2011

Multivariable Sliding Mode Control Design for Aircraft Engines

Sirirat Sangwian
Cleveland State University

MULTIVARIABLE SLIDING MODE CONTROL DESIGN FOR AIRCRAFT ENGINES

SIRIRAT SANGWIAN

Bachelor of Science in Mechanical Engineering

Cleveland State University, Cleveland, Ohio

August 2009

submitted in partial fulfillment of requirements for the degree

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

at the

CLEVELAND STATE UNIVERSITY

August 2011

This thesis has been approved
for the department of MECHANICAL ENGINEERING
and the College of Graduate Studies by:

_____

Thesis Chairperson, Hanz Richter, Ph.D.

_____

Department & Date

_____

Jerzy Sawicki, Ph.D.

_____

Department & Date

_____

Lili Dong, Ph.D.

_____

Department & Date

ACKNOWLEDGMENTS

MULTIVARIABLE SLIDING MODE CONTROL DESIGN FOR AIRCRAFT
ENGINES

SIRIRAT SANGWIAN

ABSTRACT

Many control theories are used in controlling aircraft engines. However, the multi-variable sliding mode control is not yet established in this application even though it has a lot of potential in dealing with complex and nonlinear systems such as aircraft engines. Therefore, a guideline in developing multivariable sliding mode control law for an aircraft engine is presented in this thesis. The problem of chattering in the sliding mode control is suppressed by the use of the boundary layer method. The control logic is tested by implementing NASA's Commercial Modular Aero-Propulsion System Simulation 40k (C-MAPSS40k). Simulation results are analyzed and compared to the results obtained from the baseline controller. The robust property of multivariable sliding mode control is also examined by altering the flight condition of the engine.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

Aircraft engines are complex dynamic systems with uncertain parameter variations, therefore multivariable sliding mode control is an excellent candidate to utilize as a control law for these systems. This thesis establishes practical procedures for designing multivariable sliding mode control and applies them to a turbofan engine, which is widely used in commercial aircrafts due to its efficiency and quieter operation. In addition, by utilizing the multivariable sliding mode control in creating the control law for an aircraft engine, the horizon of the control theory for aircraft engines is expanded. In this first chapter, the motivation behind the thesis is discussed along with a brief history of aircraft engine controllers. The scope of the thesis is also presented at the end of the chapter.

1.1 Motivation

Technology in aircraft engines has significantly improved from the first flight of Gloster E28/39 in 1941 to the current world's most powerful commercial jet engine GE90. The Gloster E28/39 used a Whittle engine, known as the W1, generating

850 lb-thrust, while the GE90 is recognized by the Guinness Book of World Records producing thrust at 127,900 lb-thrust [1]. With the engine technology continuously progressing, developments in aircraft engine control needs to be on par with the advance in engine technology. New engines are more complex and powerful, therefore, the engine control architecture needs to be more robust and versatile.

Current research in control theories used in aircraft engine control are focusing on multivariable control, also known as multi-input multi-output (MIMO) control. Many MIMO control methods have been utilized in controlling the aircraft engines, such as: output feedback regulator (OFR), linear quadratic Gaussian (LQG), H-infinity, and inverse Nyquist array(INA). However, a control technique such as sliding mode control, which has a lot of potential in dealing with nonlinear and complex system like aircraft engine, has been received less attention. Therefore, this thesis is developing a controller based on multivariable sliding mode control for an aircraft engine. The sliding mode controller is then tested using the turbofan engine simulation created by National Aeronautics and Space Administration (NASA).

1.2 Aircraft Engine Controller

The main function of the engine controller is to be able to accurately control the amount of thrust with respect to the position of the throttle regardless of changes in the operating environment. Other than controlling the amount of thrust, the engine controller can be used to "schedule engine geometry and augmentor fuel, provide fan and booster stall protection, control variable parasitic engine flows, improve integrated engine-airframe performance, and provide engine health monitoring and diagnostics" [2]. Sometimes, the monitoring part is separated into its own structure called a monitoring system. An aircraft engine has to be reliable since a mistake or a fault can be the difference between life and death as well as hundreds to hundreds of

thousands of dollars. Therefore, the monitoring role is as important as the controlling role. It needs to be able to detect failure in the system and compensate for that failure accordingly. The monitoring system also requires to predict an engine life or engine component life before the engine or its components are no longer in acceptable operating conditions. The major benefit of an engine controller is apparent when unexpected problems arise, fixing or tuning the controller is always an easier solution to resolve the problems compared to physically fixing, reinstalling, or redesigning the engine parts.

Designing engine controllers is not an easy task and it has undergone many changes over the past decades. Aircraft engines can vary from engine to engine due to manufacturing processes and a mismatch in engine components. Even the same engine can change over time due to deterioration. As a consequence, engine controllers need to be able to adapt and cope with each engine's characteristics and deal with the slow change in engine performance due to deterioration, a sudden shift in performance due to the operating environment, or after maintenance. In an early stage of engine control development (1940s), engine controllers were initially created using mostly mechanical parts such as cams, gear trains, linkages, and valves. These controllers then became hydromechanical controllers (HMC) when hydraulic fluid was added to the system. The control algorithm is limited due to the constraints in the mechanical parts, which hinders the development of HMC to keep up with engine technologies. In order to create more sophisticated HMC, more parts were added to the system to accommodate its complexity. As a result, HMC became bigger and heavier, which did not favor the engine design. Having more components also increased the operating and maintenance cost of the controllers. Therefore, HMC had slowly reached its limit as an aircraft engine controller. With the advances in electronic circuit technology, engine controllers have evolved and became more computerized. As a result, controller logics grew in complexity which improved controller performance and effi-

ciency, since the engine is able to operate closer to its limitations. In other words, the efficiency of the engine is greatest when the engine operates at or near its maximum limitations. Furthermore, the electronic controller also has longer service life, higher precision, higher reliability, lower weight, easier backup, and lower operating cost of the aircraft. In addition, HMC is used as a back up in some electronic controllers in case of a failure or pilot override. The electronic controllers are developed based on the specifications and applications of the aircrafts.

In the early 1970s, electronic aircraft engine controllers were developed first in analog and later in digital. These are called electronic control units (ECU). If the ECU "controls the entire operation of an engine from startup to shutdown for the engine thrust (or power)" [3], the ECU is considered to have "full authority". Over the past decades, the ECU has grown and expanded to become a standard control system for jet engines. Many functions have been added to the control to improve engine performance and diagnostics. The weight and size of controllers are also lighter and smaller due to improvement in microprocessor technologies. Common ECU are digital engine controls (DEC) and full authority digital electronic controls (FADEC), created by Pratt and Whitney, and General Electric, respectively. Moreover, the FADEC does not have an override option for the pilot. All of the engine parameters are controlled by the FADEC. Therefore, FADEC are designed with many redundancies because in case of a total FADEC failure, the pilot will have no means to control or restart the engine.

This thesis is in a developmental phase of the aircraft engine controller, where control architecture is established and analyzed in a non-real time simulation. The next step after the controller has been perfected is to create the source code for the control logic. The source code is then compiled and downloaded onto a control hardware. Finally, the control law is tested with an actual engine in a real time setting.

1.3 Scope of Thesis

This thesis focuses on developing a guideline to establish a multivariable sliding mode controller for an aircraft engine. To validate the developed control logic, the sliding mode controller is implemented into a turbofan engine simulation created by NASA called Commercial Modular Aero-Propulsion System Simulation 40k (C-MAPSS40k) [16]. The engine performance of the multivariable sliding mode controller is compared to the engine performance of the default baseline controller that comes with the C-MAPSS40k. Sliding mode control is known for its robustness, so the controller should be able to handle different types of engine operating environment. To evaluate the robust property of the controller, the engine flight condition is altered.

There are six chapters in this thesis. Chapter 2 reviews the basic knowledge of aircraft engines, where aircraft components are discussed along with type of aircrafts, engine maintenance, and the standard practice of engine station numbering. Chapter 3 presents a brief history of control methods used in aircraft engine controls, including engine characteristics and two instabilities of the engine: stall and surge. Chapter 3 also contains general information and function of C-MAPSS40k. In chapter 4, the concept of sliding mode control is introduced and organized in developing the control law based on which sliding mode control is established. The problem of chattering in sliding mode control is also discussed. Note that the control law is created in the form of state space, where there are three control inputs and three control outputs. Hence, the control law is for a MIMO system and to reduce wordiness, the prefix multivariable is dropped and the control method will be mentioned as sliding mode control. Chapter 5 applies the sliding mode controller developed in chapter 4 to a turbofan engine and implements the controller into the C-MAPSS40k. The simulation results are then discussed and compared to the results obtained from a baseline controller, which comes with the C-MAPSS40k. The robustness of the sliding mode controller is also

tested by altering the engine operating mode. To do so, the flight condition of the engine is changed. Chapter 6 concludes this thesis paper with the summary of the thesis, along with recommendation for the future work in the same subject.

CHAPTER II

AIRCRAFT ENGINES

2.1   Aircraft Engines

Aircraft engines are fascinating machines, reflecting high levels of craftsmanship, materials, and technology of the time period when they were developed. Selecting the right kind of engine to suit the aircraft depends upon the operating requirements, configuration, weight, dependability, and cost of the overall designed aircrafts. Furthermore, to design a suitable engine, "the intended application or mission, the level of technology available, the degree of subsequent growth capability required, and expected competition from other engines" need to be taken in to consideration [2]. To work with aircraft engines, one must have a strong understanding of their components and operations. As a result of these intricacies, this chapter explains functions of standard engine components, along with basic types of aircraft engines. In addition, description of specific fuel consumption, engine maintenance, and the numbering of aircraft engine station are also included in this chapter. Knowledge of these subjects is beneficial in designing and simulating the control law for the aircraft engine.

2.2 Aircraft Engine Components

Each of the engines are different in configuration, but have some common parts such as the inlet, a compressor, a combustor, a turbine, and a nozzle.

2.2.1 Inlet

The engine inlet is located in front of the compressor, where it guides the air into the engine. Since thrust, F, is dependent on the mass flow rate from $F = \dot{m}(u_e - u_0)$ where $\dot{m}$ is the inlet mass flow rate, u_0 is the flight velocity, and u_e is the exhaust velocity, it is obvious that the inlet has a significant effect on the engine net thrust. Designing the inlet is based upon the operating speed. Therefore, inlets are classified into three types according to the operating speed of the aircraft, which are the subsonic inlets, supersonic inlets, and hypersonic inlets.

2.2.2 Compressor

Once the air passes through the inlet, it is then compressed in the compressor to increase the pressure of the air before being mixed with the fuel and ignited in the combustor. Compressors come in two types: axial compressors and centrifugal compressors. The main difference between these compressors is the path that air passes through them. Air travels parallel to the axis of the rotation in the axial compressor, while it travels perpendicular to the axis of rotation in the centrifugal compressor. The same type of compressors can also be joined together to create a multistage compressor, resulting in higher level of compression. The axial compressor is more popularly used as a multistage compressor since it is easy to link them together, due to the parallel path of the air to the axis of the rotation. On the other hand, centrifugal compressors need a passage for the air to be ducted back to the axis of the compressor at each stage which complicates the system and reduces the efficiency of

the compressors. In addition, the extra passages of the air for each multistage centrifugal compressor also cause higher aircraft drag due to an increase in the engine's size. There are also some undesired characteristics that can occur at the compressor which can affect an entire engine. These instabilities are surge and stall which will be discussed more in chapter 3.

2.2.3 Combustor

After the air is pressurized, it is then combined with fuel in the combustor. The combustor is sometime called a burner, since the air-fuel mixture is ignited in the combustor. "The overall mixture of fuel and air varies over a wide range with changing engine speed, and the compressor outlet air varies in pressure and temperature with altitude and flight Mach number as well as with engine speed" [5]. The combustor consists of a case, diffuser, liner, snout, dome, fuel injector, and igniter. There are three types of combustors based on the configurations of the combustion chambers: annular, can, and can-annular. The annular is the most common type, widely used in modern aircraft engines due to its better uniform combustion, uniform exit temperature, lighter weight, and lower pressure drop compared to the other types of combustors.

2.2.4 Turbine

The exhaust gas exits from the combustor, passing through the turbine which coexists on the same shaft as the compressor. The turbine extracts energy from the exhaust gas and uses that energy to primarily drive the compressor. In some engines, this energy is also utilized to power fuel pumps, oil pumps, hydraulic pump, and similar equipment. There are two common configurations for the connection between the turbine and the compressor in the aircraft engines, which are a single-spool also known as a single-shaft, and a two-spool. The single-spool structure has only one

turbine and one compressor. On the other hand, the two-spool has two turbines positioned on two separate shafts. One turbine is a high pressure turbine attached to a high pressure compressor and the other turbine is a low pressure turbine connected to a low pressure compressor. In some rare cases, a third turbine is added to the system and the engine is called a three-spool engine. Overall, the two-spool engine has a better thermodynamic efficiency compared to the single-spool engine due to the increase in the overall pressure ratio, which is the combustor inlet pressure divided by the intake delivery pressure. On the other hand, the three-spool possesses a higher cost compared to other engine types and presents a designing challenge due to its complexities. As a consequence, the two-spool engine is a better choice and widely used in the industries.

2.2.5 Nozzle

Positioned after the turbine is the nozzle, where exhaust gas is released back to the atmosphere, producing thrust. The nozzle is a shaped tube which can come in different sizes and shapes depending on the application of the aircraft. The common types of nozzles are co-annular, convergent, and convergent-divergent. The co-annular nozzle has two separate sections: the center nozzle and the annular nozzle. The exhaust gas flows through the center nozzle while the fan flow is discharged through the annular nozzle. The fan flow is the air ducted into the engine that did not go through the combustion process. It bypasses the core engine and then merges with the exhaust gas at the nozzle. Doing so improves thrust efficiency and reduces exhaust noise. The convergent nozzle is the reduced model of the co-annular nozzle, having only the center nozzle where all of the exhaust gas is converged by the shape of the nozzle before being released back into the atmosphere. Lastly, the convergent-divergent nozzle is fairly self-explanatory. The first section of the nozzle is exactly like the convergent nozzle where the gas is converged and then diverged by the second

section of the nozzle. With this arrangement, the flow of the exhaust gas is able to obtain supersonic speeds. The smallest cross section area of this type of nozzle is called the throat, which is located between the first section and the second section of the nozzle. The throat plays an important role in determining the amount of thrust being produced by the nozzle by setting the mass flow rate, the exit velocity, and the pressure at the exit for the engine.

2.3   Type of Aircraft Engines

A basic aircraft engine can be created with an inlet, compressor, combustor, turbine, and nozzle. However, with demands in a wide range of applications and environments, aircraft engines come in many shapes, sizes, and configurations, which can be categorized into four general types: ramjet, turbojet, turbofan, and turboprop.

2.3.1   Ramjet

The ramjet is the most basic form of a jet engine [5]. It only consists of an inlet, a combustor, and a nozzle, as shown in Figure 2.1. Ramjet is also called scream jet due to its very loud operating noise. Since ramjet does not have a compressor section, it relies on the speed of the aircraft to compress the incoming air when it rams into the engine, hence the name ramjet. Ramjets cannot produce thrust at zero speed and therefore, a stationary aircraft needs an alternative propulsion system to initially move the aircraft. As a result, ramjets are rarely used in a commercial aspect. Examples of ramjets are missiles and rockets.

2.3.2   Turbojet

The turbojet is a ramjet with an additional compressor and a turbine as shown in Figure 2.2. The efficiency of the ramjet is increased as a result of the air pressure
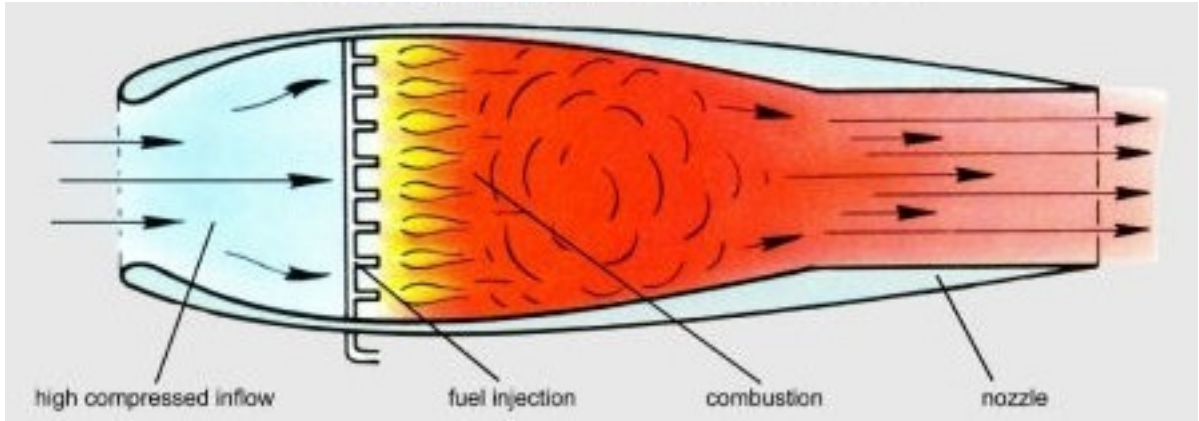
Figure 2.1: Schematic Diagram of a Ramjet Engine (from [6])

and air temperature rise before the combustion process. Additionally, turbojets can be arranged in single-spool or two-spool configurations. In the early days, turbojets used centrifugal compressors initially, but soon after, axial compressors were used to obtain smaller engine sizes. Turbojets were developed and used primarily as war planes in World War II. Nowadays, turbojets are being replaced by turbofans which have better fuel efficiency and less operating noise.
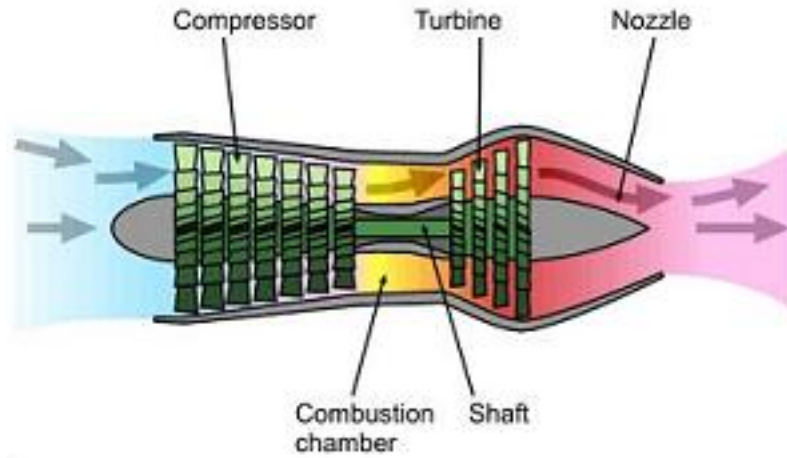
Figure 2.2: Schematic Diagram of a Turbojet Engine (from [7])

2.3.3 Turbofan

The turbofan improves propulsion efficiency of the turbojet by directing part of the intake air to bypass the core engine and adding a fan in front of the compressor as shown in Figure 2.3. The bypass air then joins with the exhausted gas from the core engine, or expands through a separate nozzle. The bypass ratio is the bypass airflow divided by the core airflow. The higher the bypass ratio is, the higher the energy that goes into the bypass section is, decreasing the mean jet velocity and enhancing the propulsive efficiency. A turbofan also has a lower exhaust velocity compared to a turbojet, resulting in less required fuel flow which means lower fuel consumption (i.e., more fuel efficiency). A turbofan can be classified into two types using the bypass ratio. The first type is low-bypass turbofan and the second type is high-bypass turbofan. The low-bypass turbofan has low bypass ratio and usually has multistage fan which creates higher exhaust gas velocity. Therefore, it has a high specific thrust which is defined as net thrust divided by intake airflow. A low-bypass turbofan has quieter noise when compared to a turbojet, but is still considered to be loud compared to the high-bypass turbofan, resulting in its high presence in fighter airplanes. On the other hand, high-bypass turbofans have high bypass ratio and low specific thrust and they are mainly used in civil aircrafts due to its quieter operating noise. The turbofan usually comes in two-spool arrangement. In addition, an afterburner, which is a second combustion chamber, can be added to a turbojet or turbofan to create more thrust. However, an afterburner consumes a lot of fuel, so it is usually used in a short period of time like during takeoff, emergencies, or combat, and as a result, an afterburner is usually used in fighter airplanes.
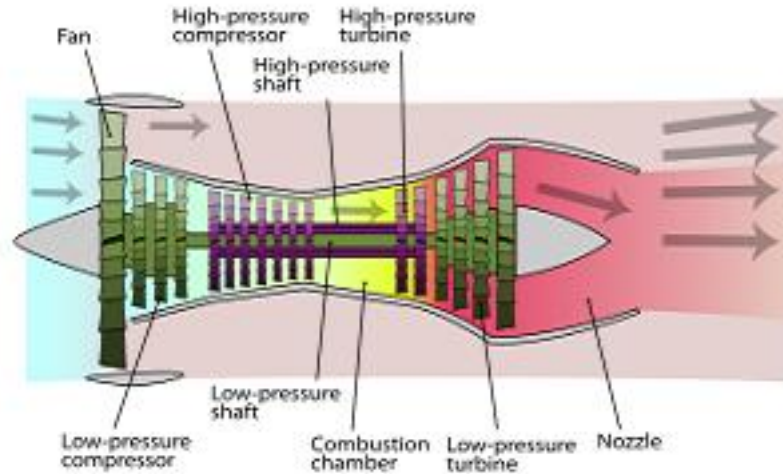
Figure 2.3: Schematic Diagram of a Turbofan Engine (from [8])

2.3.4 Turboprop

The turboprop engine is similar to a high-bypass turbofan except for one key aspect. The energy extracted from the turbine is used to drive a propeller through a series of a shaft and a gear box instead of driving a compressor, as shown in Figure 2.4. As a result, the propeller then produces most of the thrust. The turboprop is commonly used in small aircraft due to its limitation in flight speed since the propeller is exposed to the oncoming air flow and it "becomes less efficient as the speed of the aircraft increases" as well [9]. Although, turboprop can been seen in large aircrafts where the duration of flight is more emphasized than the velocity of flight.

2.4 Specific Fuel Consumption

The specific fuel consumption (SFC) is used to to determine the fuel efficiency of an engine. The SFC is an indication of how much fuel an engine is used to provide a net thrust in a given period of time. To calculate the SFC, the fuel flow rate (lbm/hr) is divided by the net thrust (lbf). Therefore, an engine with a low SFC will generate
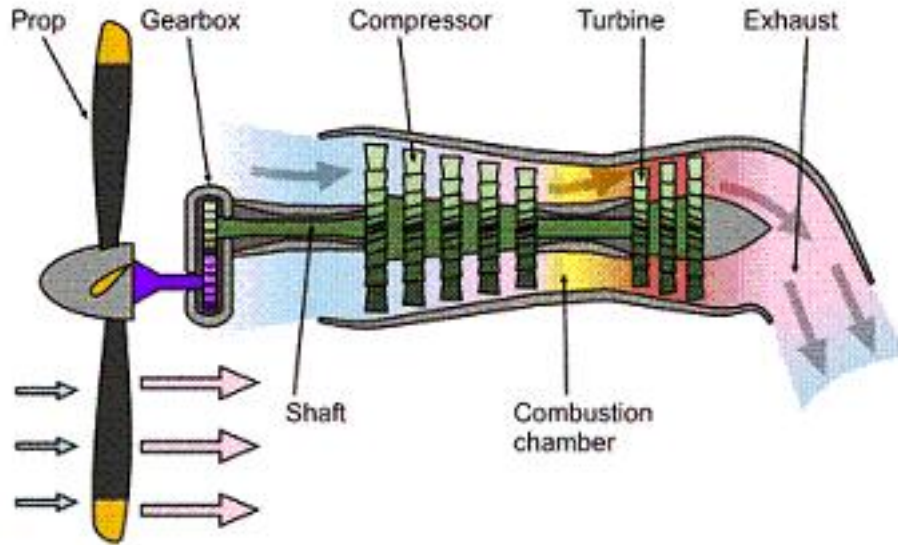
Figure 2.4: Schematic Diagram of a Turboprop Engine (from [10])

more thrust with less fuel consumption compared to an engine with a high SFC. An engine that has a low SFC has much more fuel efficiency than an engine with a high SFC. The turbojet JT3C-6 engine used in the Boeing 707-120 and Douglas DC-8/-11/-12 generating 11,200 lb-thrust has the SFC of 0.775, while the turbofan CFM56-5C4 engine used in Airbus A340-200/-300 providing 34,000 lb-thrust has the SFC of 0.33 [11]. The lower SFC value in turbofan engine confirms the improvement of propulsion efficiency of turbofan by adding the fan to the engine. The addition of the fan only slightly increases the fuel flow rate but greatly amplifies the amount of thrust generated by the engine. Note that SFC is changed with the altitude and operating modes of the engine, since the fuel flow rate and the net thrust are adjusted to suit each operating environment.

