Use Authorization

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at Idaho State University, I agree that the Library shall make it freely available for inspection. I further state that permission for extensive copying of my thesis for scholarly purposes may be granted by the Dean of the Graduate School, Dean of my academic division, or by the University Librarian. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Signature _____

Date _____

Preliminary Investigation on a Single-stage Axial Compressor Design and Stall Precursor Detection Algorithm

By

Shishir Khanal, E.I.T.

A thesis

Submitted in partial fulfilment

Of the requirements for the degree of

Master of Science in Measurement and Control Engineering

College of Science and Engineering

Idaho State University

Spring 2022

Committee Approval

To the Graduate Faculty:

The members of the committee appointed to examine the thesis of Shishir Khanal find it satisfactory and recommend it be accepted.

Dr. Marco P. Schoen, Ph.D., P.E.

Major Advisor

Dr. Anish Sebastian, Ph.D.

Co-Advisor, Committee Member

Dr. Paul Bodily, Ph.D.

Graduate Faculty Representative (GFR)

Acknowledgement

I would like to express my sincere gratitude to my thesis advisor Dr Marco P. Schoen for his continuous support throughout my master's career at Idaho State University. His enthusiasm, support, advise, resource and expertise on the compression systems helped develop my knowledge in the compression systems and come up with the design solutions while I was working on my thesis.

I would also like to thank my committee member, Dr Anish Sebastian for providing resources and expert point of view in improving the compressor and throttle sections of the compression system. I received an opportunity of efficiently optimizing the compression system as a part of the Measurement and Controls Laboratory (ME 6644) under him.

Similarly, I would like to thank Dr Bodily for agreeing to become my Graduate Faculty Representative and becoming my co-advisor for the algorithm design to detect stall precursors. I also took his class in Advanced Algorithms (CS 5512) which helped me gain confidence in programming and interest in designing algorithms.

I would like to thank my peers Mr. Cooper Dastrup and Mr. Andrew Anderson for their help in designing multiple unique stators for the system. Because of their innovative design approaches, I have achieved confidence in delegation of the project tasks that I am passionately involved in.

I would also like to thank the student 3D printing in charge of Mechanical Engineering Department Mr. Vincent Akula, Mr. Osazee Erhunmwunse and Mr. Michael Crump for patiently working with my significant number of 3D print requests and Mr. Golam Gause Jaman for teaching me how to design a Soldered Circuit Board using a given Circuit Board Diagram.

I would also like to acknowledge Dr. Ken W. Bosworth, Dr Thomas Baldwin, Mr. Farsad Dailami, Dr Taher Deemyad, Dr Steven Millward and Dr Andrew Chrysler for allowing me to enroll in their graduate level classes. The knowledge I gained from their class has directly or indirectly helped me acquire knowledge towards my thesis work.

Finally, I would like to thank my mother Narbada Ghimire Khanal, father Ishwar Prasad Khanal, grandmother Gyatri Khanal, grandfather Hari Prasad Khanal and brother Saujan Khanal for keeping me connected to my home throughout my journey of undergraduate and graduate engineering career at Idaho State University.

Table of Contents

Definitions:

1. **Rotor:** A part of a machine that turns around a central point [1].
2. **Compressor:** A machine that compresses air or other gasses [2].
3. **Controller:** A device that controls or regulates a machine or part of a machine [3].
4. **Characterization:** A set of procedures to extract a mathematical model of a system.
5. **Stall:** A situation in which there is a pressure drop inside the compressor due to the breakage of the flow in the blade region of the compressor.
6. **Stator:** A stationary part of a rotary system [4].
7. **Inception:** an act, process, or instance of beginning [5].
8. **Precursor:** A prior event that precedes the primary event.
9. **Univariate:** Involving a single variable.
10. **Normal Distribution:** A bell-shaped distribution where the majority of the data points are concentrated around their mean.
11. **Stationary Data:** Mean, variance, and autocorrelation structure do not change over time [6].

Abbreviations:

- **PDE:** Partial Differential Equation
- **IGV:** Inlet Guide Vane
- **ISU:** Idaho State University
- **DAQ**: Data Acquisition
- **AR:** Autoregressive Model
- **RLS:** Recursive Least Squares
- **GESDT:** Generalized Extreme Studentized Deviate Test
- **DAQ:** Data Acquisition System
- **RPM:** Rotations Per Minute
- **TTL:** Transistor-Transistor Logic
- **M-G**: Moore-Greitzer (Refers to the Moore Greitzer model for Compressor)

List of Figures:

List of Tables:

Preliminary Investigation on a Single-stage Axial Compressor Design and Stall Precursor Detection Algorithm

Thesis Abstract - Idaho State University (2022)

Axial compressors are rather complex systems used in a range of industrial and civil applications. For transportation, axial compressors have found a dominant role in jet engines powering a wide variety of civil airplanes. Despite the central role such systems play over a number of years in propulsion, many design and performance issues are unsolved. In particular, the efficiency and the associated risk of stall and surge present a challenge as fuel economy and pollution issues are closely related to efficiency. To further the research on these unresolved topics, a compressor research facility requires an interface to study the effects of the compressor instabilities and remedial control actions, preferably at a reasonable cost. Hence, a design of a test-bench compressor is proposed. In this work, a prototype design and implementation of such a system are detailed including the associated electronic instrumentation and data acquisition system. Additionally, the instrumentation along with the corresponding acquired signals from the different sensors are analyzed and used to characterize the proposed system. An algorithm to predict stall precursors is proposed as well. The experimental data for the algorithm, using the pressure data around the tip clearance region of the rotor blades, is acquired from such a test bench system. The two major components of the algorithm are designed and tested against various test cases.

Keywords: Axial Compressor, Flow Dynamics, Measurement Systems, Test Stand

Chapter -1: Introduction

1.1 Background

Turbomachines are a group of machinery equipment that exchange energy between a rotor and a fluid. There are three major kinds of turbomachines: pumps, turbines, and compressors. The pumps are used to supply energy to the fluids and are used to take out underground water, create vacuum chambers, etc. Turbines are used to extract energy from the fluid and they are used for energy harvest applications such as hydropower plants, windmill plants, etc. Finally, the compressors are used to supply energy to the gaseous fluid and are used extensively in the process and aerospace industry [7]. Based on the mechanism of compression, there are four major sub-categories of compressors: rotary, reciprocating, centrifugal, and axial compressors [8]. In this work, an axial compressor is studied.

In the axial compressors, the fluid flow is parallel to the axis of the rotation. The rotor and stator blades are alternately placed in a compressor where the rotor blade supplies energy to the fluid by exerting torque and the stator blades allow the air to slow down which causes the increase in potential energy and hence increasing the fluid pressure [9].

1.2 Research History

The axial flow compression system was first invented by Parsons in 1901 [10]. Currently, five major research groups are actively researching axial compression systems: Whittle Laboratory of Cambridge University, Gas Turbine Laboratory of Massachusetts Institute of Technology, Institute of Engineering Thermophysics, Georgia Institute of Technology, and Waseda University.

The most popular model for a simple compression system was proposed by Moore and Greitzer in 1986 which was able to model both the stall and surge instabilities in multistage compressors[11]. After that, McDougall studied the spike stall behavior in 1990 [12] and Day, I.J studied Modal Stall behavior in 1993 using low-speed compressors[13]. Mansoux et al., in 1994, presented a control-theoretic form of the Moore-Greitzer model and performed a Lyapunov-

based stability analysis. They concluded that the shape of the compressor characteristic determines how the stall inception occurs [14].

Historically, two major techniques have been used to study compressor instabilities: numerical methods and computational fluid dynamics methods. Hwang et al. developed a numerical model to study the first stage of a compressor system with an Inlet Guide Vanes [15]. His team concluded that numerical simulations have limited promise in compressor instability research. On the other hand, I.J. Day, one of the researchers at Whittle lab, claimed in 2015 that the computational fluid dynamics approach has more promise in the improvement of performance prediction, flow modeling, and the interpretation of the data to the underlying mechanism [16].

At Idaho State University (ISU), the first work in compressor research started under Prof. Dr. Marco P. Schoen in 2009 in the Aerospace Laboratory at Measurement and Controls Engineering Laboratory. Under the collaboration with the Institute of Engineering Thermophysics in China, the research grew from investigating the flow in the tip of the blades to investigating the real compressor data. Sterbentz et al. investigated the dynamics of the tip region of the axial compressor blade passage by using the system identification approach [17]. In 2017, Schoen and Lee proposed a subspace system identification algorithm that relates the tip air injection and the throttle movement to the overall dynamics of the compressor and also the tip air injection to the flow dynamics at the tip of the blade. In the same year, Bitikofer et al. developed a genetic algorithm-based Graphical User Interface application that could extract the Moore-Greitzer model by processing the compressor data [18].

The research work accelerated after the Aircraft Maintenance Technology Department of ISU donated jet engines Westinghouse J-32 and Lycoming T-51 in the summer of 2019. In 2019, Aung et al. used the Outliers detection method along with the Autoregressive model and devised an algorithm that could detect the spike stall precursors 32 revolutions before the stall initiation [19]. During this time, the research team acquired two jet engines from the Aircraft Maintenance Technology department of ISU to develop new algorithmic and control methods for the axial compressor engines.

1.3 Statement of Problem

Idaho State University's Mechanical Engineering program has a dedicated research group focused on Turbomachinery research specifically in the compressor sub-system of the jet engines. The risk of compressor instabilities to the engine is high. Hence, the development of a working prototype of the compressor stage is crucial not only to validate the proposed solutions to the compressor-related problems but also to investigate the unanticipated physical problems arising from engineered solutions. Hence, to research on the actual jet engine in the Jet Engine Research Laboratory, it is crucial to construct a stable prototype for a single-stage compressor that exhibits similar dynamics as the actual jet engine compressor while having a simpler design and lower cost so that the risks associated with the experiments are minimal.

However, any testbed will not only have the complexity of a physical system but also have added complexity arising from the measurement instruments and dynamics visualization so that investigators can make a real-time assessment of the parameter changes in the compressor. Hence, a DAQ system is also desired to measure the dynamics of the compressor in real-time.

1.4 Proposed Solution

The major objective of the thesis research is to design a low-cost single-stage compression system that is capable of stabilizing against the compressor instabilities and visualizing the compressor and controller dynamics to the investigator in real-time. Following are the sub-objectives of the thesis:

- To design & build a working prototype and DAQ of the single-stage axial compressor
- To construct an algorithm that can predict the stall precursors using the time-series pressure data of the compressor

This system will allow the investigators to study the effects of control law, blade geometry, and stall precursors in the optimization of the efficiency of the compressor without significantly modifying the system.

1.5 Source of Funding

The single-stage compressor system design is supported by internal funds from the Department of Mechanical Engineering and a Research Initiation Grant from the Office of Research at ISU.

Chapter -2: Single-Stage Axial Compressor

2.1 Moore-Greitzer Model

F.K. Moore and E.M Greitzer updated the original model of the E.M Greitzer for the axial compression system. The Moore-Greitzer model[20] is a nonlinear state-space model that captures the rotating stall and surge in the axial compressors. The experimental setup for the compression system is assumed as a duct with three major subsystems: compression system, plenum, and throttle system. Air is compressed in the compression section and then discharged to the plenum. The throttle system controls the flow through the system. The flow of air in the compression system is assumed to be incompressible but treated as compressible in the plenum.

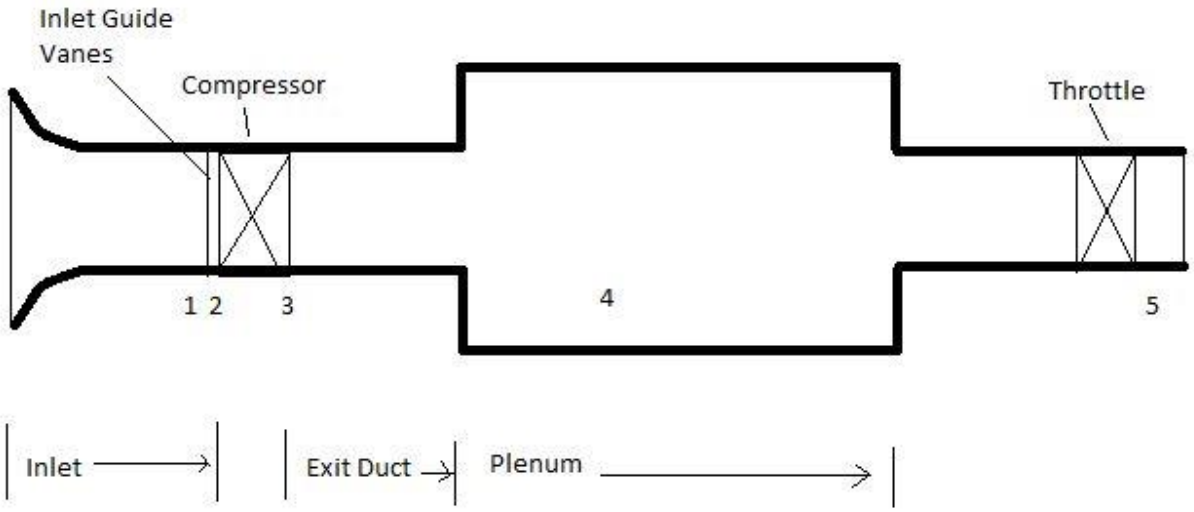


Fig 2.1.1: Moore-Greitzer Compression System Model

The pressure and flow parameters in the compression system are non-dimensionalized and expressed in terms of pressure rise coefficient and flow coefficient.

Pressure Rise Coefficient (Ψ) = $\frac{P_4 - P_{in}}{0.5 * \rho * U^2}$

Flow Coefficient (Φ) = $\frac{C}{U}$

where, P_4 = Pressure in Plenum (Unit: Pascals)

P_{in} = Inlet Pressure (Unit: Pascals)

p= Density of the air (Unit: Kg/m^3)

U = Mean Rotor Velocity (Unit: m/s)

C = Axial Flow Velocity (Unit: m/s)

The circumferential mean of the flow coefficient at station 1 in Figure 2.1 is,

$$\Phi(\zeta) = \frac{1}{2\pi}\int_0^{2\pi} \phi(\zeta,\theta), \phi(\zeta,\theta) = \Phi(\zeta) + \varphi_0(\zeta,\theta)$$

where, $\varphi_0(\zeta, \theta) = \varphi(\zeta, \theta, 0)$

$\varphi(\zeta, \theta, \eta)$ = Flow disturbance

η = nondimensional axial distance from station 1

ζ = time-parameter non-dimensionalized by the rotor speed

η is taken as a positive value for a distance on the right of and a negative value on the distance left of station 1.

