters corresponding to these 15 data sets.

Parameter	min	max
$W_f(kg/s)$	0.9	1.55
$Altitude(m)$	3000	14000
$Mach$	0.55	0.95
$Tam(C)$	-44	34

Table 3.9: Engine operational conditions ranges.

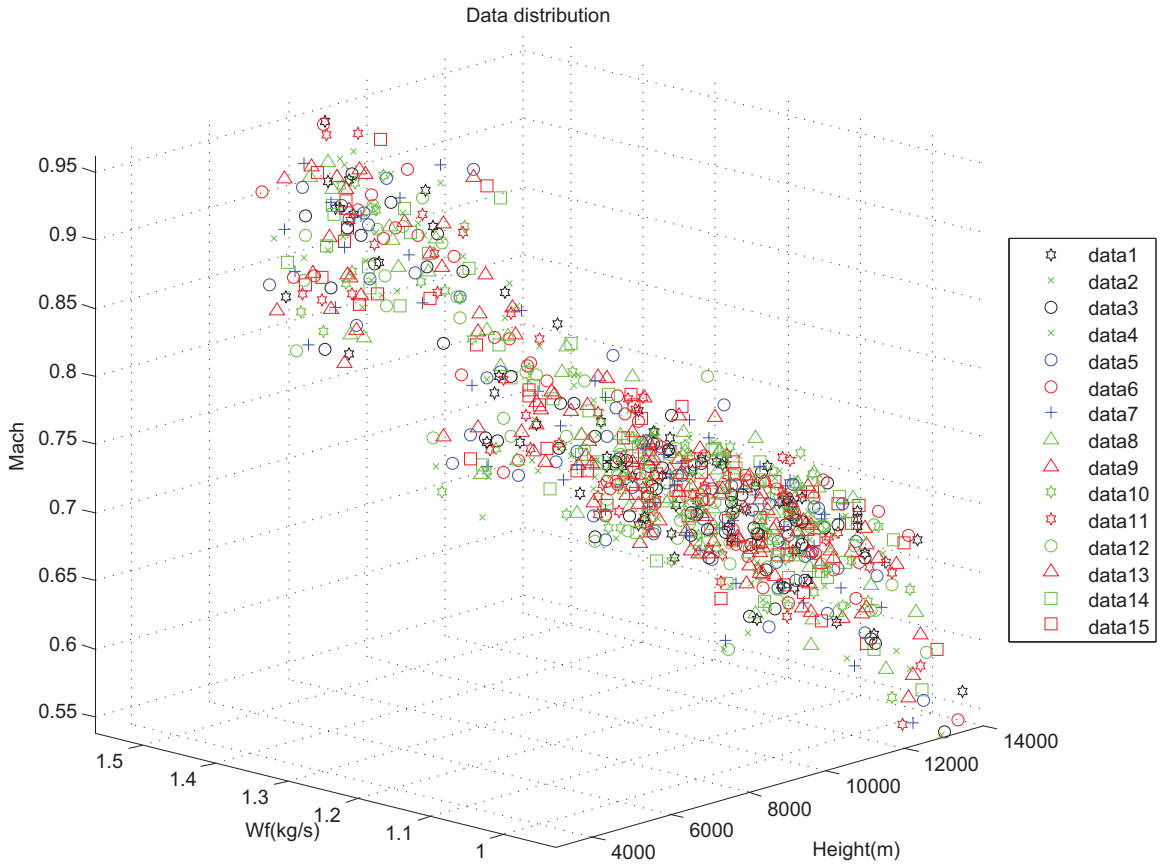

Figure 3.23: Distribution of the W_f, altitude and $Mach$ number.

Similar to the take-off mode we try to obtain two models. One by just using the external states and operational conditions, namely the altitude H, the ambient temperature T_a, the fuel flow W_f and the Mach number M as input to the model. The second model is obtained by using the engine internal states N_1 and N_2 and pressure after high pressure compressor P_3 as input in addition to previous parameters. We expect that these additional states improve the accuracy of the model. The best obtained models are shown in equations (3.2.9) and (3.2.10), respectively.

$$EGT = \frac{aW_f + B}{cH^2 - dHT_{am}^2 + E} \tag{3.2.9}$$

a	-92.099
B	-360.187
c	264.939
d	213.52
E	-89.315

$$EGT = \frac{aM^3 - bH + C}{dN_1^{3N_2}} \tag{3.2.10}$$

a	-92.14
b	-17.778
C	-203.87
d	-164.707

In obtaining these models the same settings as the take-off mode are applied (Table 3.2). Similar to the take-off mode the obtained model performance is represented by comparing its EGT output with the GSP model output. Figures 3.24 and 3.25 show the two model outputs. The maximum error between the two models and the GSP are less than 1.8% and 1.2%, respectively. It can be seen that in some points the error is higher. This is partly the result of the engine parameters that may not been considered in our modelling as well as due to modelling errors. These jumps in the error can be reduced by looking at the flights in a smaller range of operating conditions.

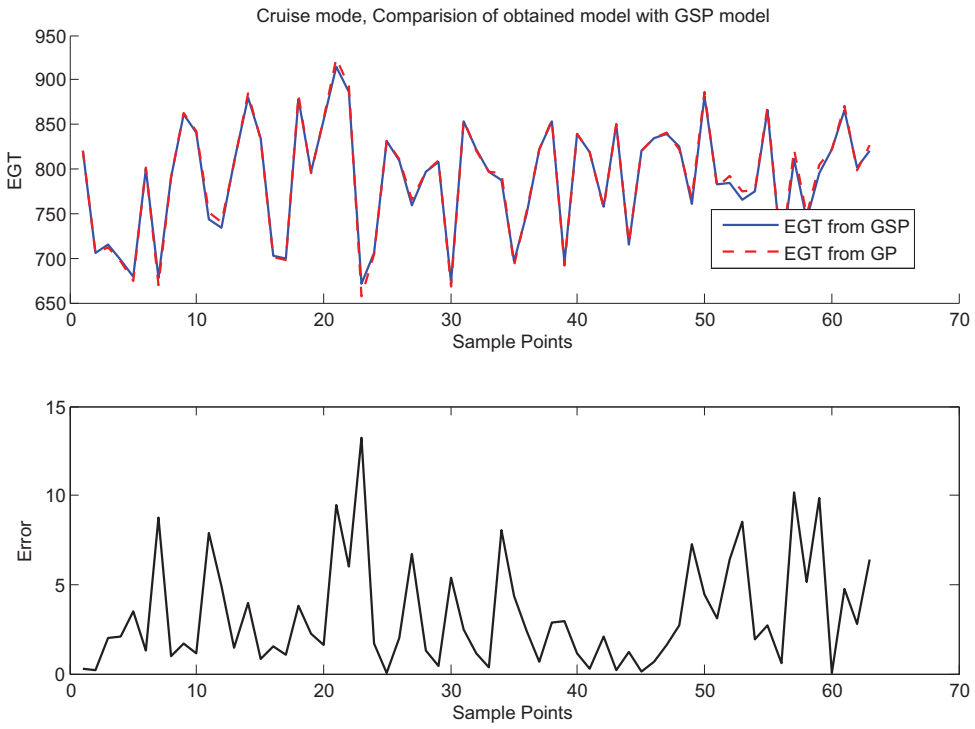

Figure 3.24: Model EGT output vs the GSP software EGT output in the Cruise mode corresponding to equation (3.2.9).

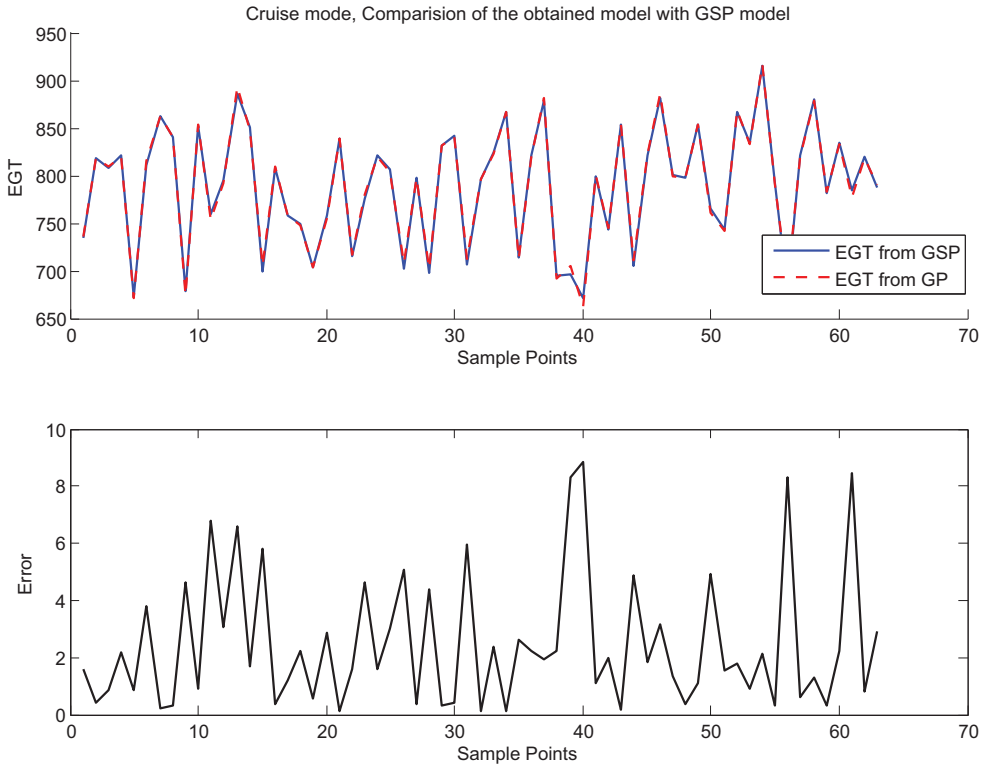

Figure 3.25: Model EGT output vs the GSP software EGT output in the Cruise mode corresponding to equation (3.2.10).

3.2.2.2 Modelling Error

We used the same strategy as in the take-off to determine the modelling error for the cruise mode. The parameters of the model structures that are obtained in the previous section were optimized for 15 different sets of data. Each data set consists of 65 flight samples in which 20 sample points were used to optimize the parameters and the rest were used as the testing data. The mean square between the GP and GSP models calculated for each data set and at the end the maximum error in the simulations were selected as thresholds to be applied in the subsequent fault detection simulations. The mean error and maximum error in each simulation is shown in Tables 3.10 and 3.11. The maximum error in all simulations is 16 C and 11 C (less than 2%) for the model of equation (3.2.9) and equation (3.2.10), respectively.

It can be seen that the maximum error in the model that is obtained by using both the external and internal states is less than the model obtained by using only the external states of the engine.

Data set	MSE	Max Error
1	2.65	13.99
2	4.20	15.16
3	2.89	11.16
4	2.15	9.39
5	3.8	15.35
6	3.11	10.97
7	2.835	9.81
8	3.40	14.25
9	3.56	10.16
10	3.25	12.94
11	2.99	10.65
12	2.59	11.39
13	4.02	11.43
14	3.35	11.44
15	4.87	15.28

Table 3.10: Mean square error and maximum error for 15 test data set by using equation (3.2.9).

Data set	MSE	Max Error
1	2.46	9.91
2	2.85	10.62
3	2.48	10.89
4	2.53	9.85
5	2.49	9.11
6	2.61	10.54
7	1.34	5.97
8	2.47	9.27
9	2.53	10.97
10	2.47	10.62
11	2.04	6.83
12	2.52	9.21
13	2.39	7.75
14	2.57	9.90
15	2.12	8.11

Table 3.11: Mean square error and maximum error for 15 test data set by using equation (3.2.10).

3.2.2.3 Fault Detection Process

The maximum modelling error that is obtained in Section 3.2.2.2 is considered as the residual threshold. Figures 3.26 and 3.27 show the modelling error for the two aforementioned models by using the internal and the external states, respectively. The dashed lines denote the maximum error in each simulation.

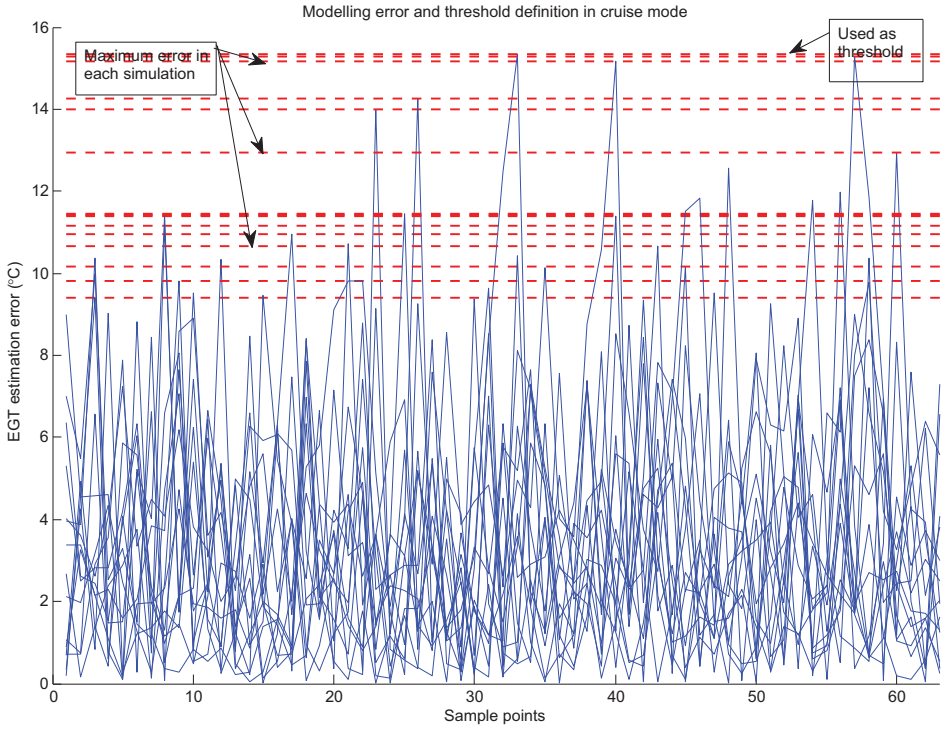

Figure 3.26: Threshold definition in the cruise mode by using the external states corresponding to equation (3.2.9).

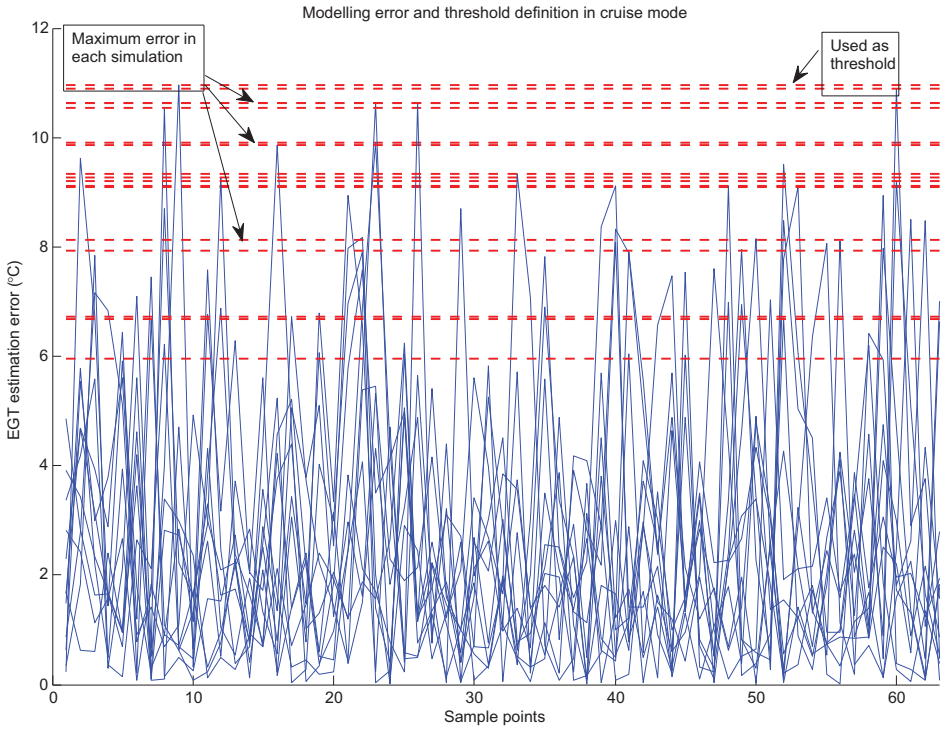

Figure 3.27: Threshold definition in cruise mode by using the internal and external states corresponding to equation (3.2.10).

If the residual passes this threshold a fault detection alarm is fired. Typical responses of the engine model to fault in the high pressure turbine efficiency HT_{eff} is shown in Figures 3.28 and 3.29. In these simulations a 3% fault in HT_{ef} is injected at the 35th sample point. It can be seen that the outputs of the GSP and the GP model deviate from each other after the fault occurrence resulting in increase in the residual (difference between two model output EGT). More results to different types of faults are presented in Section 3.2.2.4.