2.5 Aircraft Engine Maintenance

Maintenance is a crucial part to keep an aircraft engine working properly for a very long time. Maintenance can be broken down into preventive and corrective

maintenance. Preventive maintenance is a planing service which occurs periodically to eliminate failure in the engine. Inspection and repair focus on parts that are more prone to failure such as sensors and actuators, or components that deteriorate faster than others such as turbines and nozzles, since they are working in a very high temperature section of the engine. On the other hand, if the failure has already occurred in the engine, the service becomes a corrective maintenance.

In order to carry effective preventive maintenance, an aircraft engine must have a good monitoring system. The monitoring system oversees the overall health of the engine in terms of health parameters. To best indicate the engine's health, all of the parameters that impact the engine must be accounted for. However, certain health parameters may not be available due to difficulty in measuring them, which may be from the inaccessibility of the position to mounting the measuring device (due to complexity of the engine) or unavailability of the device itself. Therefore, each engine has its own specific health parameters.

2.6 Engine Station Numbering

Aircraft engine industries are massive operations, where one company may have different departments working on the same task or different companies may work together on the same task. Therefore, communication plays an important role in passing information among the agents. Companies and manufacturers usually have their own ways in numbering the engine stations, hence, confusion often arises when information is passed between agents with different standards. The Aerospace Recommended Practice (ARP 755A) has created an internationally recognized standard system in numbering engine stations for gas turbine engines. This practice has improved the quality of communication and effectively eliminated confusion in the industries. Engine components are divided into nine stations, describing by number one through

16

number nine as shown in Table 2.1.

Table 2.1: Fundamental Station Numbers [12]

Station Numbers	Description
AMB	Ambient conditions
0	Ram conditions in free stream
1	Engine intake front flange or leading edge
2	First compressor or fan front face
3	Last compressor exit face
4	Combustor exit plane
5	Last turbine exit face
6	Front face of mixer, afterburner etc.
7	Propelling nozzle inlet
8	Propelling nozzle throat
9	Propelling nozzle or exhaust diffuser exit plane

The first fundamental station number can be followed by the second digit, which indicates sub-stations between the fundamental stations. These intermediate stations are usually divided into ten stations. However, if there are more than ten stations, a third digit is used. Unfortunately, there is no standard in numbering the intermediate stations and each company usually has its own practice. In addition, different types of engines have different ways in interpreting the intermediate stations.

Nomenclature is also used with the engine station numbers to specify parameters at the particular engine location. The ARP 755A also standardizes the nomenclature as well. Examples of nomenclature with engine station numbers are: T1 indicates total temperature at the engine inlet, P2 indicates total pressure at the first compressor front face, and W31 indicates combustor inlet mass flow rate [12]. For the full list of nomenclature refers to [12].

CHAPTER III

AIRCRAFT ENGINE CONTROLS AND

C-MAPSS40K

3.1   Introduction

In the early period of engine control development (1940s), the frequency response method was used by utilizing the Bode plot to achieve desired system responses. In addition, a step response method and a root locus method were also used to improve control efficiency at a high altitude [3]. At this time, analysis of engine performance was based on a steady-state performance where engine module performance maps obtained from rig testing were primarily used in the evaluation of the engine performance.

During the 1950s-1960s, classical control such as single-loop, frequency-response, and time-domain methods were used in the design of the engine controllers. A method called successive loop-closure technique was utilized to isolate the coupling effects between control loops. Hence, control loops can be designed separately by assuming that other control loops are maintained at or near steady-state [3]. The control loops

can then be put together through successive loop-closure to create multiple control loops. Additionally, simulation of dynamic performance of the gas turbine engine was developed at the National Advisory Committee for Aeronautics (NACA) Lewis (the NACA Lewis is now the NASA Glenn Research Center). Furthermore, the Air Force Aero Propulsion Laboratory created a new efficient technique to calculate steady-state engine performance by utilizing the Newton-Raphson method, which significantly reduced the time to compute a design point [3].

When the electronic control unit (ECU) became available in the early 1970s, research in applying multivariable control design methods was already started. However, the multivariable control design was not ready for manufacturing engines yet, and all engine productions were still using single-input single-output (SISO) design in the control loop. The successive loop-closure technique was then used to join the SISO control loops together. Both frequency-domain and time-domain methods in multivariable control were also developed, but only the time-domain method was widely used. The frequency-domain method was proved to be too complex in a mathematical sense and was not as popular as the time-domain method. At this time, steady-state and transient performance simulations were used in the engine and control design.

During the 1980s, real-time simulation was developed and used, significantly expanding in later decades with the advancement of block-diagram-based modeling, automatic software generation tools, and the introduction of graphical user interfaces simulation.An advance in multivariable control design method also finds its way into the integration of the flight control and the propulsion control into one system. Prior to this innovation, these controls were usually designed separately by the airframe and engine manufacturers. By integrating the flight control and the propulsion control, pilot's responsibilities are decreased and aircraft maneuverability is enhanced [3]. In the past decade, the integration of control systems was broadened into the incorporation of subsystems, such as engine inlet and exhaust controls, with the main

engine system. The integration of subsystems further improves maneuverability.

This thesis expands the horizon of multivariable control design for engine control by implementing the multi-input sliding mode control method to the engine control logic. The engine simulation developed by NASA, called C-MAPSS40k, is used to serve as an engine model. A general idea of C-MAPSS40k will be presented in this chapter, along with the description of essential aircraft engine characteristics and components.

3.2 Aircraft Engine Characteristics

Aircraft engines are designed to operate in diverse environments where altitudes can range from sea level to 60,000 feet or higher and temperatures can range from below freezing level to 100 °F or higher. However, like any other machine, aircraft engines have limitations. Extreme inputs from a pilot can damage the structure of an aircraft, aircraft components, or aircraft engine. Therefore, an aircraft engine has an operating envelope, and within that envelope, the aircraft engine will safely operate without exceeding its structural and aerodynamic limits. The operating envelop for aircraft engines can be interpreted as an diagram of flight Mach number versus flight altitude, and the ambient temperature can be added as a third dimension for a turbojet or a turbofan engine. Figure 3.1 shows an engine operating envelope of a generic turbojet and a turbofan.

Aircraft engine controllers must be able to monitor essential engine parameters and use the information to sufficiently control the engine and ensure that nothing operates outside its limit. Operating an engine outside its limit can also instigate undesired characteristics of the system, such as stall and surge.
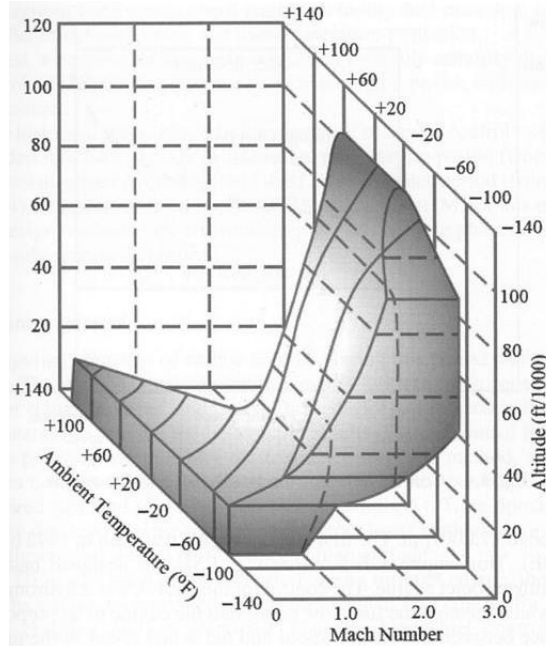
Figure 3.1: Engine Operating Envelope of Generic Turbojet and Turbofan (from [3])

3.2.1 Stall and Surge

Two common instabilities in turbine engines are surge and stall,which usually occur in the engines with axial compressors. Surge and stall commonly appear dependant of each other. An axial compressor uses rotor stages to increase pressure of air flowing into the engine. A rotor stage is composed of many small blades, which are small airfoils. These blades can be thought of as airplane wings, where stall is developed when the angle of attack exceeds the critical angle of attack which cause the air to no longer cling to the blades creating disruption in the airflow. The disruption in the airflow causes the flow through the compressor to slow down and stagnate, or even reverse in flow direction [13]. Hence, the compressor no longer effectively pressurizes the air, and the amount of thrust produced by the engine is reduced. The usual causes of stall are:

- Damage of the compressor or compressor rotors, resulting from ingestion of foreign objects into the engine or deterioration of engine components.

21

- Intake of turbulent, distorted airflow, and hot airflow (e.g., re-ingestion of hot exhaust gases).

- Operation outside of engine limits established by the manufacturer affects the rotational speed of the compressor rotors.

Stall generally occurs at high compressor rotor speeds. The area where a stall occurs is called a stall cell, which can partially or entirely cover a rotor span. Therefore, the severity of a stall can range from a momentary decrease in compression (e.g., slightly loss of power/thrust) or a total loss of compression causing the entire engine to fail and the pilot may lose control of the aircraft. Stall can also develop in a transient or a steady state. A transient stall is intermittent and can be identified with a loud explosion noise in the engine as the engine backfires and reverse of airflow occurs [13]. A transient stall is usually temporary and resolved on it own in a matter of seconds. Thus, a transient stall is generally not destructive to the engine. On the other hand, a steady state stall can cause strong repeatable vibrations through out the engine and poses significant stresses on the engine components. Immediate action from a pilot or an engine controller must take place before the effects of steady stall create damage on the engine structures, which may lead to total loss of thrust.

On the other hand, surge is a condition where flow rate is heavily fluctuated and the flow of air is reversed (it exits out through the inlet duct). Surge can occur due to the amount of airflow sent to the higher compressor stages by the lower compressor stages exceeds the higher compressor stages' limit. Hence, the higher compressor stages are choked, which builds up pressure in the compressor to the point where the lower compressor stages (in the front of the compressor) can no longer hold back the pressure causing the airflow to reversely blow out through the engine inlet. Surge can also take place as a result of a compressor stall. In addition, surge usually occurs at low compressor rotor speeds.

To combat stall and surge, an aircraft engine is equipped with variable stator vanes (VSV) and bleed valves. VSV guides incoming air into the engine at an appropriate angle to ensure that air uniformly flows through compressor rotors, preventing separation of airflow from the rotor's blades, thus avoiding the compressor stall. An engine can have many stages of VSV, where each stage is located between compressor rotors.

A stage of VSV consists of a series of vanes that individually attached to a vane arm. The vane arm is connected to a unison ring, which is controlled by an actuation system through a drag link. When the unison ring is rotated by the actuation system, each vane is pitched at the same angle. In addition, rub pads are placed at the inner radius of the unison ring to keep the unison ring in a circular shape [14]. Figure 3.2 shows a stage of VSV.

Bleed valves are attached on a compressor case where they release excess amounts of air in the compressor into the outside environment. The engine controller is able to regulate the pressure in the compressor and prevent a compressor stall and surge to occur. A bleed valve is composed of the valve, actuator, bellcrank, rollers and cams slots, and rub pads [14]. The actuator controls the valve to open or close through the bellcrank with a series of rollers and cam slots, guiding the motion of the bleed valve. The rub pads located around the inner ring of the valve also act as the shape constraint, ensuring the circular shape of the bleed valve. Figure 3.3 and 3.4 show a construction of a bleed valve.

Stall and Surge Margin

A stall margin can be used to determine stability and robustness of a compressor. "Stall margin is defined in various ways, all of which characterize the distance on the mass-flow/pressure-rise map of the compressor between the operating line of the compressor and the stall/surge line, i.e. the line beyond which the compressor
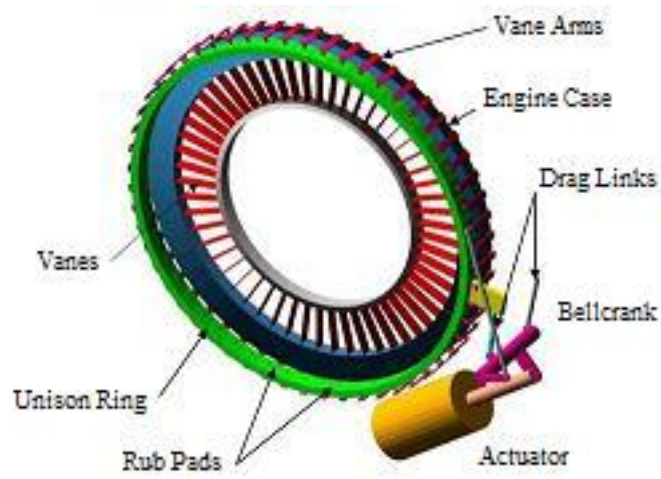
Figure 3.2: A stage of VSV (from [14])
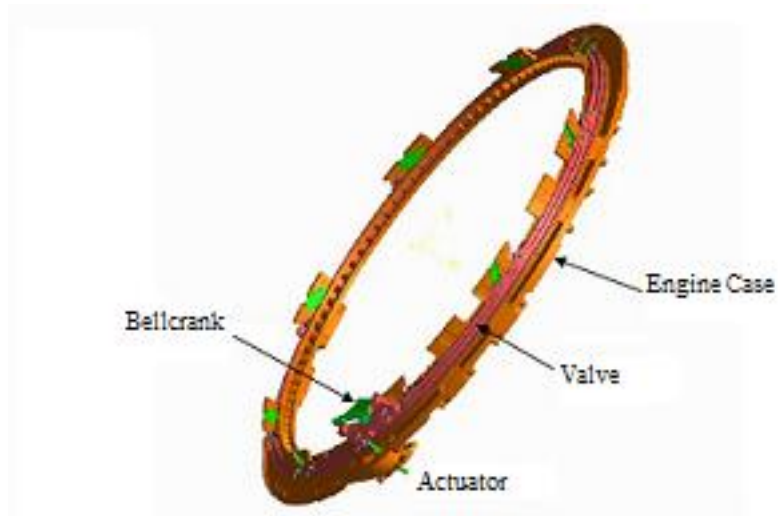
Figure 3.3: A Bleed Valve (from [14])
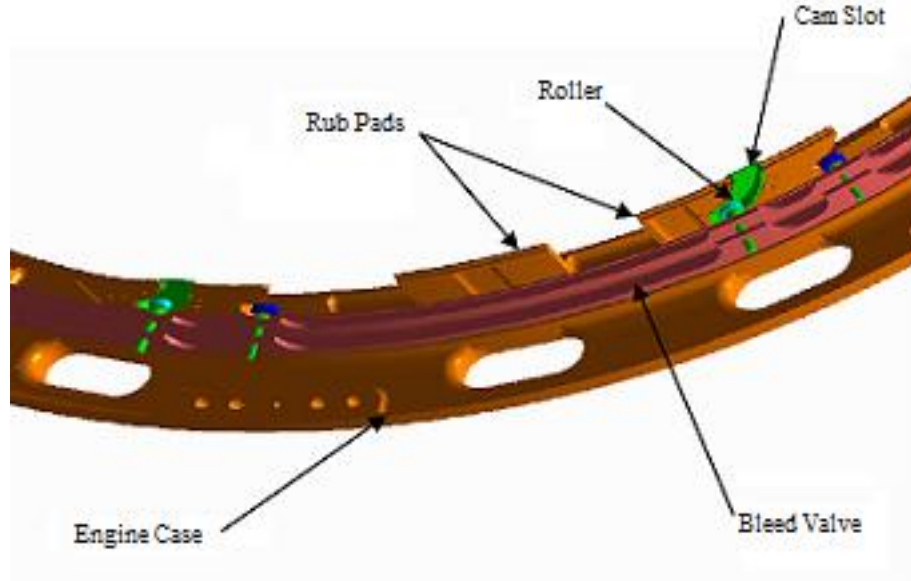
Figure 3.4: A Bleed Valve (from [14])

becomes unstable" [15]. Therefore, a stall margin indicates the boundary in which a compressor can operate without inducing a stall. The same idea applies to surge margin, but instead of stall, the compressor will surge if operating outside a surge margin. In addition, the surge margin is used to indicate the engine's ability to accelerate. Hence, the broader the surge margin, the faster the engine can accelerate. Stall and surge margins are calculated as the following [4]:

$$Stall/SurgeMargin = \frac{Pressure\ ratio_{stall/surge} - Presssure\ ratio_{operatingpoint}}{Pressure\ ratio_{operatingpoint}} * 100 \quad (\%)$$

(3.1)

Figure 3.5 shows a surge margin in a green region on a compressor map. The surge margin in Figure 3.5 is reduced due to uncertainties and variations in the engine. In which, overtime the normal operating line will move upward, while the surge line will move downward [3]. Note that a compressor map is a diagram of pressure ratio (i.e., the ratio of compressor exit static pressure to compressor inlet static pressure) versus amount of airflow, in which it can predict the behavior of the compressor. The compressor map is obtained through rig testing or a computer program.
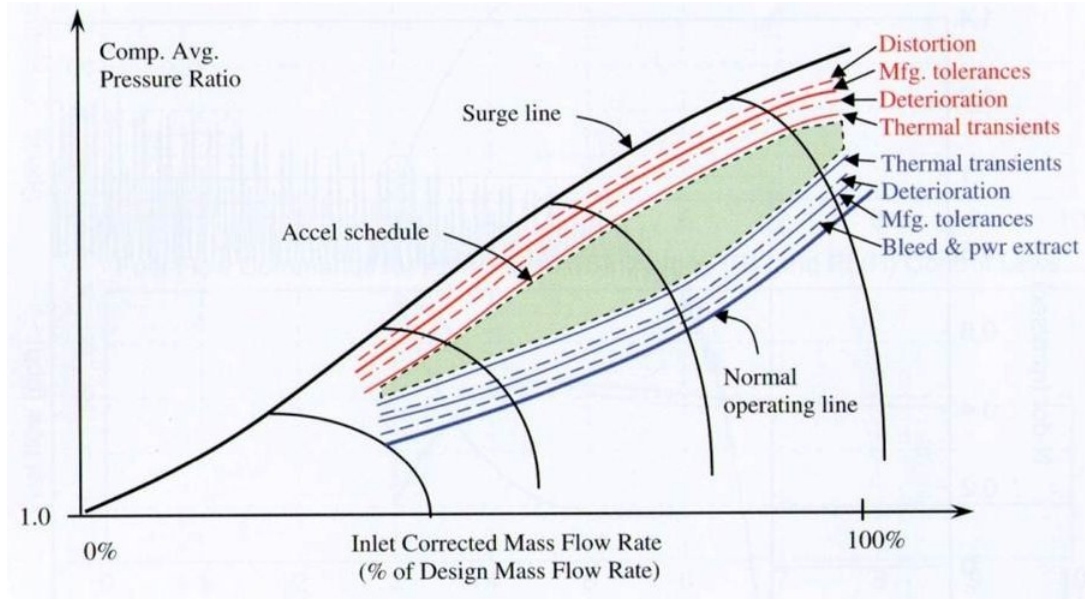
Figure 3.5: Surge Margin on a Compressor Map (from [3])

3.3   Thrust Control

The main objective of the engine controller is to accurately control thrust in regards to the position of the throttle. However, measuring thrust directly is not currently practical. The engine pressure ratio (EPR) and engine shaft rotational speed N directly correlate to thrust; hence, they are used to indicate the amount of thrust produced by the engine. Engine pressure ratio is a ratio of the nozzle pressure to compressor inlet pressure, whereas the engine shaft rotational speed, N, can be divided into $N1$ and $N2$ values. The $N1$ value is the rotational speed of the low pressure compressor, where $N2$ is the rotational speed of the high pressure compressor. Only $N1$ is used as the indication of thrust while $N2$ is used for limit and condition monitoring [17]. Both EPR and engine shaft rotational speed indicators are provided in the cockpit for cross reference. Note that the indicated values of engine shaft rotational speed are a percentage of design r.p.m. Since the EPR and engine shaft rotational speed N are good indication of thrust, they are used as parameters in engine controllers to control the amount of thrust produced by the engine. Figure
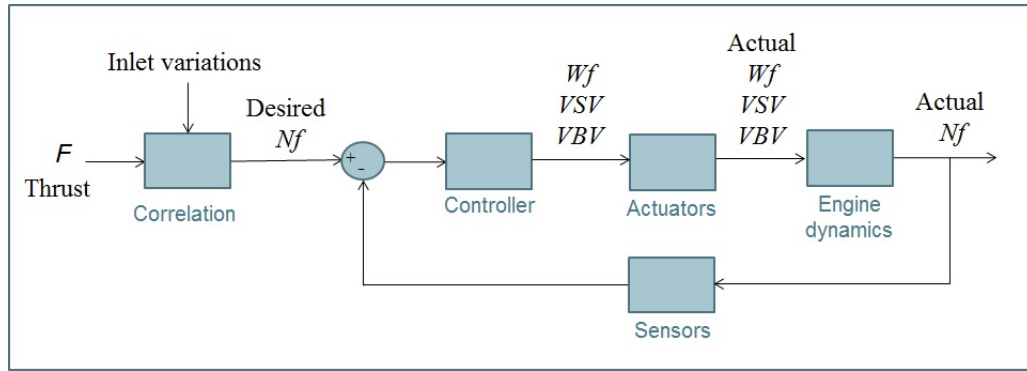
Figure 3.6: Diagram of Thrust Control Using Fan Speed N_f

3.6 shows a diagram of thrust control where the value of fan speed N_f is equivalent to $N1$. The engine controller sends the fuel flow W_f, VSV, and VBV commands to the actuators, which then regulate the W_f, VSV, and VBV to obtain the desired N_f value.

Overall, the EPR is more favorable as the thrust indication compared to the $N1$ value due to its consistent thrust indication over a long period of time. In other words, the value of EPR will always give the same value of thrust. However, the value of engine shaft rotational speed will not indicate the same value of thrust over time due to engine deterioration as the engine ages. Values of EPR and $N1$ are also used as a suggestion to identify specific breakdowns in the engine. Low values of EPR and $N1$ may be caused by flameout, engine rollback, or engine damage. Fast fluctuations of EPR and $N1$ value may be a result of an instability in the engine, such as stall and surge. Surprising high values of EPR and $N1$ may be an effect of malfunction in the fuel control.