The disturbance flow potential $\bar{\varphi}(\zeta, \theta, \eta)$ satisfies Laplace's equation,

$$\frac{\delta^2\bar{\varphi}}{\delta\theta^2} + \frac{\delta^2\bar{\varphi}}{\delta\eta^2} = 0$$

and boundary layer condition is,

$$\frac{\delta\bar{\varphi}}{\delta\eta}(\zeta,\theta,-l_F) = 0,\ l_F \to \infty \qquad (2.1.1)$$

The pressure rise coefficient for the compressor is governed by the following equation,

$$\Psi = \psi_3(\phi) - l_c\frac{d\Phi}{d\zeta} - \lambda\frac{\delta\bar{\varphi}}{\delta\zeta}\bigg|_{\eta=0} - \frac{1}{2a}\left(2\frac{\delta^2\bar{\varphi}}{\delta\eta\delta\zeta} + \frac{\delta^2\bar{\varphi}}{\delta\eta\delta\theta}\right)\bigg|_{\eta=0} \qquad (2.1.2)$$

where,

l_c = Total aerodynamic length of the compressor system (Units: meters)

λ = Exit duct length factor

For uniform and steady flow,

$$\Psi = \psi_3(\phi) = \psi_{3,0} + H\left[1 + \frac{3}{2}\left(\frac{\Phi}{W} - 1\right) - \frac{1}{2}\left(\frac{\Phi}{W} - 1\right)^3\right] \qquad (2.1.3)$$

where,

$\psi_{3,0}$ = is the local flow coefficient in the compressor (i.e. 3) in the initial state (i.e. 0)

The dynamics at the plenum is isentropic, hence the mass flow rate equation is,

$$\frac{dm_4}{dt} = \frac{dm_3}{dt} - \frac{dm_5}{dt}$$

where,

m_4 = mass flow in the plenum (Unit: kilogram)

m_3 = mass exiting the compressor to the plenum (Unit: kilogram)

m_5 = mass exiting the throttle from the plenum (Unit: kilogram)

Imposing the isentropic condition,

$$\frac{1}{V_4}\left(\frac{dm_3}{dt} - \frac{dm_5}{dt}\right) = \frac{d\rho}{dt} = \frac{1}{a_s^{\,2}}\frac{dP_4}{dt} \tag{2.1.4}$$

where,

a_s = speed of sound (Unit: m/s)

$\frac{dm}{dt}$=ρ*C*A

where,

A = Cross-Sectional Area of Duct (Unit: square meters)

The above equation can be written as,

$$\Phi - \Phi_5 = \frac{V_4 U}{a_s^{\,2} A}\frac{d\Psi}{d\zeta} = 4B^2 l_c \frac{d\Psi}{d\zeta}; B := \frac{U}{2a_s}\sqrt{\frac{V_4}{AL_c}} \tag{2.1.5}$$

where,

L_c = Physical dimension of l_c (Units: Meters)

Assuming parabolic throttle characteristic with K_5 throttle constant, the above equation becomes,

$$l_c \frac{d\Psi}{d\zeta} = \frac{1}{4B^2}(\Phi - \Phi_5(\zeta)) = \frac{1}{4B^2}(\Phi - F_5^{-1}(\Psi)), \Psi = F_5(\Phi_5) = \frac{1}{2}K_5(\Phi_5)^2 \tag{2.1.6}$$

Eq 2.2 is a Partial Differential Equation (PDE) which needs to be projected into Ordinary Differential Equation (ODE). Using the spatial harmonic form for the disturbance velocity potential below. This satisfies Laplace's Equation and the boundary condition Eq 2.1.

$$\bar{\varphi} = \sum_{n=1}^{\infty}\frac{1}{n}e^{n^*\eta}(a_n(\zeta)e^{jn\theta} + \bar{a}_n(\zeta)e^{-jn\theta}), \eta \le 0 \tag{2.1.7}$$

where,

a_n = Spatial Fourier coefficient at the IGV station (η= 0)

j = $\sqrt{-1}$ represents complex number

Using equation above,

$$\varphi_0 = \left.\frac{d\bar{\varphi}}{d\eta}\right|_{\eta=0} = \sum_{n=1}^{\infty}\varphi_{0,n} = WA_n(\zeta) * \sin(n\theta - r_n(\zeta)), A_n = \frac{2\,|\,a_n(\zeta)\,|}{W} \tag{2.1.8}$$

where,

$r_n(\zeta) = r_n * \zeta$

r_n = wave speed constant

W = semi-width of the cubic curve characteristic [11]

A_n = Amplitude of Angular Disturbance of Axial Flow Coefficient [11]

Rewriting Eq 2.6,

$$\frac{d\Psi}{d\zeta} = \frac{W/H}{4B^2}[\frac{\Phi}{W} - \frac{1}{W}F_5^{-1}(\Psi)]\frac{H}{l_c}, F_5^{-1}(\Psi) = \sqrt{\frac{2\Psi}{K_T}} \tag{2.1.9}$$

Substituting φinto Eq. 2.2 and integrating over 0 to 2π,

$$\frac{d\Phi}{d\zeta} = [\frac{\psi_3(\Phi) - \Psi}{H} - \frac{3}{4}(\frac{\Phi}{W} - 1)J_n]\frac{H}{l_c} \tag{2.1.10}$$

Substituting φ into Eq. 2.2, multiply by $\sin(n\theta - rn\zeta)$ and integrating over 0 to 2π,

$$\frac{dJ_n}{d\zeta} = J_n([1 - (\frac{\Phi}{W} - 1)^2 - \frac{1}{4}J_n]\frac{3aH}{W})(1 + \frac{a\lambda}{n})^{-1} \tag{2.1.11}$$

where,

$J_n = A_n^2$

$r_n = \frac{n}{2[1+a\lambda]}$

$a = \frac{R}{N\tau U}$

$\lambda = \frac{1}{a}(\frac{1}{2} - f)$

where,

H = Semi-height of the cubic characteristic [11]

N = Number of Stages of a Compressor

τ = Pressure-Rise lag Coefficient [11]

f = Non-dimensional wave-speed Coefficient [11]

R = Radius of Rotor

2.2 Compressor Stall and Surge

One of the major achievements of the Moore-Greitzer model is that the ODEs can isolate pure stall and pure surge conditions.

The equilibrium point (Ψ_e,Φ_e, J_e) for the axisymmetric and steady flow is governed by [20],

$$\Psi_e = \frac{1}{2}K_5(\Phi_e)^2 = \psi_3(\Phi_e), J_e = 0 \tag{2.2.1}$$

If the compressor equation is linearized about $\Phi = \Phi_e < 2W$, $\Psi = \psi_3(\Phi_e)$, $J_n = 0$, the equilibrium point is unstable due to the positive slope of $\psi_3(\Phi)$. This is shown in Figure 2.3.1 below. The disturbance at these points develops into the rotating stall, surge, or both. For pure surge,

$$\frac{dJ_n}{dt} = J_n = 0$$

For a rotating stall, the time derivatives in Eq. (2.1.9) - Eq. (2.1.11) must be zero. At Eq. (2.1.11) the prior conditions restrict J_n to be either 0 or a constant equilibrium value [20].

$$J_e = \frac{8W}{3H}\frac{d\psi_3(\Phi)}{d\Phi}\bigg|_{\Phi=\Phi_e} = 4[1-(\frac{\Phi_e}{W}-1)^2] \tag{2.2.2}$$

For $J_e \geq 0$, the allowable range for $\boldsymbol{\Phi}$ is $0 \leq \boldsymbol{\Phi} \leq 2W$. In addition, the pure rotating stall occurs at the peak at which $\boldsymbol{\Phi}_e = 2W$.

$$\frac{d\Phi}{dt} = [-\Psi + \psi_s(\Phi)]\frac{1}{l_c}, \psi_s(\Phi) = \psi_{3,0} + H[1-\frac{3}{2}(\frac{\Phi}{W}-1)+\frac{5}{2}(\frac{\Phi}{W}-1)^3] \tag{2.2.3}$$

Hence, for rotating stall, the equilibrium point (Ψ_e,Φ_e, J_e) is governed by,

$$\Psi_e = \frac{1}{2}K_5(\Phi_e)^2 = \psi_s(\Phi_e), J_e = 4[1-(\frac{\Phi_e}{W}-1)^2] \neq 0 \tag{2.2.4}$$

2.3 Compressor and Throttle Characteristics

The characterization of a compression system can be obtained by plotting the non - dimensionalized flow coefficient Φ versus non-dimensionalized pressure rise coefficient Ψ as

shown in Figure 2.2. The compressor dynamics can be represented by a cubic curve[21]. A representative of the throttles for a compression system exhibits a parabolic characteristic. When the rpm of the compressor is changed the characteristic curve moves normally i.e. if the rpm is decreased, the curve moves towards origin and when it is increased, the curve moves away from an origin. For a fixed rpm, the characteristic curve for the compressor can be traced by slowly closing the throttle from a completely open throttle position. By measuring the pressure change and the axial airflow speed and plotting the non-dimensionalized parameters, a characteristic curve for the compressor can be achieved.

The optimum performance of the compressor is achieved when the throttle curve intersects the characteristic curve at its topmost point. In the phase space of the compression system, the pure rotating stall is a stable equilibrium and the pure surge is a stable limit cycle [21]. However, a compression system can also have both effects at the same time. The optimal efficiency point is so fragile that system disturbance and uncertainties can induce bifurcation in the phase space leading to compressor instabilities. Not only that, if an attempt is made to get out of the stall by changing the airflow and the throttle opening, the compressor gets trapped in the state of hysteresis [21](represented by the shaded part in Figure 2.2).

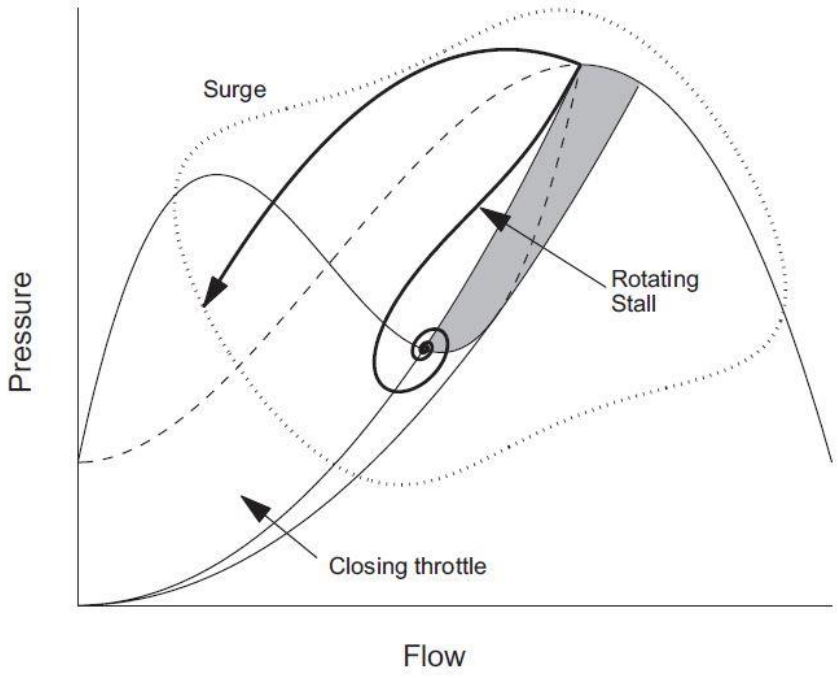


Fig 2.3.1: Compressor Characteristic Plot in Conjunction with Throttle Characteristic Plot [21]

As such, a classical approach of preventing the stall has been running the compression system sub-optimally within a specific safety margin. The tradeoff of this approach is it guarantees safety but the system is less efficient which means it uses more resources (eg, fuel) but provides a lower output (eg, mechanical work). Modern control techniques involve the fusion of robust disturbance rejection method[22], low controller effort method[23] along with nonlinear optimal control methods [21].

Chapter -3: Stall and Stall Inception

3.1 Fast Dynamics

The tip of the compressor blades must rotate without interference from the compressor walls. Hence, there is a gap between the blades and the wall of the compressor. This affects the dynamics of the compression system when the flow conditions in the compressor change. Not only that, the geometry of the blades of the compressor also determines what kind of compressor instability the system exhibits. These issues are collectively studied under the fast dynamics of the compressor.

There are two kinds of compressor stall phenomena observed in the axial compression system: modal stall and spike stall inception. The compressors that exhibit the modal stall initiate 'stall cell' as a bump in the pressure reading as shown in Figure 3.1. In the figure, the dotted line represents the points of time at each measurement point where the first oscillation on the otherwise constant reading is seen. This is referred to as a stall cell. After that, the stall waves travel circumferentially but slower than the airspeed before developing into a full stall. The modal stall inception process is comparatively easier and can be done outside of the compressor.

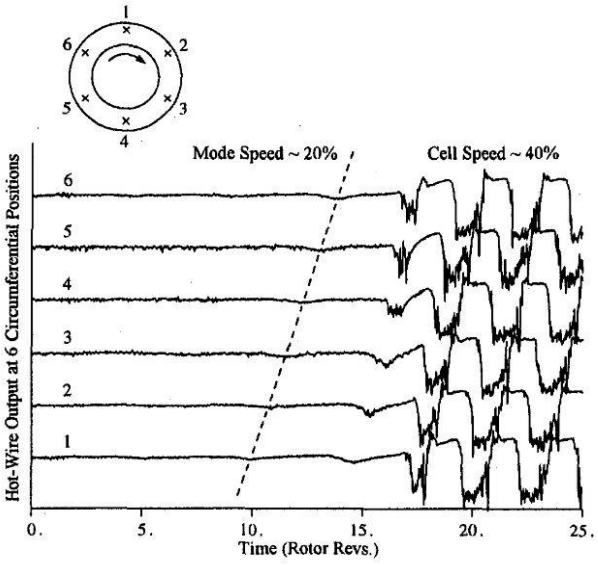


Fig 3.1.1: Modal Stall Inception [24]

On the other hand, the compressors that exhibit a spike stall inception begin the transformation into the stall by giving out a small amplitude but a sharp drop in the pressure which travels in the direction of the rotation of the compressor before fully developing into a complete stall. This can be seen in Figure 3.2. The inception of this stall is more complicated as it requires real-time data from the blades of the compressor. Also, the detection of spike stall precursors is a hot topic for further research for axial compression systems.