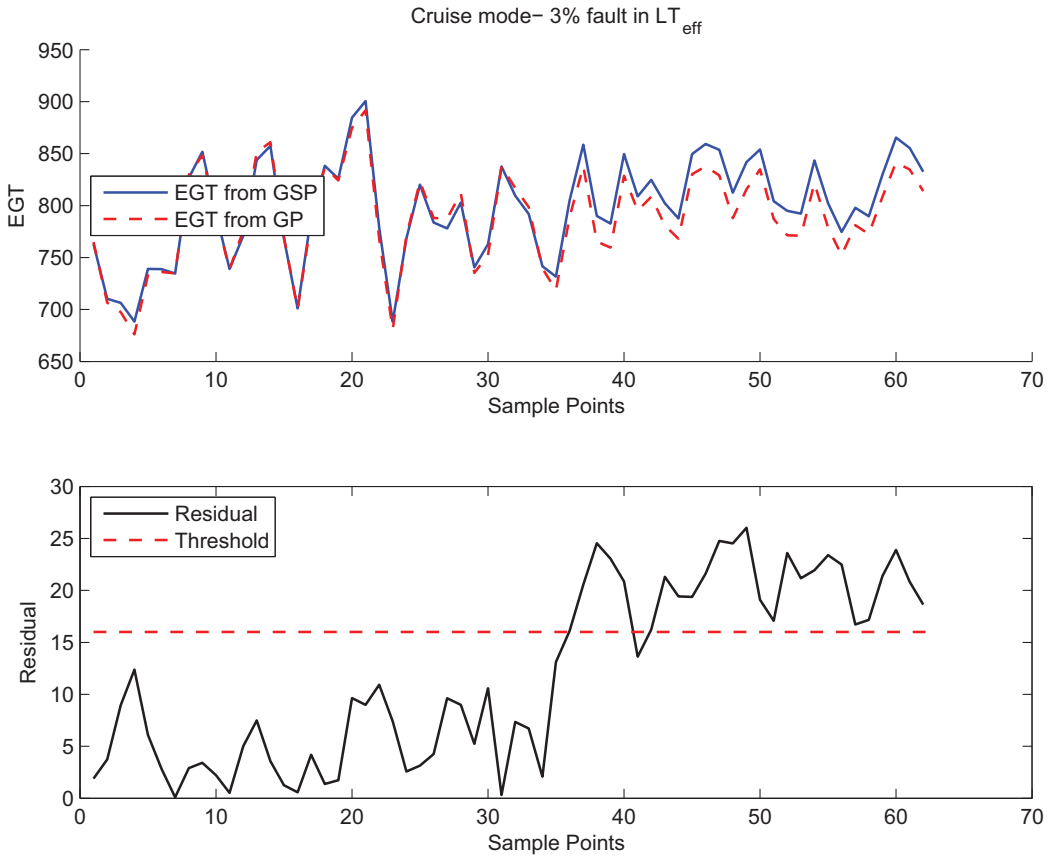

Figure 3.28: Model EGT output vs GSP software EGT output in the cruise mode with fault injected at the 50th sample point corresponding to equation (3.2.9).

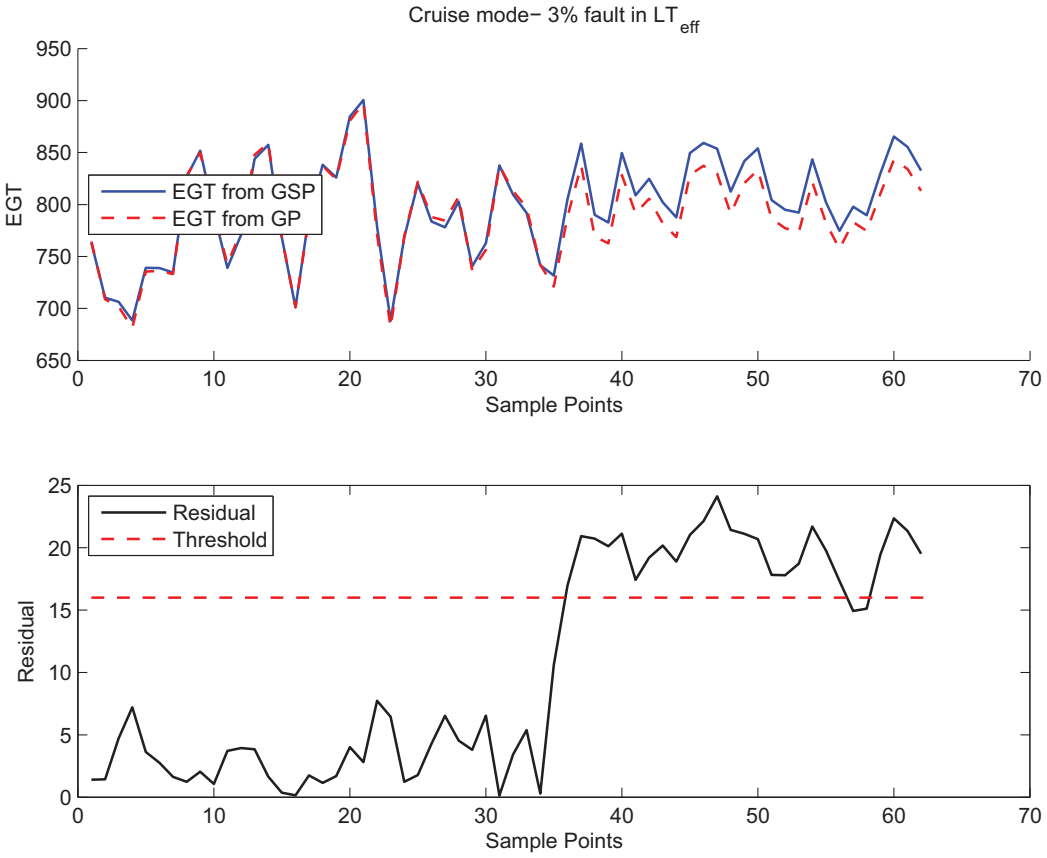

Figure 3.29: Model EGT output vs GSP software EGT output in the cruise mode with fault injected at the 50th sample point corresponding to equation (3.2.2).

3.2.2.4 Smallest Detectable Fault (Confusion Matrix)

Same as in the take-off mode the confusion matrix is used to derive the minimum detectable faults in the cruise mode. Confusion matrix is obtained statistically by simulating 15 different sets of data and different levels of fault between 1 to 9% in the corresponding components. Faults with severities more than 9% are considered too extreme and are not investigated here. Minimum detectable faults and the corresponding faults are shown in Tables 3.12 and 3.13. The results shown here are obtained by considering 70% level for the accuracy indicator and finding the minimum severity of the fault that the resultant accuracy from the simulations are more than 70%. Other levels of accuracy can also be used. The minimum detectable fault increases by considering higher levels for the accuracy indicator. Compatible with the previous arguments the minimum detectable faults in the cruise mode are significantly higher than the take-off mode. It can be seen that in most cases even with the high level of 9% fault in the engine the effects of the fault is not detectable in the cruise mode. Although the modelling error is small, nevertheless the obtained model is not precise enough to detect these changes.

Fault type	Minimum detectable fault (%)	Confusion matrix	Accuracy (%)	Precision (%)	TPR (%)	FPR (%)	TNR (%)	FNR (%)
$\mathbf{LT_{efi}}$	3%	7 3 1 4	73.33	57	70	20	80	30
$\mathbf{HT_{efi}}$	6%	9 1 2 3	80	75	90	40	60	10
$\mathbf{LC_{efi}}$	9%		–					
$\mathbf{HC_{efi}}$	6%	8 2 1 4	80	66	80	20	80	20
$\mathbf{LT_{flow}}$	9%		–					
$\mathbf{LC_{flow}}$	6%	7 4 0 4	73.3	50	63	0	100	36
$\mathbf{HT_{flow}}$	9%		–					
$\mathbf{HC_{flow}}$	7%	7 3 1 4	73.3	57	70	20	80	30

Table 3.12: Minimum detectable fault and the confusion matrix corresponding to equation (3.2.9).

Fault type	Minimum detectable fault (%)	Confusion matrix	Accuracy (%)	Precision (%)	TPR (%)	FPR (%)	TNR (%)	FNR (%)
$\mathbf{LT_{efi}}$	3%	8 2 1 4	80	66	80	20	80	20
$\mathbf{HT_{efi}}$	6%	8 2 1 4	80	66	80	20	80	20
$\mathbf{LC_{efi}}$	9%		–					
$\mathbf{HC_{efi}}$	5%	8 2 1 4	80	66	80	20	80	20
$\mathbf{LT_{flow}}$	9%		–					
$\mathbf{LC_{flow}}$	6%	9 1 2 3	80	75	90	40	60	10
$\mathbf{HT_{flow}}$	9%		–					
$\mathbf{HC_{flow}}$	6%	7 3 1 4	73.3	75	70	20	80	30

Table 3.13: Minimum detectable fault and the confusion matrix corresponding to equation (3.2.10).

For each type of fault, Figures 3.30 to 3.36 show a typical behaviour of the residual to different faults that are used to determine the minimum detectable faults in Table 3.12 and the model in equation (3.2.9). One can observe that before a fault occurs the model has a good agreement with the measured data from the GSP software but after the fault occurrence at the 35th data point the residuals start increasing until they pass the defined thresholds. Similar results were obtained for the model in equation (3.2.10).

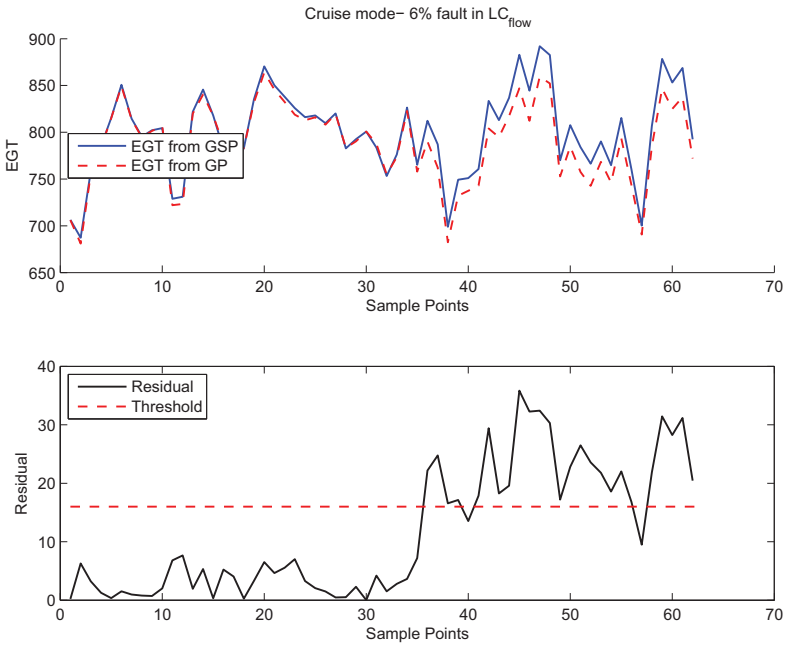

Figure 3.30: Fault in the LC_{flow}.

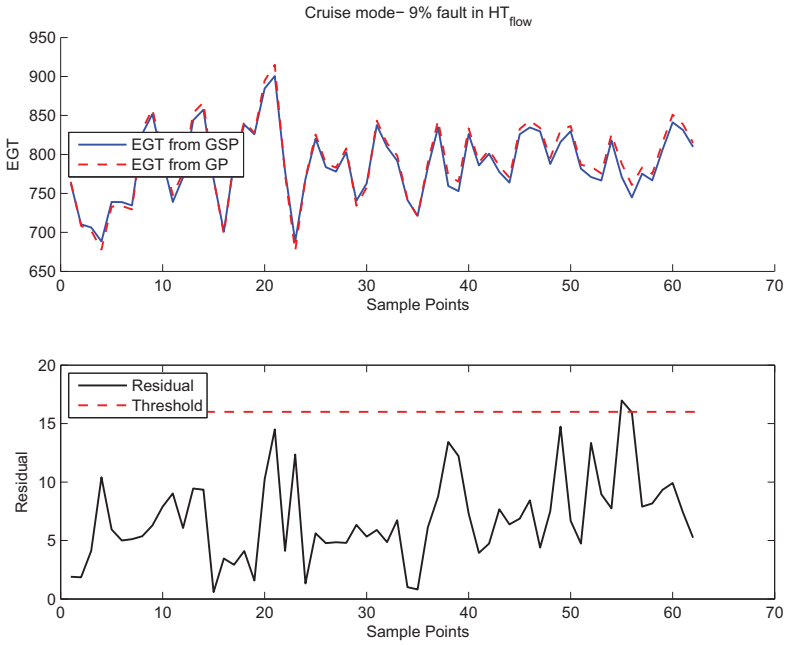

Figure 3.31: Fault in the HT_{flow}.

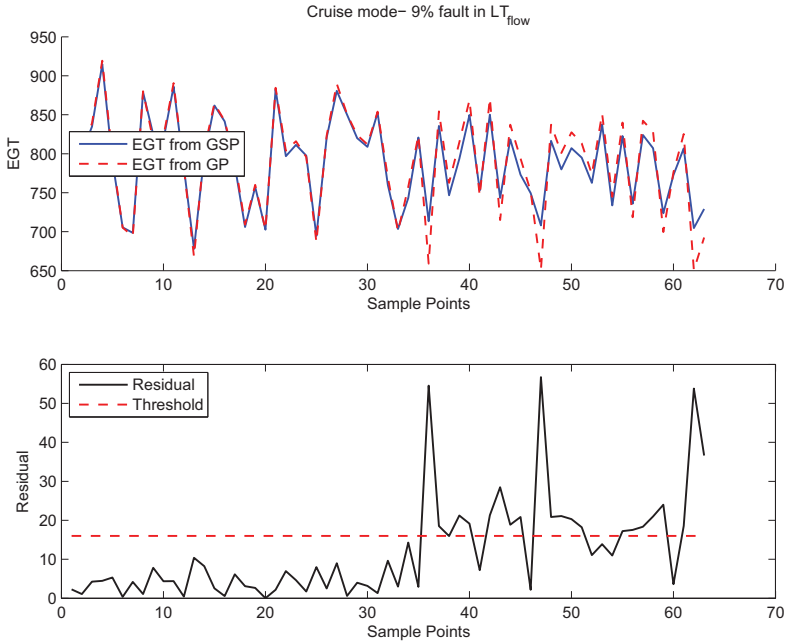

Figure 3.32: Fault in the LT_{flow}.

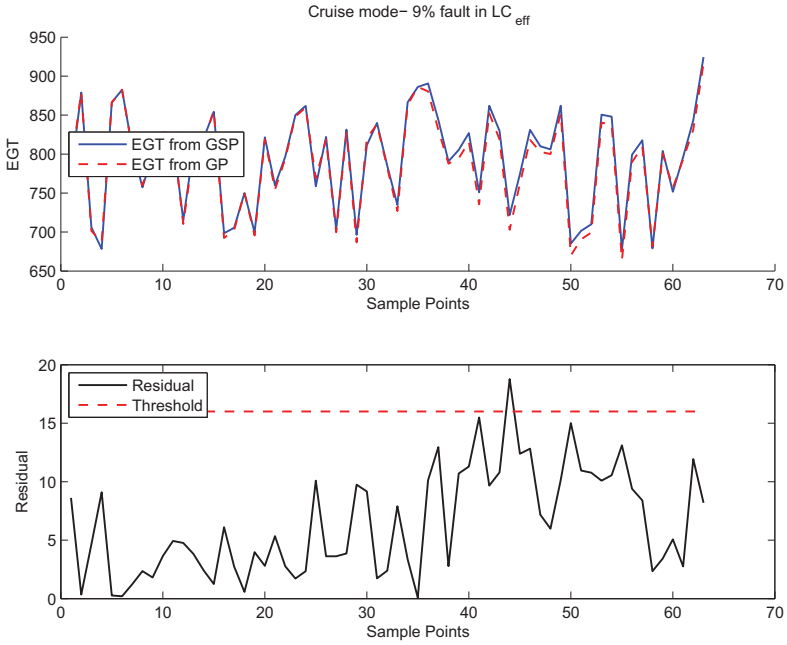

Figure 3.33: Fault in the LC_{eff}.

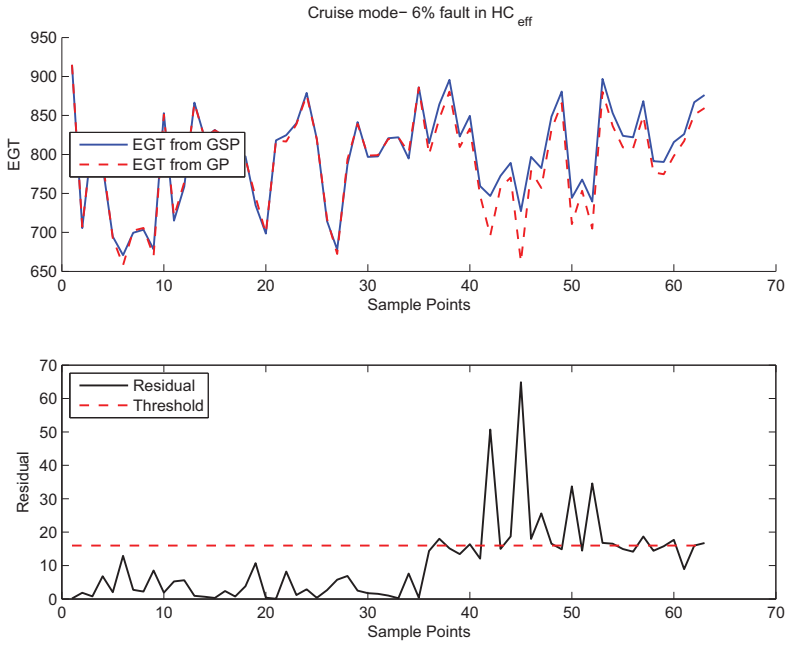

Figure 3.34: Fault in the HC_{eff}.