3.4   Commercial Modular Aero-Propulsion System Simulation 40k (C-MAPSS40k)

NASA has developed a simulation of a generic 40,000 lb-thrust twin spool, high bypass turbofan engine utilizing Matlab and Simulink program. This simulation is called Commercial Modular Aero-Propulsion System Simulation 40k (C-MAPSS40k). It is an updated version of the previous C-MAPSS, which has a maximum thrust of 90,000 lb-thrust, but it only has one control mode: fan speed. The updated version has three control modes: fan speed (N_f), core speed (N_c), and engine pressure ratio (EPR). It is obvious that N_f is basically $N1$ and N_c is equal to $N2$ as discussed in the previous section. Engine life, high-speed shaft power takeoff, and customer bleed flow rate were also added to this update version of C-MAPSS. In addition, the C-MAPSS40k throttle setting is in a degree unit of power lever angle (PLA), instead of the percentage of power used in the earlier version of C-MAPSS, and the engine control is digital rather than the continuous control used in the C-MAPSS. Note that this section only provides the most essential material about the C-MAPSS40k. For more in-depth information of C-MAPSS40k refer to [16].

Adjustment of the engine parameters can be done through the command line or the graphical user interface (GUI) shown in Figure 3.7. Modification to the engine model can be partial or complete such as replacing the entire engine. There are countless possibilities of how to modify the engine as long as all of the components involved in the change are adjusted accordingly and the change is not in conflict with the existing systems.

The C-MAPSS40k has been developed based on test data. Therefore, the responses of the engine merely represent the test data from which the engine simulation was created and not from the precise responses of an actual engine. Operating conditions and simulation inputs of C-MAPSS40k is shown in Table 3.1 and 3.2 re-
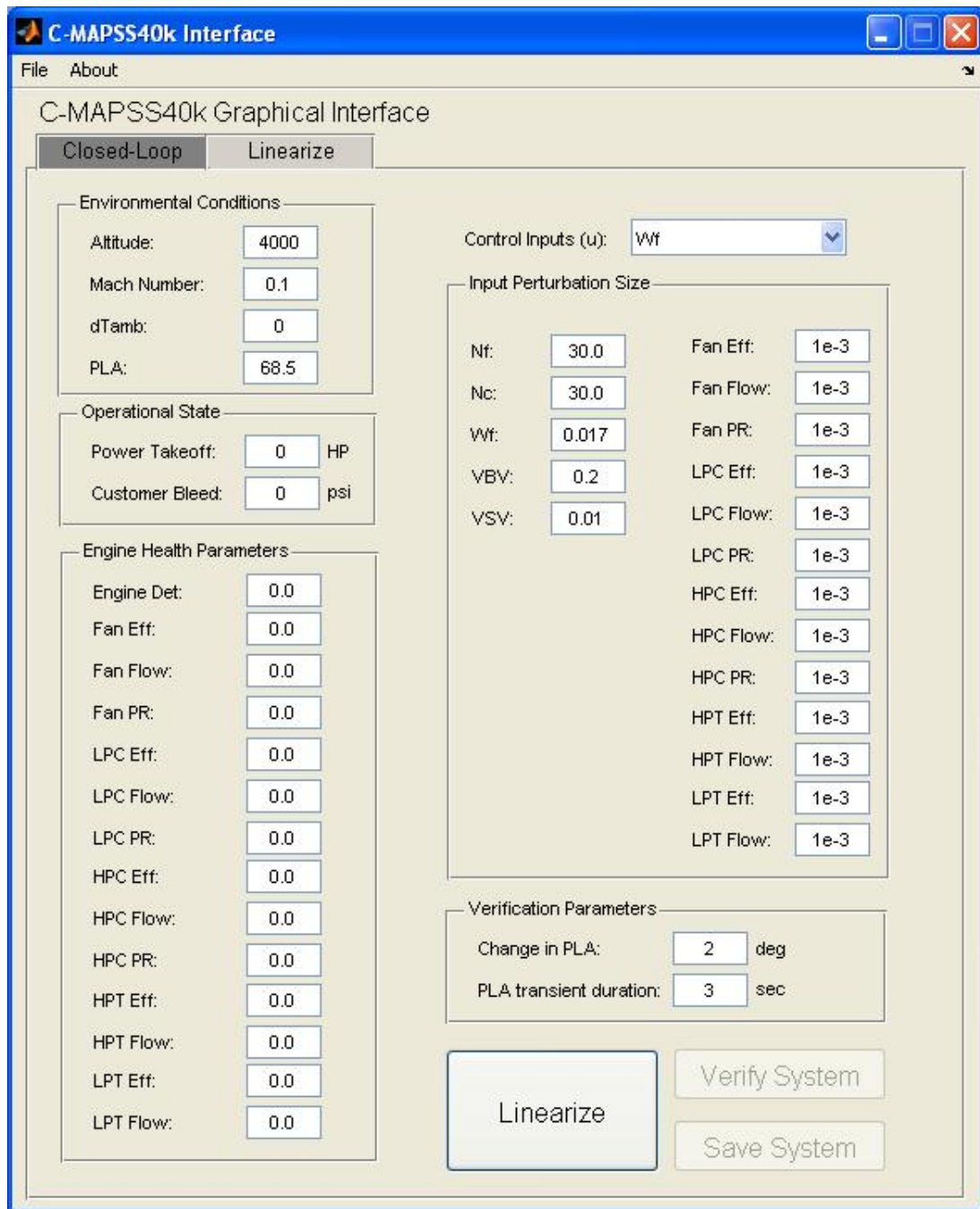
Figure 3.7: C-MAPSS40k Graphical User Interface (from [16])

spectively.

Table 3.1: Operation Conditions of C-MAPSS40k (from [16])

Conditions	C-MAPSS40k
Altitude	0 to 40,000 ft
Mach Number	0.0 to 0.85
dTamb	-30 to 50 °F
Throttle	40 to 80.5 degrees
Maximum Thrust	40,000 lbf

Table 3.2: Simulation Inputs of C-MAPSS40k (from [16])

Inputs	C-MAPSS40k
Controller	Fuel flow, Variable Bleed Valve position, and Variable Stator Vane
Controller Type	Digital
Atmospheric	Altitude, Mach number, and delta ambient temperature
Component Health Parameters	13 parameters
Control Mode	EPR, fan speed, and core speed
Engine Life	Deterioration level of 0.0 to 1.0
Operational	High-speed shaft power takeoff, and customer bleed flow rate

The delta ambient temperature (dTamb) represents "free stream static temperature minus standard atmosphere temperature" in degrees Fahrenheit [16]. The deterioration level of the engine is represented in percentage, where 0.0 indicates a new engine and 1.0 indicates an end-of-life engine.

Engine components in the C-MAPSS40k are represented in Simulink by block diagrams, which makes it easy to see how the engine systems are built and how the components are connected. Thus, engine input and engine parameter modifications are straightforward. However, not all of the engine components are defined in the Simulink. Some of the engine components, such as the inlet, LPC, and HPC, are described as Matlab executable (MEX) files (i.e., compiled C-code). By interpreting some of the engine components as MEX files, the simulation time is shorted.
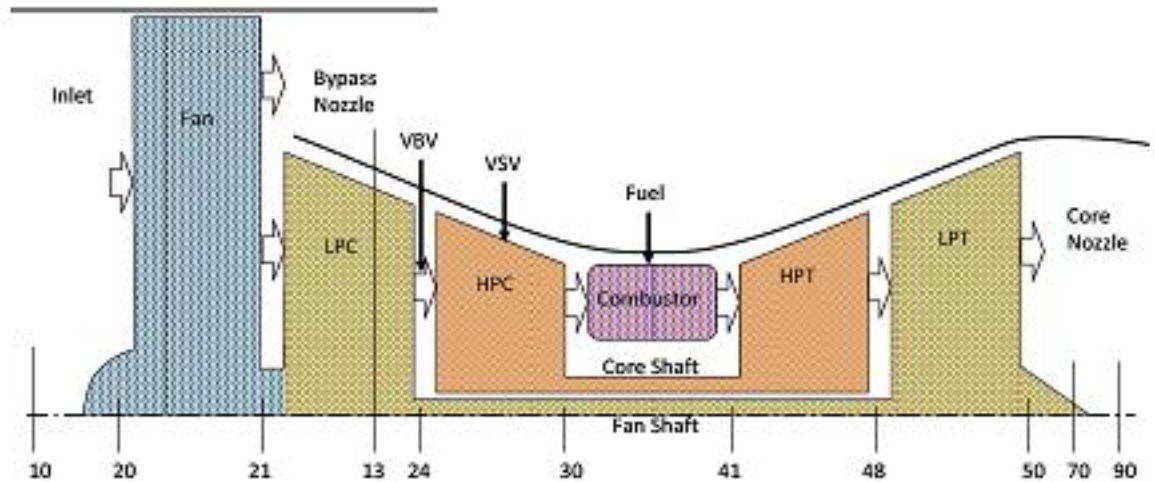
Figure 3.8: Schematic of The Twin Spool, Turbofan Engine (from [18])

3.4.1 C-MAPSS40k Engine Model

The engine in C-MAPSS40k is called the PAX200 engine, possessing three major components: open-loop engine model, discrete-time solver, and digital baseline engine controller [16]. The schematic of the twin spool turbofan engine that the PAX200 engine is modelled after is shown in Figure 3.8, along with the engine station numbers. As described earlier, an aircraft engine must operate within its operating envelope in order to ensure the safety of the flight. The PAX200 engine's operating envelope is shown in Figure 3.9.

The open-loop engine model is shown in Figure 3.10, where text represents engine control inputs. The discrete-time solver is utilized to balance the mass-flow rate which is used to express the relationship between engine components. The controller used in the C-MAPSS40k is a digital baseline engine controller that consists of three components: input schedules, power management, and actuators, as shown in Figure 3.11. The input schedules calculate the VSV command (deg) and the VBV command (%open), where the power management calculates the fuel flow rate W_f (lb/sec). The commands of VSV, VBV, and W_f are sent to the actuator models as the input signals.
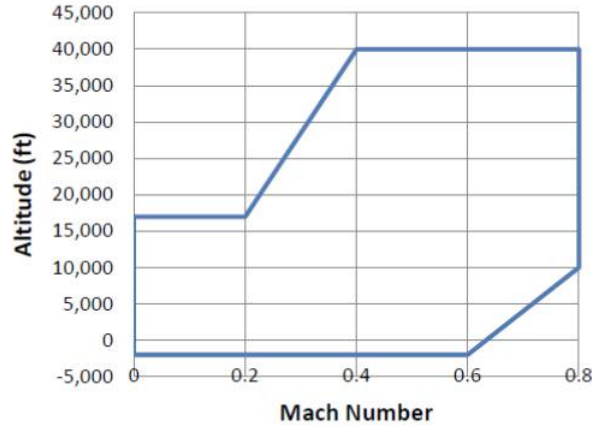
Figure 3.9: PAX200 Engine's Operating Envelope (from [16])

The actuator systems then simulate the actuators that regulate the fuel flow, VSV, and VBV. Finally, the commands from the actuator models are used as the inputs for the engine model.

A proportional fuel metering valve is used as an actuator for the fuel flow; its model in C-MAPSS40k is shown in Figure 3.12 where the non-linearities such as dead-band, rate-limiter, and stroke limit are included in the model to represent the non-linear characteristics of an actual fuel metering valve. The VSV actuator is modeled as a closed loop system, consisting of the proportional integral (PI) controller, a motor model, and a backlash block. Figure 3.13 shows the VSV actuator model implemented in C-MAPSS40k, where "VSV PID controller subsystem contains a PI controller that takes an error input (degrees) and creates a current control signal for the motor (mA)" [16]. The VBV actuator is represented as a first order system in the state space form. Using the first order system, the lag in response to the VBV command is captured by the model. The VBV actuator model used in C-MAPSS40k is shown in Figure 3.14.

In the C-MAPSS40k, the VSV are located in front of the HPC, while the VBV is located between the LPC outlet and the bypass duct. The range of the VSV command is from -62.412 to 0.61 degrees and the range of the VBV command is between 0 to 1, where zero represents fully closed and one represents fully open.

32

Figure 3.10: C-MAPSS40k Complete Block Diagram (from [16])

Figure 3.11: Diagram of the Engine Controller Used in C-MAPSS40k (from [16])
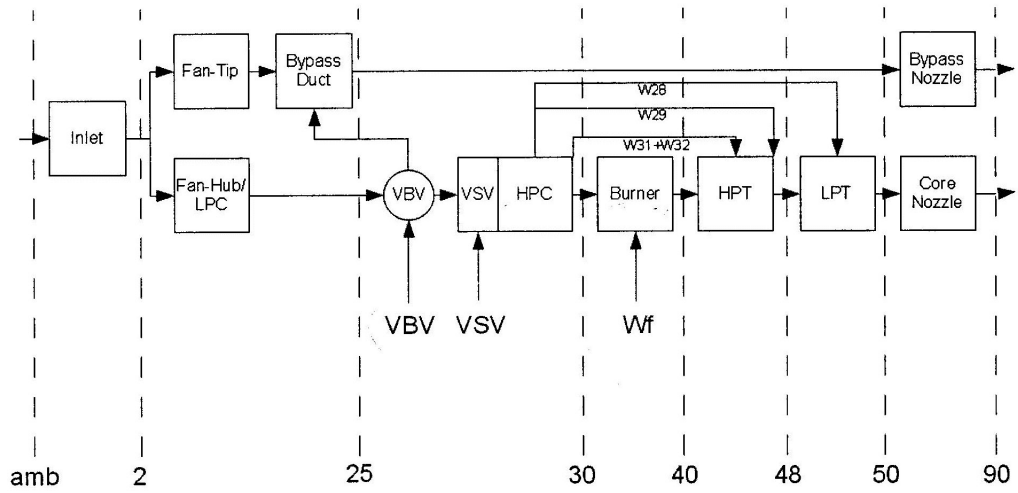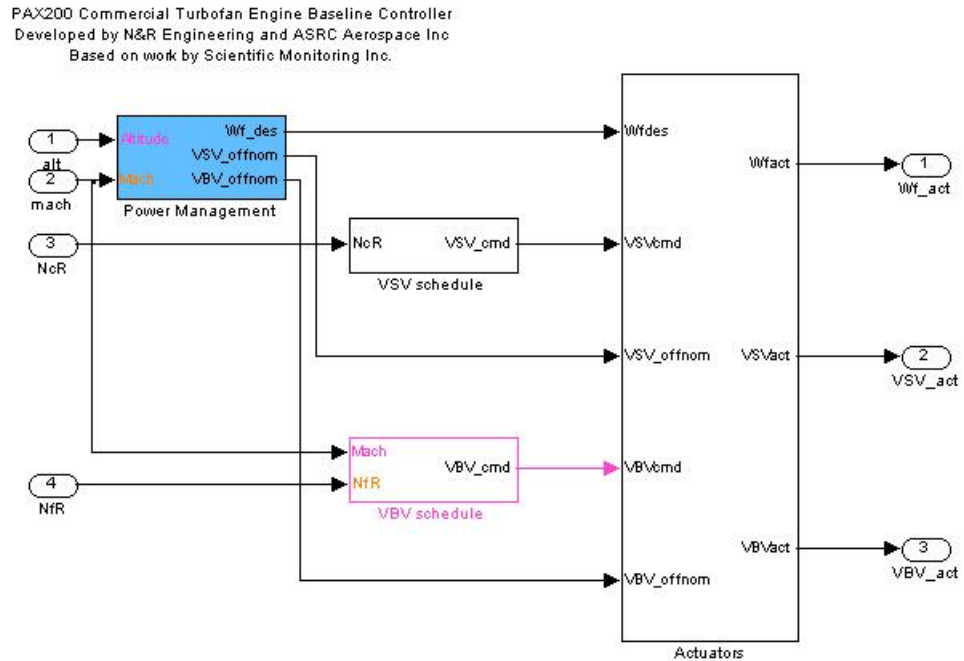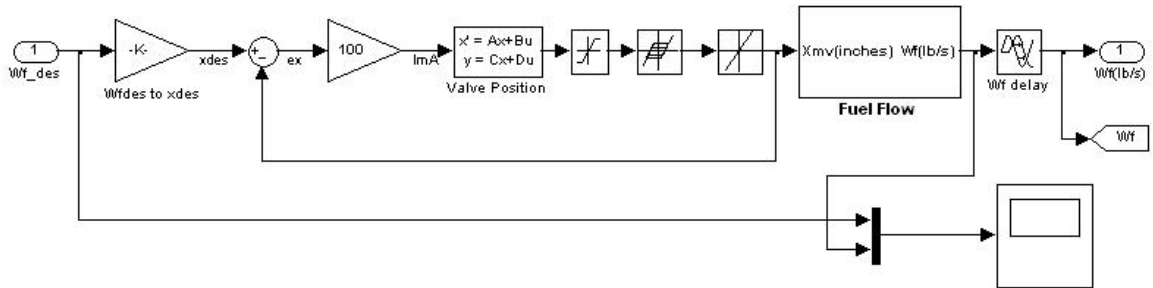
Figure 3.12: Fuel Metering Valve Diagram (from [16])
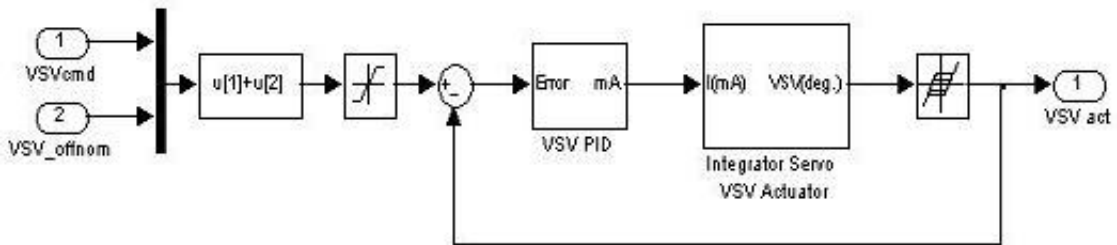
Figure 3.13: Variable Stator Vane Actuator Diagram (from [14])
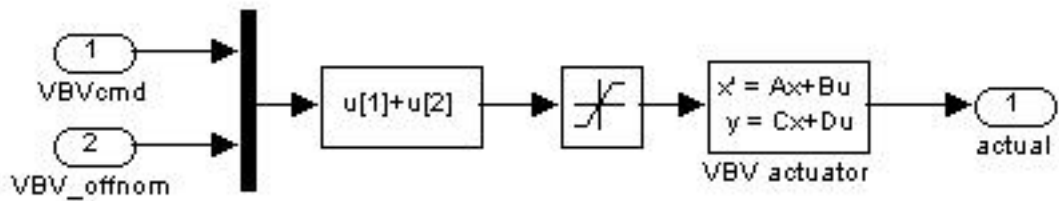
Figure 3.14: Variable Bleed Valve Actuator Diagram (from [14])

The fuel flow rate calculated by the power management is based on the specific power lever angle (PLA) that defined by the users. The power lever angle is the angle of the throttle, which corresponds to the amount of thrust demanded by the pilot. Keep in mind that the PLA does not correlate to the percent open of the throttle. "At the minimum PLA of 40 degrees, the net thrust is approximately 3,000 lbf while full power corresponds to a thrust of nearly 35,000 lbf at sea level static conditions" [16]. Hence, a PLA of 40 degrees, only equals to an approximated 8.6% of the full throttle. Table 3.3 shows the PLA values according to the power ratings.

Table 3.3: Simulation Inputs of C-MAPSS40k (from [16])

Power Rating	PLA (degrees)
Ground Idle	40
Taxi	42
Flight Idle	44
Approach	52
Max Cruise	68
Max Climb	73
Max Take-off	78

In addition, the control mode selected by the user is implemented in C-MAPSS40k inside the power management architecture. There are three control modes, which are N_f, N_c, and EPR. These control mode are represented as setpoint controllers in the PAX200 engine. When the N_f control mode is selected, the engine controller will drive the value of N_f to a specific value and the output of the N_f controller will always be selected by the control logic. The same idea applies to N_c and EPR control mode. Furthermore, the power management can interfere with the VSV and the VBV commands to guarantee that compressor stall will not occur, especially during large flight transitions.

3.4.2 Linearization

The GUI shown in Figure 3.7 gives the user easy access to the linear time-invariant model of an open-loop engine with the user specified inputs, such as altitude, Mach number, and PLA. The linearization routine within the GUI can generate two types of the engine models. Both of the engine models have the same output, but their inputs are different. One of the models has one input, which is fuel flow, while the other model has three inputs-which are fuel flow, VSV, and VBV. The model with one input assumes that VSV and VBV are always on schedule and do not change [16]. On the other hand, the model with three inputs includes VSV and VBV to represent the variations from their nominal schedule values. The GUI outputs a linear model in the state space form as the following

$$\dot{x} = Ax + Bu + Lh \tag{3.2}$$

$$y = Cx + Du + Mh \tag{3.3}$$

where:

x is the state vector, $[N_f, \ N_c]^T$

u is the input vector, $[W_f, \ VSV\_offnom, \ VBV\_offnom]^T$

h is the health parameters vector defined in Table **??**

y is the output vector defined as $[N_f, \ N_c, \ T25, \ T50, \ Ps3, \ P5, \ Fnet, \ SM\_LPC, \ SM\_HPC]^T$

Note that linearization can be done through the command line as well, but this method is not used in this thesis. With the history of the engine controllers, and the engine characteristics are described, along with the introduction of the C-MAPSS40k, the next chapter will introduce the idea of the sliding mode control and develop the sliding mode control logic to be implemented into C-MAPSS40k.

CHAPTER IV

SLIDING MODE CONTROL

4.1   Introduction

This chapter introduces the concept of sliding mode control then uses it to develop the control law to be implemented in NASA's Commercial Modular Aero-Propulsion System Simulation 40k (C-MAPSS40k). First, a description of sliding mode control is presented. Then, the concept of sliding mode control is mathematically demonstrated and its properties are presented. Once the system model and the reference model are defined, the sliding mode control law is derived. To efficiently implement the control into C-MAPSS40k, chattering phenomenal due to imperfections and unmodelled dynamics is suppressed. After chattering is dealt with, the control law will be modeled in Matlab/Simulink and set up in the C-MAPSS40k, which will be discussed later in chapter 5.

Sliding mode control is a variable structure system where the control logic switches between two extreme control laws to obtain the designed system performance. The control uses a switching function to choose between two controls based on the location of the state trajectory. A state trajectory is driven toward the sliding surface by

switching between control laws. When a state trajectory reaches the sliding surface (also called sliding manifold or hypersurface when the dimension of the surface is greater than two), the trajectory is trapped in the surface and forced to move toward state equilibrium, such as the origin. The movement of the trajectory along the surface is called sliding, hence the name sliding mode control. The biggest advantage of sliding mode control is its robustness, because the control structure is protean, making it flexible to deal with modeling imprecisions and disturbances.

In addition, when the state trajectory reaches the sliding surface, the order of the system is reduced which simplifies the complexity of the system and makes it easier to regulate. Other advantages of sliding mode control are rejection of disturbance, insensitivity to parameter variations, and simple implementation by means of power converters [19]. However, sliding mode control has a major setback: the phenomenon called "chattering". Chattering is a high frequency switching motion of the state trajectory between the two extreme control laws, which occurs in the neighborhood of the sliding surface. In an attempt to trap the state trajectory within the sliding surface, the switching functions are switched at an infinite frequency. The undesired effect of chattering creates heat as a result of rapid change between the switching functions when the control is implemented into electronic or mechanical systems. The heat can burn electronic systems, while in mechanical parts the act of rapidly changing between two extreme control laws can wear off mechanical parts and eventually damage them.