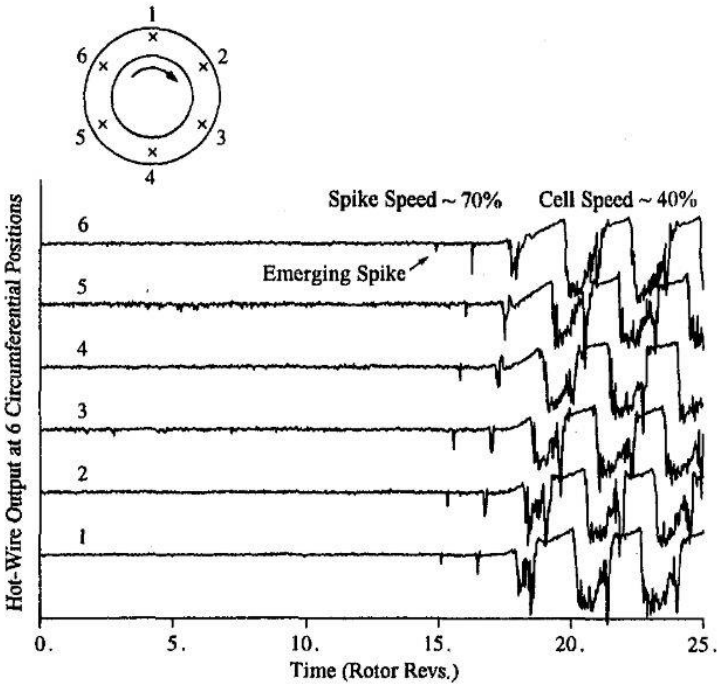


Fig 3.1.2: Spike Stall Inception [24]

Based on the location of the stall initiation point relative to the critical point, Camp and Day came up with a model of the stall inception [24]. This model allows us to inspect which kind of stall: spike or modal, the compressor goes into by looking at the complete compressor characteristic curve (Figure 3.3). If the critical rotor incidence corresponding to the stall initiation point occurs right at the peak point of the compressor, the compressor goes into a spike stall. Similarly, if the critical rotor incidence occurs towards the right of the peak point of the compressor, the compressor goes into a modal stall.

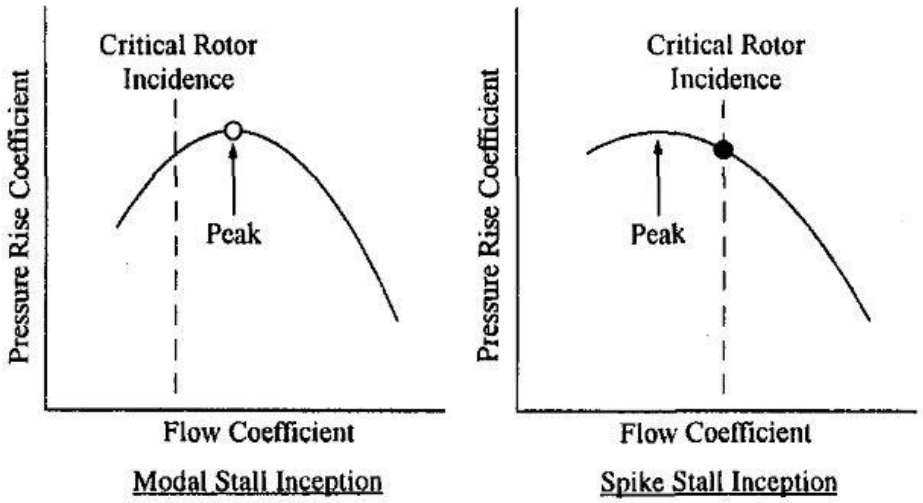


Fig 3.1.3: Camp and Day Stall Inception Model [24]

3.2 Tip Clearance

Tip clearance is the gap between the compressor blade tip and the inner wall of the compressor. The tip clearance can cause leakage which could induce vortices in the compressor section due to a microflow reversal circumferentially around the tip of the blades. Two parameters affect the tip leakage flow dynamics: the magnitude of the gap of tip clearance and the pressure difference between the two sides of the compressor. For a similar operating condition, the increase in the tip clearance gap by 1% reduces the compressor efficiency by 5% [25]. For this research, the compressor is run under a constant tip clearance gap condition.

Chapter -4: Spike-Stall Precursor Detection

4.1 Past Approaches

During a normal operation of the axial compressor, the modal oscillation can only occur if the compressor can operate beyond the zero-slope characteristic. For some compressors, the localized overloading is strong enough to induce a disturbance in the compressor even before the compressor reaches the zero-slope characteristic. Hence, the spike stall is induced due to the spikes originating from such overloading in an individual blade row [24].
Several approaches have been invented in an attempt to design a stall warning algorithm for the compressors that exhibit spike-type stalls which are named below [26].

a) Spatial Fourier Analysis of a circumferential array of pressure transducers
b) Traveling Wave energy analysis
c) Cross-Correlation of spatially adjacent signals (with and without stochasticity)
d) Windowed space-time correlation

Before the types of stalls were fully understood, the stall phenomenon was modeled as a wave which allowed the researchers to come up with a better operating point by incorporating the stall warning mechanism with the compression system [27]. In 1991, Inoue et al. conducted a statistical analysis of the pressure signal around the wall of the blades of the rotor and found that the study of the pressure signal fluctuations in that region can be useful in detecting stall precursors [28]After the spike and the modal stalls were identified, the modal stalls were also found to accompany additional disturbances which were named ‘pip’, and a statistics-based structure function was used in the off-line band-pass filtered data to monitor the gradual increase in the amplitude of the oscillations found before the stall [29]. It was later found that the modes will only form if the spikes do not form first [16]. Hence, all those disturbances were spikes forming really close to the modes in the compressor characteristic.

The research work after the 2000s has a strong impact on the experimental setup for stall precursor detection. The usage of digital electronics and modern DAQs has made it convenient to acquire a large amount of compressor data which has allowed validation of several algorithms on the stall precursor detection in the off-line data which can be incorporated into the compressor

system to be run in real-time using windowing techniques. Tahara et. al was able to detect precursors for the spike stall by monitoring the pressure at the tip region of the compressor [30].

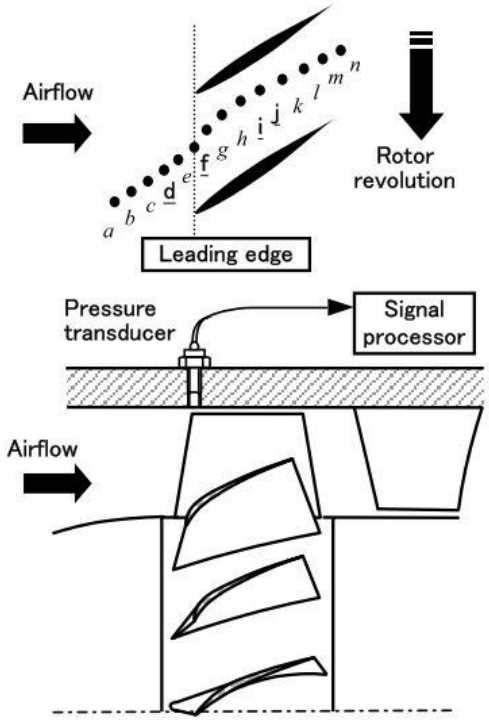


Fig 4.1.1: Experimental Setup for the Stall Precursor Detection [30]

Finally, Heinlein et. al proposed to track the outliers in the entropy of the simulated data [31].

4.2 Implemented Method

Stall phenomena of a compressor can be observed in the pressure data of the axial compression system. So, if a model for the pressure dynamics of the compressor is evaluated, it is proposed that at least one of the eigenvalues of the corresponding model should capture the stall inception dynamics from the stall precursor until the development into the full stall [19].

For this purpose, the pressure data is fitted into an AR model and the eigenvalues are observed for cues on precursors. After that, the number of outliers of each of the eigenvalues is monitored for any abrupt changes around the stall using the GESDT algorithm. The overall architecture of the algorithm is provided in Figure 4.2 below.

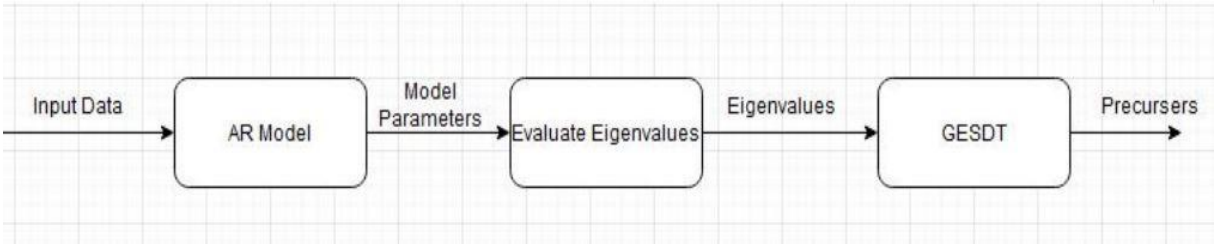


Fig 4.2.1: Overall Architecture of Stall Precursor Detection Algorithm

4.2.1 AR Model Fitting

The AR model is a popular structure used in forecasting behaviors in time series data. The procedure is relevant when there is some correlation between time series data and the values of the past and the future. By using compressor data that goes into a stall, it is possible to create a time-dependent model of the compressor and by sufficiently accommodating the order of the AR model, it is possible to capture the stall dynamics of the compressor [19].

When the compressor is running in the normal operating condition, the system is stable. Hence, the eigenvalues of the discrete AR model lie inside the unit circle. When the system goes into the stall, the system becomes unstable because the stall phenomenon drives the compressor to an equilibrium point and the surge drives the system to a limit cycle. Hence, by iteratively comparing the eigenvalues of the AR model it is possible to speculate if certain eigenvalues are tracing a locus away from the unit circle.

Finally, the AR model assumes that the data is stationary. For time-series data of a compressor going into a stall, the overall data is not stationary. However, if the data is broken down into really small time steps, it is possible to construct an AR model for every new time step and evaluate a really small drift in the position of the eigenvalues.

The AR model is evaluated using a least-squares estimation method. The estimation procedure is based on the objective of minimizing the square of the errors during the regression. However, the specific procedure requires a large amount of data to be processed by inverting a data matrix to evaluate the inverse during the evaluation of error covariance. However, since we are already

using a really small amount of data for the stationary condition, it is not possible to provide a large amount of compressor pressure data to build the model. Hence, the RLS method is appropriate for the model fitting of the AR model. The RLS method improves on the non-recursive method by using the previous estimate to evaluate the error in the estimation.

AR model is a simplified form of an ARX model. The general form of ARX model is given by,

$$\vec{y}_k = \sum_{i=1}^{p} \vec{a}_i \vec{y}_{k-i} + \sum_{i=1}^{p} \vec{b}_i \vec{u}_{k-i} + \varepsilon_k \qquad (4.2.1.1)$$

where,

$\hat{y}_k$ = output vector

u = input vector

a, b = parameter matrices

ε = stochastic factor, also called residual (used for the optimization of the model)

An AR model doesn't have $\bar{b}$ and $\bar{u}$. If we also assume the residual to be 0, the ARX model becomes an AR model and the equation (4.2.1.1) becomes:

$$\hat{\vec{y}}_k = \sum_{i=1}^{p} \vec{a}_i \vec{y}_{k-i}$$

(4.2.1.2)

Here, the carat sign (i.e. ^) represents that the vector y is an estimated vector. The parameter matrix vector is represented as:

$$\vec{\theta} = \begin{bmatrix} \vec{a}_1 & \vec{a}_2 & ..... & \vec{a}_p \end{bmatrix}^T \qquad (4.2.1.3)$$

The value of p is chosen so that we can capture the dynamics of the system using the AR model. The equation for the RLS estimation is given below,

$$\hat{\vec{\theta}}_{k+1} = \hat{\vec{\theta}}_k + \frac{\vec{P}_{k+1} \vec{\phi}^{T}_{k+1}}{1 + \vec{\phi}_{k+1} \vec{P}_{k+1} \vec{\phi}^{T}_{k+1}} (\vec{y}_{k+1} - \vec{\phi}_{k+1} \hat{\vec{\theta}}_k) \qquad (4.2.1.4)$$

where,

$\hat{\vec{\theta}}$ = Estimated parameter matrix, $\hat{\vec{\theta}} \in \mathbb{R}^{1 \times p}$

p = the order of the estimation

$\vec{P}_{k+1}$ = Inverse Correlation Matrix

The fraction term serves as a weighting factor in the error corresponding to each order of the coefficient for the AR model. In this way, the equation finds the appropriate amount of the error to adjust on the previous estimate to update to the new estimate.

The Inverse Correlation Matrix (P_k) can be evaluated recursively as,

$$\hat{\vec{P}}_{k+1} = \hat{\vec{P}}_k - \hat{\vec{P}}_k \vec{\phi}^{T}_{k+1} \left(\frac{\vec{\phi}_{k+1} \hat{\vec{P}}_k}{1 + \vec{\phi}_{k+1} \hat{\vec{P}}_k \vec{\phi}^{T}_{k+1}} \right) \tag{4.2.1.5}$$

Hence, this equation takes the error covariance matrix produced in the previous time step and updates the matrix for the current time step by subtracting the change in every element of the matrix.

Similarly, the matrix ϕ is obtained as:

$$\vec{\phi}_{k+1} = \left[\vec{y}^{T}_{k+1} \quad \vec{y}^{T}_{k} \quad \dots \quad \vec{y}^{T}_{k+1-p} \right] \tag{4.2.1.6}$$

4.2.1.1 Batch Least Squares

The equation for the RLS estimation can evaluate the parameter matrix at time step 'k+1' by using the parameter matrix and error covariance matrix at time step 'k'. The process halts after processing the last element of the data. However, to initiate the computation, the initial values for the matrix's 'P' and '$\hat{\theta}$' should be provided.