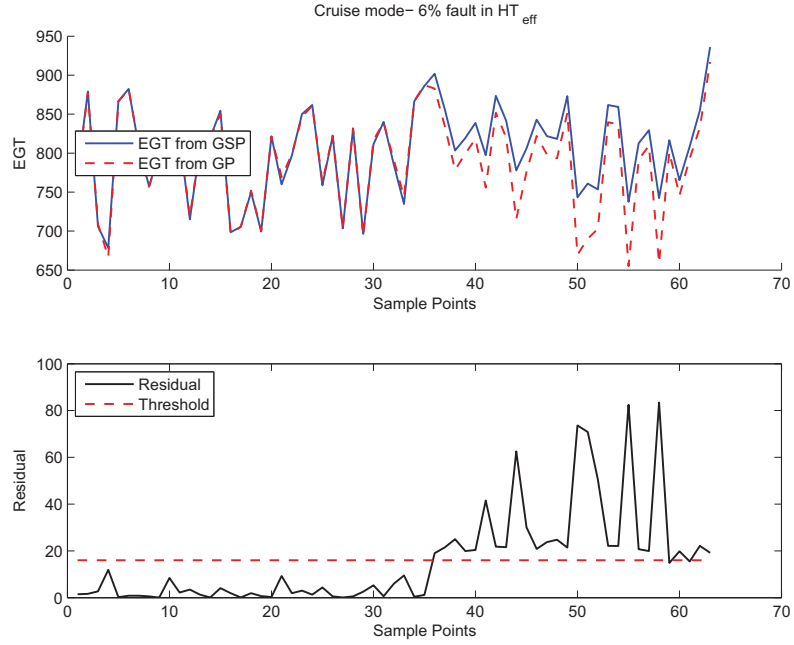

Figure 3.35: Fault in the HT_{eff}.

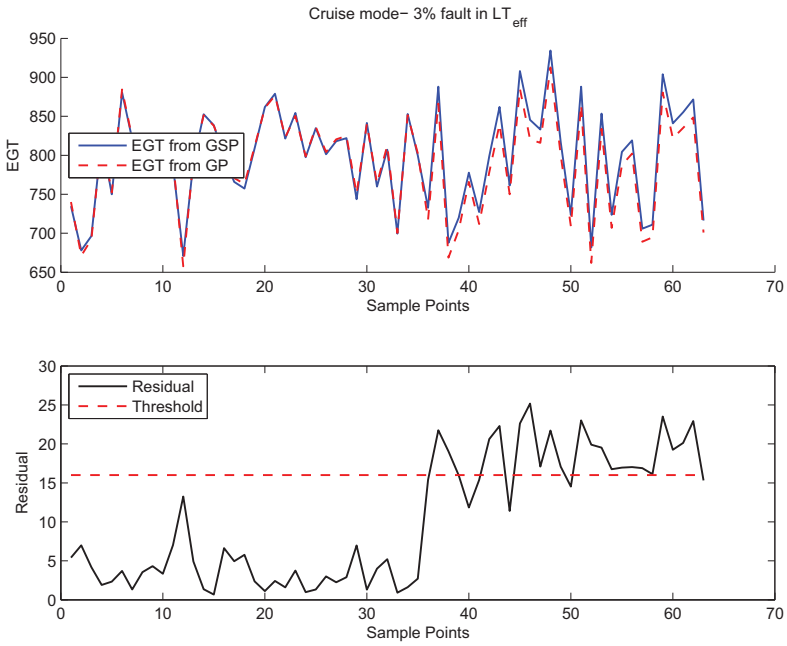

Figure 3.36: Fault in the LT_{eff}.

3.3 Chapter Contributions

The main contribution of this chapter was to provide simple mathematical models of the aircraft jet engine in the steady state working conditions for two modes of the flight operation, namely the take-off and the cruise modes. The developed models provide an efficient tool to monitor the engine condition without getting involved in the complicated dynamics of the engine. By deriving the structure of the relationships among the engine parameters it is only necessary to modify the model numerical coefficients to adjust it for the deviations in the engine.

Another advantage of this approach is the fact that a few data points were necessary to find the model parameters unlike other approaches which consist of a large number of weights and numerical parameters. This is because in addition to the numerical parameters of a model the structure of the model was also evolved and

optimized.

The other aspect of these models is extracting some analytical relations among engine states without having any *a priori* information about the engine components performance charts and the operating maps. In fact the obtained models provide and approximate the interaction of these components characteristics in a portion of their operating maps.

A fault detection (FD) scheme using the developed model was introduced and different aspects of it were investigated through numerical simulations. It was shown that the proposed FD scheme is useful for monitoring the engine health condition and is able to detect faults in a specified severity range.

3.4 Conclusions

In this chapter the GP algorithm was used to develop a set of static nonlinear models that relate EGT as a major engine degradation indicator to the other engine parameters and operational conditions. Using this approach we were able to obtain mathematical models of the engine operation without any information about the engine components characteristics such as turbine or compressor maps. The resulting models were later used to detect abrupt faults in the engine performance. It was shown that with a few snapshots of the engine variables in flight we were able to develop a model that is able to estimate the engine EGT with error less than 0.2% in the take-off and 2% in the cruise modes. As expected, the faults in the cruise mode were less observable than the faults in the take-off mode. Although the error between the generated model and the GSP output is small in the cruise mode (less than 2%), however the effect of the faults are less than this modelling error and more precise models are necessary to capture these changes for this mode.

Chapter 4

GP Algorithm for Jet Engine Fault Isolation

The next step in the FDI problem after successfully detecting the fault is the isolation task. To have a complete monitoring system fault isolation is as important as the fault detection. Isolation refers to finding the location and the type of the fault. It helps making right decisions and efficient recovery actions. A fault in the jet engine can be either actuator fault, sensor fault or component fault. In this research our focus is on engine component faults. In this chapter, a hierarchical approach is proposed for isolating different kinds of faults in a jet engine. It consists of using a series of nonlinear functions called fault indices as classifiers which step by step narrow down the possible fault type toward the actual fault location.

4.1 Methodology

In the previous chapter we used the engine EGT parameter to construct the residual and detect the fault occurrence in the engine. While for detecting the fault one residual is enough, for the task of fault isolation more residuals are necessary [86].

According to the discussion in Chapter 2 the isolation residual set can be built either as structural residuals or directional residuals. In the structural residuals approach, which has been used in this thesis, residuals have sensitivity to the specific fault or faults but are insensitive to other faults. When a fault occurs in an engine component all engine internal states such as temperatures and pressures are more or less affected. However each type of fault affects a group of engine variables more than others. For example, faults in the high pressure turbine efficiency HT_{eff} affect the T_{HT} more than other types of faults. At the same time it affects other engine states with less severity.

The main idea here is to introduce a series of nonlinear functions that are composed of the engine states and operating conditions that can amplify a fault effect by combining its effect on all the engine parameters. We call these nonlinear functions as *fault indices*. These nonlinear functions later provide the required residuals for the task of fault isolation.

In the detection part the residuals were constructed by using the difference between the developed engine model and the measured EGT from the GSP software. The maximum modeling error was statistically calculated and used as fault detection thresholds. Obviously to isolate 8 types of faults that are summarized in Table 2.2 more residuals are necessary. In the next section, we first define the notion of fault indices and subsequently use them to generate a new notion of residual to perform the fault isolation task.

4.1.1 Definition of Fault Indices and Fault Residuals

A fault index is defined as a nonlinear function that accepts engine parameters and engine operating conditions as inputs and produces a scalar output which has the following characteristics:

- For a range of input data corresponding to healthy engine operation its value remains within a constant band.

- It is sensitive to one or a group of faults while remains insensitive to the others.

As an example, and without loss of generality, consider 3 faults f_1,f_2 and f_3 belonging to two classes 1 and 2 and a fault index $F = f(W_f, N_1, N_2, T_{am}, ...)$ that is insensitive to faults f_1 and f_2 in class 1 and sensitive to fault f_3 in class 2. Insensitive means that the value of F does not change significantly if its input variables correspond to the faulty engine in class 1 and remains within a band around the F values for the healthy input data. Sensitive means that if the input data changes from healthy to faulty the output value of the function changes significantly and passes a defined band around the mean of the F healthy values. The idea is to use this deviation as a tool to classify the faults. By defining a threshold on the amount of deviation due to the fault, classification can be done as follows:

- For the previous example and a detected fault if the change in the value of the fault index F from the mean value of F for the training data is more than a predefined threshold the fault belongs to class 2

- Otherwise, if the value of the fault index remains within the threshold the fault belongs to class 1.

In the literature, it is common to define a residual as a signal that has zero or close to a zero value when there is no fault and has a non-zero value when the fault occurs. In this chapter for the task of fault isolation a residual is defined as a signal that assigns a fault to one of the two classes. It remains zero or close to zero when the detected fault belongs to one class and non-zero when it belongs to the other class.

Previously defined fault index can perform the same classifying task, however since a fault index in general is a nonlinear function of the engine variables it is not necessarily zero or close to zero when its inputs belong to the healthy data or the

insensitive faults class. However, according to the definition its value must remain within a band around a mean value that can be a value other than zero. The fault index terminology is used to prevent confusion with the standard definition of the residuals.

By knowing a fault index function, residual can be constructed by removing the mean of the fault index function value for a set of training data. The objective is to exploit GP to find a set of fault indices and subsequently construct the residuals by removing the non-zero biases from these indices in the healthy condition. This can be done by subtracting the mean value of a fault index obtained from healthy training data. Note that the only difference between the residual and the fault index is this shift.

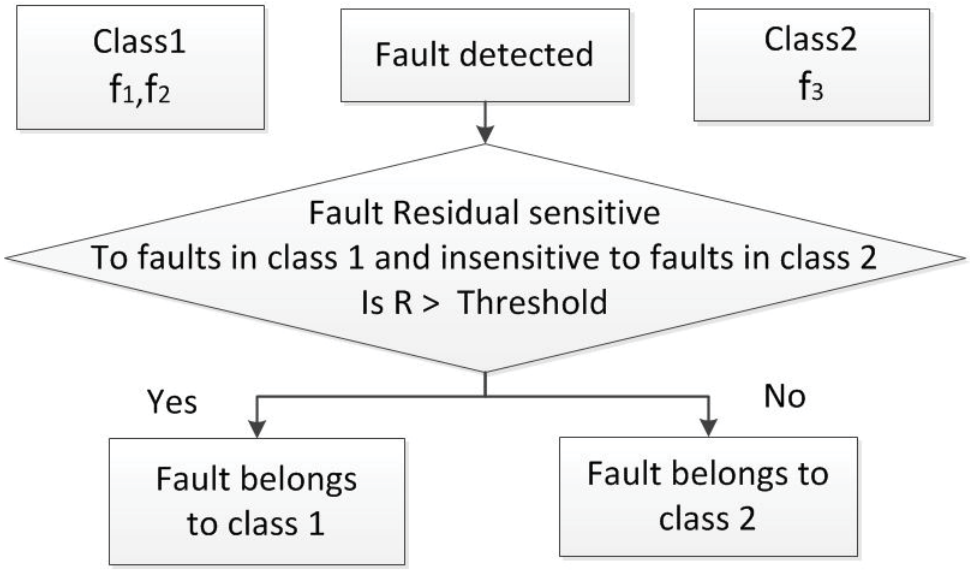

Figure 4.1: Each residual divides the faults into two groups.

4.1.2 Fault Isolation Logic

Using the argument in previous section a residual can assign a detected fault to one of the two faults classes. For the eight types of engine faults one approach could be

to try to find a residual for each fault in a way that it is only sensitive to that fault and insensitive to all the other types of faults (as in the dedicated residual scheme [86]). Consequently, fault isolation can be done simply by looking at these residuals and see which one respond to the fault. The problem with this approach is that due to the correlation among the engine variables and the fact that a fault in one engine component more or less affects all the engine variables it is difficult to find a residual that reacts to only one type of fault without reacting to all the other types of faults.

To overcome this difficulty, a hierarchical fault isolation procedure is developed by defining fault residuals for a subset of faults instead of one fault at each isolation step. In this case the GP has less constraints and more probability of finding the appropriate results. As an example, consider the fault isolation tree shown in Figure 4.2. The isolation task can be performed by defining four fault residuals and following this tree and corresponding fault residual, at each step. Starting from the top, if the value of R_1 residual is more than its predefined threshold Thr_1, the detected fault belongs to class 1, otherwise it belongs to the class 2. Knowing that the fault belongs to one of these classes one can move to the next level and examine the residual R_2 if the fault belongs to class 1 or residual R_3 if it belongs to class 2. Following this procedure one can finally isolate the type of fault.

Fault Tree Construction

Before starting to construct the fault residuals for the isolation purpose it is necessary to construct the fault isolation tree structure. Randomly dividing the faults into some classes and trying to find the residuals that classify the faults in those classes is not a proper approach since some faults have similar affects and signatures on the engine. Putting similar faults in the same class specially at the higher levels of the isolation tree provides a better contrast and more feasibility for the GP in obtaining a fault

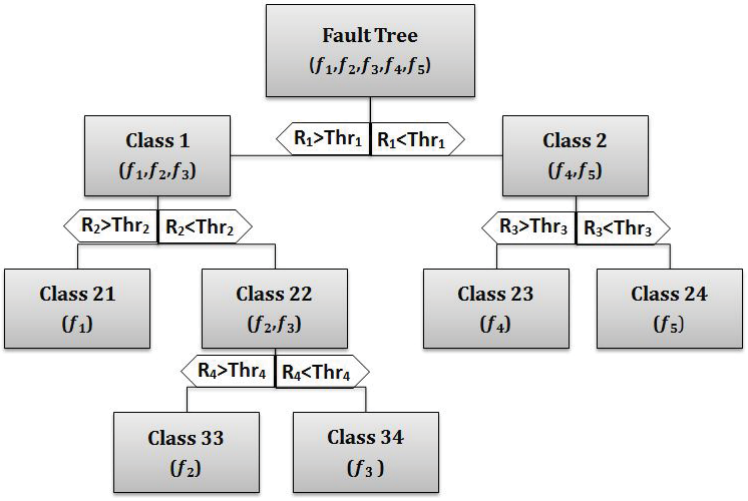

Figure 4.2: An example fault tree.

index and prevents a conflicting condition in determining the numerical coefficients of the fault index. For example, if two faults that have similar effects on the engine are placed into two different classes GP algorithm tries to construct a residual which is sensitive to the first one and insensitive to the second one. Because of the fault similarities, the GP can not produce an index function that is strongly reactive to the first fault and at the same time remains insensitive to the other one. For this purpose and to avoid conflicts an initial classification is necessary. In this thesis, the correlation analysis is used to accomplish this task.

The correlation matrix gives a measure of the amount of linear relationship between two sets of data. The correlation coefficient between two random variables X_i and X_j is defined as

$$corr(X_i, X_j) = \frac{E((X_i - \mu X_i)(X_j - \mu X_j))}{\sigma_{X_i}\sigma_{X_j}} \tag{4.1.1}$$

where E is the expected operator, μX_i and μX_j are the standard deviations and σ_{X_i} and σ_{X_j} are their covariances [87]. The implementation details and resultant classes are explained subsequently in the simulation results, Section 4.2.1.

Threshold Definition

For the same amount of fault severity in component efficiencies or flow capacities, different faults have different levels of impact on the engine variables. Faults in some components as in high pressure turbine have sever effects on the engine variables while the same severity of fault in the low pressure turbine efficiency has a minor effects on the engine variables.

In the previous section fault tree and classifier residuals were introduced. Each of these residuals classifies two or more faults into two classes by reacting to faults in the first class and not responding to faults in the second one. In the ideal case, the residual remains zero when faults in the second class occur however since any fault in the engine affects more or less all the engine variables the fault residual is not completely insensitive to faults in the second class and respond to them within a band. The width of this band depends on the severity of these faults. Consequently, a sufficiently high severity fault in the second class may lead to a response which may pass a pre-defined threshold for isolating faults in in the first class. This results in a false classification.

Consequently, a limit must be defined on the smallest isolable faults for the faults in the first class and the largest isolable fault severity for the faults in the second class. Obviously there is a trade off between the level of the threshold, the lowest severity isolable fault in the first class and the highest severity level of fault in the second class at each level of the fault isolation process.

Figure 4.3 shows a fault residual which is designed to assign a fault to one of the two LT_{flow} and HT_{flow} faults by responding to HT_{flow} and staying inactive to LT_{flow} fault. As can be seen although the residual is supposed to stay zero if the LT_{flow} fault occurs but there are small responses to this fault (compared to response to the HT_{flow} fault). A threshold can be selected at the intersection of the maximum LT_{flow}

fault severity and the minimum HT_{flow} fault severity. For this residual the two fault severities correspond to the intersection of the maximum LT_{flow} fault and minimum HT_{flow} fault that are isolable by using the residual in this example.