4.2   Concept of Sliding Mode Control

Sliding mode is a motion where the control logic switches at a high frequency, taking place in any control system with discontinuous control function and in any dynamic system with discontinuities in the motion equations. Using an example

38

of a first-order tracking relay system from [20], sliding mode characteristic can be demonstrated. The state variable is set up as

$$\dot{x} = f(x) + u \tag{4.1}$$

where $f(x)$ is a bounded function, $|f(x)| < f_0 = constant$, and the control, u, is a relay function of the tracking error defined as $e = r(t) - x$ where $r(t)$ is a reference input. The control is defined as

$$u = \begin{cases} u_0 \text{ if } e > 0 \\ -u_0 \text{ if } e < 0 \end{cases}$$

with $u_0 = constant$. This control can be written in the signum function form as $u = u_0 sgn(e)$. By differentiating the error equation, the following equation is obtained.

$$\dot{e} = \dot{r} - f(x) - u_0 sgn(e) \tag{4.2}$$

If $u_0 > f_0 + |\dot{r}|$, then e and $\dot{e}$ have different signs, which means that tracking error decreases at a finite rate. When the tracking error is equal to zero, discontinuity occurs in the control since the sgn function is undefined at $e = 0$. The motion after the tracking error reaches zero is called sliding mode [20].

The next example is a system with discontinuities in its motion equation, in which sliding mode behavior is also present. Using Coulomb friction example from [20], the motion equation is defined as

$$m\ddot{x} + kx = -u_f(\dot{x}) \tag{4.3}$$

with $x(t)$ defined as displacement and k defined as spring stiffness. The friction

force, u_f is discontinue and given as

$$u_f = u_0 sgn(\dot{x}) \tag{4.4}$$

where u_0 is a constant. By substituting equation (4.4) into equation (4.3) and rearranging to obtain

$$m\ddot{x} = -u_0 sgn(\dot{x}) - kx \tag{4.5}$$

It is seen that if the friction force, u_f, is less than the spring force, then the mass alternately moves in a positive and a negative direction. However, if the friction force is greater than the spring force, then the mass cannot move which means $x(t)= \dot{x}=0$. This condition creates slide mode since $\dot{x} = 0$ is not defined in the signum function in the friction force equation.

Sliding mode control also possesses invariance property, which means the control has low sensitivity to disturbances and plant parameter variations. To show the invariance property of sliding mode control, the example from [20] is demonstrated. The system is a second order time invariant relay system and its equation is determined as

$$\ddot{x} + a_2 \dot{x} + a_1 x = u + f(t) \tag{4.6}$$

$$u = -M sgn(s) \tag{4.7}$$

$$s = \dot{x} + cx \tag{4.8}$$

where a_1, a_2, M, and c are constant variables. Bounded disturbance is represented by $f(t)$. When $s = 0$, the system undergoes sliding mode and the motion equation

can be described as

$$\dot{x} + cx = 0 \qquad (4.9)$$

The solution to equation (4.9) is $x(t) = x(t_1)e^{-c(t-t_1)}$ where t_1 is the time when the state trajectory reaches switching surface and is trapped in the surface. It is obvious that the solution does not depend on the plant parameters or the disturbance. This invariance property of sliding mode control adds robustness to the controller and makes it attractive to systems with uncertainty conditions. Furthermore, sliding mode control also creates order reduction in the system once the state trajectory reaches the sliding surface (i.e., $s = 0$), making the system simpler and easier to regulate. Using the previous second order time invariant relay system, the order reduction property of sliding mode control can be shown. When the sliding mode is induced, $s = 0$ and $\dot{x} + cx = 0$ are obtained, hence

$$\dot{x} = -cx \qquad (4.10)$$

$$\ddot{x} = -c\dot{x} \qquad (4.11)$$

By substituting equation (4.10) and (4.11) into equation (4.6), the next equation is obtained.

$$-c\dot{x} + (a_1 - a_2c)x = u + f(t) \qquad (4.12)$$

Equation (4.12) shows that the system equation is of reduced order. The reduction of the system order is equal to the number of inputs. In this case, the order of the system is reduced by one since there is only one input. In addition, sliding mode dynamics are based on the switching function equations and are independent of the control logic.

41

With many advantages, sliding mode control is a suitable algorithm to deal with complex high order systems under uncertainty conditions. However, implementing the sliding mode control into a real system can pose problems, since the ideal sliding mode may not be obtained due to imperfections of the switching device, dead zone, hysteresis, or unmodelled system dynamics, etc. Therefore, the control can only force the state trajectories to the vicinity of the switching surface, which may lead to chattering. Fortunately, many strategies have been developed to eliminate chattering. This thesis focuses on the boundary layer method in suppressing the chattering, which will be discussed after the design of the control logic is presented.

4.3   Sliding Mode Control Design

After the system model and the reference model are determined, designing a sliding mode control consists of three steps. The first step is to define the sliding surfaces that will give desired performances in the system. The second step is designing the control logic that will induce the state trajectories into the intersection of sliding surfaces with any initial conditions. The final step is to ensure that the trajectories will stay on the sliding surfaces once sliding mode occurs, so in other word $s = 0$ after $t = t_s$ where t_s is the time that trajectories reach sliding surface. The plant is modeled in state space representation as the following

$$\dot{x} = Ax + Bu \qquad (4.13)$$

with tracked outputs

$$y = Cx + Du$$

The reference model is defined as

$$\dot{x}_m = Ax_m + Bu_m \tag{4.14}$$

where the reference control law is

$$u_m = -Kx_m + Pr \tag{4.15}$$

By substituting equation (4.15) into (4.14), the reference model becomes a closed loop model:

$$\dot{x}_m = (A - BK)x_m + BPr \tag{4.16}$$

with outputs

$$y_m = Cx_m + Du_m \tag{4.17}$$

The desired outputs are defined as r. The gain matrix K is calculated from utilizing the linear quadratic regulator (LQR) to ensure that $(A - BK)$ is a stable matrix while giving an optimal control [21]. In addition, the state variables are normalized to change in the same range by choosing Q and R appropriately. The gain matrix P is chosen with a condition in which at steady state (where $\dot{x}_m = 0$), the reference outputs, y_m will equal the desired outputs, r. When $\dot{x}_m$ becomes zero, the equation (4.16) is solved for x_m, yielding

$$x_m = -(A - BK)^{-1}BPr \tag{4.18}$$

By substituting equation (4.18) into equation (4.17), the next equation is obtained.

$$y_m = [D - (C - DK)(A - BK)^{-1}B]Pr \tag{4.19}$$

Since y_m and r are equal at a steady state, the term $[D-(C-DK)(A-BK)^{-1}B]P$ must be equal to identity matrix, I. Therefore, P can be calculated as

$$P = [D - (C - DK)(A - BK)^{-1}B]^{-1} \qquad (4.20)$$

4.3.1 Defining Sliding Surface

Once the system model and the reference model are described, the first step is to define sliding variable s

$$s = Ge \qquad (4.21)$$

where G is a gain matrix calculated by choosing the desired sliding mode poles that will give desired properties in sliding mode. As a consequence, matrix G establishes the orientation of the sliding surfaces, where s is equal to zero. Hence, the tracking error e will be forced to go to zero as well. A Matlab function called "vplace" is created based on methods described in [22], and it is utilized to determine G. The inputs for the function "vplace" are desired sliding mode poles and partial of the regular form of the state space matrix A. The regular form of matrix A is calculated with the Mathlab command demonstrated in [22] as well, which is shown in Appendix A. The tracking error e is defined as

$$e = x - x_m \qquad (4.22)$$

Differentiate equation (4.21) yielding

$$\dot{s} = G\dot{e} \qquad (4.23)$$

By setting $\dot{s}$ equal to $-\eta sgn(s)$, where η is a real number and a positive constant, sliding variable s will converge to zero.

4.3.2 Designing the Control Law

The second step in designing the sliding mode control is to ensure that state trajectory will be induced into the sliding surface. The value of the sliding variable s can be thought of as a distance between the state trajectory and sliding surface. As a result, the absolute value of s is needed to be reduced over time in order for state trajectory to reach sliding surface. Using a Lyapunov function v, the origin of the state plane is asymptotically stable (i.e., s will always decrease) if derivative of Lyapunov function is negative definite. Lyapunov function is defined as

$$v = \frac{1}{2}s^2$$

$$\dot{v} = s\dot{s}$$

$$\dot{v} = s(-\eta sgn(s))$$

- when $s > 0$, $-\eta sgn(s) = -\eta$

  and $\dot{v} = -s\eta < 0$

- when $s < 0$, $-\eta sgn(s) = \eta$

  and $\dot{v} = -s\eta < 0$

Therefore, $\dot{v} < 0$ whenever $s \neq 0$ and distance between the state trajectory and the sliding surface is always reduced. In other words, the state trajectory is able to reach sliding surface. Next, the control law is derived through the sliding surface equation to assure that the state trajectory will be induced into the sliding surface. Since $\dot{e}$ is equal to $\dot{x} - \dot{x}_m$, equations (4.13) and equations (4.14) can be used to

45

substitute $\dot{e}$ in equation (4.23), resulting in

$$-\eta sgn(s) = G(Ax + Bu - Ax_m - Bu_m) \qquad (4.24)$$

Solve for u and the following equation is obtained:

$$u = (GB)^{-1}[-GA(x - x_m) + GBu_m - \eta sgn(s)] \qquad (4.25)$$

Hence, the invert of GB must exist. Substituting equation (4.22) into equation (4.25) and the control law for the plant can be obtained as

$$u = (GB)^{-1}[-GAe + GBu_m - \eta sgn(s)] \qquad (4.26)$$

This control law will ensure that state trajectories will induce into the sliding surfaces. Furthermore, instead of having to force the state trajectories into the intersection of the sliding surfaces, each state trajectory is equipped with its own sliding surface. This will simplify the control law and enable the designer to tune the sliding surfaces separately.

4.3.3 Maintaining Trajectories on The Sliding Surfaces

To guarantee that state trajectories will stay on the sliding surface once sliding mode occurs, the tracking error dynamic is analyzed.

$$\dot{e} = Ax + Bu - Ax_m - Bu_m$$

$$\dot{e} = Ae + Bu - Bu_m \qquad (4.27)$$

By substituting control u into equation (4.27), tracking error dynamic equation is

written as

$$\dot{e} = Ae + B(GB)^{-1}[-GAe + GBu_m - \eta sgn(s)] - Bu_m$$

$$\dot{e} = [A - B(GB)^{-1}GA]e + B(GB)^{-1}GBu_m - B(GB)^{-1}\eta sgn(s) - Bu_m$$

The terms $B(GB)^{-1}GBu_m$ and Bu_m canceled out and error dynamics can be represented as

$$\dot{e} = A_{eq}e - B(GB)^{-1}\eta sgn(s)$$

where $A_{eq} = [A - B(GB)^{-1}GA]$. If all of eigenvalues of A_{eq} are in the left hand side of state space plane, stable sliding mode is obtained. Therefore, when sliding variable s is equal to zero (i.e., the state trajectory is on the sliding surface), tracking error e will tend to zero since $\dot{e} = A_{eq}e$. This means that once the state trajectories are induced into sliding surfaces, they will stay on the surfaces since the reduced order system is stable. Note that there will be zero eigenvalues present in the A_{eq} matrix. The number of zero eigenvalues of A_{eq} matrix is equal to the number of input because a zero eigenvalue represents a reduction of system order which is equal to the number of input as well. In this case, there are three inputs, therefore, three zero eigenvalues exist.

When the control law is implemented into the actual system, the unmodelled dynamics or system imperfections can cause chattering in the system. Since chattering is undesired in the controlled environment, it will be dealt with in the next section.

4.4 Chattering Problem

Chattering can occur in the system due to unmodelled dynamics or imperfections in the system. The unmodelled dynamics can be excited by the discontinuities in the control [20], therefore they must be carefully modeled into the system. However,

without the knowledge of these dynamics, designing the system using boundary layer control can be used, but with the trade off of disturbance rejection properties. In the presence of imperfection in the system, such as time delays, chattering can also appear. Delay "is the most relevant to any electronic implementation of the switching device, including both analog and digital circuits, and microprocessor code executions" [23]. The chattering due to delays can be suppressed using discrete-time control design techniques. In the scope of this thesis, chattering due to unmodelled dynamics is dealt with, and boundary layer approach is used.

Boundary layer approach replaces a discontinuous signum function with a saturation function, therefore, the control signal becomes continuous and chattering is eliminated. The saturation function approximates the signum function in a boundary layer of the sliding surface $s = 0$ [20]. As a consequence, ideal sliding mode is no longer taking place and state trajectories are confined to a vicinity of the sliding surfaces instead of exact tracking. The saturation function is defined as

$$sgn(s) = \frac{\eta}{\Phi} sat(s) \tag{4.28}$$

where η is the sliding gain. A high value of η will drive the value of the sliding variable s to zero faster than the lower value of η. However, it is more likely that the system response will experience overshoot as the sliding variable s goes to zero more quickly. The value of Φ represents the width of the area extending from the sliding surface where the state trajectory will be trapped in.

4.5 Remarks

In this chapter, the concept of sliding mode is presented, along with mathematical examples and its undesired characteristic of chattering. The advantages of the sliding mode, such as invariance property and order reduction, are also demonstrated. The

control law is then developed in a state space form, following three simple steps. Lastly, the chattering effect is suppressed using the boundary layer method. Now the control law is ready to be applied to the aircraft engine system. The next chapter will model the control logic in Matlab/Simulink and implement it into the C-MAPSS40k.

CHAPTER V

IMPLEMENTATION OF SLIDING MODE CONTROL

5.1 Introduction

In this chapter, the control is further defined in the aircraft engine environment. Applicable matrix values and dimensions will be assigned to the variables in the control logic. After all of the parameters are defined, the control law will be simulated in Matlab/Simulink where it will be tested with constant reference inputs to verify that the controller is functioning properly. Then, it will be implemented into the C-MAPSS40k, where the tuning process will take place and the robustness of the control will be examined. Performance of the engine based on the multivariable sliding mode control is also compared to the default baseline engine controller, which is comprised of the schedule input and the power management structure.

5.2 Sliding Mode Control for Aircraft Engine

The developed sliding mode controller will regulate thrust according to specific values of fan speed N_f, core speed N_c, and stall margin of high pressure compressor $SMHPC$. The desired N_f and N_c values are related to the amount of thrust, while the desired $SMHPC$ value ensures that the engine operates within its limits. The desired values of N_f, N_c, and $SMHPC$ will be used as inputs to create the reference model. The controller's job is forcing the system states to follow the reference model by using the control law. Therefore, the control law signals are the commands of fuel flow rate W_f, VSV, and VBV that will generate the desired values of N_f, N_c, and $SMHPC$. Note that the controller in this thesis is created based on a linearized model of the C-MAPSS40k engine at a ground idle condition, where the altitude is set to 1,000 feet with the Mach number of 0.1 and power lever angle (PLA) equal to 40 degrees.

5.2.1 Plant Model

First the plant in chapter 4 is augmented to create a type-one system, in which "one" indicates the number of integrations (in this case is one integration). As a consequence, a steady-state tracking error of zero can be produced when step inputs are fed into the system [21]. Figure 5.1 shows how the controller is constructed with the augmented plant.

To create an augmented plant, equation (4.13) can be determined as

$$\dot{x}_{aug} = A_{aug}x_{aug} + B_{aug}v_r \tag{5.1}$$

with tracked outputs

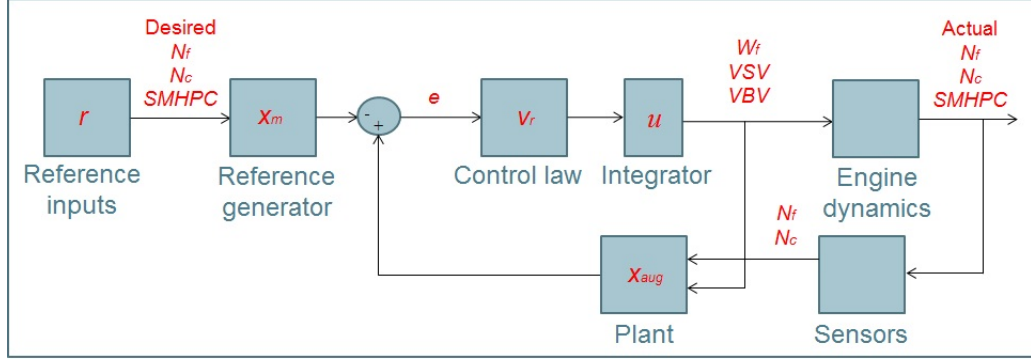$$y = C_z x_{aug}, \quad D_v \text{is equal to zero}$$

Figure 5.1: Diagram of a Sliding Mode Controller With an Augmented Plant

where

$$A_{aug} = \begin{bmatrix} A_z & B_v \\ 0 & 0 \end{bmatrix}$$

$$B_{aug} = \begin{bmatrix} 0 \\ I \end{bmatrix} \qquad I = \text{ identity matrix}$$

$$C_z = C_{aug}T_{xu} \qquad C_{aug} = \begin{bmatrix} C & D \end{bmatrix} \tag{5.2}$$

$$x_{aug} = \begin{bmatrix} z \\ v \end{bmatrix} \qquad z \text{ and } v \text{ are scaled version of state variable } x \text{ and control } u \text{ repectively}$$

$$\dot{x}_{aug} = \begin{bmatrix} \dot{z} \\ \dot{v} \end{bmatrix}$$

hence equation (5.1) can be rewritten as

$$
\begin{bmatrix} \dot{z} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} A_z & B_v \\ 0 & 0 \end{bmatrix} \begin{bmatrix} z \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} v_r
\tag{5.3}
$$

The subscript "aug" corresponds to the format of the A and C matrices that are augmented. The rest of the plant variables are subscripted with "aug" as well to prevent confusion with regular state space matrices. In addition, the whole system is scaled, where z is used to denote the scaled state variable x and v denotes the scaled control u. The state variable x represents engine fan speed N_f and engine core speed N_c, whereas the control u represents actuator commands, which are fuel flow W_f, variable stator vanes VSV, and variable bleed valve VBV commands. The magnitudes of state variable x and control u are quite different in themselves and between them, therefore, scaling the system enables these variables to be in the same range. In order to scale the system, transformation matrices T_x, T_u, and T_{xu} are used. Variables associate with state vector are scaled using T_x, while variables related to the input vector are scaled using T_u. Since the augmented matrix C_{aug} is a combination of C and D matrices, scaling the C_{aug} is done through the transformation matrix T_{xu} defined as

$$
T_{xu} = \begin{bmatrix} T_x & 0 \\ 0 & T_u \end{bmatrix}
$$

To properly scale the system matrices, the state variable x and the control u (in term of the transformation matrices) are substituted in a generic state space equation for the proper forms of the scaled matrices to emerge. These forms are then used to appropriately scale the matrices. This procedure is shown as the following:

$$
z = T_x x
$$

$$x = T_x^{-1} z \tag{5.4}$$

$$\dot{x} = T_x^{-1} \dot{z} \tag{5.5}$$

For control u

$$v = T_u u$$

$$u = T_u^{-1} v \tag{5.6}$$

Substitute equations (5.4), (5.5), and (5.6) into the regular plant state space $\dot{x} = Ax + Bu$ result in

$$T_x^{-1} \dot{z} = A T_x^{-1} z + B T_u^{-1} v$$

$$\dot{z} = T_x A T_x^{-1} z + T_x B T_u^{-1} v \tag{5.7}$$

It can be seen from equation (5.7) that $A_z = T_x A T_x^{-1}$ and $B_v = T_x B T_u^{-1}$. Values of matrices A, B, C, and D are obtained from linearizing the engine system through the C-MAPSS40k graphical user interface (GUI) in a ground idle condition where the altitude is set to 1,000 feet with the Mach number of 0.1 and power lever angle (PLA) equal to 40 degrees. Also, C and D matrices only include the first row, second row, and ninth row from the linearized C and D matrices from C-MAPSS. These three rows represent N_f, Nc, and the stall margin of the high pressure compressor $SMHPC$, respectively.

5.2.2 Reference Model and The Control Law

The reference model described in equation (4.14) is redefined with the scaled augmented matrices as

$$\dot{x}_m = A_{aug}x_m + B_{aug}u_m \qquad (5.8)$$

where the reference control law is

$$u_m = -Kx_m + Pr$$

Therefore, P can be calculated as

$$P = -C_z(A_{aug} - B_{aug}K)^{-1}B_{aug}{}^{-1} \qquad (5.9)$$

Hence, the control law defined in equation (4.26) is equal to

$$v_r = (GB_{aug})^{-1}[-GA_{aug}e + GB_{aug}u_m - \eta sgn(s)]$$

Note that the control v_r in equation (5.1) is the control rate, $\dot{v}$. In addition, A_{eq} described in the tracking error dynamic in chapter 4 is redefined as $A_{eq} = [A_{aug} - B_{aug}(GB)^{-1}GA_{aug}]$

5.3 Values and Dimensions Assignment of System Parameters

The control law is redefined to make it suitable for the engine system and now matrix dimensions are assigned to the matrix variables in the control logic. For simplicity, matrix dimensions are summarized in Table 5.1, where n represents the number of states and m represents the number of inputs. In this case, there are five

Table 5.1: Dimension of Matrix Parameters

Variable	Dimension
A_{aug}	nxn
B_{aug}	nxm
C_z	mxn
G	mxn
K	mxn
P	mxm
e	$mx1$
r	$mx1$
s	$mx1$
u_m	$mx1$
v_r	$mx1$
x_{aug}	$nx1$
x_m	$nx1$

states (N_f, N_c, and three zero states) and three inputs (fuel flow rate W_f, variable stator vane angle VSV, and variable bleed valve position VBV).

The values of linearized A, B, C, and D matrices obtained from C-MAPSS40k in a ground idle condition are shown in Table 5.2 along with the values of the transformation matrices. Note that C and D matrices only contain the first, second, and ninth row of the original linearized state space.

The gain matrices K, G, and P are calculated as the following:

- K is determined via linear quadratic regulator (LQR), which is an optimal method where state variable deviations and the control effort are weighted [21]. The calculated matrix K will minimize the quadratic cost function:

$$J(u) = \int_0^\infty (x^T Q x + u^T R u + 2 x^T N u) dt$$

Where Q is defined as an identity matrix of 5x5, and R is equal to a 3x3 diagonal matrix with diagonal values of 1, 1, and 10. By choosing the values of Q and R appropriately, the state variables are normalized to change in the same range.

Table 5.2: Values of Linearized State Space Matrices and Transformation Matrices

Variable	Value		
A	$\begin{bmatrix} -3.3808 & 1.2954 \\ 0.4444 & -3.0501 \end{bmatrix}$		
B	$1 * 10^3 \begin{bmatrix} 0.6678 & -0.0392 & -0.0142 \\ 1.3340 & 0.1173 & -0.0268 \end{bmatrix}$		
C	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0.0158 & -0.0037 \end{bmatrix}$		
D	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -10.9483 & 0.1837 & -0.4766 \end{bmatrix}$		
T_x	$\begin{bmatrix} \frac{1}{500} & 0 \\ 0 & \frac{1}{1000} \end{bmatrix}$		
T_u	$\begin{bmatrix} \frac{1}{5} & 0 & 0 \\ 0 & \frac{1}{100} & 0 \\ 0 & 0 & 1 \end{bmatrix}$		

- G is determined through a pole placement method using a Matlab function obtained from [22]. This Matlab function is called "vplace". The selected pole will give desired properties in sliding mode. The desired poles are eventually the sliding mode poles. The number of poles correspond to the number of states minus the number of inputs. Therefore, there are two sliding mode poles in this system. Values of the desired poles are -3 and -2.5.