Assume that we would like to evaluate the batch least-squares at k = 2* p, the matrix equation for 'P_k' is:

$$\vec{P}_k = (\vec{\Phi}^{T}_k \vec{\Phi}_k)^{-1} \tag{4.2.1.1.1}$$

Similarly, the expression for '$\hat{\theta}$' is,

$$\hat{\vec{\theta}}_k = (\vec{\Phi}^{T}_k \vec{\Phi}_k)^{-1} \vec{\Phi}^{T}_k \vec{Y} = \overline{P}_k \vec{\Phi}^{T}_k \vec{Y} \tag{4.2.1.1.2}$$

Here, the matrices 'Φ' and 'Y' are defined below,

$$\vec{\Phi}_k = \begin{bmatrix} \vec{y}_p^T & \vec{y}_{p-1}^T & \vec{y}_{p-2}^T & \cdots & \vec{y}_1^T \\ \vec{y}_{p+1}^T & \vec{y}_p^T & \overline{y}_{p-1}^T & \cdots & \vec{y}_2^T \\ \vec{y}_{p+2}^T & \vec{y}_{p+1}^T & \overline{y}_p^T & \cdots & \vec{y}_3^T \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \vec{y}_{2p}^T & \vec{y}_{2p-1}^T & \overline{y}_{2p-2}^T & \cdots & \vec{y}_p^T \end{bmatrix} \quad (4.2.1.1.3)$$

$$\vec{Y}_k = \begin{bmatrix} \vec{y}_{2p}^T & \vec{y}_{2p-1}^T & \vec{y}_{2p-2}^T & \cdots\cdots & \vec{y}_p^T \end{bmatrix} \quad (4.2.1.1.4)$$

4.2.1.2 Pseudocode for AR Model Fitting:

function ARmodel ([y_1, ….., y_n], p)
Input: Common blade pressure data [y_1, ….., y_n]
Total number of desired eigenvalues ‘p’<<n
Output: Parameter matrix θ = [a_1 a_2 …. a_p]
(previousP, previousθ) = batchleastsquares ([y_1, ….., y_{2p}], p)
modelparameters = previousθ
for i = (2p + 1) to n:
ϕ = buildphi ([y_i, ….., y_{i-p}])
P = evaluateP (previousP, ϕ)
θ = evaluateθ(previousP, previousθ, ϕ, y_i)
modelparameters.append (θ)
previousP = P
previousθ = θ
end
return modelparameters

function batchleastsquares ([y_1, …., y_n], p)
Input: Initial pressure data points [y_1, …...., y_n]
Total number of desired eigenvalues ‘p’<<n
Output: (initialP, initialθ)

$\boldsymbol{\Phi}$ = populatematrix ([y_1, ….., y_n])

initialP = inverse(transpose($\boldsymbol{\Phi}$)*$\boldsymbol{\Phi}$)

Y = [y_n, ….., y_{n-p}]

initialθ = initialP*transpose($\boldsymbol{\Phi}$)*Y

return (initialP, initialθ)

4.2.1.3 Code Implementation and Test of RLS Algorithm

A source code of the python implementation of the RLS algorithm is provided in Appendix 2. The test of the algorithm is done by creating synthetic data using a known difference equation. Then, the algorithm is implemented on the data to create model parameters and the model parameter is then used to build a differential equation which is also used to generate a second set of data. After that, the original data is plotted on top of the estimated data. Following parameters were used in the creation of the plot and the following results were obtained:

Case-1: When the number of original eigenvalues is less than the number of guessed eigenvalues

Original number of data generated: 10,000

Difference equation used:

y[i] = - 0.15*y[i-1] - 0.92*y[i-2] + random.uniform(0,0.025), y[0] = 10, y[1] = 15

Number of eigenvalues requested: 5

Sets of model parameters generated by RLS: 9900

Model parameter number used to generate estimated data: 9900th

Difference equation used to evaluate the estimated model:

yp[i] = param[0] * yp[i-1] + param[1] * yp[i-2] + param[2] * yp[i-3] + param[3] * yp[i-4] + param[4] * yp[i-5] + random.uniform(0,0.025), yp[0] = 10, yp[1] = 15

Plot:

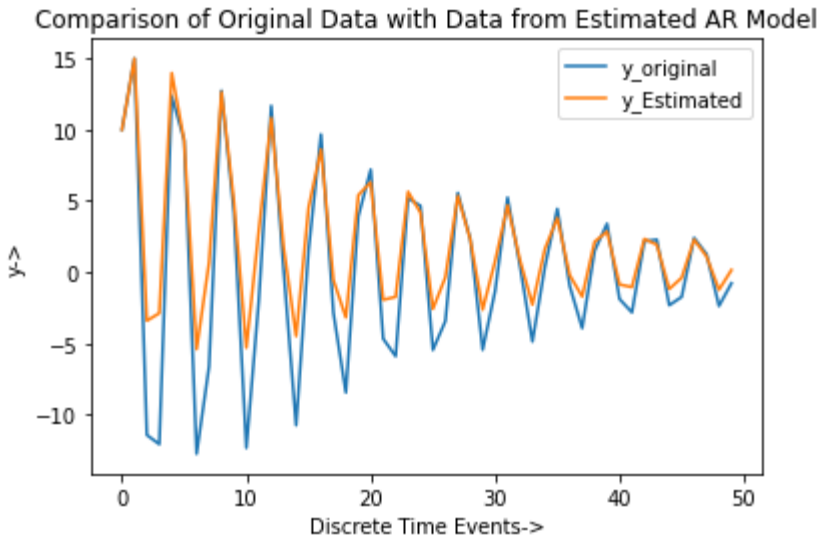


Fig 4.2.1.3.1: Comparison Between Original Data and Estimated Data for Case 1

Case 2: When the number of original eigenvalues is equal to the number of guessed eigenvalues

Original number of data generated: 10,000

Difference equation used:

y[i] = - 0.15 * y[i-1] - 0.92 * y[i-2] + random.uniform(0,0.025), y[0] = 10, y[1] = 15

Number of eigenvalues requested: 2

Sets of model parameters generated by RLS: 9900

Model parameter number used to generate estimated data: 9900th

Difference equation used to evaluate the estimated model:

yp[i] = param[0] * yp[i-1] + param[1] * yp[i-2] + random.uniform(0,0.025), y[0] = 10, y[1] = 15

Plot:

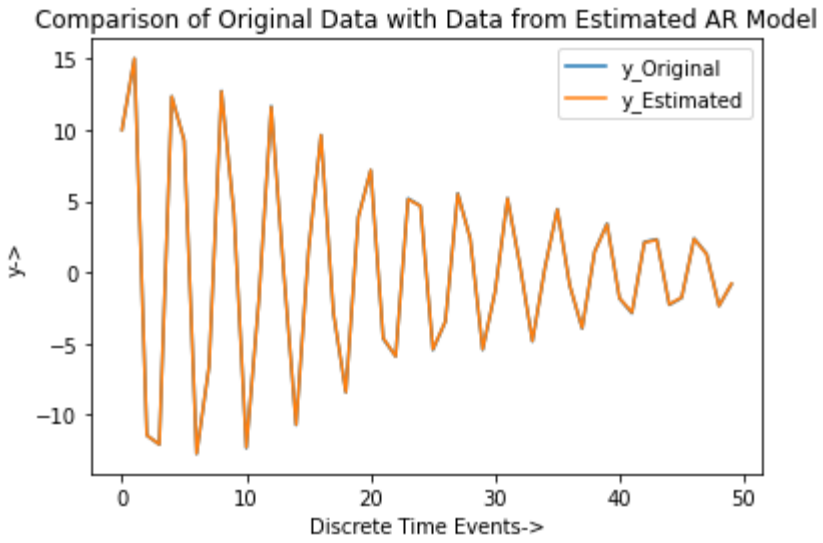


Fig 4.2.1.3.2: Comparison Between Original Data and Estimated Data for Case 2

In this case, the original data and the estimated data are greatly similar resulting in high accuracy.

Case 3: When the number of original eigenvalues is greater than the number of guessed eigenvalues

Original number of data generated: 10,000

Difference equation used:

y[i] = - 0.15 * y[i-1] - 0.92 * y[i-2] - 0.22 * y[i-3] + random.uniform(0,0.025), y[0] = 10, y[1] = 15

Number of eigenvalues requested: 2

Sets of model parameters generated by RLS: 9900

Model parameter number used to generate estimated data: 9900th

Difference equation used to evaluate the estimated model:

yp[i] = param[0] * yp[i-1]+param[1] * yp[i-2] + random.uniform(0,0.025), y[0] = 10, y[1] = 15

Plot:

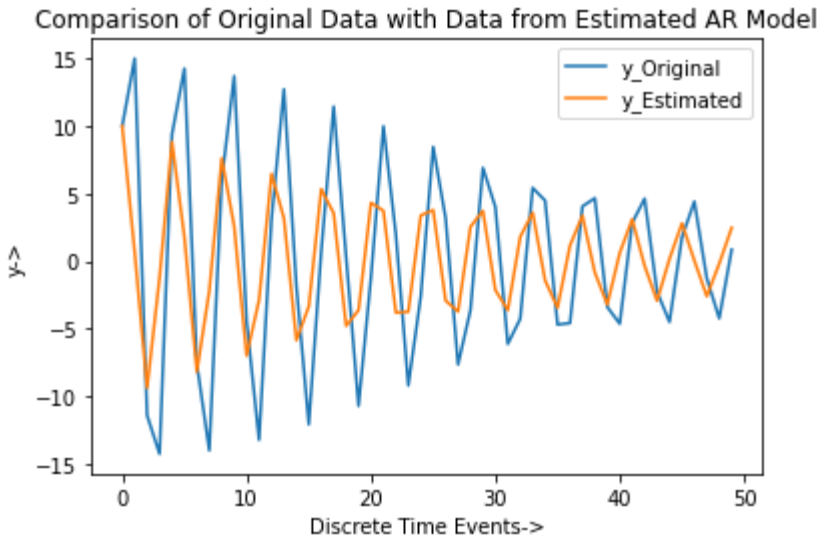


Fig 4.2.1.3.3: Comparison Between Original Data and Estimated Data for Case 3

4.2.2 GESDT Algorithm

The GESDT algorithm[32] is used to detect multiple outliers in a univariate data set. The major assumption about the data set is that it is an approximating a normal distribution.

For a predefined value of potential outliers' 'r', GESDT performs 'r' successive tests for the number of outliers defined for the hypothesis:

H_0: there are no outliers in the data set

H_a: there are 'r' outliers in the data set

The high-level procedure of the algorithm involves evaluating test statistic 'R_i' and critical region 'λ_i' where i is iterated from 1 to the prespecified value 'r'. The data point corresponding to the test statistic 'R_i' is an outlier if $R_i > \lambda_i$

4.2.2.1 Test Statistic

The major purpose of the test statistic is to find out the most "misbehaved" data points in the data set to compare with the studentized distribution data set. First, the variance of each of the

data points is evaluated, and the maximum of the variance is used in the equation below to evaluate the test statistic:

$$R_i = \frac{\max_i | x_j - \overline{X} |}{\sigma} \quad (4.2.2.1.1)$$

where,

X = Mean of the data set

σ = Standard Deviation of the data set

i = 1,....,r

j = 1,...,n

n = size of the data set

For every successive iteration of 'i', the data point x_j corresponding to '$\max_i |x_j - \overline{X}|$' is removed from the data set and a new R_i is evaluated for the new data set.

4.2.2.2 Critical Value

The critical values are simply evaluated using the equation below:

$$\lambda_i = \frac{(n-i) * t_{p,n-i-1}}{\sqrt{((n-i-1) + (t_{p,n-i-1})^2) * (n-i+1)}} \quad (4.2.2.2.1)$$

where,

α is the significance level for the outliers taken as 0.05

p = 1- $\frac{\alpha}{2(n-i+1)}$

i = 1,....,r

$t_{p,v}$ = 100pth percentage points from the studentized t-distribution with v degrees of freedom

n = size of data set

4.2.2.3 Pseudocode of GESDT Algorithm

function gesdt ([x_1, ….., x_n], r)

Input: AR model parameters [x_1, …..., x_p]

Number of potential outliers ‘r’

Output: Number of outliers in the data set

R = evalautetesstatistic ([x_1, ….., x_n], r)

λ = evalautecriticalvalues (r)

outlier = 0

for i = 0 to r-1:

  if R[i] > λ[i]:

    outlier += 1

return outlier

function evalautetesstatistic ([x_1, ….., x_n], r)

Input: AR model parameters [x_1, ….., x_p]

Number of potential outliers ‘r’

Output: Test statistics [R_1, R_2,....,R_r]

data = [x_1, ….., x_n]

for i = 0 to r - 1:

  maxvariance, maxvarianceindex = computevariance(data)

  standarddeviation = std(data)

  R[i] = maxvariance/standarddeviation

  data = deleteelement(data, maxvarianceindex)

return R

function computevariance ([x_1, …...., x_n])

Input: array of model parameters [x_1, …...., x_n]

Output: maximum variance element max(j) |x_j - Mean| and the respective index

array = [x_1, …...., x_n]

mean = mean(array)

for j = 0 to |array| - 1:

  variance[i] = |oldarray[j] - mean|

maxvariance, index = max(variance)

return (maxvariance, index)

function computecriticalvalues (r, n)

Input: Number of potential outliers ‘r’ and the size of dataset ‘n’

Output: Critical Values [λ_1, λ_2,..., λ_r]

α = 0.05

for i = 0 to r-1:

    p = 1 - α/(2*(n-i+1))

    t = computetdistribution(p, n-i-1)

    λ[i] = ((n-i) *t) / sqrt((n-i-1+t^2) *(n-i+1))

return λ

4.2.2.4 Code Implementation and Test of GESDT Algorithm

A source code of the python implementation of the RLS algorithm is provided in Appendix 3. The test is done by creating 100 random points between 0 and 1 and manually substituting 5 values with integer and double values. Then the dataset is passed through the algorithm along with the maximum number of outliers set to 5 and the significance level for the outliers set to 0.05. The algorithm counts the number of outliers that it can find and gives out the number which is printed in the output terminal.