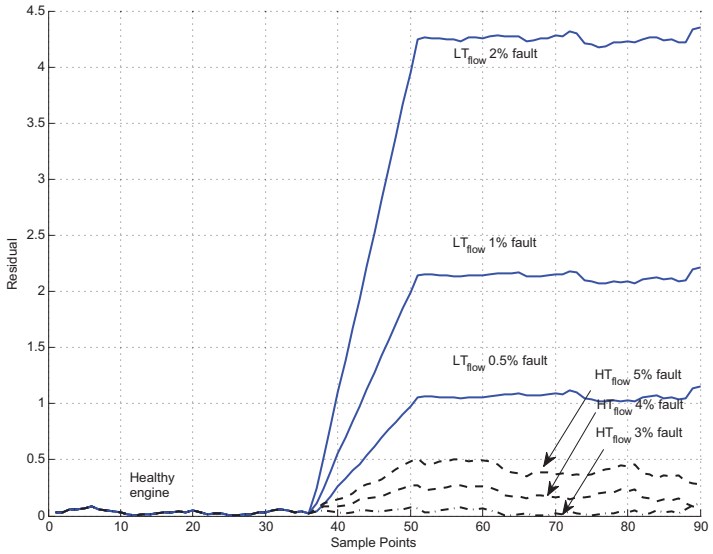

Figure 4.3: Defining the threshold by decreasing and increasing the faults severities in the two classes.

For the cases that the classes contain more than one fault, in order to define a threshold related to each residual, the severity of the least sensitive fault in the first class can be reduced while at the same time the amount of fault in the most sensitive fault in the second class is increased until the resulting residual value from the two faults converge to each other. In this case, the value of the residual is considered as the threshold, and the severity of the fault in the first class as the minimum isolable fault for that fault and the severity of fault in the second class as the maximum isolable fault for that fault. Similarly, to determine the minimum or the maximum isolable fault for the other faults in the two classes the fault severities can be decreased or increased until they reach to the previously determined threshold.

4.1.3 GP Implementation

In the same manner as in the previous chapter, the objective is to use genetic programming technique to construct the fault indices and subsequently fault residuals and optimize them in such a way that each one has the most sensitivity to one class of faults while at the same time it has the lowest possible sensitivity to the other classes of faults.

In the simulations it is assumed that all engine variables as shown in Table 2.3 are available. Each residual is a function of all or part of these variables. During the course of the algorithm run less significant variables on the target faults would be eliminated automatically. The temperature T_{cc} after the combustion chamber is not considered since in practice one usually does not have physically sensory data due to very high temperature at this stage. Different types of faults in the jet engine components are modelled by changes in the efficiency and flow capacity of the main components.

The same settings as in the fault detection phase are used for the GP parameters and optimizer parameters. These settings are provided in Table 3.2. The main difference here is in the definition of the fitness function. To find each fault index it is necessary to run the GP with an specified fitness function.

4.1.3.1 Fitness Function Definition

The key factor in obtaining an effective fault index function with highest sensitivity to a class of faults and lowest sensitivity to the faults in other classes is to appropriately modify the fitness function in the GP algorithm. A special fitness function should be defined such that during the algorithm run better candidates gain more grade than the others. The total fitness consists of four parts as shown in equation (4.1.2). Each part is designed to comply with one of the objectives

$$Fitness = fit_1 + fit_2 + fit_3 + fit_4 \tag{4.1.2}$$

where $fit_i, i = 1, ..., 4$ denote fitness functions that are defined as follows:

The fitness function fit_1: Suppose that the fault index that one is looking for is responsible for recognizing the faults belonging to an assumed class A by only reacting to the faults belonging to class A and remains inactive to faults in the other considered classes.

In this case, an fault index function $F = f(N_1, N_2, T_{HT}, W_f, ...)$ must have highest deviation from the healthy condition if the input engine variables to this index function belong to the faults in class A. This can be fulfilled by considering the equation (4.1.3). In this equation, $X^k_{Healthy}$ and $X^k_{fault_j}$ are the vector of engine variables in the k^{th} snapshot and $F(X^k_{Healthy})$ and $F(X^k_{fault_j})$ are the corresponding values of a candidate fault index for that k^{th} snapshot as inputs, respectively, $n^{healthy}_{data}$ is the number of snapshots in the healthy training data and $n^{fault_j}_{data}$ is the number of snapshots in the $fault_j$ training data, and n_{ClassA} is the number of faults belonging to calss A

$$fit_1 = \sum_{j=1}^{n_{ClassA}} |\frac{1}{n^{fault_j}_{data}} \sum_{k=1}^{n^{fault_j}_{data}} F(X^k_{fault_j}) - \frac{1}{n^{fault_j}_{data}} \sum_{k=1}^{n^{fault_j}_{data}} F(X^k_{Healthy})| \tag{4.1.3}$$

The fitness function fit_2: At the same time the fault index corresponding to class A must have also the least deviation from the healthy value if the input engine variables to it belong to the faults in a class rather than A. Equation (4.1.4) achieves this aim. In this equation $X^k_{fault_j}$ is the vector of engine variables in the k^{th} snapshot and n_{others} is the number of faults belonging to the other classes than A

$$fit_2 = \left[\sum_{j=1}^{n_{others}} |\frac{1}{n_{data}^{fault_j}} \sum_{k=1}^{n_{data}^{fault_j}} F(X_{fault_j}^k) - \frac{1}{n_{data}^{fault_j}} \sum_{k=1}^{n_{data}^{fault_j}} F(X_{Healthy}^k)| \right]^{-1} \tag{4.1.4}$$

The fitness function fit_3: Using only the two previous defined fitness functions for finding the fault indices can result in a signal with large oscillations. The fault index should have fluctuations as small as possible. Large oscillations of a fault index function makes it difficult to define an efficient threshold resulting in poor events detection under low severity faults. It also increases the number of false alarms. To overcome this problem we add the term fit_3 shown in equation (4.1.5) to our fitness function which basically attempts to minimize the variance of the corresponding index. Consequently, individuals with less fluctuations would be assigned higher fitnesses values. In this equation n_{fault} is the total number of faults to be classified by F, $[F]_{fault_j}$ and $[F]_{Healthy}$ are the vectors of fault index values for the faulty input snapshot data, that is $\left[F(X_{fault_j}^1), F(X_{fault_j}^2), ..., F(X_{fault_j}^{n_{data}^{fault}})\right]$ and healthy input data $\left[F(X_{Healthy}^1), F(X_{Healthy}^2), ..., F(X_{Healthy}^{n_{data}^{healthy}})\right]$, respectively

$$fit_3 = \left[\sum_{j=1}^{n_{fault}} (VAR([F]_{fault_j})) + VAR([F]_{Healthy}) \right]^{-1} \tag{4.1.5}$$

The fitness function fit_4: The simplex optimization algorithm implemented for finding the numerical coefficients of the individual models is an unconstrained optimization and there are no limitations on the values that the coefficients can take on. Consequently, the algorithm may increase the fitness only by giving the numerical coefficients a very large or very small values. To make the model coefficients remain within a limited band we introduce equation (4.1.6). In this equation $\|x\|$ is the Euclidean length of a vector x, $V_{coefficients}$ is the vector of a model numerical coefficients.

The first term in this equation prevents the optimizer to converge to very large values for numerical coefficients of a candidate structure and the second terms prevents the optimizer to converge to very small values during the optimization process

$$fit_4 = \|V_{coefficients}\| + \left[\|V_{coefficients}\|\right]^{-1} \tag{4.1.6}$$

The best solution is considered the one that has the maximum total fitness value. During the optimization process the algorithm tries to find the coefficients corresponding to each structure in a way that the fitness has its maximum value. Finally, when one of the algorithm termination criteria namely, the maximum iterations or termination fitness value is satisfied the algorithm stops and the best solution with highest fitness value will be selected as the fault index of the corresponding classification.

4.2 Simulation Results

As described earlier the isolation task can be fulfilled by constructing a fault tree and then using a set of classifying residuals to move through the fault tree from high levels towards the lower branches and finally recognizing the actual fault location. In this section, first the eight types of engine faults are regrouped and arranged in a fault tree using the correlation analysis performed on the training data. Then GP is implemented to drive seven classifying residuals required for the developed fault tree. Because of the weak detection capability of the model developed for the cruise mode the isolation is only performed for the take-off mode.

4.2.1 Fault Tree Construction

In order to construct the fault tree one can use the correlation analysis to capture the similarities that exist among the engine faults. Fault classes can be built by putting

similar faults in the same class. Towards this end, for each of the eight types of faults and the healthy engine a series of data snapshots were produced using the GSP software corresponding to the same operating conditions. The severity of faults in all data are 2% and each data set contains 100 snapshots. Each snapshot is a vector consisting of the engine variables as shown in Table 2.3. For each engine variable, the change of that variable with respect to its value in a healthy engine was calculated.

For example, for the fault in the high pressure compressor efficiency HT_{eff} and the spool speed parameter N_1 we have a set of n changes ($\triangle N_1$) corresponding to n data snapshots as follows

$$\triangle N_1 = N_{1_{HT_{eff}}} - N_{1_{Helthy}} \tag{4.2.1}$$

A correlation matrix is found for the changes for each variable and among different types of faults. This analysis results in 10 correlation matrices for 10 engine variables. Each element in the correlation matrix shows the correlation between two faults for the corresponding engine variable. A typical correlation matrix for the engine variable N_1 is shown in Table 4.1.

	LT_{eff}	**HT_{eff}**	**LC_{eff}**	**HC_{eff}**	**LT_{flow}**	**LC_{flow}**	**HT_{flow}**	**HC_{flow}**
LT_{eff}	1	0.83	0.56	0.81	-0.13	-0.64	-0.59	-0.024
HT_{eff}	0.83	1	0.61	0.98	0.07	-0.65	-0.56	0.07
LC_{eff}	0.56	0.61	1	0.58	-0.29	-0.45	-0.36	-0.05
HC_{eff}	0.81	0.98	0.58	1	0.13	-0.65	-0.58	0.05
LT_{flow}	-0.13	0.07	-0.29	0.13	1	0.26	0.095	.20
LC_{flow}	-0.64	-0.65	-0.45	-0.65	0.26	1	0.69	0.52
HT_{flow}	-0.59	-0.56	-0.36	-0.58	0.09	0.69	1	0.17
HC_{flow}	-0.024	0.07	-0.05	0.05	0.20	0.52	0.17	1

Table 4.1: The N_1 parameter correlation matrix for different faults.

It can be seen that strong correlations exist for some faults. In the next step, the correlation matrices are rounded off with a threshold on the amount of the correlation between each fault pair. Specifically, the value 0.8 is applied as the threshold for

rounding off the correlations. This value is used since it was observed that with smaller value of the threshold as seen in Table 4.3 a large number of similar values are present which makes it impossible to detect clear classes.

Each correlation between a pair of faults with higher values than this threshold is assigned the value of 1, and otherwise 0 (Table 4.2). This procedure is repeated for all the correlation matrices corresponding to all the 10 engine variables.

	$\mathbf{LT_{eff}}$	$\mathbf{HT_{eff}}$	$\mathbf{LC_{eff}}$	$\mathbf{HC_{eff}}$	$\mathbf{LT_{flow}}$	$\mathbf{LC_{flow}}$	$\mathbf{HT_{flow}}$	$\mathbf{HC_{flow}}$
$\mathbf{LT_{eff}}$	1	1	0	1	0	0	0	0
$\mathbf{HT_{eff}}$	1	1	0	1	0	0	0	0
$\mathbf{LC_{eff}}$	0	0	1	0	0	0	0	0
$\mathbf{HC_{eff}}$	1	1	0	1	0	0	0	0
$\mathbf{LT_{flow}}$	0	0	0	0	1	0	0	0
$\mathbf{LC_{flow}}$	0	0	0	0	0	1	0	0
$\mathbf{HT_{flow}}$	0	0	0	0	0	0	1	0
$\mathbf{HC_{flow}}$	0	0	0	0	0	0	0	1

Table 4.2: The N_1 parameter correlation matrix for different faults.

In the next step, for each pair of faults the number of ones in the rounded off correlation matrices are counted and sorted in a matrix. This matrix is shown in Table 4.3. Each element of this table shows the number of high correlations among two faults for all the engine variables. For example, the value 5 in the raw 4 and column 5 shows that LT_{flow} and HC_{eff} faults have strong correlation in 5 engine variables. Fault pairs with a larger value in this table have stronger correlations in more engine variables and more similar effects on the engine performance.

	$\mathbf{LT_{eff}}$	$\mathbf{HT_{eff}}$	$\mathbf{LC_{eff}}$	$\mathbf{HC_{eff}}$	$\mathbf{LT_{flow}}$	$\mathbf{LC_{flow}}$	$\mathbf{HT_{flow}}$	$\mathbf{HC_{flow}}$
$\mathbf{LT_{eff}}$	10	2	1	2	0	1	0	1
$\mathbf{HT_{eff}}$	2	10	1	10	4	0	0	0
$\mathbf{LC_{eff}}$	1	1	10	1	0	0	0	0
$\mathbf{HC_{eff}}$	2	10	1	10	5	0	0	0
$\mathbf{LT_{flow}}$	0	4	0	5	10	0	3	0
$\mathbf{LC_{flow}}$	1	0	0	0	0	10	0	7
$\mathbf{HT_{flow}}$	0	0	0	0	0	0	10	0
$\mathbf{HC_{flow}}$	1	0	0	0	0	7	0	10

Table 4.3: Number of the highly correlated fault pairs.

By looking at this table one can note that HT_{eff}, HC_{eff},and LT_{flow} have high values of highly correlated variables. This means that they have similar manifestation in the engine. This similarity in fault signatures can make their separation more difficult. LC_{flow} and HC_{flow} have also high values of correlated variables and are similar. Other variables have small values of correlated variables and do not have much similarities. Based on this analysis, in our fault isolation tree and at the top isolation levels we try to put similar faults in the same class. Therefore , we avoid differentiating similar faults at these levels which have fewer feasible search spaces, and consequently increasing the algorithm success in finding the acceptable results. The different classes at each level of the fault isolation is shown in Figure 4.4.

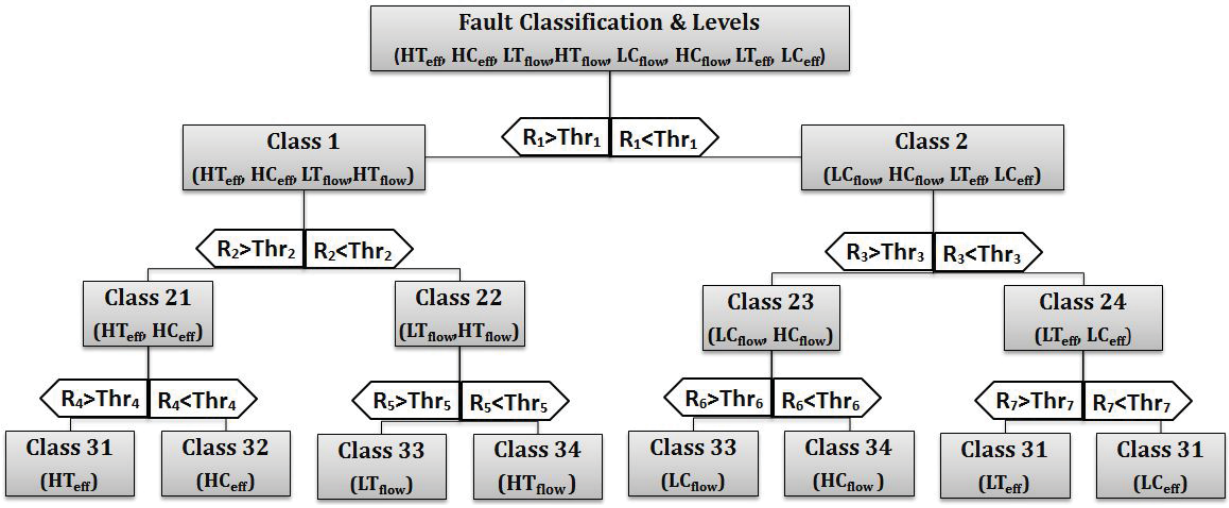

Figure 4.4: Faults isolation tree and hierarchy of the corresponding fault residuals.

By constructing the fault tree at each level of the fault isolation we divide the faults into two groups and use one residual to narrow down the possible faults to one of the classes. Seven fault residuals are defined for the fault isolation as shown in Figure 4.4. Each residual is defined explicitely as follos:

$$R_i = F_i - E\{F_i\}, i = 1, ..., 7 \tag{4.2.2}$$

where F_i denotes the i^{th} fault index. At each level, the corresponding residual narrows down the possible number of faults to the smaller number of faults in one of the classes as one proceeds to the next lower level.

4.2.2 Take-off Results

In previous section eight types of engine faults were arranged in a fault tree. Referring to Figure 4.4 in order to completely isolate a fault seven residuals are necessary. In this section first by using the GP algorithm and the fitness function defined is Section 4.1.3.1 seven isolation residuals are constructed. Subsequently the threshold values related to each residual is determined based on the discussion in Section 4.1.2. Table

4.4 summarizes the seven fault indices $F_i, i = 1, ..., 7$, that are obtained by running the GP algorithm using the training data and the corresponding fitnesses functions. As mentioned before these nonlinear functions are called fault indices since their values in healthy situation does not vary around zero. In the simulation results presented in Figures 4.5 to 4.11 the residuals are constructed from these fault indices and by subtracting the mean values of each index function corresponding to the train data.