- P is calculated using equation (5.9).

5.4   Control Law Verification

The control law is modeled in Matlab Simulink as shown in Figure 5.2. The reference inputs are the difference between the desired values (i.e., target values) and the initial values (i.e., zero value) of fan speed N_f, core speed N_c, and stall margin of the high pressure compressor $SMHPC$. These values are obtained via C-MAPSS40k closed-loop GUI at the altitude of 1,000 ft, Mach number of 0.1, delta

ambient temperature of 0, power lever angle of [55 55 75 75] in the time vector [0 15 20 30]. The engine mode is also set to fan speed control mode and the engine limiters are disabled. The difference between desired value and the initial values is called the delta value. Table 5.3 shows the desired, initial, and delta values of N_f, N_c, and $SMHPC$.

Table 5.3: Defined Desired, Initial and Delta Values of N_f, N_c, and SMHPC

Variable Name	Value	Δ Value
Nf_target	3739.4 rpm	
Nf_zro	3109.1 rpm	$\Delta N_f = 630.3$
Nc_target	11417 rpm	
Nc_zro	10584 rpm	$\Delta N_c = 833$
SMHPC_target	20	
SMHPC_zro	30.94	Δ SMHPC = -10.94

It can be seen from Figure 5.2 that the saturation functions used instead of the signum functions are multiplied by two gain blocks. These gains, called phi and eta (discussed in Chapter 4), are used to adjust the saturation functions to suppress chattering and overshoot in the system outputs. The values of phi are 0.1, 0.05, and 0.1, and the values of eta are 15, 5, and 25 for the first, second, and third saturation function respectively. Note that the values of phi are used as the reciprocal. To simulate the control law modelled in Figure 5.2, the Matlab command shown in appendix A are executed to define parameter values and run the Simulink file. The control model is simulated for 30 seconds and the results are shown in Figure 5.3 through Figure 5.14.

It can be seen from Figure 5.3 to Figure 5.5 that the actual outputs perfectly tracked the desired values of ΔN_f, ΔN_c, and $\Delta SMHPC$ shown in Table 5.3. The system states also accurately tracked the reference states as seen in Figure 5.6 to Figure 5.10. The values of W_f, and VBV commands (Figure 5.11 and Figure 5.13) are in an acceptable range. However, the VSV command (Figure 5.12) are out of its range due to the value of the fifth reference state representing VSV command being

Figure 5.2: Diagram of The Sliding Mode Control Modelled in Simulink
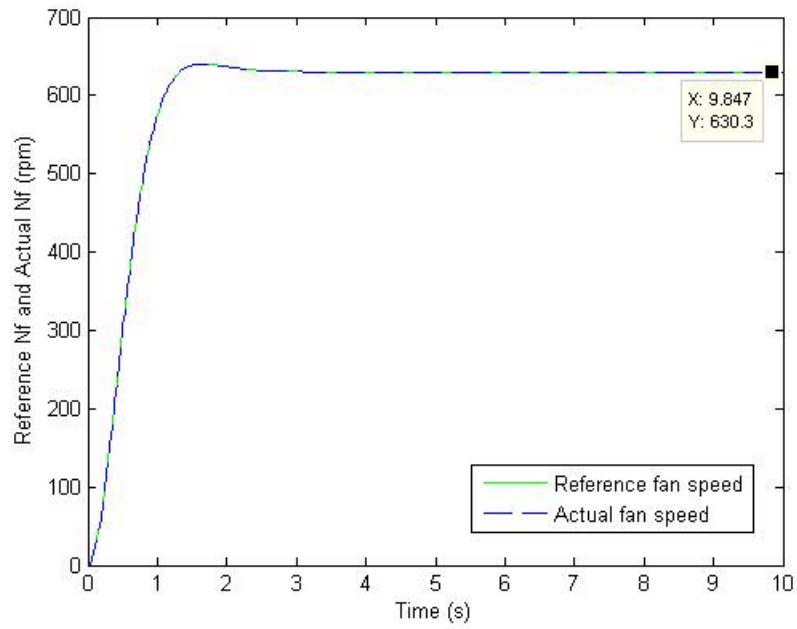
Figure 5.3: Desired and Actual Values of ΔN_f vs. Time
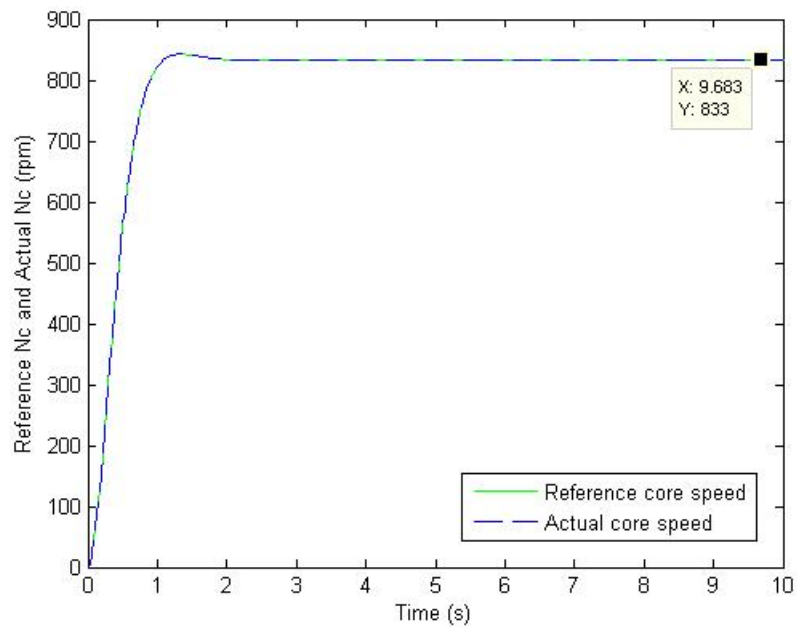
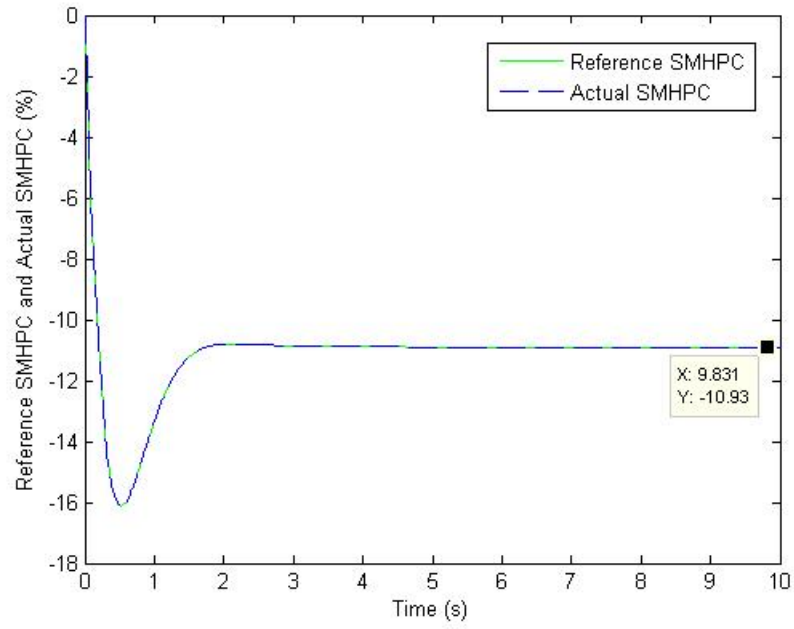Figure 5.4: Desired and Actual Values of ΔN_c vs. Time

Figure 5.5: Desired and Actual Values of $\Delta SMHPC$ vs. Time
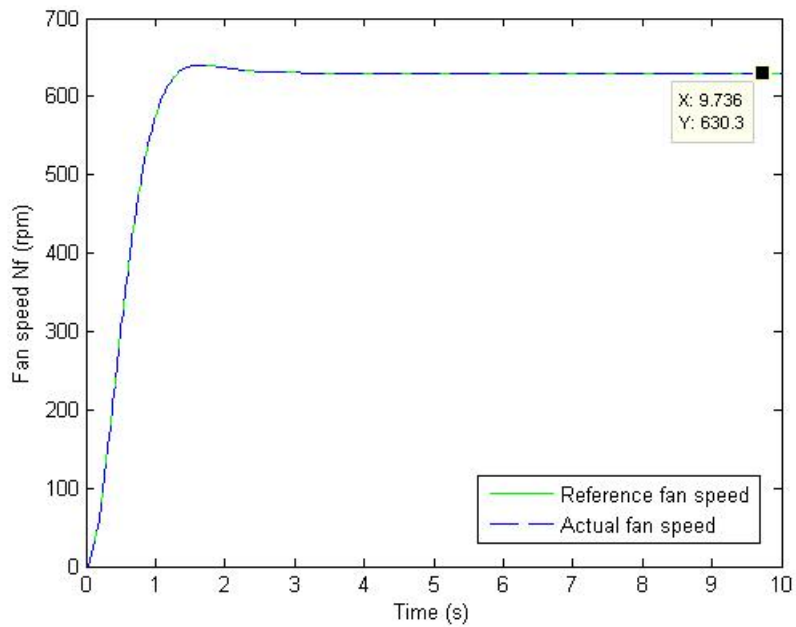

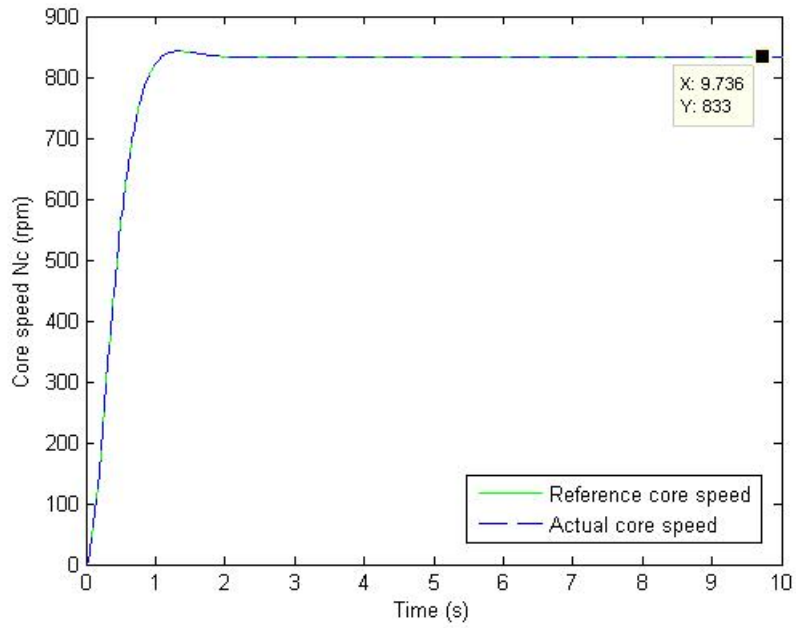Figure 5.6: Reference State and Actual State of ΔN_f vs. Time

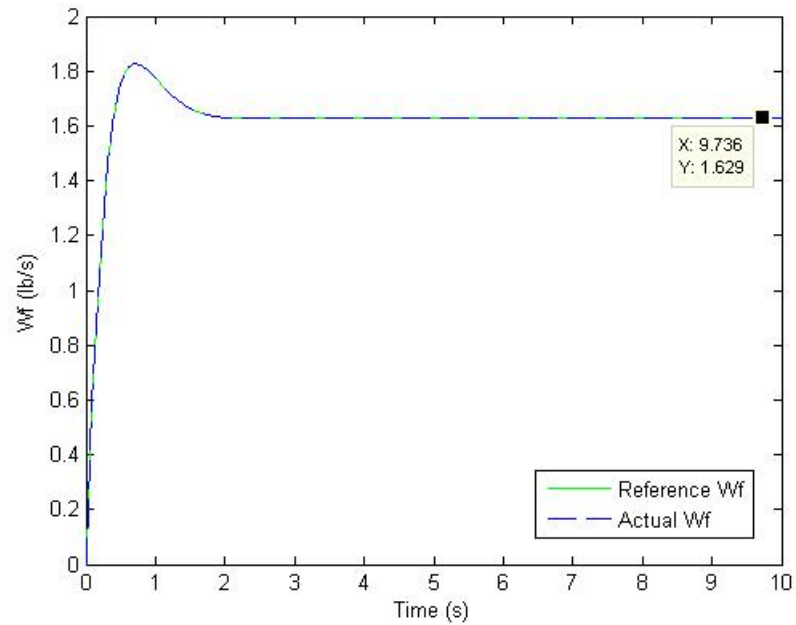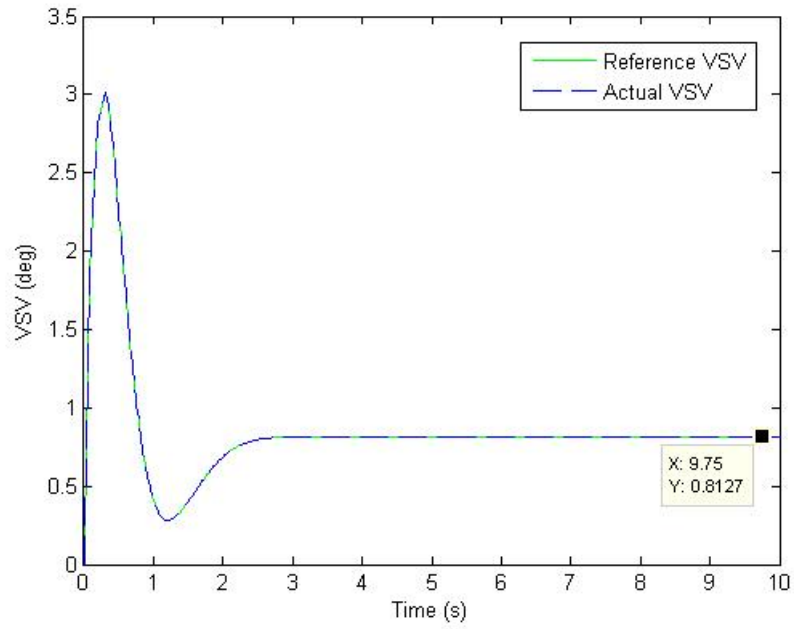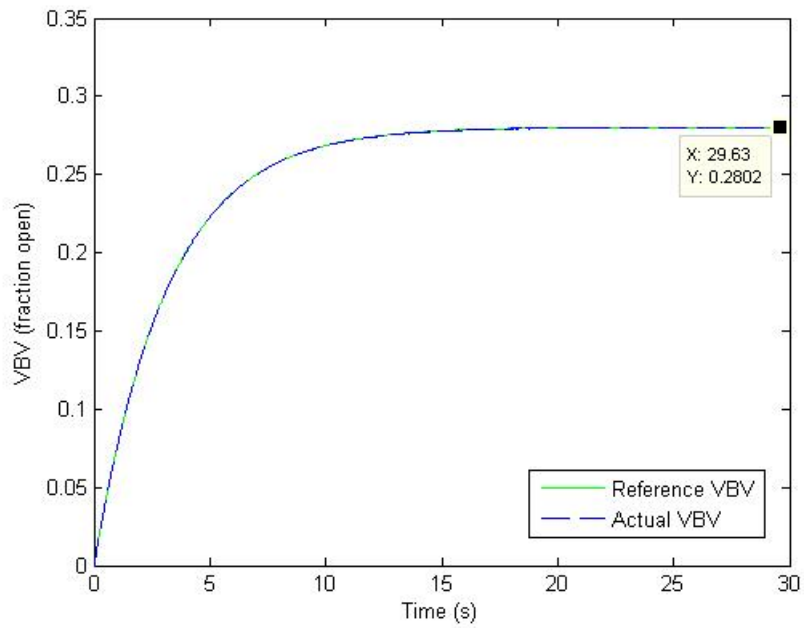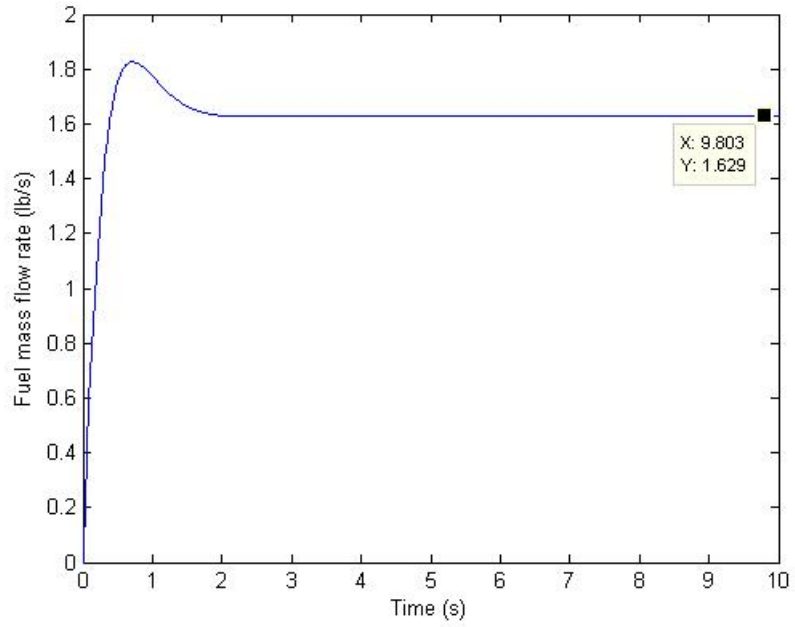Figure 5.7: Reference State and Actual State of ΔN_c vs. Time

Figure 5.8: Reference State and Actual State of ΔW_f vs. Time

Figure 5.9: Reference State and Actual State of VSV Command vs. Time

Figure 5.10: Reference State and Actual State of VBV Command vs. Time

Figure 5.11: Fuel Flow Rate Command vs. Time
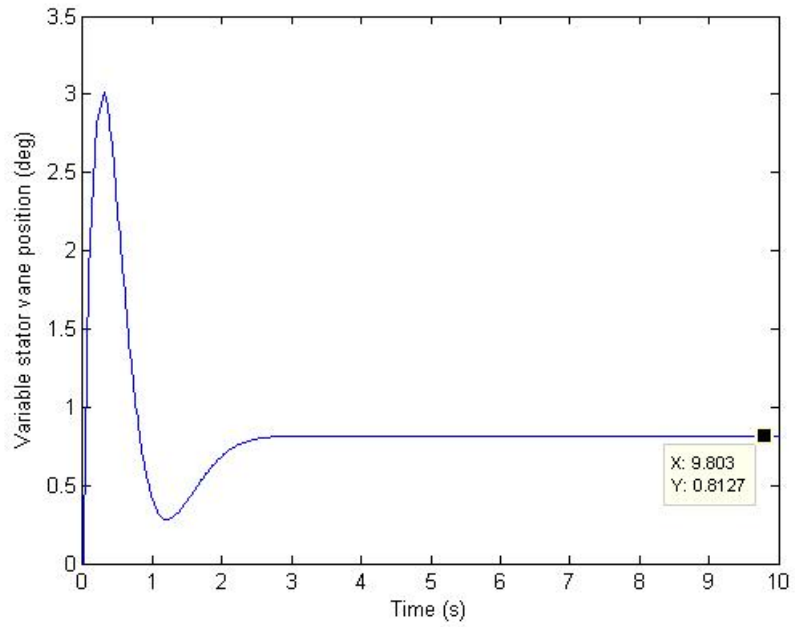
Figure 5.12: VSV Command vs. Time
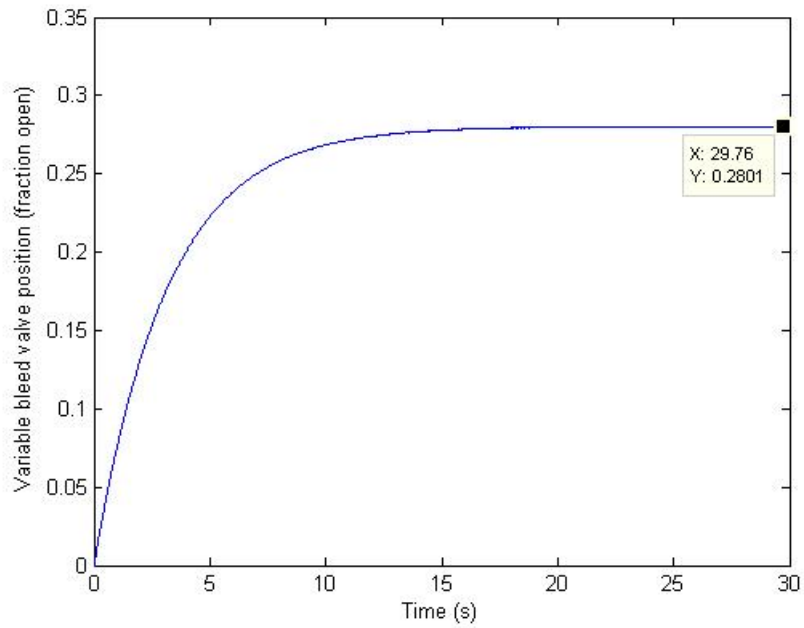
Figure 5.13: VBV Command vs. Time
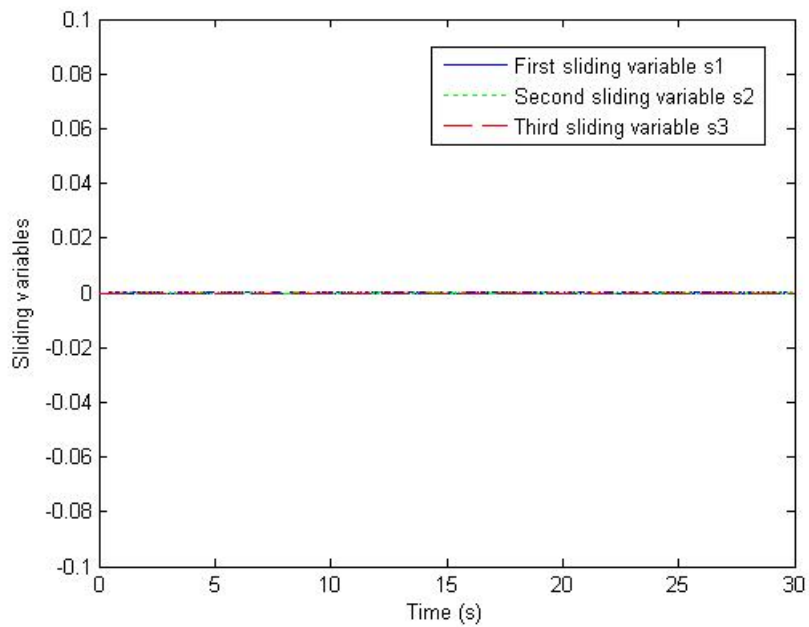
Figure 5.14: Sliding Variables vs. Time)

out of the range. This is due to the use of the linearized engine model. When the control law is implemented in the C-MAPSS, the VSV command will fall within its range since the the actual engine model is used. Finally, the values of sliding variables shown in Figure 5.14 are very close to zero, confirming that the control law is working correctly. It must be restated that the values of sliding variable will not equal to zero during sliding mode since the sgn function is replaced with the saturation function. Therefore, the values of sliding variable are only trapped in the vicinity of the sliding surface where $s = 0$.

The results obtained from simulating the control law modelled in Simulink verify that the developed control logic is working properly. This modelled controller is now ready to be implemented in the C-MAPSS40k.

5.5   Sliding Mode Controller in C-MAPSS40k

The digital baseline controller shown in Figure 3.11 is replaced with the sliding mode controller as seen in Figure 5.15. Figure 5.16 shows the sliding mode control architecture inside the SMC block in Figure 5.15.