Result:

```
In [1]: runfile('C:/Users/Shishir/Documents/GitHub/Compressor-Research/Stall Precurser Detection/GESDT.py',
wdir='C:/Users/Shishir/Documents/GitHub/Compressor-Research/Stall Precurser Detection')
5
```

Fig 4.2.2.4.1: Output Result of the GESDT Algorithm

Hence, the algorithm can accurately predict the outliers in a dataset.

Chapter -5: Compression System Design

The physical system design of the single-stage axial compressor system consists of a compressor, plenum, and throttle. The majority of the sensing is performed at the compressor mount section that lies between the compressor and the plenum section of the compressor.

5.1 Subsystem Design

The majority of the components that comprise the compression system is made up of 3D printed parts that are manually designed in SolidWorks(TM). The diameter of the duct is 92 mm and the length of the physical system is 1127.3 mm.

Table 5.1.1: List of Off-the Shelf Hardware Used in the System Design

Items	**Product Name**
Compressor	Dr MadThrust 90 mm EDF
Compressor Controller	YEP Brushless ESC 120A
Compressor Controller Power Supply	DCP3010D
Throttle Motor	Adafruit NEMA-12 Stepper Motor
Throttle Motor Controller	Adafruit Motor Shield V2

5.1.1 Compressor

An off-the-shelf EDF is modified and used as a compressor for the compression system. The internal diameter of this compressor has dictated the diameter of the compression system. The EDF only contains the rotor. Hence the posterior metal covering part of the compressor is replaced with a 3D printed design to accommodate a stator that would together form a single rotor-stator pair for the compression system.

Fig 5.1.1.1: Compressor with an Attached Stator

5.1.1A Rotor and Motor

The 12-blades rotor that came along with the EDF is used as-is. Based on the datasheet, the rotor is built using CNC alloy. The blade dimensions were measured and a SolidWorks model was developed for the rotor with a cord length value of 13 mm and thickness of 2.18 mm. The motor used had a labeled continuous power of 4800 W, maximum RPM of 48000, and maximum thrust of 5.4 kg. The physical dimension of the motor housed inside the duct of the compressor is 37 mm in diameter and 103 mm in length.

Fig 5.1.1A.1: Model of the Rotor[1]

5.1.1B Stator

[1] Modeled by Andrew Anderson

Some 3D-printed stators were designed to integrate with the rotor section of the compressor. The stator section was designed such that the stators will be placed next to the rotor section. The stators are designed to accommodate the motor support beams.

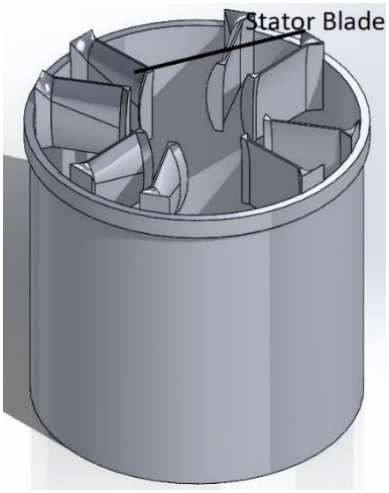


Fig 5.1.1B.1: Stator Blade with 0^0 Angle[2]

5.1.1C Motor Controller

A motor controller was used to control the rpm of the compressor. A program was written in Arduino Mega 2560(Appendix: 9) to map the voltage values from the potentiometer to the rpm adjustment signal into the compressor motor controller. Similarly, an emergency stop button was incorporated into the code as an interrupt to halt the operation of the compressor in an event of an emergency. (Power supply to controller)

[2] Designed by Andrew Anderson

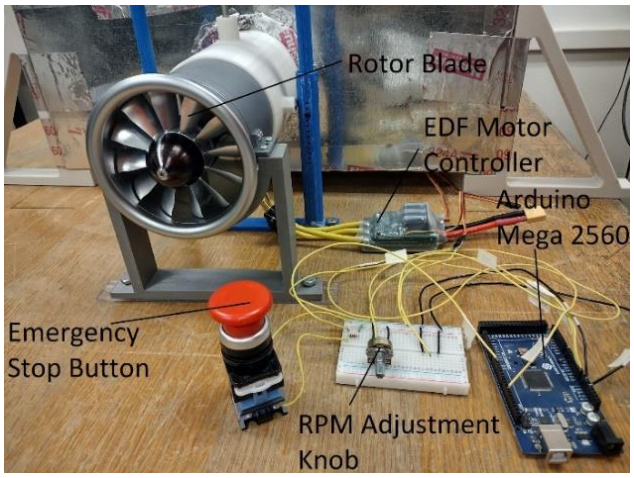


Fig 5.1.1C.1: Setup of Control System for the Compressor Motor

5.1.2 Plenum

A rectangular plenum with dimensions 609.6 mm L by 609.6 mm W by 203.2 H is built using galvanized steel sheet metal. The supports for the plenum were 3D printed to restrain the movement of the plenum. On the breadth face of the plenum, the compressor and throttle mount are attached using bolts and nuts. The compressibility of air inside the plenum affects the dynamics of the compression system and different volumes of the air have different compressibility values. So changing the volume of the plenum affects the dynamics of the compression system.

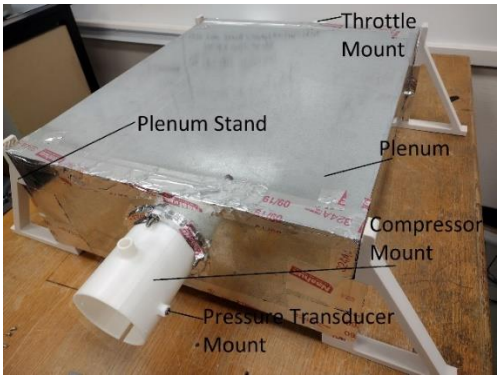


Fig 5.1.2.1 Physical Plenum with Compressor Mount[3]

[3] Plenum Designed by Undergraduate Senior Design Team MOTR

5.1.3 Throttle

A cone-shaped throttle valve is designed and built to obstruct the flow of air coming out of the throttle mount. A mechanism is designed that converts the rotatory motion of the threaded rod into the linear motion of the nose cone. The threaded rod is actuated using a stepper motor and the motor is programmed using an Arduino Mega 2560 and motor controller shield. The mechanism is surrounded by a cylindrical enclosure to prevent the air escape out of the duct. Finally, the air coming out of the throttle is vented to the ceiling using an exhaust pipe.

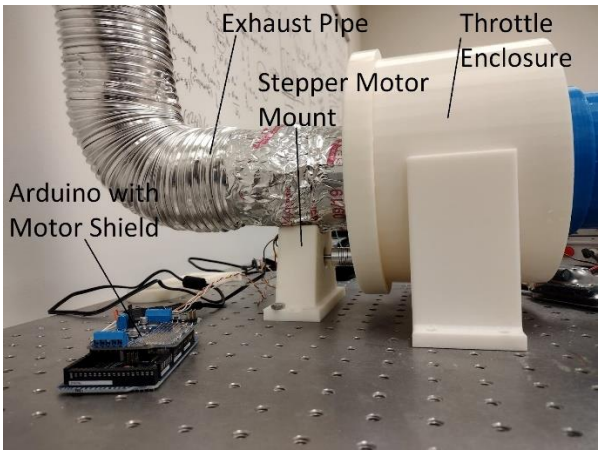


Fig 5.1.3.1: Setup of the Throttle and Exhaust Valve

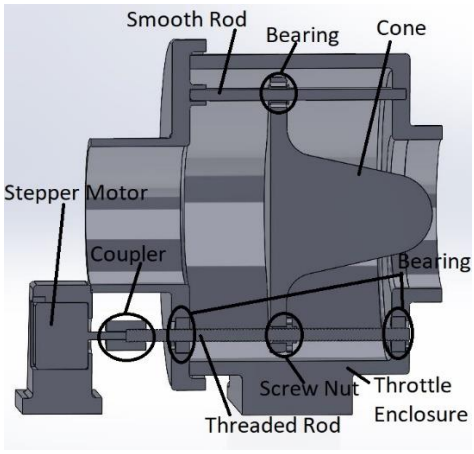


Fig 5.1.3.2: Mechanism of Throttle

5.2 Measurement Electronics

To trace out the characteristic of the compressor and measure the efficiency, parameters pressure, rpm, temperature, and torque are needed to be measured [33]. Similarly, a DAQ is required to collect the data from the sensor and a data processing software is desired that can process the data in real-time to output the desired curves in real-time. Hence, the following sensors and DAQ systems are proposed.

Table 5.2.1: List of Measurement Electronics Used in the System

Items	**Product Name**	**Additional Information**
Pressure Sensor Transducer	Kulite XCS-190SM – 30A	SN-A: 8460-2-259

		SN-B: 8524-4-145
Pressure Sensor Signal Conditioner	Kulite KSC-1-BU4	SN-A: 202181-025 SN-B: 202181-070
Pressure Sensor Power Supply	TekPower TP3005T	NA
Flow Sensor	DegreeC F450	NA
Torque Sensor	HBK T22/ 50 Nm	SN: 420034047
Op-AMP for Voltage Amplifier	TLV2471AIP	NA
RPM Sensor	Monarch ROS-W	NA

5.2.1 Pressure Sensor

The pressure rise in a single stage of a compressor is expected to be no more than 4%. If the normal atmospheric pressure is 12.4 psi, the maximum pressure rise that can be observed in a single-stage compressor will be less than 12.9 psi. Hence, the pressure sensor should be accurate with good sensitivity to the small changes in pressure.

Hence, two pressure transducers were chosen: one that would measure the pressure inside the compressor and another one that would measure the pressure at the inlet of the compressor. Both the transducer signals were passed through their respective signal conditioners to filter and amplify the pressure signal. The pressure transducer is mounted on: the compressor mount and next to the stator stage, one at a time, to investigate if the change in the cross-section due to the presence of the motor in the compressor duct has a significant role in the pressure rise reading of the compressor. The sensitivity of the transducers used was 6.731 mV/psi and 6.738 mV/psi.

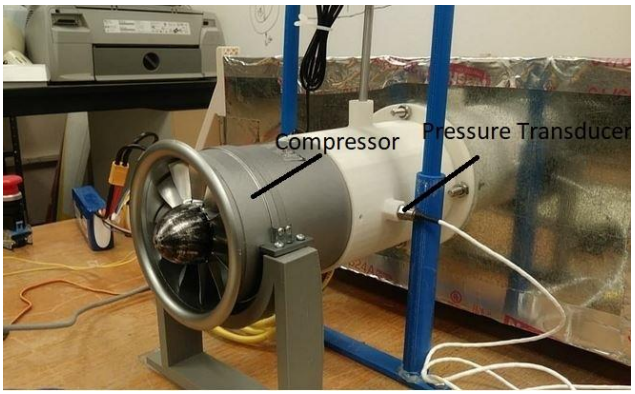


Fig 5.2.1.1: Setup of Pressure Transducer

Following equations were used to convert the sensor signal into a pressure signal:

Pressure_psi = sensitivity * sensor_output (5.2.1.1)

The ‘Zero In Adjust’ on the signal conditioner was used to change the DC offset of the output voltage from the signal conditioner.

5.2.2 Flow Sensor

A flow sensor was mounted perpendicular to the direction of the airflow that would measure the airspeed and the temperature of the air moving inside the compressor. The airspeed sensor has a sensitivity of 3.97 mm/s per mV and the temperature sensor has a sensitivity of 0.012 ^{0}C per mV. Similarly, the sensor has a response time of 400 ms.

Fig 5.2.2.1: Setup of Flowmeter

Following equations were used to convert the sensor signals into flow signals:

airspeed_m/s = speed_sensitivity * sensor_output (5.2.2.1)

Temperature_celcius = temp_sensitivity * sensor_output (5.2.2.2)

5.2.3 Torque Sensor

The rotating shaft of the motor of the compressor is connected to a Torquemeter using two couplers and a transmission shaft. The sensitivity of the sensor is 9.9953 Nm/V for clockwise torque and -9.9788 Nm/V for counterclockwise torque.

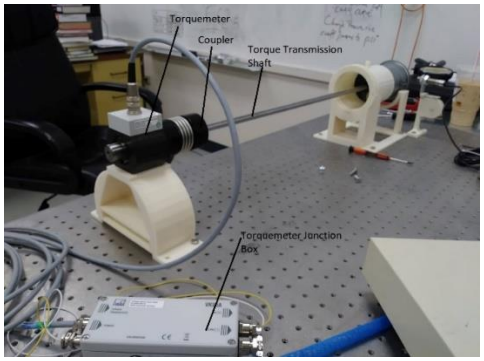


Fig 5.2.3.1: Setup of Torquemeter

Following equations were used to convert the sensor signal into torque signal:

torque_Nm = torque_sensitivity * sensor_output (5.2.3.1)

For the single-stage compressor, the expected torque is between 0 Nm to 0.99 Nm, and hence the expected output voltage from the Torquemeter sensor is 0.01V to 0.1V. Hence, to receive the torque signal without noise, a voltage amplifier module was designed that would energize using the voltage port from Arduino, have a port to connect the low amplitude torque signal, and automatically supply the amplified voltage to the analog input channel of the Arduino board. The average experimental gain of the voltage amplification module between 0.01V to 0.1V range was found to be 48.02.