Fault indices
$F_1 = \dfrac{aN_2}{2P_{HC} + bP_{HT}} + 2$
$F_2 = (2T_{LC} + 2 - (2aT_{LT}^4 - 2W_f^2 + bP_{HT} - 1)^2)^2$
$F_3 = \dfrac{2(aN_2^3 + 4T_{LC}T_{LT})}{bT_{HT}T_{LT}}$
$F_4 = \dfrac{4P_{HT} + aW_f - 2P_{LC}T_{HT}}{bT_{LT}}$
$F_5 = \dfrac{(aT_{HC}P_{HT} + bW_f)}{(cP_{LC}N_2^2)}$
$F_6 = \dfrac{aM}{bP_{LT}T_{HT}^2 - cP_{LP} - 1}$
$F_7 = \dfrac{aT_{HC}^2 - 2P_{HT} + 4T_{LC}P_{HC}}{2P_{HT}^2(bT_{LT} - 1)}$

Table 4.4: Fault indices for seven levels of fault isolation.

As can be seen each fault index function is a nonlinear function of a subset of engine variables and operation conditions. In fact, as the algorithm is running, it gradually eliminates the less effective variables and keeps the most important ones. Every fault index model contains a few number of numerical parameters. These coefficients are optimized during the algorithm run using the simplex algorithm as described in Chapter 2. These coefficients and their values are summarized in Table

4.5.

Fault Indices Parameters			
Parameters	**a**	**b**	**c**
$\mathbf{F_1}$	-0.1831	-1.9551	
$\mathbf{F_2}$	5.3373	-9.1079	
$\mathbf{F_3}$	5.9717	-0.2906	
$\mathbf{F_4}$	2.7551	-7.8149	
$\mathbf{F_5}$	-6.6837	1.0690	-0.1187
$\mathbf{F_6}$	10.158	-16.2151	-8.3866
$\mathbf{F_7}$	0.5566	-0.0413	

Table 4.5: Fault indices numerical coefficients values.

To reduce noise and false alarms we have also used a windowing filter to smoothen the residuals. The length of this window in Figures 4.5 to 4.11 are selected to be 15 points. Figures 4.5 to 4.11 show the behaviour of these residuals for different faults when a fault occurs at the sample point 50. In Figure 4.5 the residual for the first level of fault isolation tree R_1 has reacted to four faults ($LT_{flow}, HT_{eff}, HC_{eff}, HT_{flow}$) belonging to the first class in the fault tree and remained close to zero to the four other faults ($LT_{eff}, LC_{eff}, LC_{flow}, HC_{flow}$) belonging to the second class. Consequently, by observing this residual, one can conclude that a fault is within the first class if it starts to grow and pass the predefined threshold or belongs to the second class if the residual stays below the threshold (as explicitly in Section 4.2.2.1). Note that here it is assumed that the fault detection has already been performed and fault occurrence is already detected.

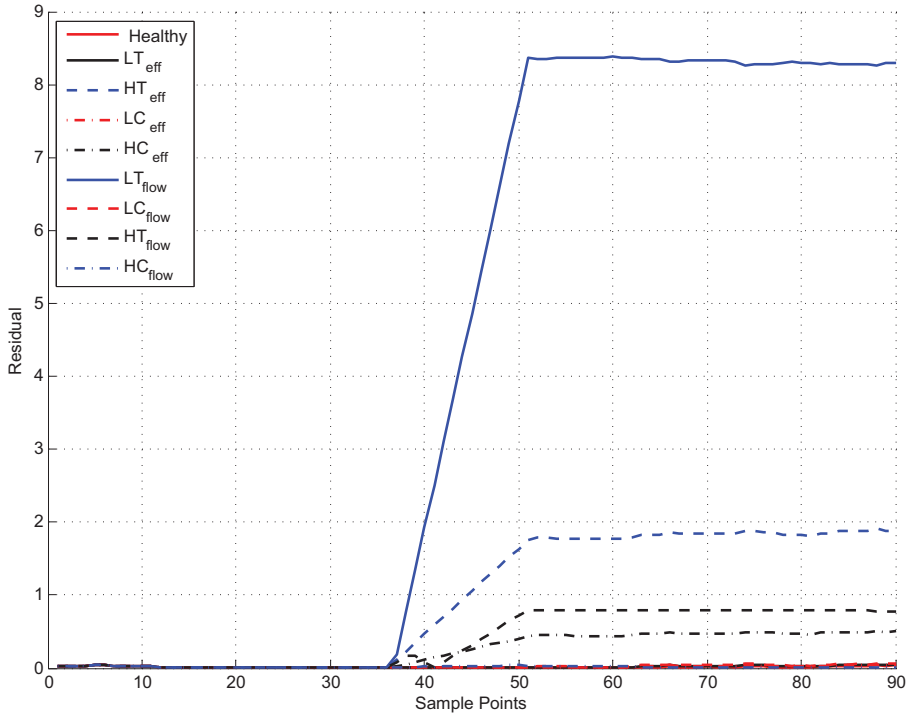

Figure 4.5: Residual R_1 response to different types of faults.

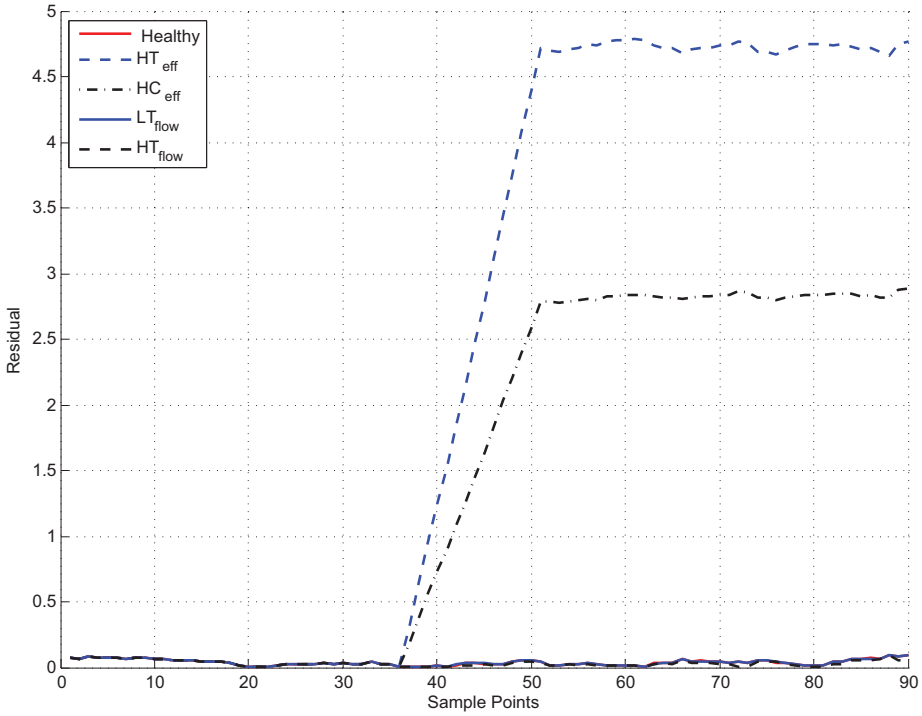

Figure 4.6: Residual R_2 response to different types of faults.

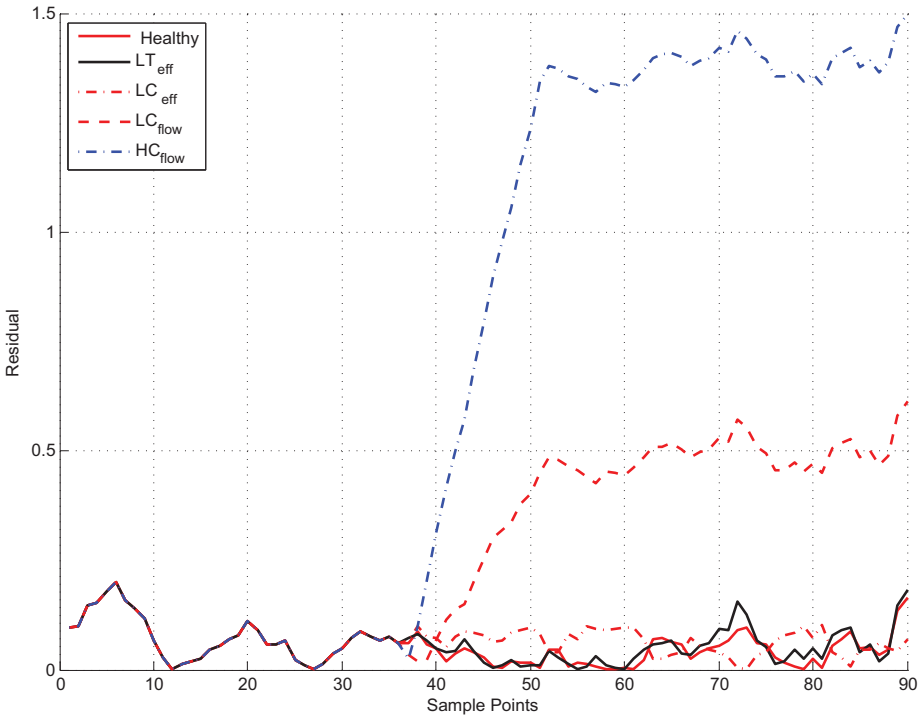

Figure 4.7: Residual R_3 response to different types of faults.

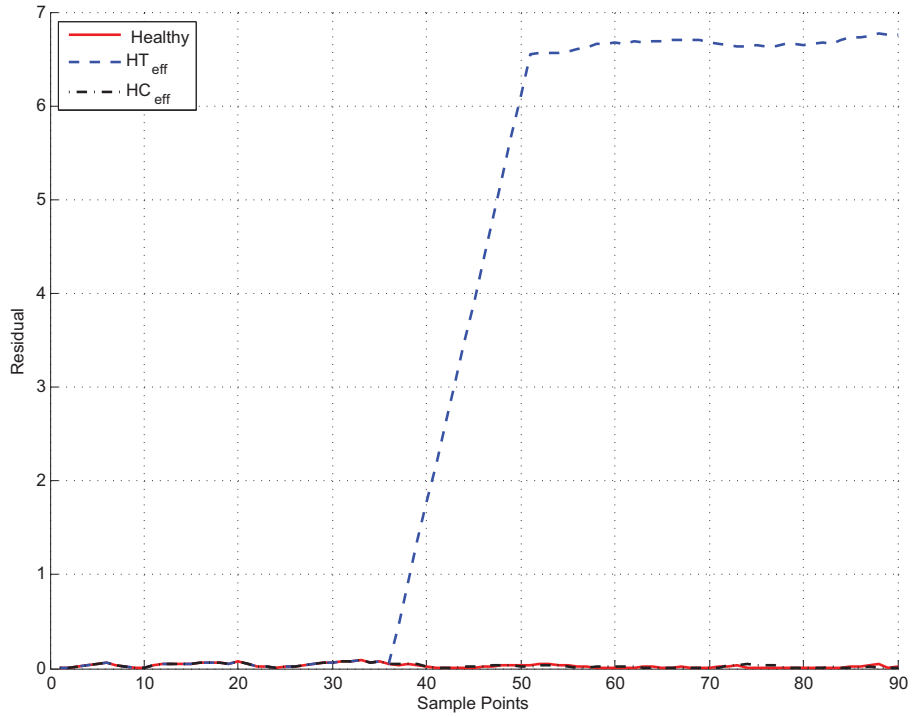

Figure 4.8: Residual R_4 response to different types of faults.

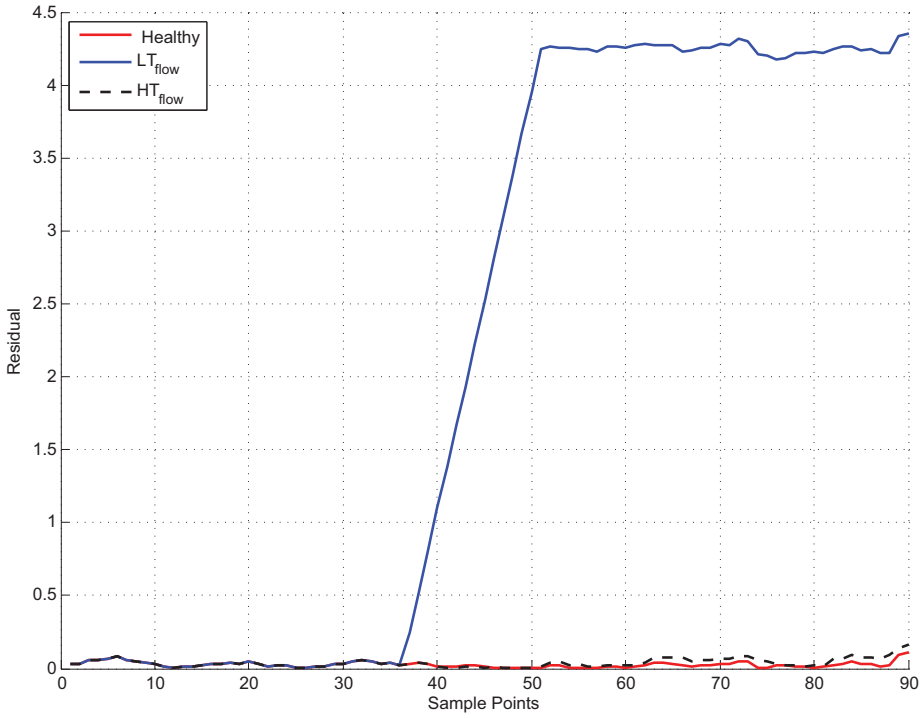

Figure 4.9: Residual R_5 response to different types of faults.

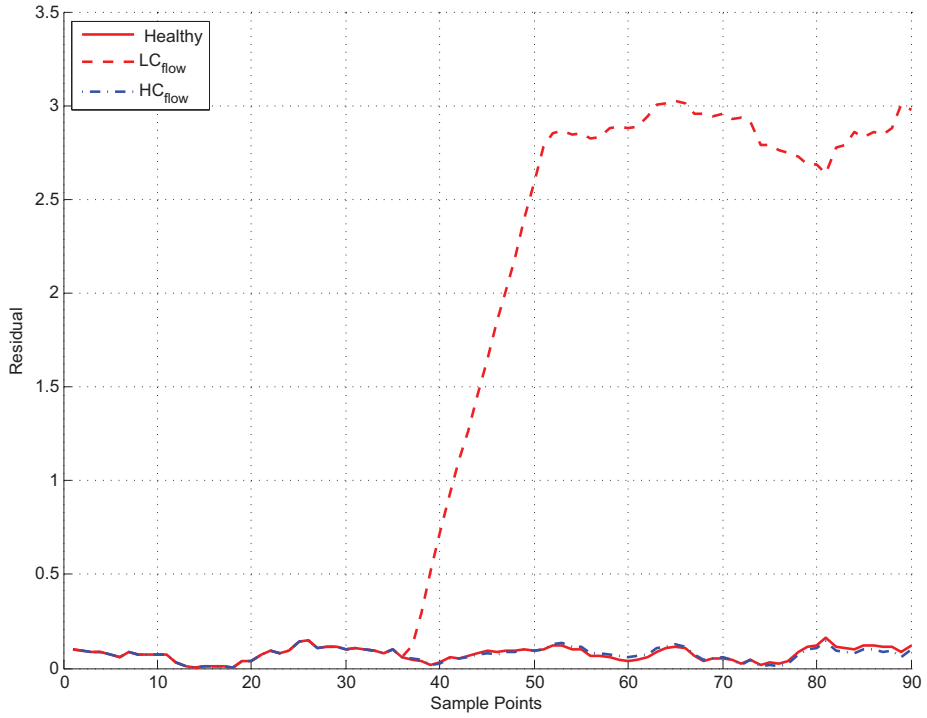

Figure 4.10: Residual R_6 response to different types of faults.

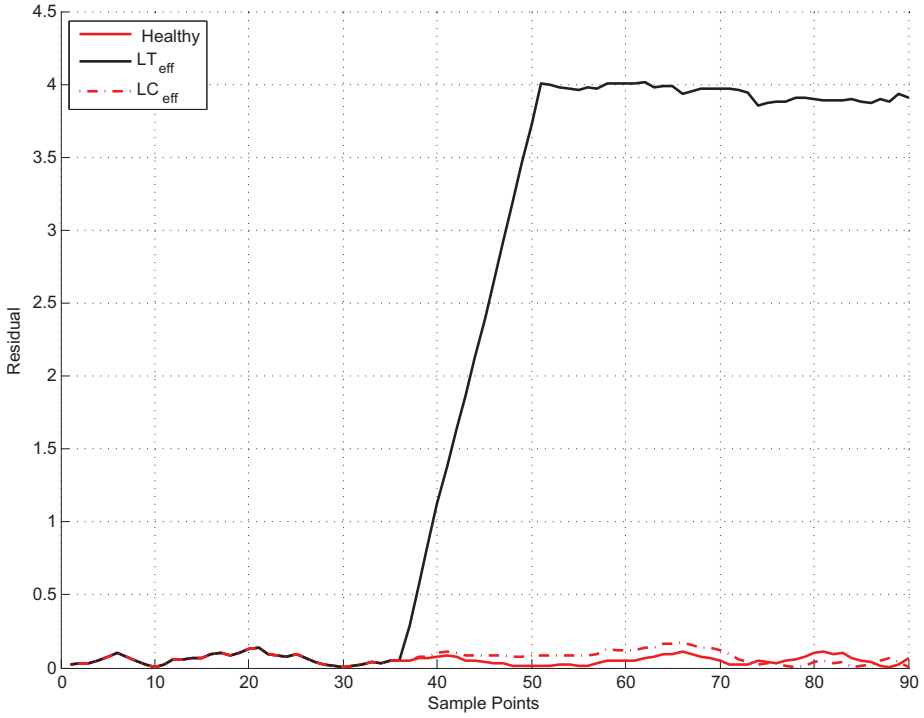

Figure 4.11: Residual R_7 response to different types of faults.