Since the inputs $(N_f, N_c, and SMHPC)$ of the reference model are incremental values (i.e., the amount of increment needed to add to the initial value in order to achieve the desired value), the control values $(W_f, VSV, and VBV)$ are also incremental values as well. Therefore, in Figure 5.15 the initial values of W_f, VSV, and VBV are added to the delta values of W_f, VSV, and VBV before being fed into the Actuators block.

In order to run the C-MAPSS40k with the sliding mode controller, some of the engine parameters are needed to be defined through the GUI. The settings in the GUI are determined as the following: the altitude of 1,000 ft, Mach number of 0.1, dTamb of 0, the PLA vector = [55 55 75 75], fan speed control mode is selected, and
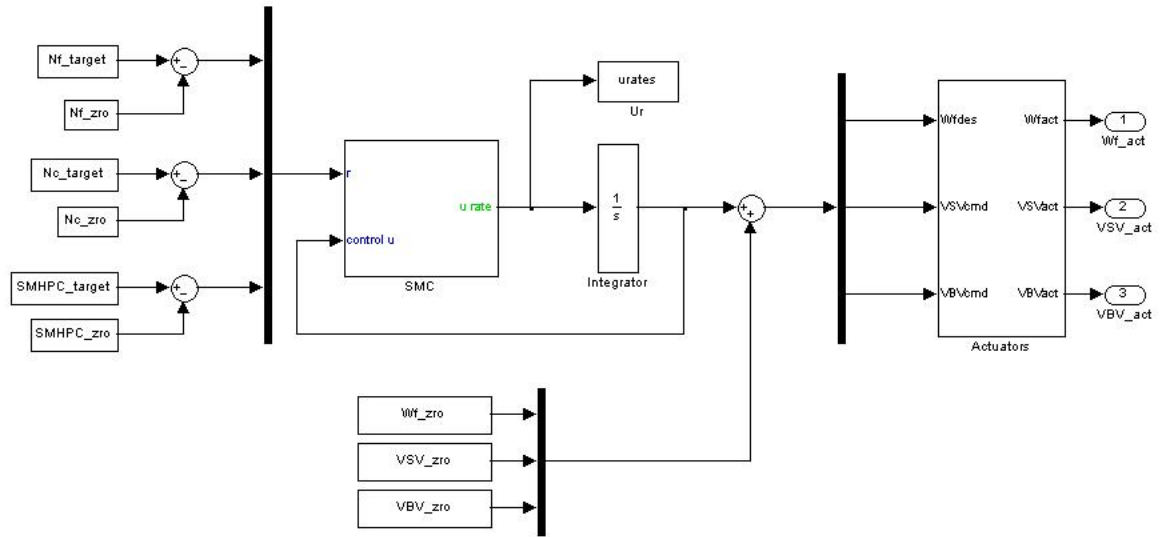
Figure 5.15: Diagram of the Sliding Mode Controller Implemented in the C-MAPSS40k

engine limiters are in-active. The sliding mode control C-MAPSS40k is then executed by the Matlab command shown in appendix A. The controller is adjusted to give a better performance by changing the values of Φ and η. Through trial and error, the best values of Φ are 11, 13.2, and 10, whereas the best values of η are 48, 25, and 25. The system is simulated for 50 seconds and the simulation results are shown in Figure 5.17 through Figure 5.23.

The simulation results show that sliding mode controller is able to drive the N_f, N_c, and $SMHPC$ values close to the desired values, since the values of sliding variables (Figure 5.23) are very close to zero. Due to the use of the saturation function instead of the signum function to suppress chattering, the obtained values of N_f, N_c, and $SMHPC$ are not equal to the exact desired values because ideal sliding mode no longer occurs. Percent errors are calculated for N_f, N_c, and $SMHPC$, which are 0.1224%, -0.2972%, and 0.0360%, respectively. The values of percent errors are very small, therefore the performance of the sliding mode control implemented in the C-MAPSS40k is acceptable. Figures 5.20 through 5.22 show values of the command
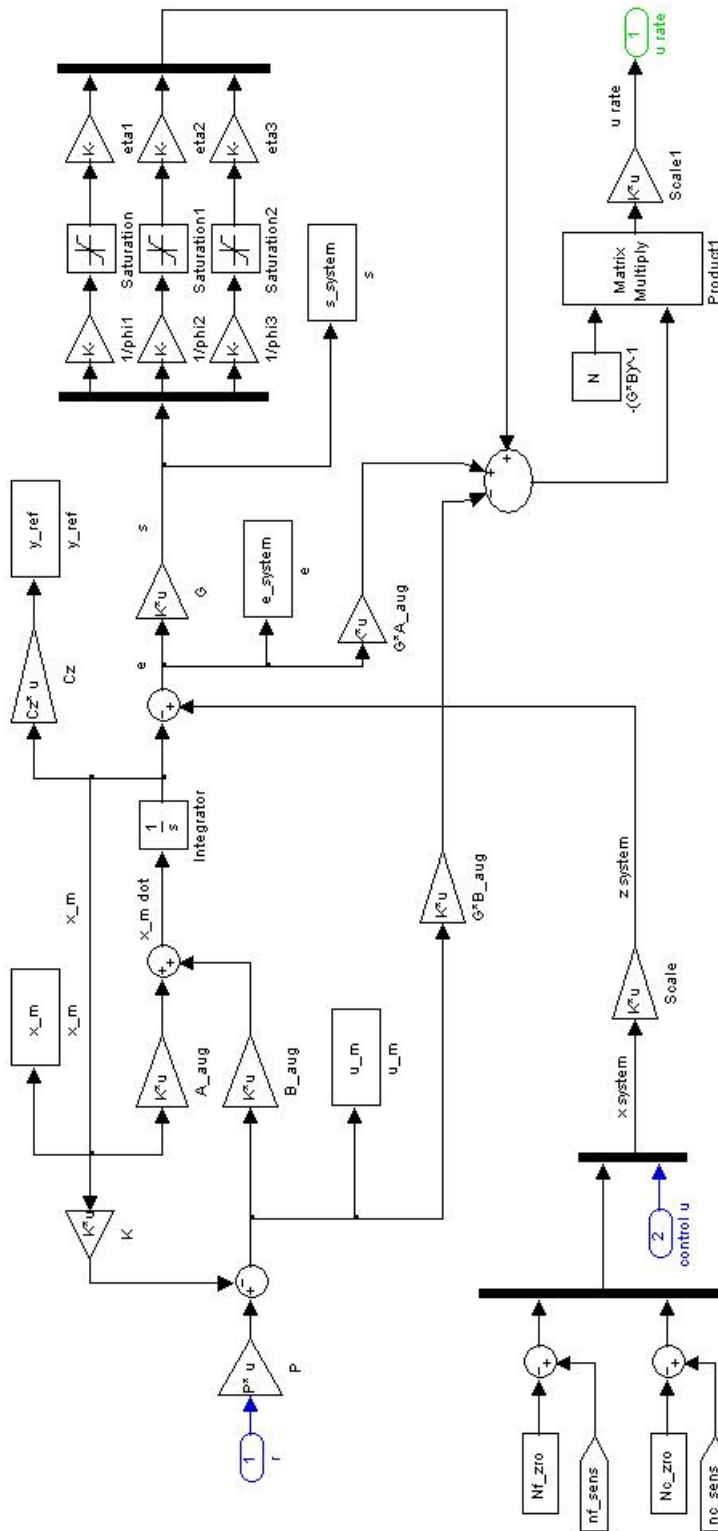
67

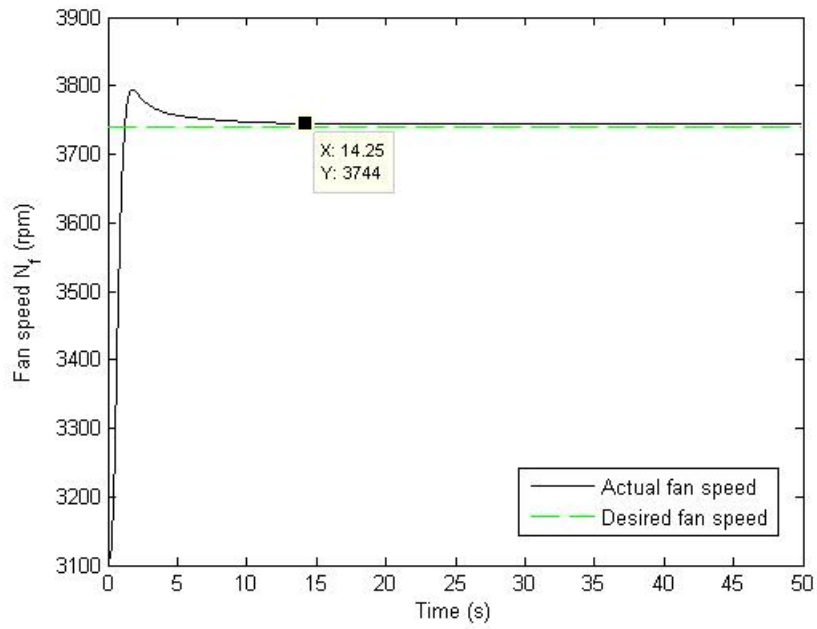Figure 5.16: Diagram of the Sliding Mode Control Logic Inside the SMC Block

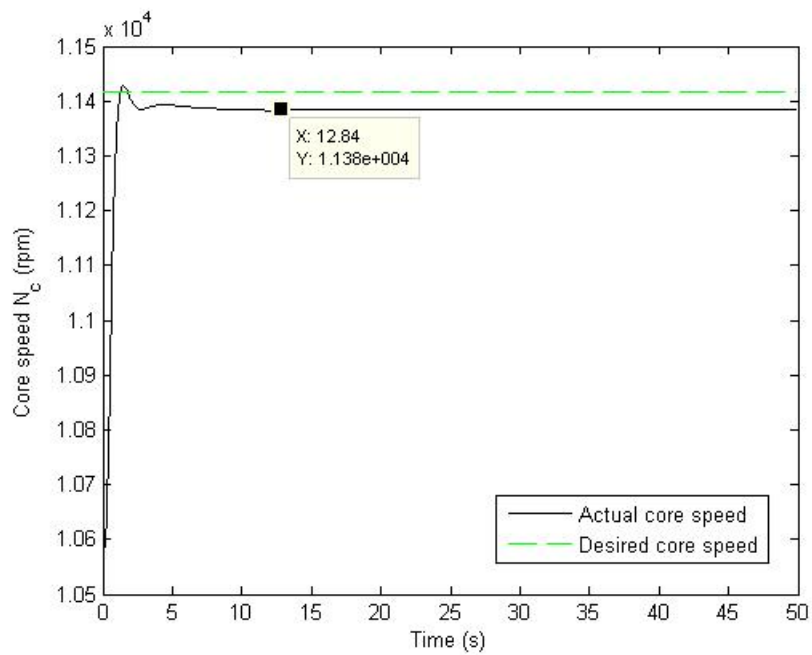Figure 5.17: Fan Speed N_f vs. Time
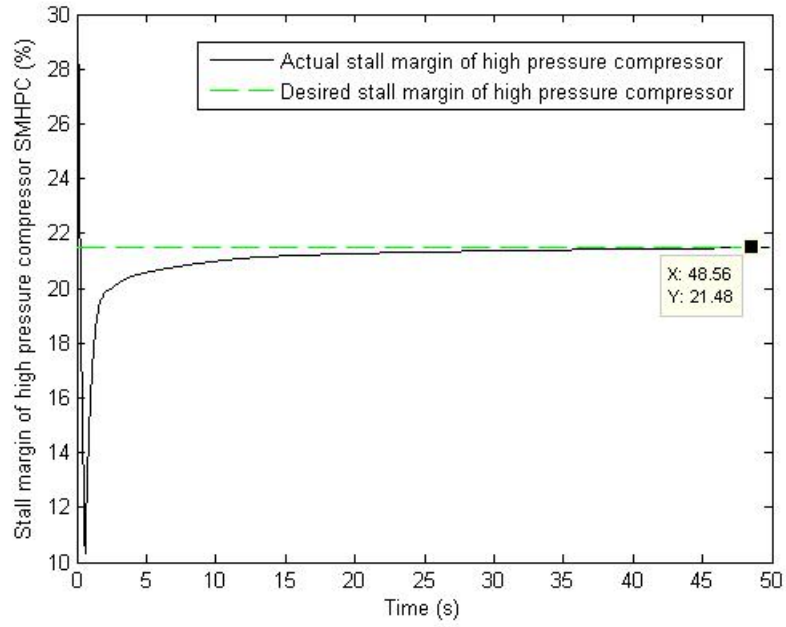

Figure 5.18: Core Speed N_c vs. Time

Figure 5.19: Stall Margin of High Pressure Compressor (SMHPC) vs. Time
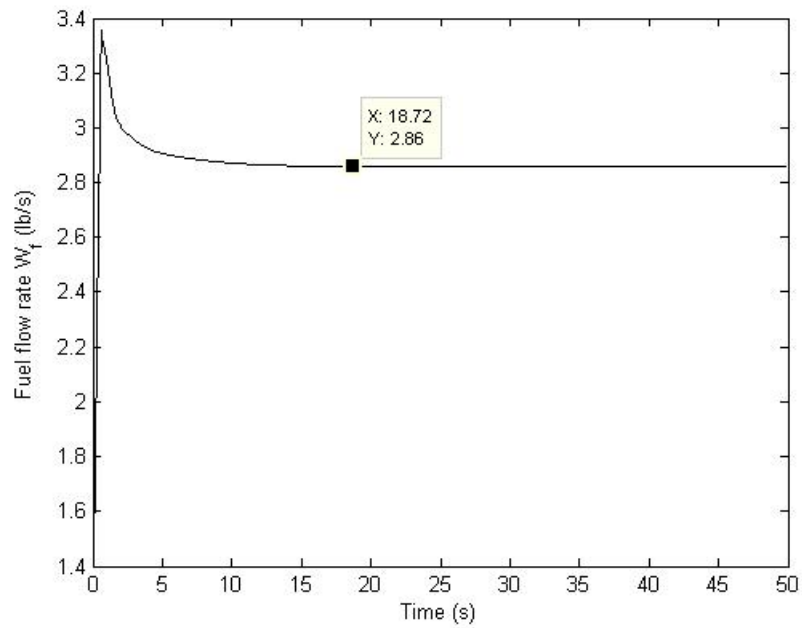

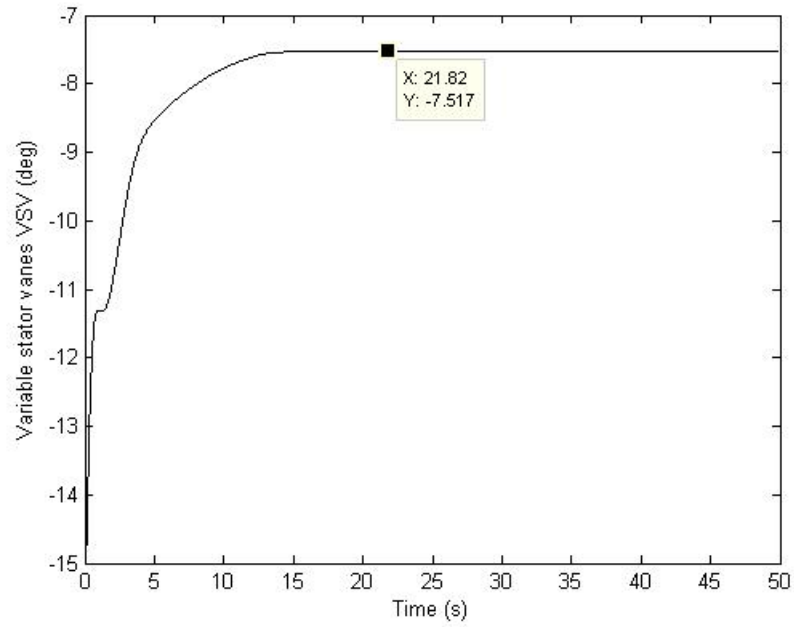Figure 5.20: Control Input of Fuel Flow Rate vs. Time

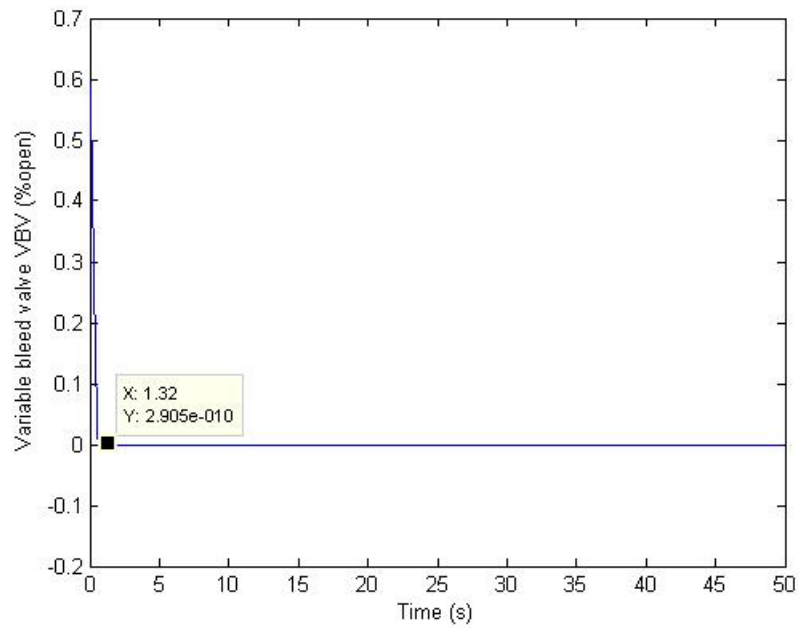Figure 5.21: Control Input of Variable Stator Vanes vs. Time

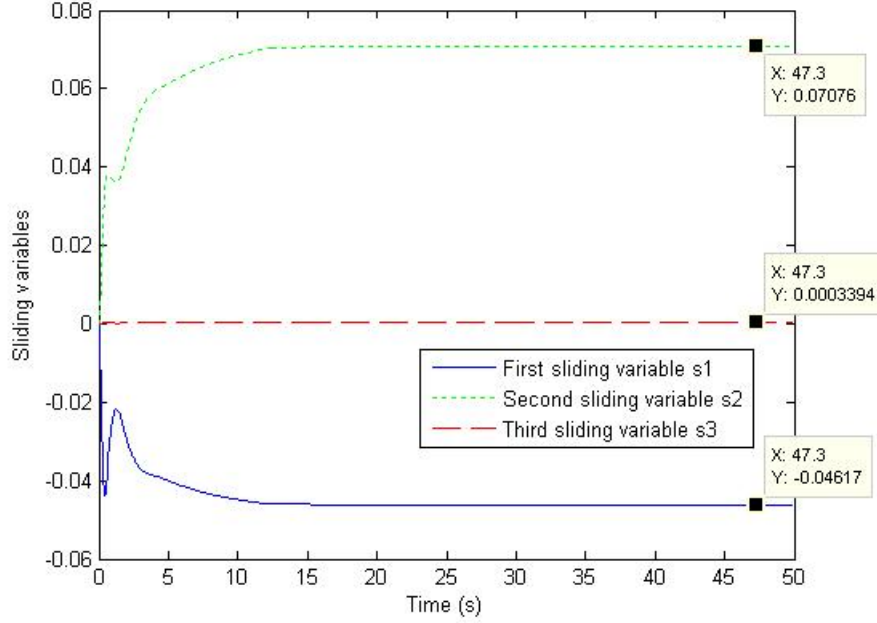Figure 5.22: Control Input of Variable Bleed Valve vs. Time

71

Figure 5.23: Sliding Variables vs. Time

inputs of W_f, VSV, and VBV that injected into the actuators. To verify if the value of W_f is reasonable, the specific fuel consumption (SFC) is calculated as follows:

$$SFC = \frac{W_f \ (lb/hr)}{F_{net} \ (lbf)} \tag{5.10}$$

Where F_{net} is the net thrust. Hence, SFC for the W_f of 2.86 lb/s and F_{net}=28,813 lbf is equal to 0.3573. The average value of SFC for the turbofan engine is approximately 0.3 [24], therefore the value of W_f is justified. Furthermore, the VSV and the VBV values are also within their operational ranges of -62.412 to 0.61 degrees for VSV, and 0 to 1 for VBV value. The VBV value is very close to zero, meaning the VBV can be interpreted as being in the fully closed position. Overall, the sliding mode controller is able to achieve the desired values of N_f, N_c, and $SMHPC$ with very small percent errors while maintaining the engine's integrity. In the next section, the performance of the sliding mode controller is compared with the digital baseline controller that comes with the C-MAPSS40k.

5.6 Comparison Between the Sliding Mode Controller and the Baseline Controller

The C-MAPSS40k is run with the same condition used for running the sliding mode controller in the previous section. This time, however, the controller is the digital baseline controller, which comes with the C-MAPSS40k. The simulation is executed for 50 seconds, and the results are shown in Figure 5.24 through Figure 5.29.

It is important to remember that the transient response of the engine started at 15 seconds instead of zero second. Figure 5.24 and Figure 5.25 show that the baseline controller is able to perfectly drive the values of N_f and N_c to their desired values. However, the value of $SMHPC$ is not settling and still increases over time. The simulation is re-run for 100 seconds to clearly see the tend of the $SMHPC$ value as shown in Figure 5.26. The fuel flow rate is equal to 2.832 lb/s (Figure 5.27) and the net thrust is equal to 28,712 lbf, therefore SFC is calculated to be 0.3551. In Figure 5.28 and Figure 5.29, the values of VSV and VBV commands are in their operational ranges. The VBV value is very close to zero, thus, the VBV is nearly in the closed position. Comparisons between the settling time of system parameters between sliding mode controller and the baseline controller are shown in Table 5.4.