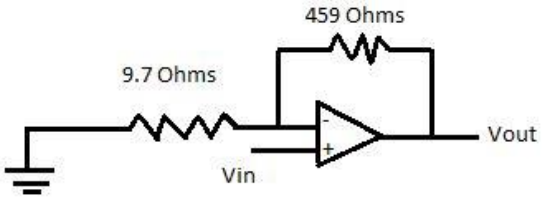


Fig 5.2.3.2: Schematic of Amplification Circuit

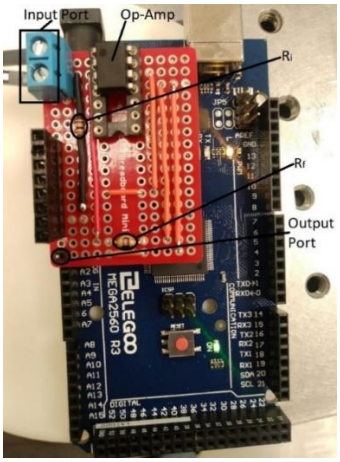


Fig 5.2.3.3: Voltage Amplification Module

5.2.4 Rotation Speed Sensor

To encode the RPM information of the compressor, an optical rpm sensor is used that converts the information as a TTL pulse by using the concept of dispersion of light. Most of the surface of the inlet centerpiece of the compressor is painted black except for a strip that would reflect the light coming out of the emitter into the detector of the sensor. The rising edge of the pulse is counted once every revolution to convert the pulse signal into the RPM signal. Simulink(TM)'s 'Tachometer' block uses the following equation to convert pulses to rpm:

$$\text{Rpm\_speed} = \frac{number\ of\ pulses}{sampling\ time} * 60 \qquad (5.2.4.1)$$

The following settings are used to sample the rpm signal:

(Digital)Pin Number: 2

Interrupt mode: Rising

Sample Time: 0.1s

The test is conducted by clicking 'Monitor and Tune' after selecting the hardware settings. The output of the rpm sensor is not smooth. Hence, a 'Low Pass Filter' block is used in the default settings provided in the Simulink to smooth out the signal. However, a tradeoff is observed in this step. The filter smooths out the signal but using the block adds delay in the signal.

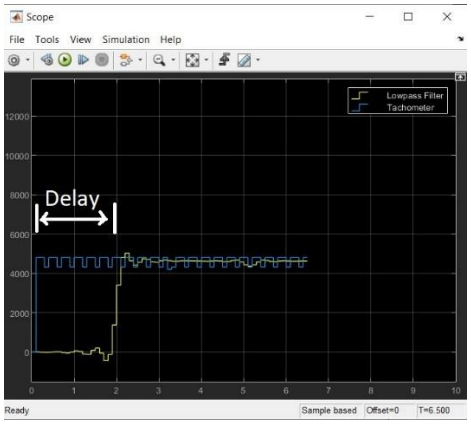


Fig 5.2.4.1: Comparison of Original and Smoothed RPM Signal

5.3 Data Acquisition

The output from the sensors is collected using the analog and digital ports of Arduino Mega 2560. The acquired signals were then processed in real-time using the graphical programming language, Simulink. Simulink blocks were created to perform the following tasks:

5.3A Differential Pressure Measurement

5.3B Characteristic Curve Measurement

5.3C Efficiency Measurement

5.3A Differential Pressure Measurement

To measure the pressure rise coefficient, the difference in the pressure at the inlet and inside the compressor has to be evaluated. Hence, the difference in the pressure is simply evaluated using the equation below:

$$\Delta\text{Pressure} = \text{Pressure inside the compressor} - \text{Pressure at the inlet} \quad (5.3A.1)$$

5.3B Characteristic Curve Measurement

The measurement of the characteristic of the compressor involves plotting the pressure rise coefficient (Ψ) vs flow coefficient (Φ) where,

$$\Psi = \frac{\Delta\text{Pressure}}{0.5 * (\text{Density of air}) * (\text{Mean rotor speed})^2} \quad (5.3B.1)$$

$$\Phi = \frac{Axial\ Flow\ Speed}{Mean\ Rotor\ Speed} \quad (5.3B.2)$$

Here,

ΔPressure is measured using Eq. 5.1

The density of air is assumed as a constant value of 1.225(49/40) kgm^{-3}

(Density of air at 15 ^{0}C at the sea level)

Mean Rotor Speed = mean radius(r) * angular speed (ω) = $\frac{r * 2 * \pi * RPM}{60}$(m/s)

Mean radius = $\frac{\frac{90mm}{2} - \frac{37mm}{2}}{2} + \frac{37mm}{2} \cong 31.75$ mm $\cong 0.0318$ m

Mean Radius is used as a constant value in the equation.

The axial Flow speed is measured using the flowmeter sensor.

5.3C Efficiency Measurement

The measurement of the efficiency of the compressor requires sensing of temperature at the inlet, pressure at the inlet and the exit, mass flow rate at the inlet, torque on the compressor shaft, and the rpm of the shaft. The equation for the efficiency of the compressor [33] is:

$$\eta = \frac{\kappa}{\kappa - 1} * R * T_{inlet} \left(\left(\frac{P_{exit}}{P_{inlet}} \right)^{\frac{\kappa - 1}{\kappa}} - 1 \right) * \left(\frac{\dot{m} * 60}{2\pi * RPM * \tau} \right) \quad (5.3C.1)$$

where,

κ = Adiabatic index of air = 1.4

R = Specific Gas Constant = 287.5 $JKg^{-1} K^{-1}$ (for air)[34]

T_{inlet} = Temperature at the inlet (Unit: Kelvin)

P_{exit} = Pressure at the exit of the compressor stage (Measured using pressure sensor) (Unit: Pascals)

P_{inlet} = Pressure at the inlet of the compressor stage (Measured using pressure sensor) (Unit: Pascals)

$\dot{m}$ = mass flow rate of the compressor at the inlet= ϱ * A* C (Unit: Kgs^{-1})

where,

ϱ = Density of air = 1.225(49/40) kgm^{-3} [cite]

A = Cross sectional area at the inlet = $4((90\ mm)^2 - (37\ mm)^2) \cong 5286.5\ mm^2 \cong 0.0053\ m^2$

C = Air speed at the inlet (Measured using flow sensor) (Unit: ms^{-1})

τ = Torque exerted on the compressor shaft (Measured using Torquemeter) (Unit: Nm)

The Simulink Code for the measurement of a real-time efficiency of a compressor is provided in Appendix: 7.

5.4 System Integration

A compressor is created by connecting the rotor stage with the 3D-printed stator. The compressor is supported using a 3D printed stand. The compressor stage acts like a pump with positive damping [35]. The end of the compressor is connected to the 3D printed compressor mount. The compressor is connected with the compressor mount using a tight fit and it is made

sure that the diameter of the duct remains tentatively the same. The compressor mount is designed to accommodate a nut for the pressure sensor and a probe for the flow sensor. The probe for the flow sensor is supported in place by using a 3D printed flowmeter mount. The mass of air up to this section inside the duct acts as inertia [35]. The duct after the compressor mount is opened to the plenum which is equivalent to the spring, according to the mass-spring-damper system representation of a compressor [35].

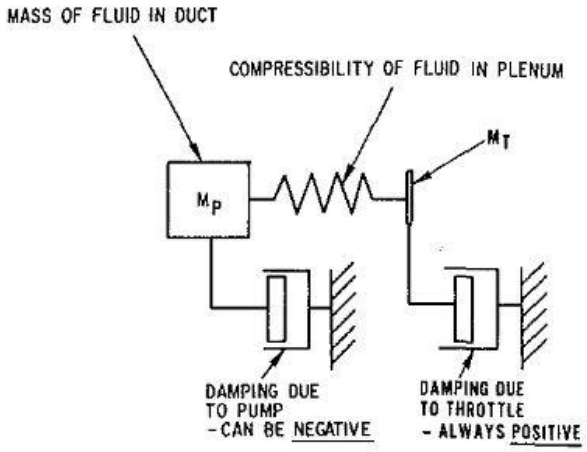


Fig 5.4.1: Mass-Spring-Damper equivalent of Pumping System [35]

Similarly, the plenum is connected to a 3D printed throttle mount on the other side which has the same diameter as the duct of the compressor. The duct is opened to the throttle which acts as a damper with positive damping[35]. The throttle section also contains a 3D printed housing for the throttle actuator. The end of the throttle is connected with an exhaust pipe which vents the air out of the throttle into the false ceiling of the laboratory.

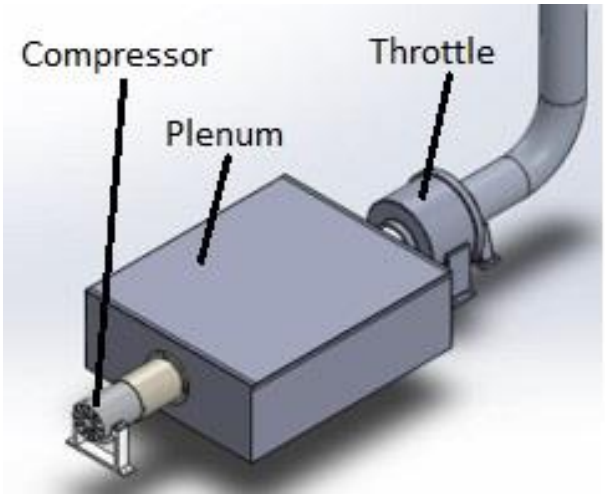


Fig 5.4.2: SolidWorks$^{(TM)}$ Assembly of the M-G Compressor

The connections of the sub-systems are sealed using aluminum-based tape to prevent air leakage. The system is designed such that it is easy to replace subsystem components. For example, if the stator needs to be replaced, it can be done by only disconnecting the compressor section.

Chapter - 6: System and Subsystem Test

The Single-State Moore-Greitzer compression is an electromechanical system that consists of multiple electrical, mechanical, and software subsystems designed to work together to reliably and repeatably generate the characteristic of a compression system. Hence, several tests are conducted to make sure that the sensors and the subsystems are working the way they are supposed to.

6.1 Sensor Testing

Although it is desired that the sensors provide a direct measurement of the parameter of interest, they have their dynamics that allow the sensors to quantize the information to be decoded into the physical parameter of interest. Because of this, the physical quantities can be extracted out of sensors using linear or nonlinear measurement models depending on the sensor dynamics. However, the dynamics also allow the sensors to have errors. The errors demonstrated by the sensors are broadly classified into stochastic errors and deterministic errors. The errors exhibited by the sensors used in the experiment are provided below.

6.1.1 Run to Run Bias in Pressure Sensor

Run to run bias is a stochastic type of sensor error where the sensor produces a slightly different reading every time the sensor is run under the same mechanical condition [36]. This is typically a difficult type of sensor error to tackle and can’t be overcome before starting the sensor [36].

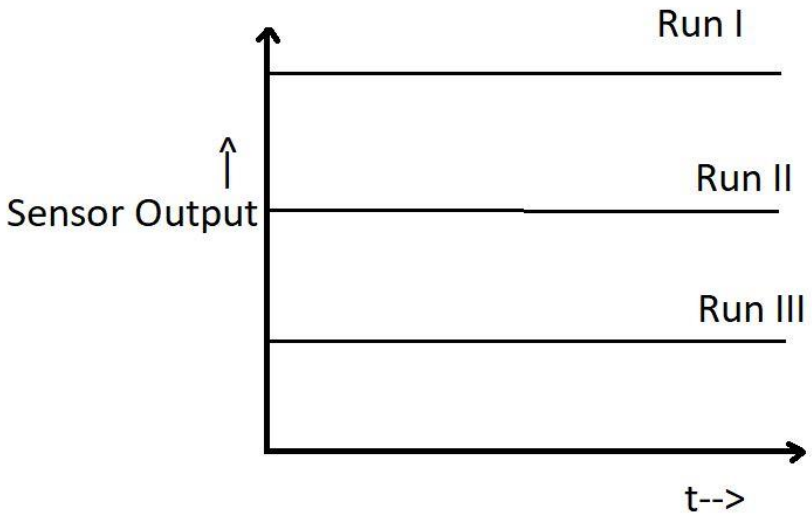


Fig 6.1.1.1: Example Plot of Run to Run Bias

Both signal conditioners connected to the pressure transducer exhibited the run to run bias in the initial value of pressure. When the sensor is first turned on, it is expected that the pressure transducers output the atmospheric pressure value on that day. However, the pressure transducers output a slightly different value every time the sensor is run and then change as per the sensitivity value of the sensor on top of that initial value. During the characterization experiment, this creates a non-zero pressure difference value even when the compressor is not running.
One way to mitigate this issue is by using 'Zero-in Adjust' to manually set the initial value of the sensor to the right value. The right value can be found by having a digital barometer in the lab to know what the atmospheric pressure is on that particular day.

6.1.2 Spike Sensor Fault in RPM Sensor

Sensor faults are usually caused by mechanical or hardware issues. This can cause the sensor to output a completely erroneous measurement that is even outside the noise profile. Using such measurements in the control system can cause the controller to destabilize[36]. Hence, the sensors used in the control system should be free of sensor faults.

During the instrumentation of the RPM sensor, it was found that the RPM outputted by the sensor using the Tachometer block in Simulink exhibited a spike sensor fault.

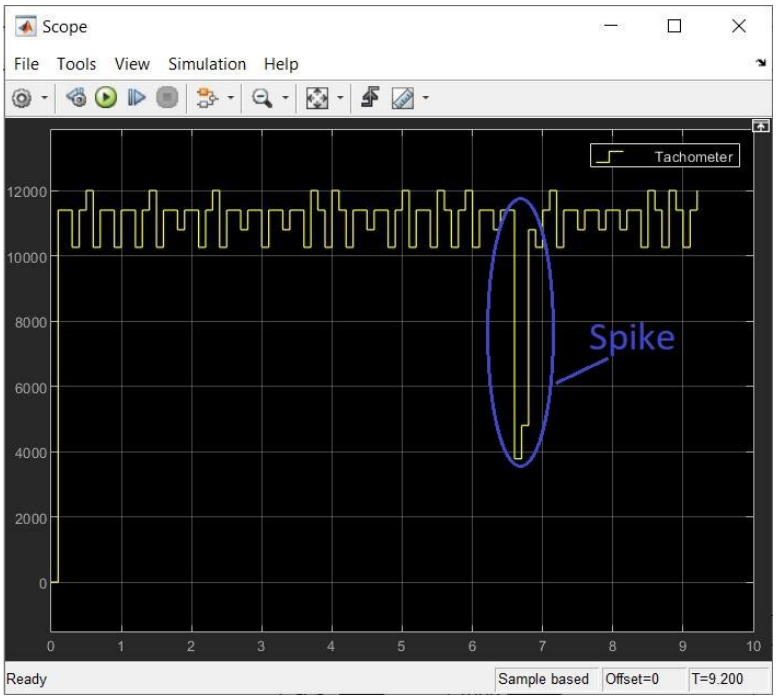


Fig 6.1.2.1: Spike Sensor Fault in the RPM Output

Some ways to mitigate sensor faults are by making sure the rpm meter is tightly fit in the RPM mount, by excluding the faulty measurements by processing the data using innovation checking or Normalized Innovation Squared (NIS) methods[36]. A better Tachometer system can be built by purchasing a panel tachometer box that is supposed to be used along with this sensor.