Training Data

Following the discussion in the previous section, one needs seven residuals to isolate all the eight types of faults. Towards this end, nine series of data are used for extracting the fault indices functions and their corresponding residuals and finding the numerical coefficients values. One set of data corresponds to the healthy engine working at different operating conditions. Each of the other data sets corresponds to a fault in one engine component. All data correspond to either 3% change in the component efficiency or the component flow capacity as given by equation (4.2.3). Each data set contains 100 engine variable snapshots. These sample points were randomly selected from the same range as take-off detection that was shown in Table 3.3.

$$
\begin{aligned}
S_{Healthy} &= \{X|X \textit{ belongs to healthy engine}\} \\
S_{HT_{eff}} &= \{X|X \textit{ belongs to engine with } 3\% \textit{fault in } HT_{eff}\} \\
S_{HC_{eff}} &= \{X|X \textit{ belongs to engine with } 3\% \textit{fault in } HC_{eff}\} \\
S_{LT_{eff}} &= \{X|X \textit{ belongs to engine with } 3\% \textit{fault in } LT_{eff}\} \\
S_{LC_{eff}} &= \{X|X \textit{ belongs to engine with } 3\% \textit{fault in } LC_{eff}\} \\
S_{HT_{flow}} &= \{X|X \textit{ belongs to engine with } 3\% \textit{fault in } HT_{flow}\} \\
S_{HC_{flow}} &= \{X|X \textit{ belongs to engine with } 3\% \textit{fault in } HC_{flow}\} \\
S_{LT_{flow}} &= \{X|X \textit{ belongs to engine with } 3\% \textit{fault in } LT_{flow}\} \\
S_{LC_{flow}} &= \{X|X \textit{ belongs to engine with } 3\% \textit{fault in } LC_{flow}\}
\end{aligned}
\tag{4.2.3}
$$

Fitnesses Definitions

During the GP run for determining a fault index function the goodness of each individual structure is determined by using fitness function and the training data. In determining the fitness value of a candidate structure the data sets shown in equation

(4.2.4) are calculated from the training data sets shown in equation (4.2.3) and for each value of the numerical coefficients. These values are then used in determining the fitness of the structure. The fitnesses defined for finding each one of fault indices functions that are represented in Table 4.4 are shown in equations (4.2.5) to (4.2.11). In these equations $mean(.)$ represents the mean value of the data and $|.|$ represents the absolute value of its argument. To calculate the fitness of a candidate structure the simplex optimizer changes the numerical coefficients of the structure and recalculates the values in equations (4.2.4) and the corresponding fitness to obtain the best fitness value and returns the maximum obtained fitness and corresponding numerical coefficients. This fitness is then considered as the fitness of that structure candidate in the GP algorithm. Specifically, we have

$$\begin{aligned}
S_F^{Healthy} &= \{F(X)|X \in S_{Healthy}\} \\
S_F^{HT_{eff}} &= \{F(X)|X \in S_{HT_{eff}}\} \\
S_F^{HC_{eff}} &= \{F(X)|X \in S_{HC_{eff}}\} \\
S_F^{LT_{eff}} &= \{F(X)|X \in S_{LT_{eff}}\} \\
S_F^{LC_{eff}} &= \{F(X)|X \in S_{LC_{eff}}\} \\
S_F^{HT_{flow}} &= \{F(X)|X \in S_{HT_{flow}}\} \\
S_F^{HC_{flow}} &= \{F(X)|X \in S_{HC_{flow}}\} \\
S_F^{LT_{flow}} &= \{F(X)|X \in S_{LT_{flow}}\} \\
S_F^{LC_{flow}} &= \{F(X)|X \in S_{LC_{flow}}\}
\end{aligned} \tag{4.2.4}$$

Fault index function F_1:

$$
\begin{aligned}
fit_1 &= |mean(S_{F_1}^{HT_{eff}}) - mean(S_{F_1}^{Healthy})| + |mean(S_{F_1}^{HC_{eff}}) - mean(S_{F_1}^{Healthy})| \\
&\quad + |mean(S_{F_1}^{LT_{flow}}) - mean(S_{F_1}^{Healthy})| + |mean(S_{F_1}^{HT_{flow}}) - mean(S_{F_1}^{Healthy})| \\
fit_2 &= \Bigg[|mean(S_{F_1}^{LC_{flow}}) - mean(S_{F_1}^{Healthy}) + |mean(S_{F_1}^{HC_{flow}}) - mean(S_{F_1}^{Healthy})| \\
&\quad + |mean(S_{F_1}^{LT_{eff}}) - mean(S_{F_1}^{Healthy})| + |mean(S_{F_1}^{LC_{eff}}) - mean(S_{F_1}^{Healthy})|\Bigg]^{-1} \\
fit_3 &= \Bigg[Var(S_{F_1}^{Healthy}) + Var(S_{F_1}^{HC_{eff}}) + Var(S_{F_1}^{HT_{eff}}) + Var(S_{F_1}^{LT_{eff}}) + Var(S_{F_1}^{LC_{eff}}) \\
&\quad + Var(S_{F_1}^{HT_{flow}}) + Var(S_{F_1}^{HT_{flow}}) + Var(S_{F_1}^{HC_{flow}}) + Var(S_{F_1}^{LT_{flow}})\Bigg]^{-1} \\
fit_4 &= \|numericalcoefficients\| + \Bigg[\|numericalcoefficients\|\Bigg]^{-1} \\
fit &= fit_1 + fit_2 + fit_3 + fit_4
\end{aligned}
\tag{4.2.5}
$$

Fault index function F_2:

$$
\begin{aligned}
fit_1 &= |mean(S_{F_2}^{HT_{eff}}) - mean(S_{F_2}^{Healthy})| + |mean(S_{F_2}^{HC_{eff}}) - mean(S_{F_2}^{Healthy})| \\
fit_2 &= \Bigg[|mean(S_{F_2}^{LT_{flow}}) - mean(S_{F_2}^{Healthy})| + |mean(S_{F_2}^{HT_{flow}}) - mean(S_{F_2}^{Healthy})|\Bigg]^{-1} \\
fit_3 &= \Bigg[Var(S_{F_2}^{Healthy}) + Var(S_{F_2}^{HT_{eff}}) + Var(S_{F_2}^{HC_{eff}}) + Var(S_{F_2}^{LT_{flow}}) + Var(S_{F_2}^{HT_{flow}})\Bigg]^{-1} \\
fit_4 &= \|numerical\ coefficients\| + \Bigg[\|numericalcoefficients\|\Bigg]^{-1} \\
fit &= fit_1 + fit_2 + fit_3 + fit_4
\end{aligned}
\tag{4.2.6}
$$

Fault index function F_3:

$$
\begin{aligned}
fit_1 &= |mean(S_{F_3}^{LC_{flow}}) - mean(S_{F_3}^{Healthy})| + |mean(S_{F_3}^{HC_{flow}}) - mean(S_{F_3}^{Healthy})| \\
fit_2 &= \left[|mean(S_{F_3}^{LT_{eff}}) - mean(S_{F_3}^{Healthy})| + |mean(S_{F_3}^{LC_{eff}}) - mean(S_{F_3}^{Healthy})|\right]^{-1} \\
fit_3 &= \left[Var(S_{F_3}^{Healthy}) + Var(S_{F_3}^{LC_{flow}}) + Var(S_{F_3}^{HC_{flow}}) + Var(S_{F_3}^{LT_{eff}}) + Var(S_{F_3}^{LC_{eff}})\right]^{-1} \\
fit_4 &= \|numerical\ coefficients\| + \left[\|numericalcoefficients\|\right]^{-1} \\
fit &= fit_1 + fit_2 + fit_3 + fit_4
\end{aligned}
\tag{4.2.7}
$$

Fault index function F_4:

$$
\begin{aligned}
fit_1 &= |mean(S_{F_4}^{HT_{eff}}) - mean(S_{F_4}^{Healthy})| \\
fit_2 &= \left[|mean(S_{F_4}^{HC_{eff}}) - mean(S_{F_4}^{Healthy})|\right]^{-1} \\
fit_3 &= \left[Var(S_{F_4}^{Healthy}) + Var(S_{F_4}^{HT_{eff}}) + Var(S_{F_4}^{HC_{eff}})\right]^{-1} \\
fit_4 &= \|numerical\ coefficients\| + \left[\|numericalcoefficients\|\right]^{-1} \\
fit &= fit_1 + fit_2 + fit_3 + fit_4
\end{aligned}
\tag{4.2.8}
$$

Fault index function F_5:

$$
\begin{aligned}
fit_1 &= |mean(S_{F_5}^{LT_{flow}}) - mean(S_{F_5}^{Healthy})| \\
fit_2 &= \left[|mean(S_{F_5}^{HT_{flow}}) - mean(S_{F_5}^{Healthy})|\right]^{-1} \\
fit_3 &= \left[Var(S_{F_5}^{Healthy}) + Var(S_{F_5}^{LT_{flow}}) + Var(S_{F_5}^{HT_{flow}})\right]^{-1} \\
fit_4 &= \|numerical\ coefficients\| + \left[\|numericalcoefficients\|\right]^{-1} \\
fit &= fit_1 + fit_2 + fit_3 + fit_4
\end{aligned}
\tag{4.2.9}
$$

Fault index function F_6:

$$
\begin{aligned}
fit_1 &= |mean(S_{F_6}^{LC_{flow}}) - mean(S_{F_6}^{Healthy})| \\
fit_2 &= \left[|mean(S_{F_6}^{HC_{flow}}) - mean(S_{F_6}^{Healthy})|\right]^{-1} \\
fit_3 &= \left[Var(S_{F_6}^{Healthy}) + Var(S_{F_6}^{LC_{flow}}) + Var(S_{F_6}^{HC_{flow}})\right]^{-1} \\
fit_4 &= \|numerical\ coefficients\| + \left[\|numericalcoefficients\|\right]^{-1} \\
fit &= fit_1 + fit_2 + fit_3 + fit_4
\end{aligned}
\tag{4.2.10}
$$

Fault index function F_7:

$$
\begin{aligned}
fit_1 &= |mean(S_{F_7}^{LT_{eff}}) - mean(S_{F_7}^{Healthy})| \\
fit_2 &= \left[|mean(S_{F_7}^{LC_{eff}}) - mean(S_{F_7}^{Healthy})|\right]^{-1} \\
fit_3 &= \left[Var(S_{F_7}^{Healthy}) + Var(S_{F_7}^{LT_{eff}}) + Var(S_{F_7}^{LC_{eff}})\right]^{-1} \\
fit_4 &= \|numerical\ coefficients\| + \left[\|numericalcoefficients\|\right]^{-1} \\
fit &= fit_1 + fit_2 + fit_3 + fit_4
\end{aligned}
\tag{4.2.11}
$$

4.2.2.1 Threshold Definition

Following the discussion in Section 4.1.2 in order to define the threshold related to each residual, the severity of the least sensitive fault in the first class (class that the residual is sensitive to its faults) is reduced while at the same time the amount of fault in the most sensitive fault in the second class (class that residual is insensitive to its faults) is increased until the resulting residual value from the two faults converge to one value. This situation corresponds to the worst case scenario. Tables 4.6 and 4.7 summarize the determined thresholds and the minimum and maximum isolable faults for the residuals.

Fault indices	Threshold
$\mathbf{R_1}$	0.12
$\mathbf{R_2}$	0.25
$\mathbf{R_3}$	0.2
$\mathbf{R_4}$	1.4
$\mathbf{R_5}$	0.6
$\mathbf{R_6}$	0.3
$\mathbf{R_7}$	0.6

Table 4.6: Fault indices thresholds.

Fault residuals		Fault types							
		$\mathbf{HT_{eff}}$	$\mathbf{HC_{eff}}$	$\mathbf{LT_{flow}}$	$\mathbf{HT_{flow}}$	$\mathbf{LC_{flow}}$	$\mathbf{HC_{flow}}$	$\mathbf{LT_{eff}}$	$\mathbf{LC_{eff}}$
$\mathbf{R_1}$	**min**	0.3%	0.1%	0.5%	1%				
	max					8%	8%	9%	8%
$\mathbf{R_2}$	**min**	0.2%	0.1%						
	max			6%	6%				
$\mathbf{R_3}$	**min**					1%	0.5%		
	max							5%	6%
$\mathbf{R_4}$	**min**	0.2%							
	max		10%						
$\mathbf{R_5}$	**min**			0.5%					
	max				8%				
$\mathbf{R_6}$	**min**					0.4%			
	max						10%		
$\mathbf{R_7}$	**min**							0.5%	
	max								10%

Table 4.7: Isolation residuals minimum and maximum detectable faults. There are no limits for the blank cells.

4.2.2.2 Performance Evaluation (Confusion Matrix)

Similar to the detection part confusion matrix has been used to analyze the performance of the isolation algorithm. To evaluate the performance of the fault isolation in correctly isolating the faults a confusion matrix is constructed based on a series of simulations by using different sets of data corresponding to different operating conditions and different fault severities.

For each type of fault 6 fault severities were randomly selected between 0.1% to 10% and for each one 100 snapshots were generated using GSP software that resulted in 48 simulations. The selected fault severities for each type of fault were selected between the minimum and the maximum isolable fault severities as shown in Table 4.7. The resulting confusion matrix is shown in Table 4.8.

In this confusion matrix the t.p denotes the number of correct detection and isolation. f.p denotes the number of correct detection but faults isolation. f.p and f.n denote the number of faults not detected but correctly isolated and the number of faults not detected and not isolated correctly.

Confusion matrix	Accuracy (%)	Precision (%)	TPR (%)	FPR (%)	TNR (%)	FNR (%)
33 6 7 2	68	25	84	77	22	15

Table 4.8: Fault isolation confusion matrix.

To show the performance of the detection and isolation schemes two typical fault scenarios are shown here. In the first scenario it is assumed that a 2% fault in the low pressure compressor flow capacity LC_{flow} has occurred at the 50 data point. Figure 4.12 shows the detection residual. The fault is correctly detected since the residual has passed the detection threshold. Figures 4.13 to 4.15 show the residuals R_1, R_3 and R_6 responses. In the isolation step the first classifier residual R_1 has not responded to this fault and remained below its threshold (refer to Table 4.6). Therefore, the fault belongs to the right branch of the fault tree in Figure 4.4. In the second level the residual R_3 has responded to the fault which shows that the fault is either LC_{flow} or HC_{flow}, and finally at the last level R_6 has responded to the fault and has passed the threshold that implies that the detected fault is LC_{flow}.

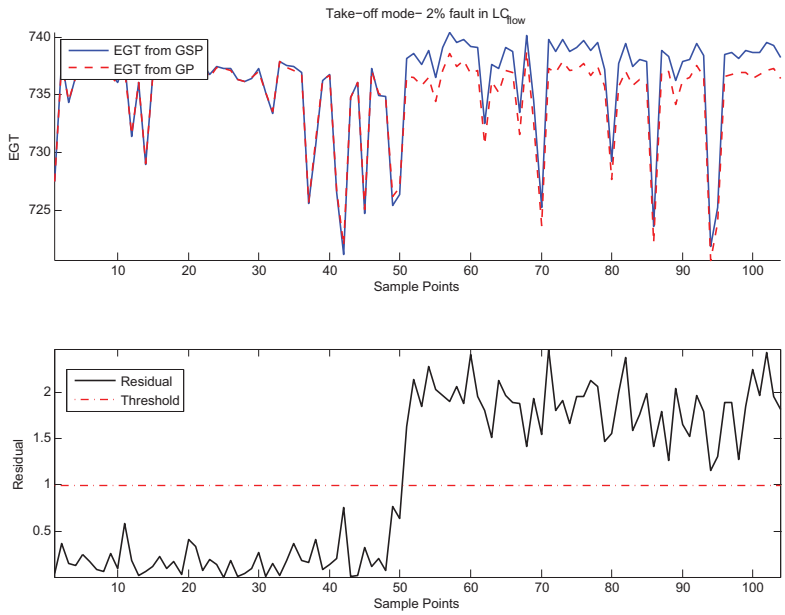

Figure 4.12: Detection of a 2% fault in low pressure compressor flow capacity. LC_{flow}

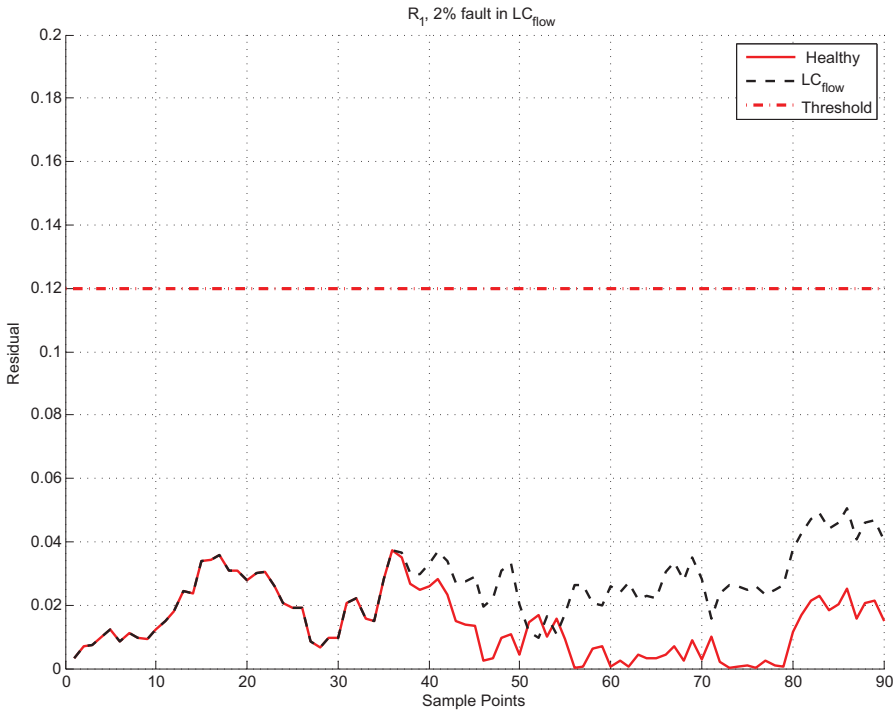

Figure 4.13: R_1 residual response to a 2% fault in LC_{flow}

.