Table 5.4: Settling Time of System Parameters

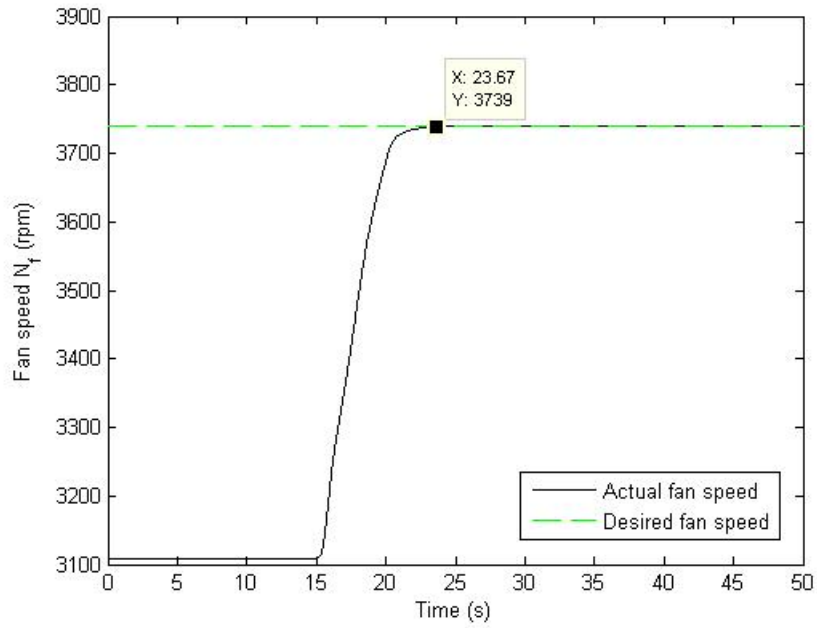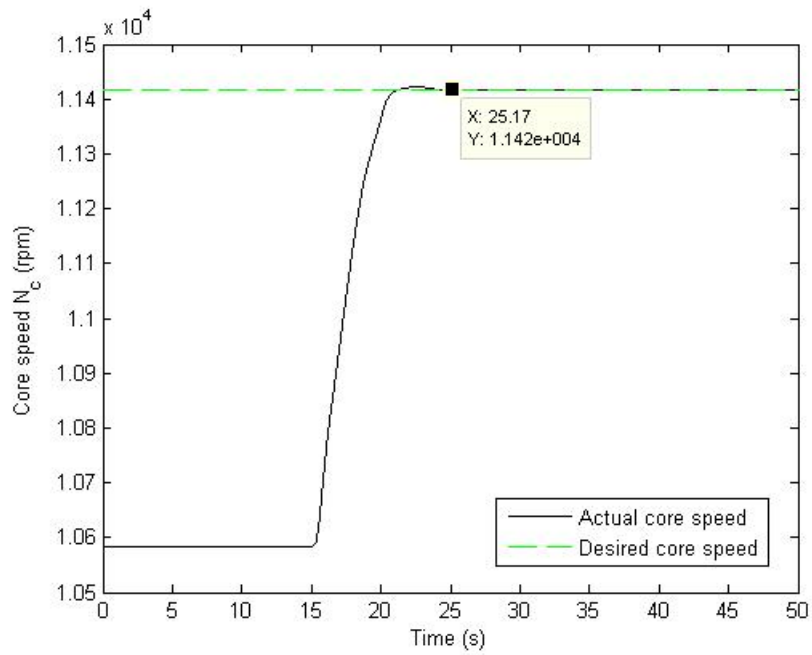| Settling Time (s)		
Variable	Sliding Mode Controller	Baseline Controller
N_f	14.25	8.67
N_c	12.84	10.17
$SMHPC$	N/A	N/A
W_f	18.72	10.97
VSV	21.82	30.8
VBV	1.32	14.77

Figure 5.24: Fan Speed N_f vs. Time

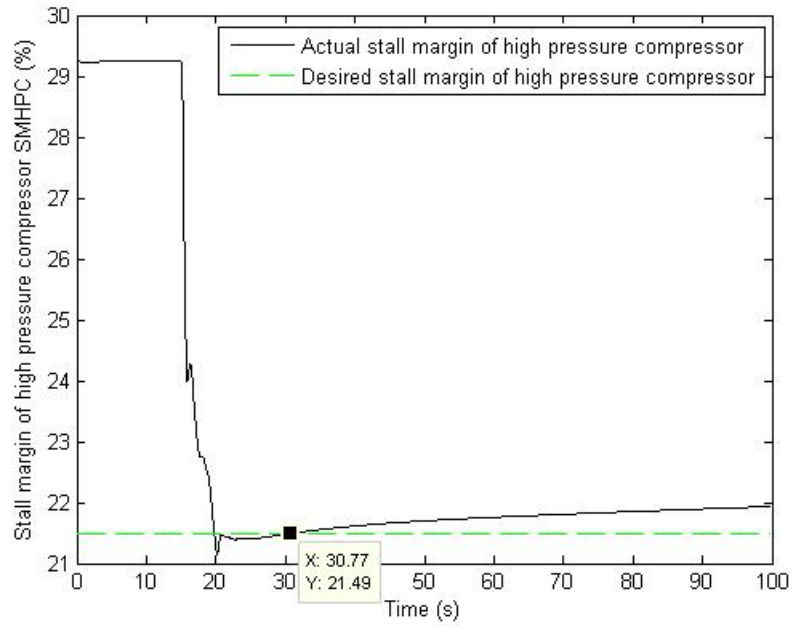Figure 5.25: Core Speed N_c vs. Time

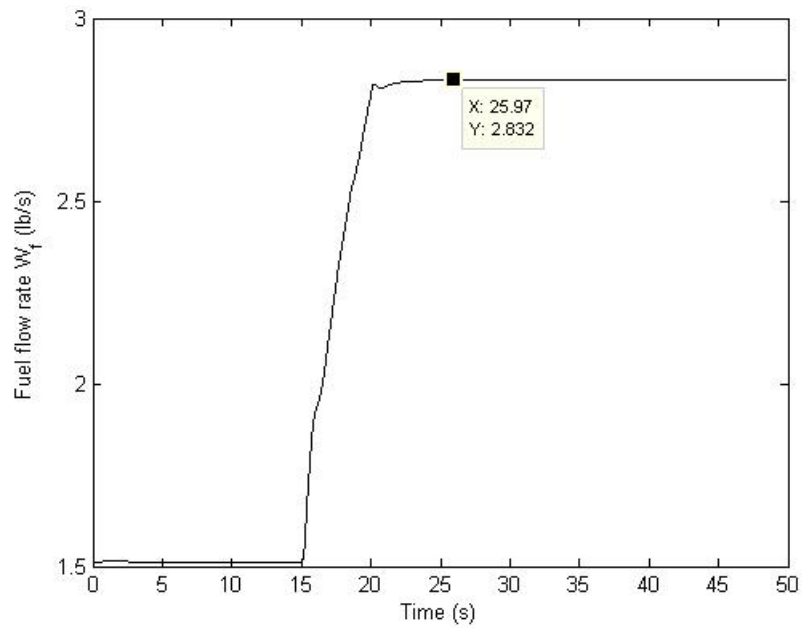Figure 5.26: Stall Margin of High Pressure Compressor (SMHPC) vs. Time

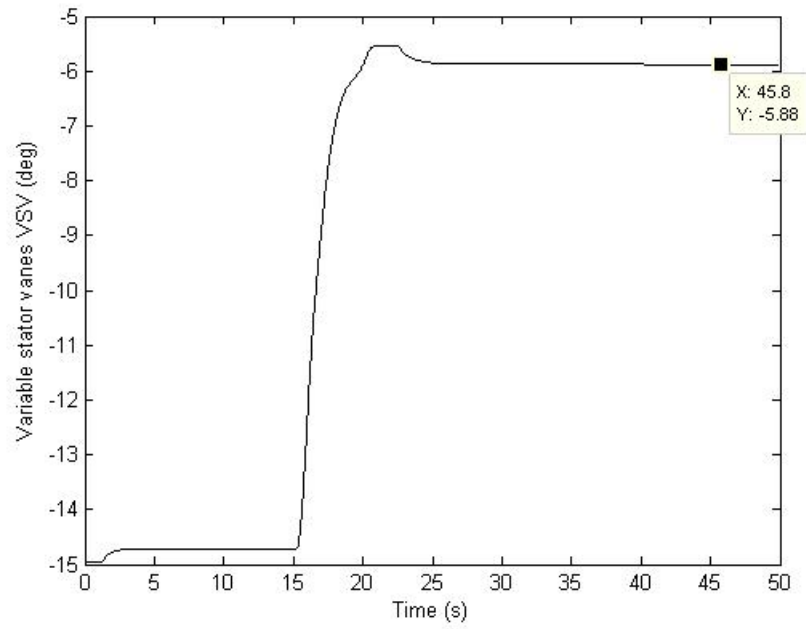Figure 5.27: Control Input of Fuel Flow Rate vs. Time

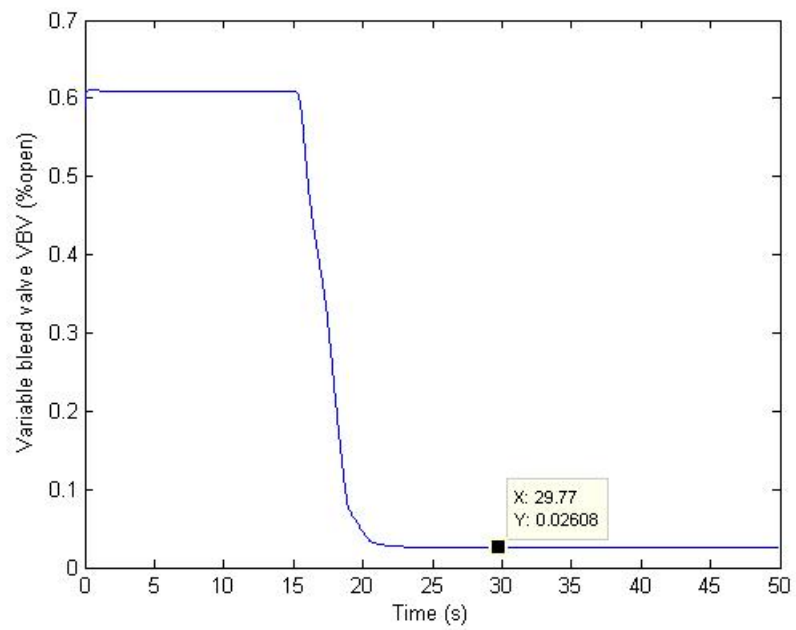Figure 5.28: Control Input of Variable Stator Vanes vs. Time

Figure 5.29: Control Input of Variable Bleed Valve vs. Time

It can be seen from Table 5.4 that the sliding mode controller has longer settling times for N_f, N_c, and W_f compared to the baseline controller. On the other hand, settling times for VSV and VBV are shorter in sliding mode controller than the baseline controller. From the control law perspective, sliding mode controller simplify the complexity of the baseline controller by eliminating the power management for the W_f and the schedule control for the VSV and the VBV. Therefore, there is no need for the the lookup tables used in the schedule control, which saves time and effort in making these tables. As long as the initial and the desired values of the N_f, N_c, and $SMHPC$ for specific flight conditions are known, sliding mode control logic developed in this thesis is a viable candidate to control the turbofan engine.

5.7 Evaluate the Robustness of Sliding Mode Control

The developed sliding mode controller is created based on the linearized model of the engine at the ground idle condition. To evaluate the robustness of the designed controller, two other conditions are applied to the engine. These conditions are

- Maximum climb: altitude of 6,000 ft, Mach number of 0.4, and PLA of 73

  where

  Nf_target = 3,814.2 rpm

  Nf_zro = 3,178.5 rpm

  Nc_target = 11,442 rpm

  Nc_zro = 10,576 rpm

  SMHPC_zro = 31.2156 %

  SMHPC_target = 21.6898 %

- Maximum take-off: altitude of 1,000 ft, Mach number of 0.15, and PLA of 78

where

Nf_target = 3,803.4 rpm

Nf_zro = 3,116.2 rpm

Nc_target = 11,527 rpm

Nc_zro = 10,584 rpm

SMHPC_zro = 31.1528 %

SMHPC_target = 20.7210 %

The target and the zero values of N_f, N_c, and $SMHPC$ for the ground idle condition defined in the Matlab file are replaced with one of these conditions while the rest of the controller parameters (that based on the ground idle condition) stay the same. The system is simulated for 50 seconds. Figure 5.30 through Figure 5.36 show simulation results for the maximum climb condition, while Figures 5.37 through 5.43 show the simulation results for the maximum take-off condition.

The sliding mode controller is able to cope with the change in the system parameters and drive the values of N_f, N_c, and SMHPC close to the desired values. Due to non-ideal sliding mode, the value of sliding variables (Figure 5.36 and Figure 5.43) are only enforced in the vicinity of the ideal sliding surface of s=0. This hinders the perfect tracking of the system. The percent errors for N_f, N_c, and SMHPC values are calculated and shown in Table 5.5. The percent errors are less than 10%, therefore, the values of N_f, N_c, and SMHPC are in an acceptable range. Values of W_f, VSV, and VBV from both conditions are also in their operational ranges. The simulation results from both of the conditions, indicating that the sliding mode controller is capable to cope with the change in the condition of flight. Therefore, the sliding mode controller still maintains its robust property.

Figure 5.30: Fan Speed N_f vs. Time

Figure 5.31: Core Speed N_c vs. Time

Figure 5.32: Stall Margin of High Pressure Compressor (SMHPC) vs. Time

Figure 5.33: Control Input of Fuel Flow Rate vs. Time

Figure 5.34: Control Input of Variable Stator Vanes vs. Time
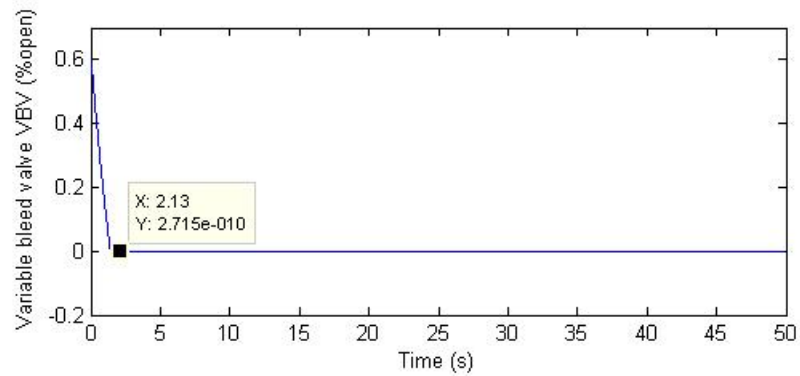
Figure 5.35: Control Input of Variable Bleed Valve vs. Time
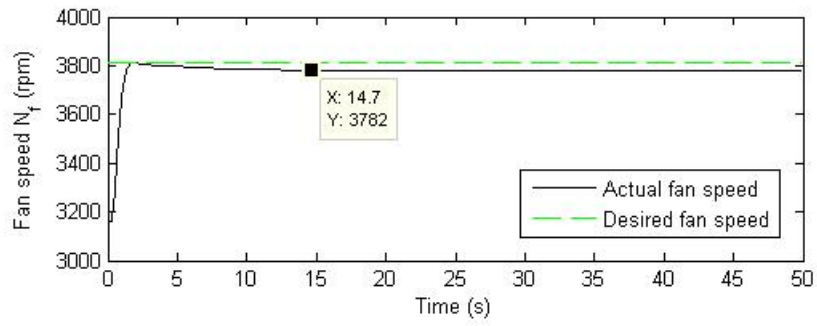
80
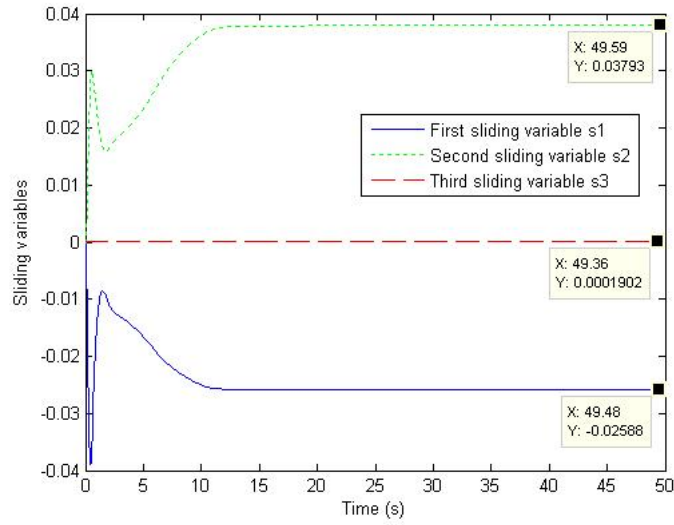
Figure 5.36: Sliding Variables vs. Time

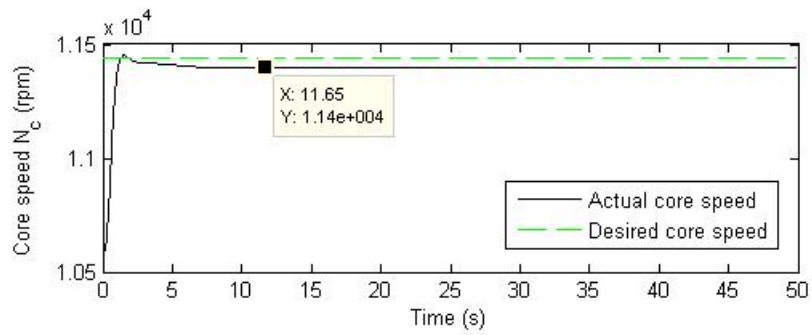Figure 5.37: Fan Speed N_f vs. Time
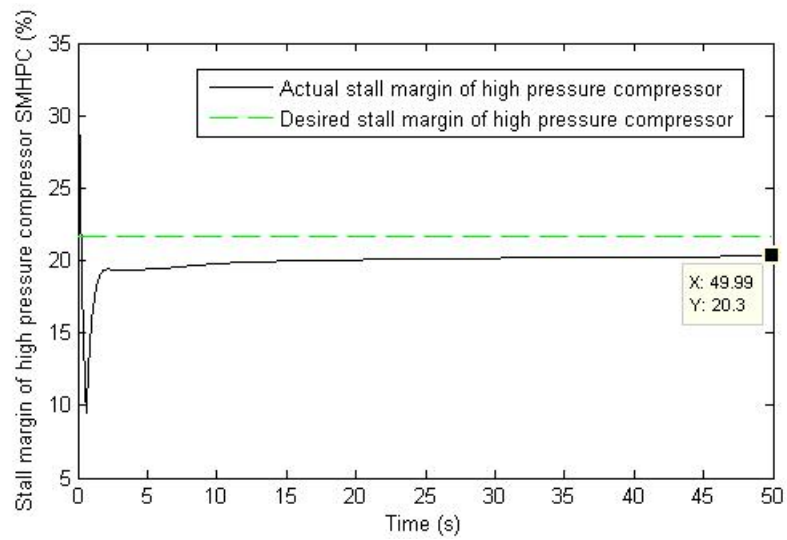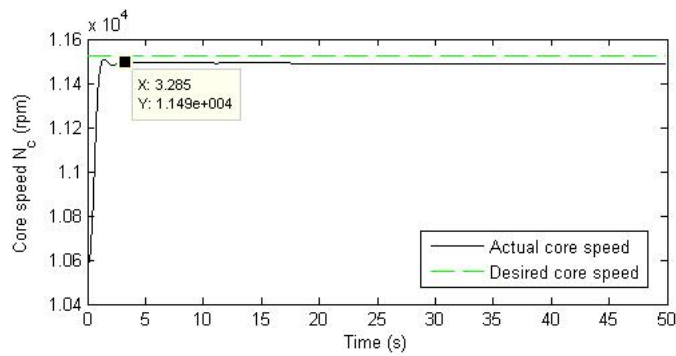

Figure 5.38: Core Speed N_c vs. Time
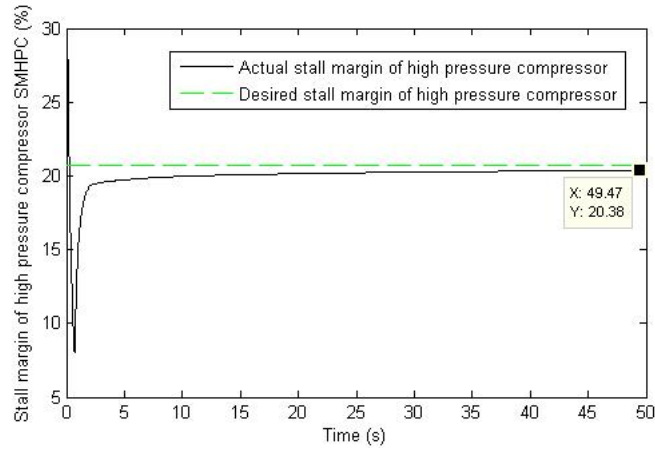
Figure 5.39: Stall Margin of High Pressure Compressor (SMHPC) vs. Time
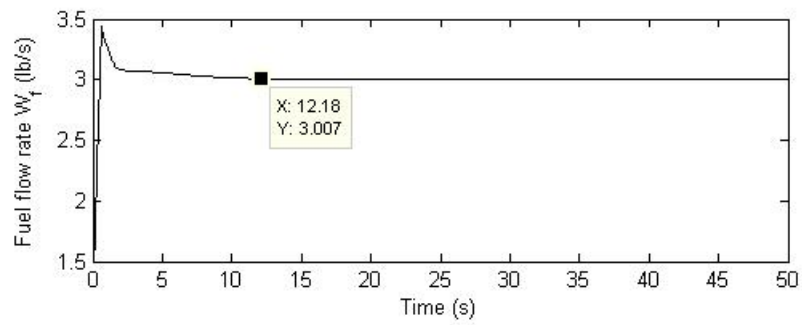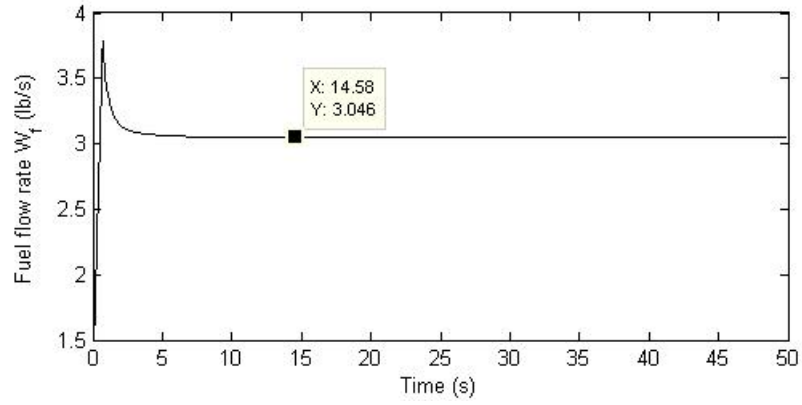

Figure 5.40: Control Input of Fuel Flow Rate vs. Time
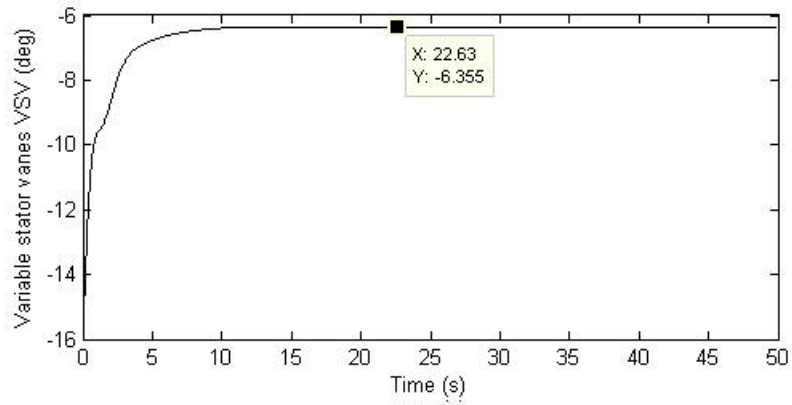
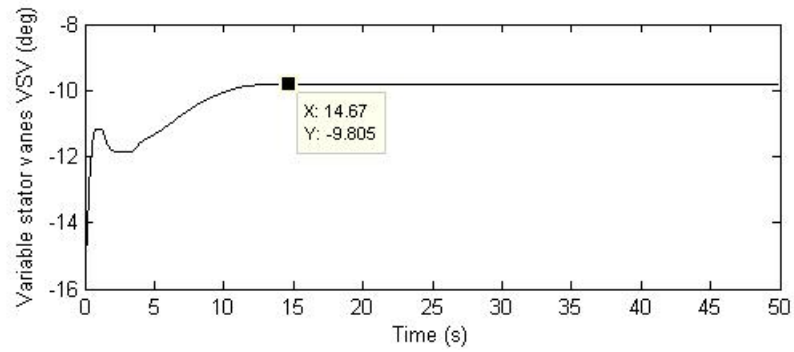Figure 5.41: Control Input of Variable Stator Vanes vs. Time