6.2 Stator Testing

Several stator designs were created to pair with the rotor to investigate the pressure rise. The objective of these tests is to compare the maximum pressure rise observed with these 3D printed stators with the original section without the stator stage. Figure 6.2.1 shows the assembly of the stator designs used in the stator testing.

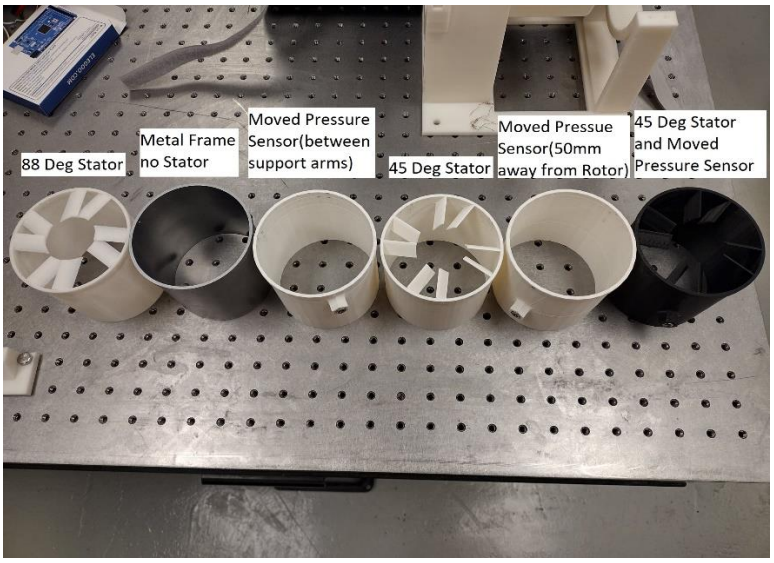


Fig 6.2.1: Assembly of Stator Designs Used in Stator Testing[4]

The experimental setup of the system is designed such that the compressor is directly connected to the throttle of the compressor using a new 3D printed compressor mount. The mount is connected to the throttle by press-fitting and the compressor by loose-fitting aided with the aluminum tape.

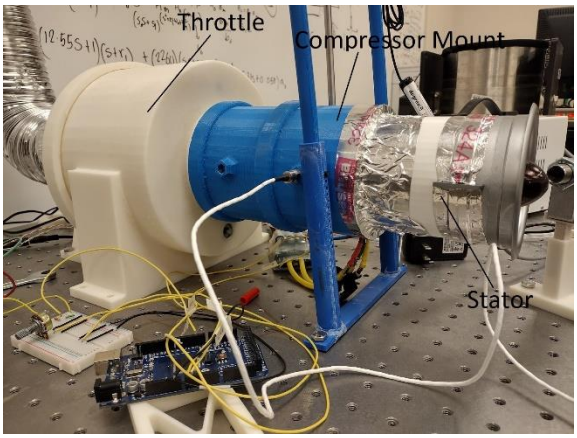


Fig 6.2.2: Setup for Stator Evaluation Experiment

Fig 6.2.3: Setup of Rotor-Stator Pair

For each experiment, the setup is only changed on the stator section with various designs of the stators. The initial pressure from the pressure sensor is recorded and then the compressor is run

[4] Stators Designed by Shishir Khanal, Andrew Anderson and Cooper Dastrup

from 0 to 13,000 RPM and the maximum pressure value is recorded. These two values are then recorded to obtain the maximum pressure rise.

Table 6.2.1: Pressure Rise Evaluation for the Proposed Stator Designs

Stator Designs	Initial Pressure (psi)	Maximum Pressure (psi)	% pressure rise
88^0 stator	13.07	13.29	1.6
	13.11	13.32	1.6
Metal frame no stator	13.43	13.69	1.9
	13.36	13.65	2.7
Moved Pressure Sensor (between the support arms)	12.71	12.85	1.1
	12.71	12.82	0.9
45^0 Stator	13.18	13.47	2.2
	13.18	13.47	2.2
Moved Pressure Sensor (50mm away from rotor) and no stator	12.71	12.89	1.42
	12.71	12.89	1.42
45^0 stator and moved (50 mm away from rotor) pressure sensor	12.96	13.47	3.9
	12.85	13.11	2.02

Among the 3D-printed stators/no stators, the 45^0 stators with the pressor sensor mount located 50 mm away from the showed the maximum pressure rise. This was expected because the location allows the cross-sectional area of the compressor to remain constant while the original mounting location of the pressure sensor diffuses the pressure because of an increase in the cross-sectional area due to the end of the section occupied by the compressor motor. In addition, this part also contains 45^0 stators which, by itself, has provided some pressure rise.

Although these pressures rise values are promising, these pressures rise percentages shouldn't be taken on their face values because these percentages represent the range of pressure rise that each design can provide but also added is the noise from the sensor. The result of the optimal part out

of this part is the one that gives the maximum pressure rise but its actual percentage pressure rise might look different when noise is eliminated.

6.3 Characterization Test

The characterization experiments deal with the plotting of the compression system and throttle characteristics on the phase space with the non-dimensionalized state variables pressure rise coefficient and the flow coefficient. The characterization experiments use the same Simulink codes but the experiments are designed differently. When the compressor is turned off, the rotor speed, airspeed, and pressure difference are 0. Hence, the pressure rise coefficient and flow coefficient has an indeterminate form zero divided by zero. For the characterization experiments, the indeterminate forms before running the compressor are assumed to be 0. Hence, the system is assumed to be at the origin when the compressor is not running. In the pressure rise evaluation tests tabulated in Table 6.2.1, 45^0 stators and moved pressure sensor and metal frame, no stator showed the most promising results. Hence, these two mounts were used in the characterization experiments.

6.3.1 Throttle Characterization

The physical setup for the throttle characterization is shown in Figure 6.3. The position of the throttle is kept constant during the experiment and the rpm of the compressor is changed. The real-time phase plots of the pressure rise coefficient versus flow coefficient are recorded in the Simulink in real-time.

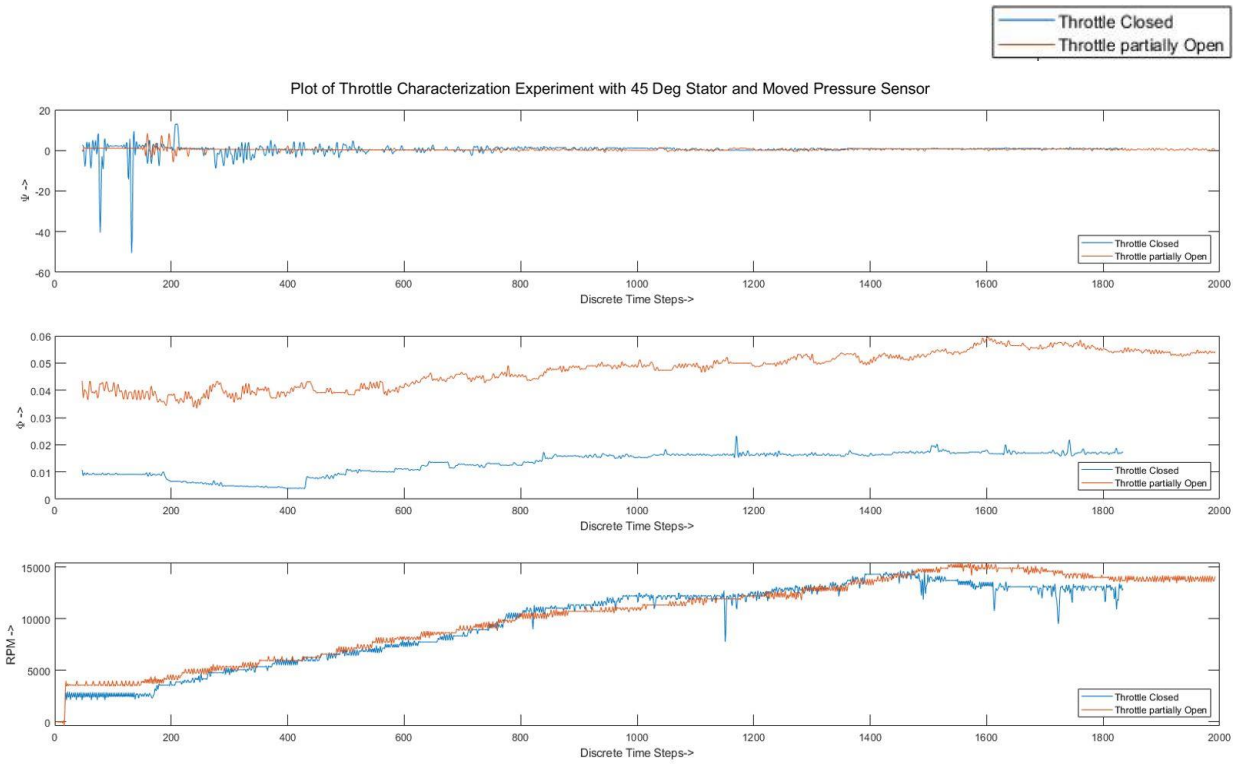


Fig 6.3.1.1: Plot of Throttle Characteristic Parameters at two Positions of Throttle with 45^0 Stator and Moved Pressure Sensor

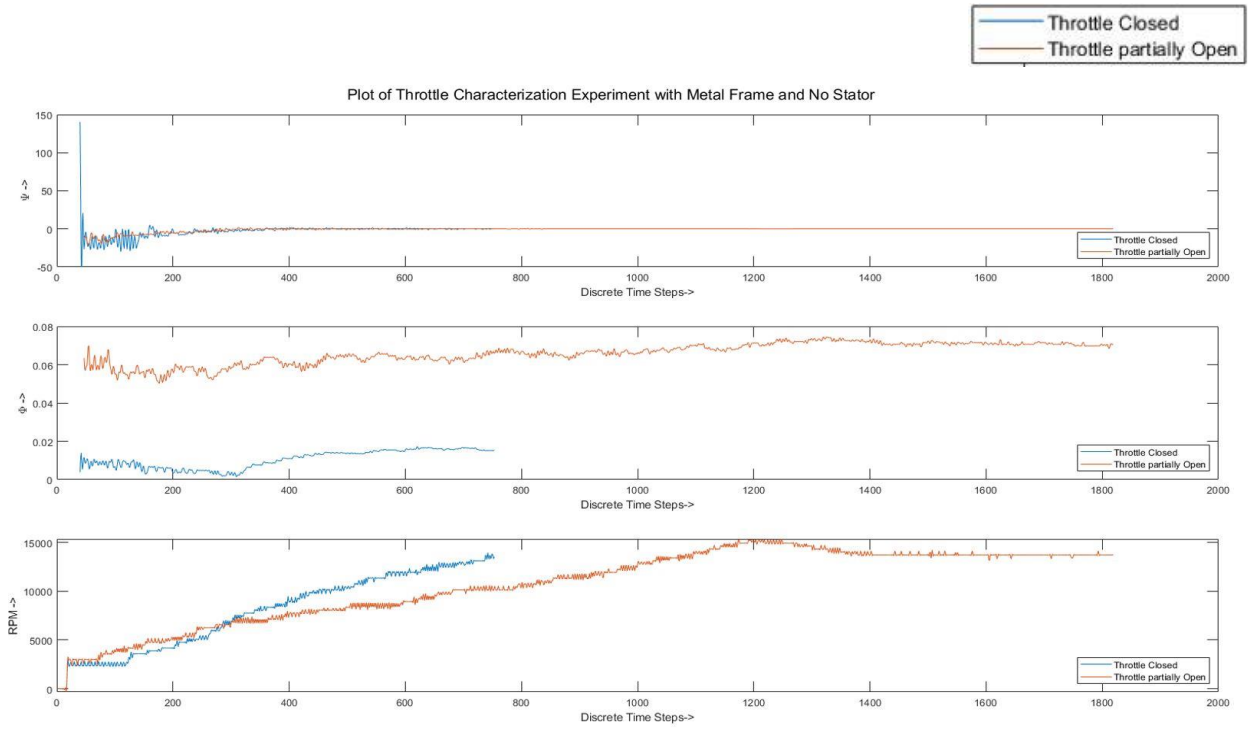


Fig 6.3.1.2: Plot of Throttle Characteristic Parameters at two Positions of Throttle with Metal Frame and no Stator

In this test, the two experiments: Throttle Closed and Throttle Partially Open were tuned to the same maximum rpm but they were not coordinated to run for the same amount of time. Hence, one experiment has more data than the other which is shown in Figure 6.3.1.1. The Flow Coefficient 'Φ' is expected to have smaller values when the throttle is closed than when it is partially open indicating Pressure Rise Coefficient 'Ψ' increases much faster than the Flow Coefficient. However, it was expected that the Pressure Rise Coefficient would increase much faster for the throttle closed position than in the partially open position. Unfortunately, the Pressure Rise Coefficient was found to almost be constant. The Kulite technical support recommends 2000 samples per second for a time-domain analysis but because Tachometer Block only worked for 10 samples per second sampling rate, the characterization experiment was performed at 10 Hz. This might have caused aliasing in the sensor measurements. Figure 6.3.1.2 also shows the same kind of qualitative results.

6.3.2 Compressor Characterization

The setup of the experiment is shown in Figures 5.4.2 and 6.3.2.1. In this experiment, the throttle is initially kept in a completely open position and later actuated during the test. The compressor is turned on and runs at a fixed rpm. While the rpm remains constant, the throttle is slowly closed using the stepper motor actuator and the phase space characteristic is measured using the Arduino DAQ in Simulink$^{(TM)}$ using the block code provided in Appendix 5. The characterization experiment is also done with the metal frame and no stator and 3D printed 45^0 stators with the pressure sensor moved to 50mm away from the rotor. Two sets of tests were conducted and the results are provided in Figures 6.3.2.2 and 6.3.2.3 below.