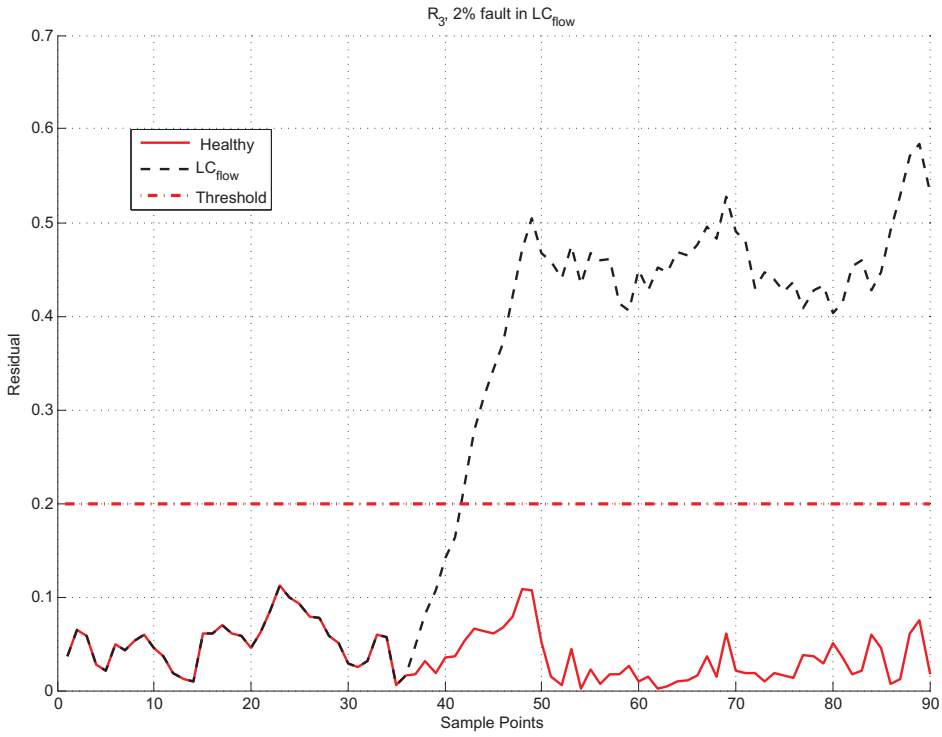

Figure 4.14: R_3 residual response to a 2% fault in LC_{flow}

.

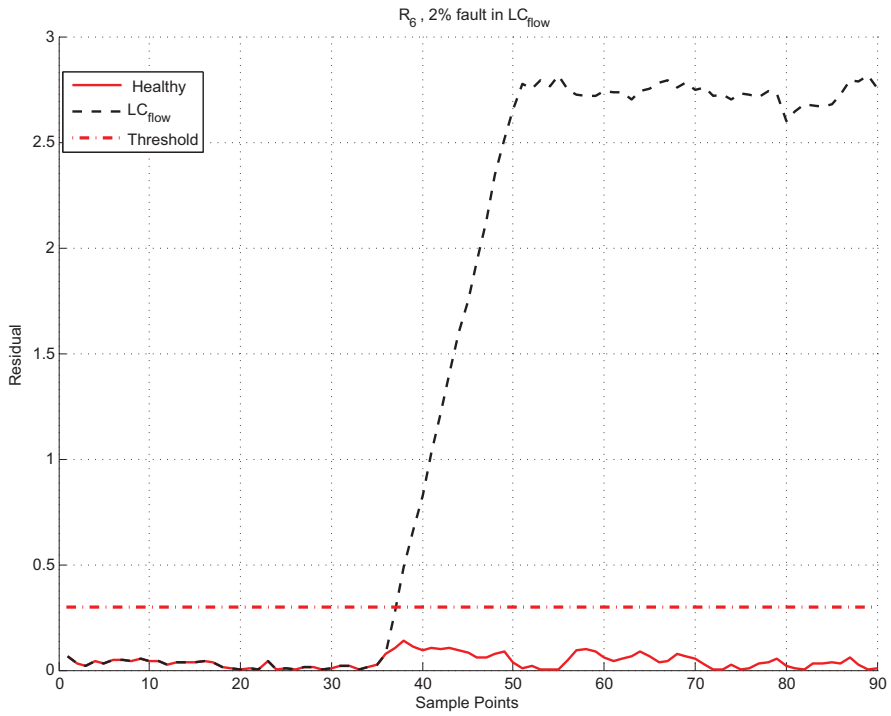

Figure 4.15: R_6 residual response to a 2% fault in LC_{flow}

.

The second scenario shows the case that a 0.5% fault has ocured in the high pressure turbine efficiency HT_{eff} at the 50 data point. Figure 4.16 shows the detection residual. The fault is correctly detected since the residual has passed the detection threshold. Figures 4.17 to 4.19 show the residuals R_1, R_2 and R_4 responses. In this case the R_1 has responded to this fault and has passed the corresponding threshold. It implies that the fault belongs to the left branch (Class1 in Figure 4.4). In the second level the residual R_2 has also responded to the fault which shows that the fault is either HT_{eff} or HC_{eff}, and finally at the last level R_4 has responded to the fault that implies that the detected fault is HT_{eff}.

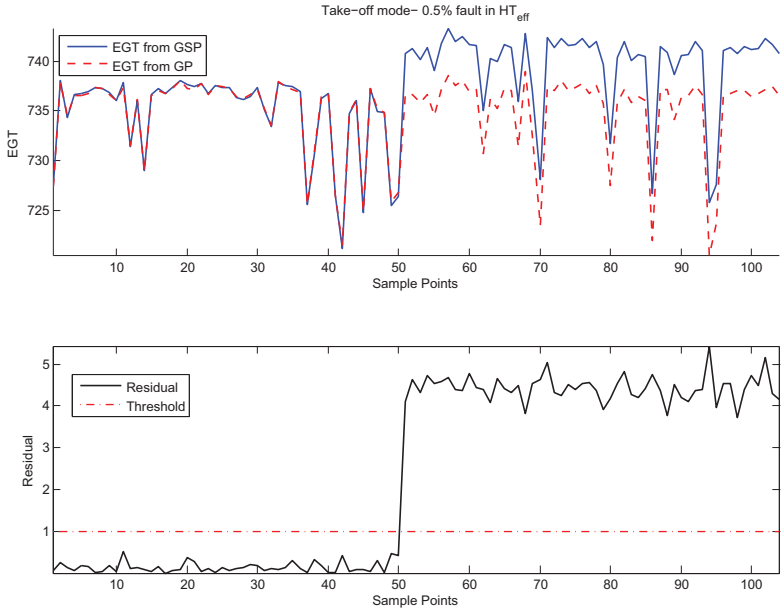

Figure 4.16: Detection of 0.5% fault in high pressure turbine efficiency HT_{eff}

.

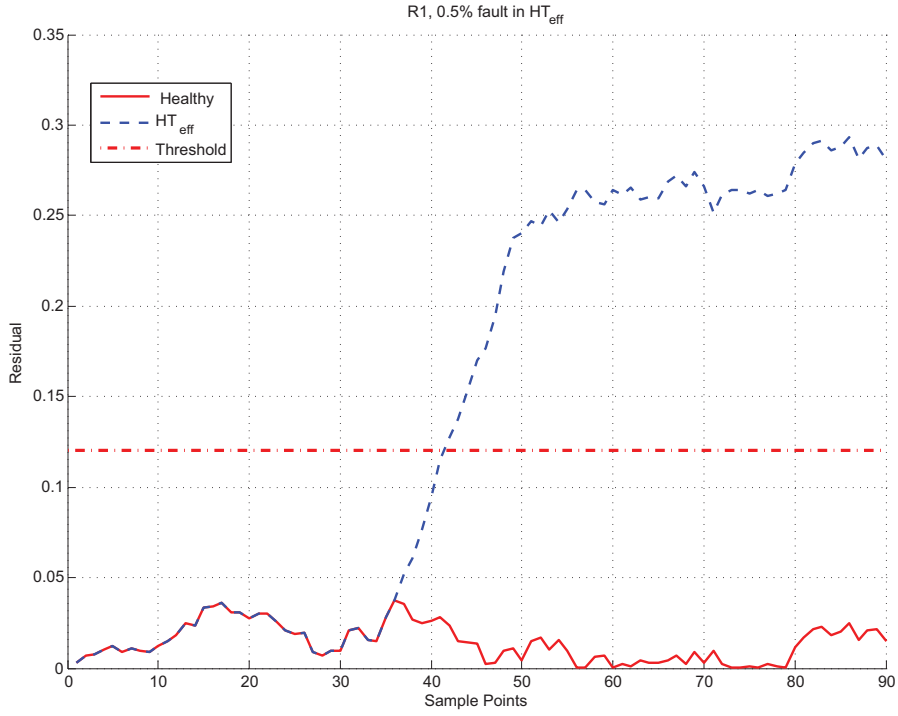

Figure 4.17: R_1 residual response to a 0.5% fault in HT_{eff}

.

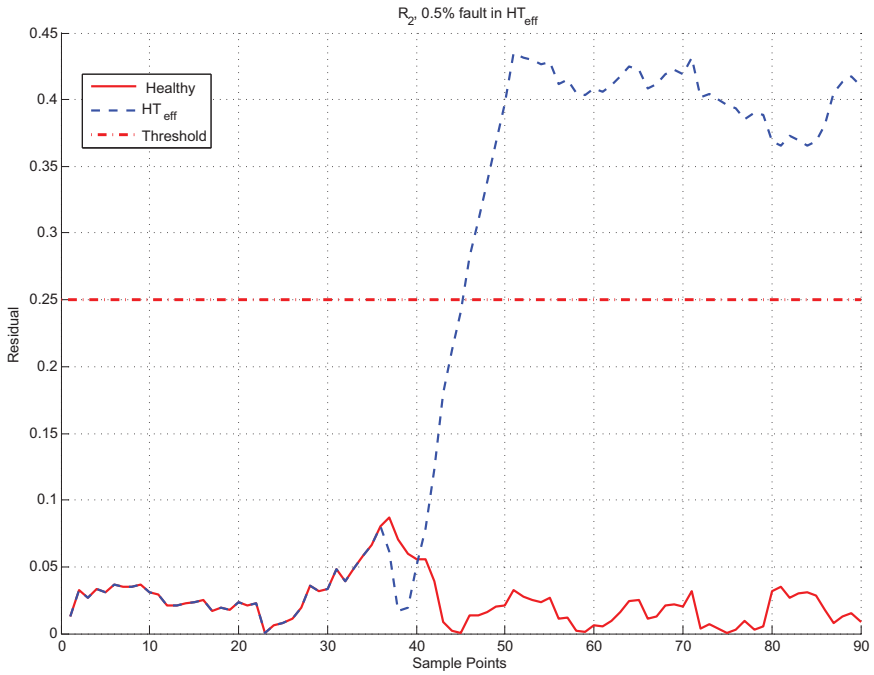

Figure 4.18: R_2 residual response to a 0.5% fault in HT_{eff}

.

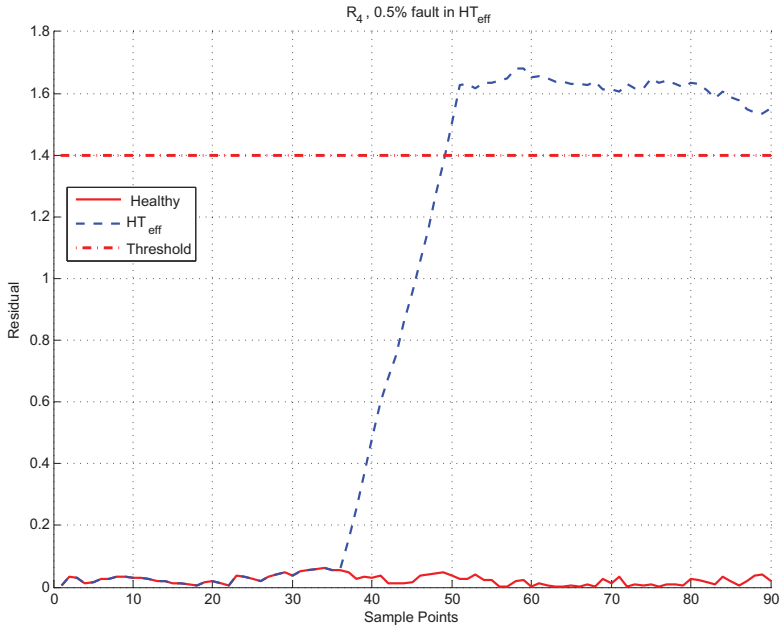

Figure 4.19: R_4 residual response to a 0.5% fault in HT_{eff}

.

4.3 Chapter Contributions

Fault isolation goal was carried out by developing a hierarchical approach in which a series of residuals are used step by step to eliminate the number of potential fault cases in the engine to ultimately converge to the correct fault type. These residuals attempt to amplify the signature of a fault in the engine by combining the effects of the fault on different engine parameters. Seven analytical expressions are obtained as isolation residuals to isolate the eight types of faults in a dual spool engine. Each of the residuals is optimized to have the maximum sensitivity to a class of faults and minimum sensitivity to the other faults.

4.4 Conclusions

In this chapter, a hierarchical fault isolation scheme was developed for the jet engine. In this approach a series of residuals are used to step by step eliminate the number

of potential faults in the engine to ultimately converge to the correct fault type in the engine. Each of the residuals has maximum sensitivity to the related detected fault and minimum sensitivity to the other faults. This approach provides an efficient and simple way to isolate faults. In contrast to other intelligent-based approaches such as neural networks where the models are basically black boxes in proposed method the residuals are analytical functions that can be inspected and analyzed with available engine mathematical models. On the other hand, by using this method we can increase the isolation efficiency of the scheme by combining the fault effects on all the engine variables.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

The main objective of this research was to combine model-based and intelligent-based approaches in the context of health monitoring of jet engines. Towards this end, genetic programming (GP) technique was used which has benefited from the capabilities of the genetic algorithm as a powerful data driven methodology and simultaneously provides analytical models. For the detection task, the GP algorithm was exploited to develop a set of static nonlinear models that relate the EGT as a major engine degradation indicator to the other engine parameters and operational conditions. Using this approach we were able to obtain mathematical models for the engine operation without any information about the engine component characteristics.

The resulting models were later used to detect abrupt faults in the engine performance. It was shown that with a few snapshots of the engine variables in flight we were able to develop a model that is able to estimate the engine EGT with errors less than 0.2% in the take-off and 2% in the cruise modes. Four mathematical models were presented for estimating the engine EGT in the take-off and cruise modes of the aircraft flight. As expected the faults in the cruise mode were less observable than

the faults in the take-off mode. Although the error between the generated model and the GSP output was small (less than 2%) the effects of faults were less than this modeling error. Fault isolation was carried out by developing a fault isolation tree in which a series of fault indices and corresponding residuals were introduced to step by step eliminate the number of potential faults in the engine to ultimately converge to the correct fault type. The residuals attempt to amplify the signature of a fault in the engine by combining the effects of the fault on the engine parameters. Seven analytical expressions were obtained as fault indices and residuals to isolate the eight types of faults in the dual spool engine.

5.2 Future Work

In this work the applicability of the GP technique in the off-line health monitoring of the aircraft engine was demonstrated by applying it to the simulated data using the GSP software. However, its capability in finding acceptable models needs to be verified with experimental data and real flight parameters.

The GP approach was implemented by using a set of fixed settings for the number of iterations and the operators probabilities. Investigating the effects of changing these parameters on the results is another aspect that needs more research. In addition, in this work basic mathematical operators were used for constructing the model structures. Another enhancement to current research could be to apply more complicated operators and functions to construct the models.

Provided models are static models and were developed by assuming that the engine has reached its steady state condition at its operating point. This assumption has limited the applicability of this approach to off-line analysis. The GP can be adjusted to be also able to model the engine dynamics. One way of doing this could be by considering the values of the engine parameters from the previous flights in the

modeling phase. This is also left as a topic of future research.

One limitation of the GP algorithm compared to neural networks is its less flexibility. Neural networks use a series of simple activation functions connected together using weight functions and has the ability to match itself to the training data by changing its number of neurons and weights. However the GP algorithm uses a defined set of mathematical operators and has less flexibility in changing its structure. To increase the efficiency and flexibility of the GP algorithm one can combine it with neural networks. It can be done by using neural networks as building blocks of the mathematical model of the engine in the GP instead of directly using engine states and operating conditions. At the low level a neural network uses parts of the engine states and operating conditions as input and produces an output. At the high level the GP algorithm can use these neural networks to construct different engine models by combining them using its defined mathematical operator set. This technique enhances the GP in finding more complicated models and at the same time reduces the complexity of the neural networks. It also provides the opportunity to better model the engine dynamics by using dynamical neural networks inside the building blocks.