82
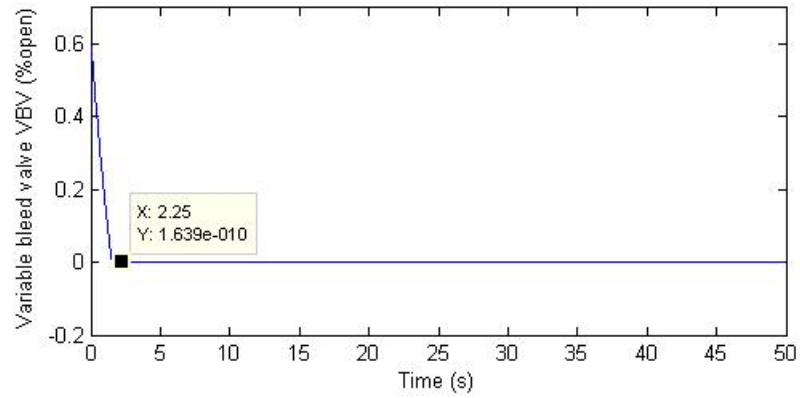
Figure 5.42: Control Input of Variable Bleed Valve vs. Time
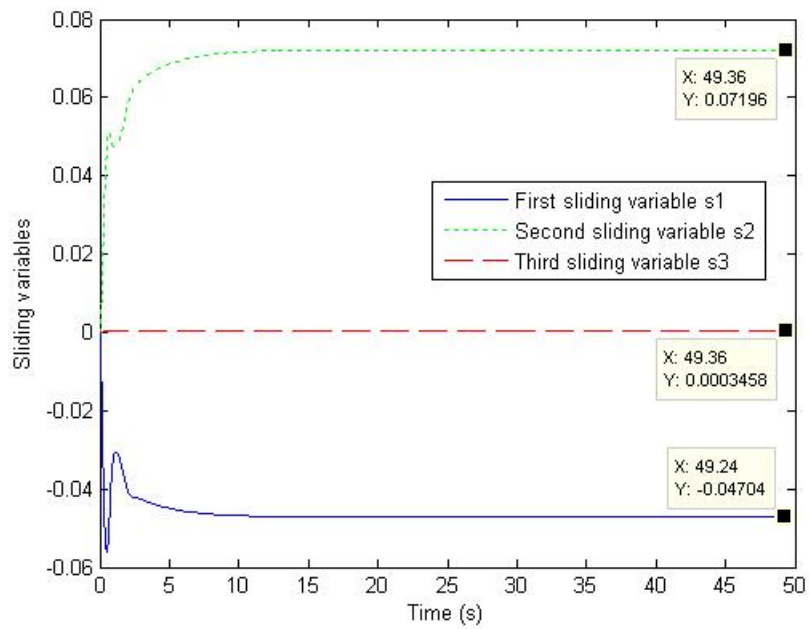
Figure 5.43: Sliding Variables vs. Time
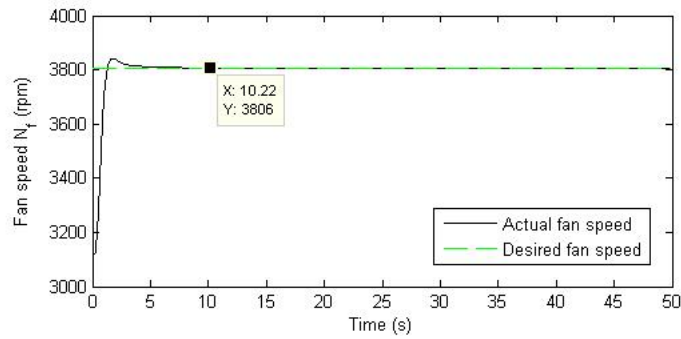
Table 5.5: Percent Error of N_f, N_c, and SMHPC

Percent Error (%)		
Variable	Maximum Climb	Maximum Take-Off
N_f	-0.8362	0.0596
N_c	-0.3805	-0.3142
$SMHPC$	-6.4136	-1.6241

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

6.1   Conclusions

This thesis has introduced and developed a guideline in creating a controller for an aircraft engine using a multivariable sliding mode control. The chattering issue, which is a major set back for sliding mode control, is successfully suppressed utilizing the boundary layer method. The sliding mode controller is implemented in the C-MAPSS40k to validate its functionality. The simulation results proves that the controller is able to control the turbofan engine, even when flight conditions changed.

By utilizing the sliding mode controller, the control architect of the engine is simplified and the need for the lookup tables used in baseline controller is eliminated. Additionally, the sliding mode controller can compensate for changes in engine characteristics over time due to deterioration, while the baseline controller does not. However, the baseline controller is able to achieve perfect tracking whereas the sliding mode controller can only drive the system parameters close to the desired values. Ideal sliding mode no longer occurs in the designed controller and perfect tracking cannot be accomplished because the signum function is replaced with the saturation

function in order to remove the chattering effect. Even though the exact desired values are not acquired by the sliding mode controller, the calculated percent errors for W_f, VSV, and VBV are very small. Ultimately, the sliding mode controller is fully capable of controlling the aircraft engine.

However, there are two drawbacks in the developed sliding mode controller. The first drawback is the construct of the reference model that based on the linearized model of the C-MAPSS40k engine at a particular condition. Even though the sliding mode control is robust, if the condition of the engine is far too different than the condition (that the reference model is base upon), the actuators will start to saturate and the engine will no longer work properly. To prevent this from happening, the reference model can be a gain scheduled or ridded of by utilizing an adaptive control. Doing so will create the second drawback in designed sliding mode controller, because it defeats the purpose to eliminate the gain schedule in the baseline control in the first place.

6.2 Future Work

This thesis is just the beginning of the multivariable sliding mode control in the application of an aircraft engine. Even though the sliding mode controller was tested in the C-MAPSS40k, it still needs to be run in a real time simulator. Hence, improvement in the performance of the designed controller may need. In addition, the sliding mode control technique utilized in this thesis is only a fundamental element of the control method. A more complex and sophisticated sliding mode control such as the integral sliding mode control can be used in developing the control logic to optimize the controller, resulting in better performance. Integral sliding mode control eliminates the reaching phase of the system trajectories and replaces it with the sliding surfaces where system trajectories are induced into sliding mode right away.

In short, the entire system response is guaranteed the robustness of the sliding mode control. Integral sliding mode control may also be employed to suppress chattering as well. Other methods in dealing with the chattering, such as the observer-based, disturbance rejection, and regular form, can also be imposed on the designed controller instead of the boundary layer approach. Moreover, the actuator models may even be integrated into the control logic to prevent chattering and improve the controller's performance. However, the control law must be carefully constructed, since it is a multi-input multi-output system and certain matrix operations may not be available.

This thesis can be used as ground work for the multivariable sliding mode controller for an aircraft engine, its limitless applications extended as far as a designer's imagination can go. With so many possibilities laying ahead of the multivariable sliding mode control, the control method may one day even be capable of managing the most powerful of commercial jet engines while raising the engine efficiency and lessen the pilot's workload.

BIBLIOGRAPHY

[1] Guinness World Records. Most powerful jet engine. `http://www.guinnessworldrecords.com/search/details/most-powerful-jetengine/57158.htm`. Accessed February 2011.

[2] H. Austin Spang III and Harold Brown. *Conrol of Jet Engines*. Control Engineering Practice 7, 1999.

[3] Link C. Jaw and Jack D. Mattingly. *Aircraft Engine Controls: Design, System Analysis, and Health Monitoring*. American Institute of Aeronautics and Astronautics, 2009.

[4] Ryan D. May, Jeffrey Csank, Thomas M. Lavelle, Jonathan S. Litt, and Ten-Huei Guo. A high-fidelity simulation of a generic commercial aircraft engine and controller. In *46th Joint Propulsion Conference and Exhibit*, July 2010.

[5] Jack L. Kerrebrock. *Aircraft Engines and Gas Turbines*. The MIT Press, 1992.

[6] Online Digital Education Connection. Propulsion and the x-43a. `http://www.odec.ca/projects/2004/flor4a0/public\_html/propulsionx43.htm`. Accessed February 2011.

[7] File: Turbojet operation-axial flow.png. `http://en.wikipedia.org/wiki/File:Turbojet\_operation-\_axial\_flow.png`. Accessed February 2011.

[8] File: Turbofan operation.svg. `http://en.wikipedia.org/wiki/File:Turbofan\_operation.svg`. Accessed February 2011.

[9] Tom Benson. Turboprop engine. `http://www.grc.nasa.gov/WWW/K-12/airplane/aturbp.html`. Accessed February 2011.

[10] File: Turboprop operation-en.svg. `http://en.wikipedia.org/wiki/File:Turboprop\_operation-en.svg`. Accessed February 2011.

[11] Nathan Meier. Civil turbojet/turbofan specifications. `http://www.jet-engine.net/civtfspec.html`. Accessed June 2011.

[12] Philip P. Walsh and Paul Fletcher. *Gas Turbine Performance*. Blackwell Publishing, 2004.

[13] *Pilot's Handbook of Aeronautical Knowledge*. U.S. Department of Transportation, Federal Aviation Administration, 2003.

[14] Dave Riesland. Aircraft engine analysis using adams. Technical report, European ADAMS User Conference, 2000.

[15] Zack Warfield, James D. Paduano, and Douglas G. MacMartin. Stall margin improvement using feedback control to mitigate inlet distortion unsteadiness. In *AIAA Flow Control Conference*, June 2002.

[16] Ryan D. May, Jeffrey Csank, Jonathan S. Litt, and Ten-Huei Guo. *Commercial Modular Aero-Propulsion System Simulation 40k (C-MAPSS40k) User's Guide*. National Aeronautics and Space Administration, September 2010.

[17] *Airplane Turbofan Engine Operation and Malfunctions Basic Familiarization for Flight Crews*. Federal Aviation Administration, 2009.

[18] Jonathan A. DeCastro, Jonathan S. Litt, and Dean K. Frederick. A modular aero-propulsion system simulation of a large commercial aircraft engine. In *44th Joint Propulsion Conference and Exhibit*, July 2008.

[19] Vadim I. Utkin. Application oriented trends in sliding mode control theory. *Institute of Robotics and System Dynamics, DLR, Wessling*, 1993.

[20] Vadim Utkin, Jürgen Guldner, and Jingxin Shi. *Sliding Mode Control in Electromechanical Systems*. Taylor and Francis, 1999.

[21] Richard C. Dorf and Robert H. Bishop. *Modern Control Systems*. Pearson Prentice Hall, 2004.

[22] Christopher Edwards and Sarah K Spurgeon. *Sliding Mode Control: Theory and Applications*. Taylor and Francis, 1998.

[23] K. David Young, Vadim I. Utkin, and Ümit Özgüner. A control engineer's guide to sliding mode control. *IEEE Transactions on Control Systems Technology*, 1999.

[24] Lloyd Jenkinson, Paul Simpkin, and Darren Rhodes. Data b: Engine data file: Cfmi and iae. `http://www.elsevierdirect.com/companions/9780340741528/appendices/data-b/table-1/default.html`. Accessed June 2011.

APPENDIX

APPENDIX A

MATLAB PROGRAMS

Vplace Function

vplace.m

```
function K=vplace(A,B,p)
[nn,mm]=size(B);


rnk=rank(B);
if rnk==mm,
    K=place(A,B,p);
else
   [Q,Btilde]=qr(B');
   Btilde=Btilde';
   Btilde=Btilde(:,1:rnk);
   Ktilde=place(A,Btilde,p);
   K=Q*[Ktilde;zeros(mm-rnk,nn)];
end

return
```

Matlab Commands for Verification of Sliding Mode Controller

SMC_MIMO_track_test.m

```
%Designs integral SMC tracker near Ground Idle

%Alt=1000ft, M=0.1, PLAstart=55, PLAfinal=75


clc


%Define matrices obtained from linearizing the engine through C-MAPSS 40k

%A, B, C, and D near ground idle condition

%(C and D include SMHPC output in third row)

A=[-3.3808 1.2954; 0.4444 -3.0501];

B=1.0e+003*[0.6678 -0.0392 -0.0142; 1.3340 0.1173 -0.0268];

C=[1 0; 0 1; 0.0158 -0.0037];

D=[0 0 0; 0 0 0; -10.9483 0.1837 -0.4766];


%Define transformation matrices Tx, Tu, and Txu. Then calculate Az and Bv

%matrices

Tx=[1/500 0; 0 1/1000];

Tu=[1/5 0 0; 0 1/100 0; 0 0 1];

scaleTxTu=[Tx zeros(2,3); zeros(3,2) Tu];

scaleTui=inv(Tu);

scaleTxTui=inv([Tx zeros(2,3); zeros(3,2) Tu]);


A_z=Tx*A*inv(Tx);

B_z=Tx*B*inv(Tu);
```

```
%Assemble augmented plant

A_aug=[A_z B_z;zeros(3,5)];

B_aug=[zeros(2,3);eye(3)];

C_aug=[C D];

Cz=C_aug*inv(scaleTxTu);


%Change A and B matrix to regular form

[nn,mm]=size(B_aug);

[Tr temp]=qr(B_aug);

Tr=Tr';

Tr=[Tr(mm+1:nn,:);Tr(1:mm,:)];

clear temp

Areg=Tr*A_aug*Tr';

Breg=Tr*B_aug;


A11=Areg(1:nn-mm,1:nn-mm);

A12=Areg(1:nn-mm,nn-mm+1:nn);

A21=Areg(nn-mm+1:nn,1:nn-mm);

A22=Areg(nn-mm+1:nn,nn-mm+1:nn);

B2=Breg(nn-mm+1:nn,1:mm);


A_augRf=Areg;

B_augRf=Breg;


%Calculate matrix G

%Desired n-m sliding mode poles
```

```
%In this case is 5 states - 3 inputs = 2 pole

DP=0.1*[-30, -25];

M=vplace(A11,A12,DP);

G=[M eye(3,3)]*Tr;

J=G*A_aug;

N=-1*inv(G*B_aug);


%Checking eigenvalues of Aeq.

%Number of zero eigenvalues equals to m input

%In this case, there should be three zero eigenvalues

%The rest of the eigenvalues should have negative values

Aeq=A_aug-(B_aug*inv(G*B_aug))*G*A_aug;

Aeq_Eigenvalue=eig(Aeq)

GBaug=G*B_aug;


%Design reference generator via LQR

%Reference generator implemented in z-coordinates

Q=eye(5);

R=diag([1 1 10]);

K=lqr(A_aug,B_aug,Q,R);


%Calculate reference prefilter

P=inv(-Cz*inv(A_aug-B_aug*K)*B_aug);


%Define target values

%Obtained from flight condition:

%Alt 1000, Mach# 0.1, DTamb 0, PLA [55 55 75 75]
```

```matlab
%Fan speed control mode, and in-active engine limiters

Nf_target=3739.4;

Nf_zro=3109.1;

Nc_target=11417;

Nc_zro=10584;

SMHPC_target=20;

SMHPC_zro=30.94;


%Define phi and eta for the saturation functions

phi1=0.1;

phi2=0.05;

phi3=0.1;


eta1=15;

eta2=5;

eta3=25;


%Simulate the system

sim('SMC_track_MIMO_test');


%Plot system outputs against reference outputs

figure(1)

plot(Time,y_ref(:,1),'k',Time,y(:,1),'b')

axis([0 10 0 700])

%title('First reference output and actual output vs. time')

xlabel('Time (s)')

ylabel('yref1 and y1')
```

```
legend('First reference output','First actual output')


figure(2)

plot(Time,y_ref(:,2),'k',Time,y(:,2),'b')

axis([0 10 0 900])

%title('Second reference output and actual output vs. time')

xlabel('Time (s)')

ylabel('yref2 and y2')

legend('Second reference output','Second actual output')


figure(3)

plot(Time,y_ref(:,3),'k',Time,y(:,3),'b')

axis([0 10 -18 0])

%title('Third reference output and actual output vs. time')

xlabel('Time (s)')

ylabel('yref3 and y3')

legend('Third reference output','Third actual output')


%Plot system states against reference states

figure(4)

plot(Time,x_m(:,1),'k',Time,x(:,1),'b')

axis([0 10 0 700])

%title('System and reference fan speed vs. time')

xlabel('Time (s)')

ylabel('Fan speed Nf (rpm)')

legend('Reference fan speed','Actual fan speed')
```

```
figure(5)

plot(Time,x_m(:,2),'k',Time,x(:,2),'b')

axis([0 10 0 900])

%title('System and reference core speed vs. time')

xlabel('Time (s)')

ylabel('Core speed Nc (rpm)')

legend('Reference core speed','Actual core speed')


figure(6)

plot(Time,x_m(:,3),'k',Time,x(:,3),'b')

axis([0 10 0 2])

%title('System and reference fuel mass flow rate command vs. time')

xlabel('Time (s)')

ylabel('Wf (lb/s)')

legend('Reference Wf','Actual Wf')


figure(7)

plot(Time,x_m(:,4),'k',Time,x(:,4),'b')

axis([0 10 0 3.5])

%title('System and reference variable stator vane angle command vs. time')

xlabel('Time (s)')

ylabel('VSV (deg)')

legend('Reference VSV','Actual VSV')


figure(8)

plot(Time,x_m(:,5),'k',Time,x(:,5),'b')

%title('System and reference variable bleed valve position vs. time')
```

```
xlabel('Time (s)')

ylabel('VBV (fraction open)')

legend('Reference VBV','Actual VBV')


%Plot system commands against time

figure(9)

plot(Time,u(:,1))

axis([0 10 0 2])

%title('Fuel mass flow rate command vs. time')

xlabel('Time (s)')

ylabel('Fuel mass flow rate (lb/s)')


figure(10)

plot(Time,u(:,2))

axis([0 10 0 3.5])

%title('Variable stator vane position command vs. time')

xlabel('Time (s)')

ylabel('Variable stator vane position (deg)')


figure(11)

plot(Time,u(:,3))

%title('Variable bleed valve position command vs. time')

xlabel('Time (s)')

ylabel('Variable bleed valve position (fraction open)')


%Plot sliding function against time

figure(12);
```

```
plot(Time,s_system(:,1),'b')

hold on

plot(Time,s_system(:,2),'g:')

plot(Time,s_system(:,3),'r--')

axis([0 30 -0.1 0.1])

legend('First sliding variable s1','Second sliding variable s2',

'Third sliding variable s3')

ylabel('Sliding variables')

xlabel('Time (s)')

%title('Sliding Functions')
```

Matlab Commands for Simulation of Sliding Mode Control in C-MAPSS40k
SMC_MIMOtrack.m

```
%Designs integral SMC tracker near Ground Idle

%Alt=1000ft, M=0.1, PLAstart=55, PLAfinal=75


%In order to run this code, some necessary papameters for the system

%model is needed to obtain through C-MAPSS by runing C-MAPSS 40k GUI and

%simulate native controller using the following: PLA vector=[55 55 75 75],

%fan speed control mode, and in-active engine limiters.


clc %Clear command window

close all %Close all the open figure windows


%Define matrices obtained from linearizing the engine through C-MAPSS 40k
```

```matlab
%A, B, C, and D near ground idle condition
%(C and D include SMHPC output the third row)
A=[-3.3808 1.2954; 0.4444 -3.0501];
B=1.0e+003*[0.6678 -0.0392 -0.0142; 1.3340 0.1173 -0.0268];
C=[1 0; 0 1; 0.0158 -0.0037];
D=[0 0 0; 0 0 0; -10.9483 0.1837 -0.4766];


%Define transformation matrices Tx, Tu, and Txu. Then calculate Az and Bv
%matrices
Tx=[1/500 0; 0 1/1000];
Tu=[1/5 0 0; 0 1/100 0; 0 0 1];
scaleTxTu=[Tx zeros(2,3); zeros(3,2) Tu];
scaleTui=inv(Tu);
scaleTxTui=inv([Tx zeros(2,3); zeros(3,2) Tu]);


A_z=Tx*A*inv(Tx);
B_z=Tx*B*inv(Tu);


%Assemble augmented plant
A_aug=[A_z B_z;zeros(3,5)];
B_aug=[zeros(2,3);eye(3)];
C_aug=[C D];
Cz=C_aug*inv(scaleTxTu);


%Change A and B matrix to regular form
[nn,mm]=size(B_aug);
[Tr temp]=qr(B_aug);
```

```
Tr=Tr';

Tr=[Tr(mm+1:nn,:);Tr(1:mm,:)];

clear temp

Areg=Tr*A_aug*Tr';

Breg=Tr*B_aug;


A11=Areg(1:nn-mm,1:nn-mm);

A12=Areg(1:nn-mm,nn-mm+1:nn);

A21=Areg(nn-mm+1:nn,1:nn-mm);

A22=Areg(nn-mm+1:nn,nn-mm+1:nn);

B2=Breg(nn-mm+1:nn,1:mm);


A_augRf=Areg;

B_augRf=Breg;


%Calculate matrix G

%Desired n-m sliding mode poles

%In this case is 5 states - 3 inputs = 2 pole

DP=0.1*[-30, -25];

M=vplace(A11,A12,DP);

G=[M eye(3,3)]*Tr;

J=G*A_aug;

N=-1*inv(G*B_aug);


%Checking eigenvalues of Aeq.

%Number of zero eigenvalues equals to m input

%In this case, there should be three zero eigenvalues
```

```matlab
%The rest of the eigenvalues should have negative values
Aeq=A_aug-(B_aug*inv(G*B_aug))*G*A_aug;
Aeq_Eigenvalue=eig(Aeq)
GBaug=G*B_aug;


%Design reference generator via LQR
%Reference generator implemented in z-coordinates
Q=eye(5);
R=diag([1 1 10]);
K=lqr(A_aug,B_aug,Q,R);


%Calculate reference prefilter
P=inv(-Cz*inv(A_aug-B_aug*K)*B_aug);


%Original target values
Nf_target=3739.4;
Nc_target=11417;
SMHPC_zro=out_SM_HPC(1);
SMHPC_target=20;


%Define phi and eta for the saturation functions
phi1=11;
phi2=13.2;
phi3=10;


eta1=48;
eta2=25;
```

```
eta3=25;


%Simulate the system

sim('PAX200_SMC_track_MIMO');


%Plotting Section

figure(1)

plot(out_t,out_Nf,'k')

hold on

plot([0 30],[Nf_target Nf_target],'g--')

ylabel('Fan speed N_f (rpm)')

xlabel('Time (s)')


figure(2)

plot(out_t,out_Nc,'k')

hold on

plot([0 30],[Nc_target Nc_target],'g--')

ylabel('Core speed N_c (rpm)')

xlabel('Time (s)')


figure(3)

plot(out_t,out_SM_HPC,'k')

hold on

plot([0 30],[SMHPC_target SMHPC_target],'g--')

ylabel('Stall margin of high pressure compressor SMHPC (%)')

xlabel('Time (s)')
```

```matlab
figure(4)

plot(out_t,out_Wf,'k')

ylabel('Fuel flow rate W_f (lb/s)')

xlabel('Time (s)')


figure(5)

plot(out_t,out_VSV,'k')

ylabel('Variable stator vanes VSV (deg)')

xlabel('Time (s)')


figure(6)

plot(out_t,out_VBV,'k')

ylabel('Variable bleed valve VBV (%open)')

xlabel('Time (s)')


figure(7)

plot(out_t,s_system(:,1),'b')

hold on

plot(out_t,s_system(:,2),'g:')

plot(out_t,s_system(:,3),'r--')

legend('First sliding variable s1','Second sliding variable s2',

'Third sliding variable s3')

ylabel('Sliding variables')

xlabel('Time (s)')


%Calculate %error

Nf_error=(out_Nf(end)-Nf_target)/Nf_target*100
```

```
Nc_error=(out_Nc(end)-Nc_target)/Nc_target*100

SMHPC_error=(out_SM_HPC(end)-SMHPC_target)/SMHPC_target*100
```