Bibliography

[1] N. Daroogheh, A. Baniamerian, H. Nayyeri, and K. Khorasani, "Deterioration detection and health monitoring in aircraft jet engines," in *Proceedings of 2012 ASME International Mechanical Engineering Congress & Exposition,Houston, Texas, USA*, 2012.

[2] D. Malladi and J. Speyer, "A generalized shiryayev sequential probability ratio test for change detection and isolation," *IEEE Transactions on Automatic Control*, vol. 44, no. 8, pp. 1522 –1534, Aug 1999.

[3] V. Patel, V. Kadirkamanathan, G. Kulikov, V. Arkov, and T. Breikin, "Gas turbine engine condition monitoring using statistical and neural network methods," in *Modeling and Signal Processing for Fault Diagnosis (Digest No.: 1996/260), IEE Colloquium on*, sep 1996, pp. 1/1 –1/6.

[4] I. Hwang, S. Kim, Y. Kim, and C. Seah, "A survey of fault detection, isolation, and reconfiguration methods," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 3, pp. 636 –653, May 2010.

[5] Y. Zhang and J. Jiang, "Bibliographical review on reconfigurable fault-tolerant control systems," *Annual Reviews in Control*, vol. 32, no. 2, pp. 229 – 252, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1367578808000345

[6] R. Patton and J. Chen, "On eigenstructure assignment for robust fault diagnosis," *International Journal of Robust and Nonlinear Control*, vol. 10, no. 14, pp. 1193–1208, 2000.

[7] P. Frank and X. Ding, "Survey of robust residual generation and evaluation methods in observer-based fault detection systems," *Journal of Process Control*, vol. 7, no. 6, pp. 403–424, 1997.

[8] L. Van Eykeren, Q. Chu, and J. Mulder, "Sensor fault detection and isolation using adaptive extended kalman filter," in *Fault Detection, Supervision and Safety of Technical Processes*, vol. 8, no. 1, 2012, pp. 1155–1160.

[9] J. Chen, R. Patton, and H. Zhang, "Design of unknown input observers and robust fault detection filters," *International Journal of Control*, vol. 63, no. 1, pp. 85–105, 1996.

[10] J. Chen and R. Patton, "Robust residual generation using unknown input observers," *Robust Model-based Fault Diagnosis for Dynamic Systems*, pp. 65–108, 1999.

[11] D. Wang and K. Lum, "Adaptive unknown input observer approach for aircraft actuator fault detection and isolation," *International Journal of Adaptive Control and Signal Processing*, vol. 21, no. 1, pp. 31–48, 2007.

[12] J. Gertler, "Fault detection and isolation using parity relations," *Control Engineering Practice*, vol. 5, no. 5, pp. 653–661, 1997.

[13] R. Patton and J. Chen, "Review of parity space approaches to fault diagnosis for aerospace systems," *Journal of Guidance Control Dynamics*, vol. 17, pp. 278–285, 1994.

[14] H. Mohamed Basri, K. Lias, W. Wan Zainal Abidin, K. Tay, and H. Zen, "Fault detection using dynamic parity space approach," in *2012 IEEE International Conference on Power Engineering and Optimization (PEOCO)*, 2012, pp. 52–56.

[15] J. Stoustrup and H. H Niemann, "Fault estimationa standard problem approach," *International Journal of Robust and Nonlinear Control*, vol. 12, no. 8, pp. 649–673, 2002.

[16] I. Hwang, S. Kim, Y. Kim, and C. Seah, "A survey of fault detection, isolation, and reconfiguration methods," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 3, pp. 636–653, 2010.

[17] R. J. Patton and J. Chen, "Robust fault detection of jet engine sensor systems using eigenstructure assignment," *J. Guidance, Contr., Dyn.*, vol. 15, no. 6, pp. 1491 –1497, 1992.

[18] R. Patton, J. Chen, and H. Zhang, "Modelling methods for improving robustness in fault diagnosis of jet engine system," in *Decision and Control, 1992., Proceedings of the 31st IEEE Conference on*, 1992, pp. 2330 –2335 vol.2.

[19] J. Luo, M. Namburu, K. Pattipati, L. Qiao, M. Kawamoto, and S. Chigusa, "Model-based prognostic techniques [maintenance applications]," in *AUTOTESTCON 2003. IEEE Systems Readiness Technology Conference. Proceedings*, Sept. 2003, pp. 330 – 340.

[20] S. B. Johnson, T. Gormley, S. Kessler, C. Mott, A. Patterson-Hine, K. Reichard, and P. Scandura Jr, *System Health Management: With Aerospace Applications*. Wiley, 2011, vol. 34.

[21] R. Patton, J. Chen, and H. Y. Zhang, "Modelling methods for improving robustness in fault diagnosis of jet engine system," in *Decision and Control, 1992., Proceedings of the 31st IEEE Conference on*, 1992, pp. 2330–2335 vol.2.

[22] S. Ofsthun and T. Wilmering, "Model-driven development of integrated health management architectures," in *Aerospace Conference, 2004. Proceedings. 2004 IEEE*, vol. 6, March 2004, pp. 3692 – 3705.

[23] A. Babbar and V. Syrmos, "Data driven approach for fault detection and identification using competitive learning techniques," in *European Control Conf.*, 2007.

[24] X. Yin, J. He, , and Z. Zhou, "Using neural network for fault diagnosis," in *International Joint Conference on Neural Networks, Como, Italy*, July 2000, pp. 217–220.

[25] G. Betta, C. Liguori, and A. Pietrosanto, "An advanced neural-network-based instrument fault detection and isolation scheme," *IEEE Transactions on Instrumentation and Measurement*, vol. 47, no. 2, pp. 507 –512, Apr 1998.

[26] J. Gertler, *Fault detection and diagnosis in engineering systems*. Marcel Dekker, 1998.

[27] Y. Li, "Performance-analysis-based gas turbine diagnostics: A review," *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, vol. 216, no. 5, pp. 363–377, 2002.

[28] L. Urban, "Gas path analysis applied to turbine engine condition monitoring," in *Proceedings of the AIAA/SAE 8th Joint Propulsion Specialist Conference, 72-1082*. AIAA No., 1991, pp. 72–1082.

[29] A. Volponi, "Foundations of gas path analysis, parts i & ii," in *Lecture Notes for the Von Karman Institute Lecture Series*, Jan. 2003, pp. 13–17.

[30] T. Kobayashi and D. L. Simon, "Application of a bank of kalman filters for aircraft engine fault diagnostics," in *NASA Report 212526*, 2003.

[31] E. Naderi, N. Meskin, and K. Khorasani, "Nonlinear fault diagnosis of jet engines by using a multiple model-based approach." ASME, 2011.

[32] D. Simon, "A comparison of filtering approaches for aircraft engine health estimation," *Aerospace Science and Technology*, vol. 12, no. 4, pp. 276 – 284, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1270963807000818

[33] R. Joly, S. Ogaji, R. Singh, and S. Probert, "Gas-turbine diagnostics using artificial neural-networks for a high bypass ratio military turbofan engine," *Applied energy*, vol. 78, no. 4, pp. 397–418, 2004.

[34] S. Simani and C. Fantuzzi, "Fault diagnosis in power plant using neural networks," *Information Sciences*, vol. 127, no. 3, pp. 125–136, 2000.

[35] C. Romessis, K. Mathioudakis *et al.*, "Bayesian network approach for gas path fault diagnosis," *Journal of Engineering for Gas Turbines and Power(Transactions of the ASME)*, vol. 128, no. 1, pp. 64–72, 2006.

[36] M. Zedda and R. Singh, "Fault diagnosis of a turbofan engine using neural networks: a quantitative approach," in *The 34th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit,(Cleveland, OH)*, 1998.

[37] G. Torella and G. Lombardo, "Utilization of neural networks for gas turbine engines," in *ISABE- International Symposium on Air Breathing Engines, 12 th, Melbourne, Australia*, 1995, pp. 358–366.

[38] J. D. Addison, S. Wermter, and J. MacIntyre, "Effectiveness of feature extraction in neural network architectures for novelty detection," in *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, vol. 2. IET, 1999, pp. 976–981.

[39] L. Marinai, D. Probert, and R. Singh, "Prospects for aero gas-turbine diagnostics: a review," *Applied energy*, vol. 79, no. 1, pp. 109–126, 2004.

[40] D.-H. Seo, T.-S. Roh, and D.-W. Choi, "Defect diagnostics of gas turbine engine using hybrid svm-ann with module system in off-design condition," *Journal of mechanical science and technology*, vol. 23, no. 3, pp. 677–685, 2009.

[41] R. Mohammadi, S. Hashtrudi-Zad, and K. Khorasani, "Hybrid fault diagnosis: Application to a gas turbine engine." ASME, 2009.

[42] A. J. Volponi, H. DePold, R. Ganguli, and C. Daguang, "The use of kalman filter and neural network methodologies in gas turbine performance diagnostics: a comparative study," *Journal of engineering for gas turbines and power*, vol. 125, no. 4, pp. 917–924, 2003.

[43] A. Babbar, V. Syrmos, E. Ortiz, and M. Arita, "Advanced diagnostics and prognostics for engine health monitoring," in *Aerospace conference, 2009 IEEE*, March 2009, pp. 1 –10.

[44] V. Morgenstern, B. Upadhyaya, and M. Benedetti, "Signal anomaly detection using modified cusum method," in *Decision and Control, 1988., Proceedings of the 27th IEEE Conference on*, Dec 1988, pp. 2340 –2341 vol.3.

[45] A. Willsky and H. Jones, "A generalized likelihood ratio approach to the detection and estimation of jumps in linear systems," *IEEE Transactions on Automatic Control*, vol. 21, no. 1, pp. 108 – 112, Feb 1976.

[46] M. Imhoff, M. Bauer, U. Gather, and D. Lohlein, "Statistical pattern detection in univariate time series of intensive care on-line monitoring data," *Intensive Care Med.*, vol. 24, pp. 1305–1314, 2004.

[47] N. Ramirez-Beltran and J. Montes, "Neural networks for on-line parameter change detection in time series model," *Computer and Industrial Engineering*, vol. 33, pp. 337–340, 1997.

[48] K. Kumar and B. Wu, "Detection of change points in time series analysis with fuzzy statistics," *International Journal of System Science*, vol. 32, pp. 1185–1192, 2001.

[49] H. X., H. Qiu, and N. Iyer, "Multivariate change detection for time series data in aircraft engine fault diagnostics," in *IEEE International Conference on Systems, Man and Cybernetics, 2007. ISIC.* IEEE, 2007, pp. 2484–2489.

[50] A. Gulati, D. Taylor, and R. Singh, "Multiple operating point analysis using genetic algorithm optimization for gas turbine diagnostics," in *ISOABE, ISABE-International Symposium on Air Breathing Engines, 15 th, Bangalore, India*, 2001.

[51] M. Zedda and R. Singh, "Gas turbine engine and sensor fault diagnosis using optimization techniques," *Journal of Propulsion and Power*, vol. 18, no. 5, pp. 1019–1025, 2002.

[52] M. Witczak, A. Obuchowicz, and J. Korbicz, "Genetic programming based approaches to identification and fault diagnosis of non-linear dynamic systems," *International Journal of Control*, vol. 75, no. 13, pp. 1012–1031, 2002. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/00207170210156224

[53] R. Patton, P. Frank, and R. Clark, *Issues of fault diagnosis for dynamic systems*. Springer, 2000.

[54] O. Nelles, *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*. Springer, 2000.

[55] G. Gray, D. J. Murray-smith, Y. Li, and K. C. Sharman, "Nonlinear model structure identification using genetic programming," vol. 6, pp. 1341–1352, 1996.

[56] J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.

[57] C. Banks, "Searching for lyapunov functions using genetic programming," *Virginia Polytech Institute, unpublished*, 2004.

[58] J. McGough, A. Christianson, and R. Hoover, "Symbolic computation of lyapunov functions using evolutionary algorithms," in *Proceedings of the 12th IASTED International Conference*, vol. 15, 2010, p. 17.

[59] K. Bettenhausen, P. Marenbach, S. Freyer, H. Rettenmaier, and U. Nieken, "Self-organizing structured modelling of a biotechnological fed-batch fermentation by means of genetic programming," in *Genetic Algorithms in Engineering Systems: Innovations and Applications,*, Sep 1995, pp. 481 –486.

[60] P. Marenbach and K. Bettenhausen, "Signal path oriented approach for generation of dynamic process models," in *Proc. 1st Anual Conf. on Genetic Programming*, 1996, pp. 327 –332.

[61] A. Alcázar and K. Sharman, "Some applications of genetic programming in digital signal processing," in *Late Breaking Papers at the Genetic Programming 1996 Conference Stanford University*. Citeseer, 1996, pp. 24–31.

[62] G. Olague and L. Trujillo, "Evolutionary-computer-assisted design of image operators that detect interest points using genetic programming," *Image Vision Comput.*, vol. 29, no. 7, pp. 484–498, Jun. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.imavis.2011.03.004

[63] J. Madár, J. Abonyi, and F. Szeifert, "Genetic programming for the identification of nonlinear input-output models," *Industrial & engineering chemistry research*, vol. 44, no. 9, pp. 3178–3186, 2005.

[64] W. Weimer, T. Nguyen, C. Le Goues, and S. Forrest, "Automatically finding patches using genetic programming," in *Proceedings of the 31st International Conference on Software Engineering*, ser. ICSE '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 364–374. [Online]. Available: http://dx.doi.org/10.1109/ICSE.2009.5070536

[65] A. Shintemirov, W. Tang, and Q. Wu, "Power transformer fault classification based on dissolved gas analysis by implementing bootstrap and genetic programming," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 39, no. 1, pp. 69 –79, Jan. 2009.

[66] Z. Zhang, K. Fang, and W. Huang, "A genetic programming based fuzzy model for fault diagnosis of power transformers," in *Intelligent Networks and Intelligent Systems (ICINIS), 2010 3rd International Conference on.* IEEE, 2010, pp. 455–458.

[67] W. Visser and M. Broonhead, "Gsp, a generic object-oriented gas turbine simulation environment," *ASME TURBO EXPO 2000, Munich, Germany*, 2000.

[68] R. Isermann, *Fault-diagnosis systems: an introduction from fault detection to fault tolerance.* Springer, 2005.

[69] M. Goosens, F. Mittelbach, and A. Samarin, *Modelling and estimation strategies for fault diagnosis of non-linear systems. Lecture notes in control and information sciences*. Berlin: Springer, 2007.

[70] K. Hunecke, *Jet engines: fundamentals of theory, design and operation*. Zenith Press, 1997.

[71] "http://upload.wikimedia.org/wikipedia/commons/7/75/turbofan_operation .svg," accessed on 28/1/2013.

[72] Z. Husain, *Air Breathing Engines*. IK International Pvt Ltd, 2010.

[73] D. Shanno *et al.*, "Conditioning of quasi-newton methods for function minimization," *Mathematics of computation*, vol. 24, no. 111, pp. 647–656, 1970.

[74] "http://sounak4u.weebly.com/gas-power-cycle.html," accessed on 28/1/2013.

[75] *GSP user manual*. National Aerospace Laboratory NLR Anthony Fokkerweg, 2 1006 BM Amsterdam The Netherlands, January 2013.

[76] G. Olague and L. Trujillo, "Evolutionary-computer-assisted design of image operators that detect interest points using genetic programming," *Image and Vision Computing*, vol. 29, pp. 484–498, June 2011.

[77] K. D. Jong, *Evolutionary Computation: A Unified Approach*. Cambridge, MA: MIT Press, 2001.

[78] R. Poli, W. Langdon, and N. McPhee, *A field guide to genetic programming*. Lulu Enterprises Uk Limited, 2008.

[79] W. Press, T. S., V. W., and F. B., *Numerical Recipes in C*. Cambridge University Press., 1992.

[80] J. Lagarias, J. Reeds, M. Wright, and P. Wright, "Convergence properties of the nelder–mead simplex method in low dimensions," *SIAM Journal on Optimization*, vol. 9, no. 1, pp. 112–147, 1998.

[81] R. Lewis, V. Torczon, and M. Trosset, "Direct search methods: then and now," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1, pp. 191–207, 2000.

[82] S. Singer and J. Nelder, "Nelder-mead algorithm," vol. 4, no. 2, p. 2928, 2009.

[83] "Gas turbine performance simulation," *National Aerospace Laboratory NLR*, 2012.

[84] W. Visser, O. Kogenhop, and M. Oostveen, "A generic approach for gas turbine adaptive modeling," *Proceedings of ASME Turbo Expo. Viena*, 2004.

[85] V. Klee and G. J. Minty, "How good is the simplex algorithm?" in *Inequalities, III (Proc. Third Sympos., Univ. California, Los Angeles, Calif., 1969; dedicated to the memory of Theodore S. Motzkin)*. New York: Academic Press, 1972, pp. 159–175.

[86] N. Meskin and K. Khorasani, *Fault Detection and Isolation: Multi-Vehicle Unmanned Systems*. Springer, 2011.

[87] G. Box, J. Hunter, and W. Hunter, *Statistics for experimenters: design, innovation, and discovery*. Wiley Online Library, 2005, vol. 13.