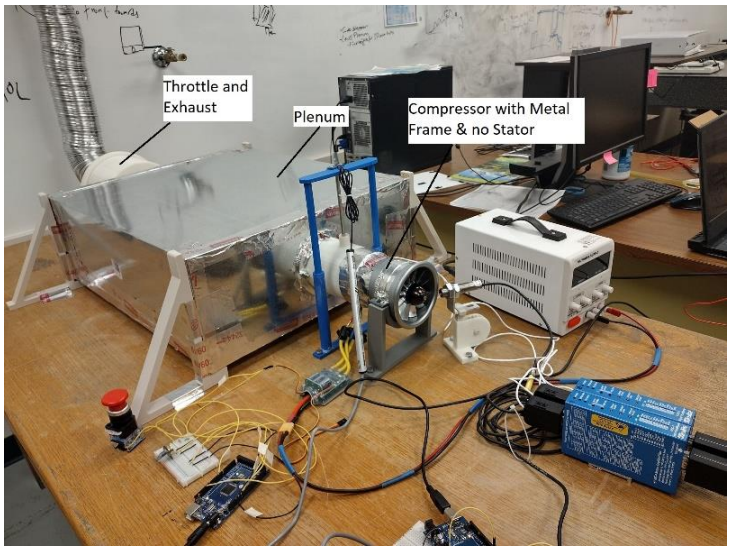


Fig 6.3.2.1: Physical Setup of the M-G Compressor Prototype

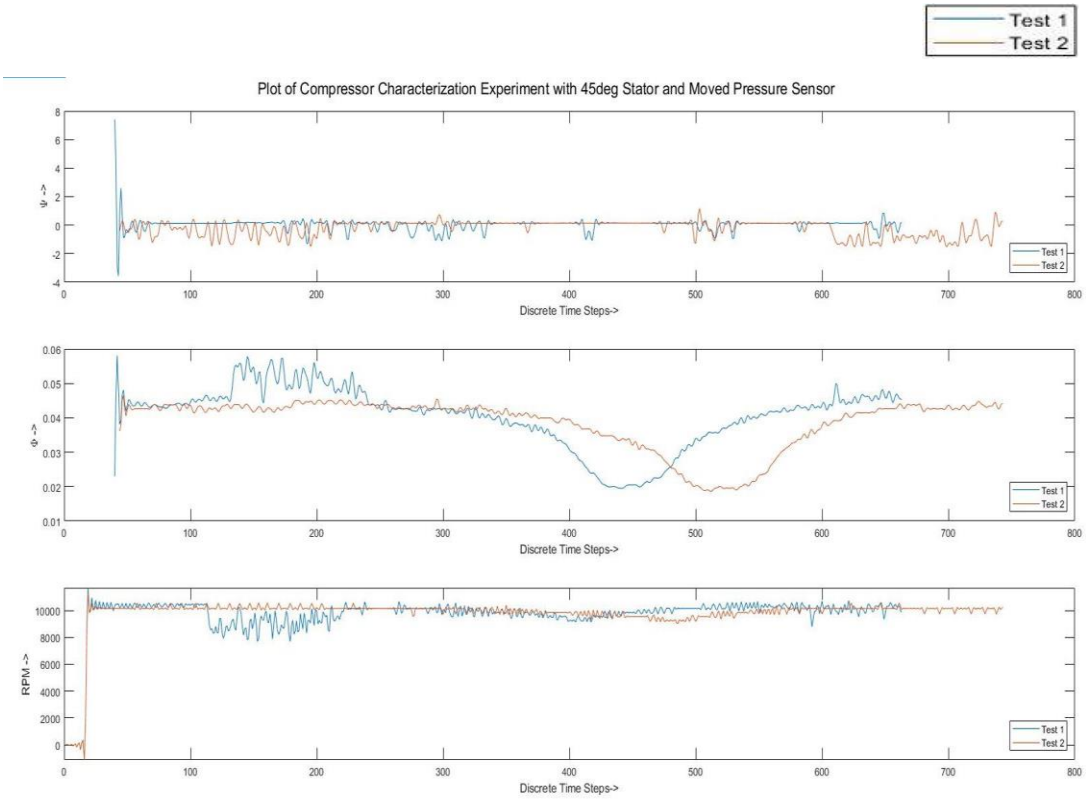


Fig 6.3.2.2: Compressor Characteristic with 45 deg Stator and Moved Pressure Sensor

In a stall situation, both the Pressure Rise Coefficient and the Flow Coefficient are supposed to sharply drop down the values. Unfortunately, no significant drop in the Pressure Rise Coefficient is seen in Figure 6.3.2.2. This could be because of the aliasing of the sensor data during the sampling. Measurement is aliased if the sampling is not in accordance to the Nyquist frequency criterion. The official recommendation for sampling of spectrally rich pressure signal is 2500 samples per second but because RPM sensor only worked in sampling rate of 10 samples per second, the pressure signal might have aliased. Also, a significant drop in the Flow Coefficient is observed. Similarly, a small dip in the RPM signal was also observed at the time when the Flow Coefficient starts to drop down.

In Figure 6.3.2.3, a drop in the Pressure Rise coefficient was seen in Test 2 which immediately follows the drop in the Flow Coefficient. Hence, this could be indicative of a compressor stall. The rpm dip in front of the drop in the Flow Coefficient is also seen in this experiment.

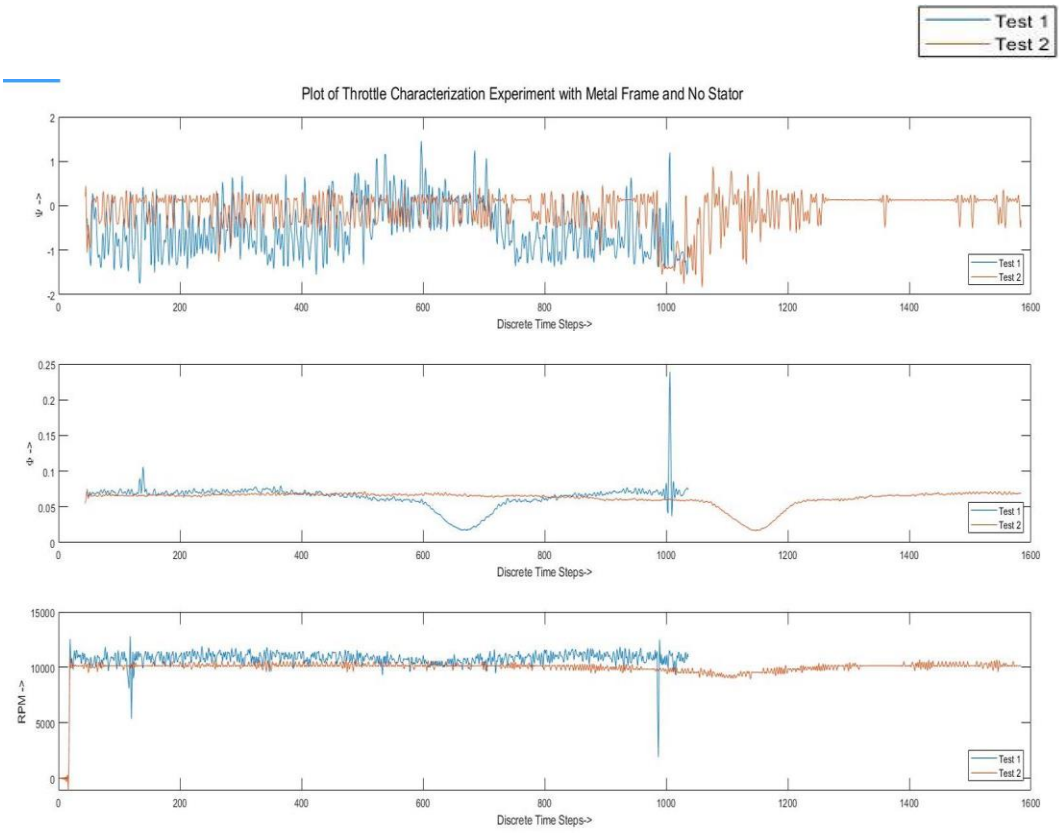


Fig 6.3.2.3: Compressor Characteristic with Metal Frame and no Stator

Chapter - 7: Conclusion and Research Directions

In this thesis draft, the results of the work put into the development of the testbench prototype and stall precursor detection algorithm in the last two years are detailed. In the prototype development work, various stator designs are introduced, new instrumentation systems are developed and a new throttle is designed. Similarly, on the stall precursor detection aspect, the python class files for Recursive least squares and Generalized Extreme Studentized Deviate Test are developed.

The subsections below provide recommendations for improving the current system and completing the algorithm design:

7.1 System Design

The following recommendations are made to improve the current design of the single-stage axial compressor system:

7.1.1 Compressor System:

- It is recommended that a new compressor be designed with the following considerations:
    - The characteristics of the motor used to run the rotor are mostly unknown. The motor has a rated KV of 1000. The KV and Torque are inversely proportional [37]. Provided that the rated rpm of the compressor is 48,000, the motor is designed for high RPM applications. Given that a compression system needs both torque and speed to create pressure inside the duct, either a compressor with a known characteristic be purchased or the characteristic of the current compressor be evaluated in-house using the rpm and Torquemeter sensors.
    - The design of the off-the-shelf compressor has the motor support arms exactly where the stators need to be placed. As a result, for a 12-blade rotor stage, only an 8-blade stator stage can be housed. Hence, if the motor

support stage is designed either in front of the rotor stage (perhaps as an inlet guide vanes) or after leaving some space for the stator stage, the compressor stage can be improved.

- To measure the efficiency of the compressor, the pressure, temperature, and airspeed at the inlet also need to be measured. Hence, the new system also needs to accommodate mounts and adjustments to place an airspeed and temperature sensor probe and a bolt to mount a pressure sensor.
- Vo's thesis dissertation manual discusses how blade geometry affects what kind of stall (spike or modal), the compressor is going to run into. This can be kept in mind while designing rotor blades for the compressor.

7.1.2 Throttle

- The conical section that regulates the opening of the throttle is currently designed to operate the throttle least aggressively. The speed of the actuator stepper motor and the slope of the cone can be modified to change the aggressiveness of the throttle.
- When the throttle is completely closed, the system is not designed to prevent the throttle from letting any air inside the throttle. If a complete seal needs to be created, a rubber attachment can be placed around the base of the throttling cone.
- The current design of the throttle doesn't allow the throttle to be taken apart and re-attached. An upgrade on the current design could have threads on the connection of front and back components and some modifications on the design that would allow the throttle to be taken apart for modifications and re-assembled together.

7.2 Electronic Instrumentation

Highly sensitive measurement instruments were purchased and designed for the system. Table 7.2.1 lists all the measurement instruments and states if they have been tested to provide the right values.

Table 7.2.1: List of the Sensors with the Validation Information

Measurement Instruments	**Validated (Yes/No)?**	**Validated Using (if Yes)**
Pressure Sensor A	**Yes**	Manually using the instructions from Kulite Technical Support
Pressure Sensor B	**Yes**	Manually using the instructions from Kulite Technical Support
Optical Tachometer	**Yes**	Tested against the handheld Tachometer Sensor
Op Amp Module	**Yes**	Compared experimental gain with the theoretical gain (Appendix: 7)
Airspeed Sensor	**No**	NA
Torquemeter	**No**	NA

Hence, the following recommendations, are made to improve the instrumentation and DAQ systems for the single-stage axial compressors:

- Experiments need to be designed to test the airspeed sensor and Torquemeter sensors.
- It is seen that the Arduino is not the best device to use as a Data Acquisition system as it adds noise to the measurements. It is recommended that the Arduino be replaced with a better DAQ.
- Panel Tachometer can be used to sample RPM and DAQ can be sampled higher rate to avoid aliasing of Pressure sensor.

Finally, a recommendation is made (Table 7.2.2) for the purchase of the instruments to accommodate the current status of the measurement system and future measurement and control system designs:

Table 7.2.2 Purchase Recommendation

Items	Remarks
Thermistor	To Validate and Calibrate Temperature Measurement Sensors
Barometer	To Tune Pressure Sensors
AirSpeed Sensor (DegreeC F450)	To Measure Efficiency
Handheld Anemometers	To Validate AirSpeed Sensors
Panel Tachometer (Monarch Panel Tachometer for ROS)	To isolate sampling of RPM sensor

7.3 Algorithm Design

For the completion of the algorithm, the main class file needs to be created that imports the RLS and GESDT classes. This main class needs to evaluate the 15^{th} eigenvalue for each of the models generated by the RLS algorithm on the times series pressure data. The 15^{th} eigenvalues should then be passed through either a growing window or a moving window algorithm to evaluate outliers in the 15th eigenvalue using the GESDT algorithm.

The first stage objectives for the system are focused on developing the system that exhibits stall and developing an instrumentation system that can capture the dynamics. The next phase of the research involves the implementation of real-time stall precursor detection, and control schemes to prevent and/or counter stall and/or surge to improve the efficiency of the compressor.

Interesting research could also be in the implementation of an active control strategy that uses air injection at the tip of the rotor blades that prevents tip leakage flow. This can prevent the compressor from going into the stall within the blade section hence completely postponing the propagation of the stall.

Bibliography:

[1] "Definition of Rotor." https://www.oxfordlearnersdictionaries.com/us/definition/english/rotor?q=rotor.

[2] "Definition of Compressor." https://www.oxfordlearnersdictionaries.com/us/definition/english/compressor?q=compressor.

[3] "Definition of Controller." https://www.oxfordlearnersdictionaries.com/us/definition/english/controller?q=controller.

[4] "Definition of Stator." https://en.wikipedia.org/wiki/Stator.

[5] "Definition of Inception." https://www.merriam-webster.com/dictionary/inception.

[6] "Stationarity." https://www.itl.nist.gov/div898/handbook/pmc/section4/pmc442.htm.

[7] "Application of Turbomachines." https://en.wikipedia.org/wiki/Turbomachinery.

[8] "Types of Compressors." https://en.wikipedia.org/wiki/Compressor#Air_bubble_compressor.

[9] "Mechanism of Axial Compressors." https://en.wikipedia.org/wiki/Axial_compressor.

[10] "History of Axial Compressors." https://www.turbomachinerymag.com/view/parsons-designs-first-axial-flow-compressor-2.

[11] E. M. Greitzer and F. K. Moore, "A theory of post-stall transients in axial compression systems: Part I—development of equations," *J. Eng. Gas Turbines Power*, vol. 108, no. 2, pp. 231–239, 1986, doi: 10.1115/1.3239893.

[12] N. McDougall, N. Cumpsty, and T. Hynes, "Stall inception in axial compressors," 1990, Accessed: Feb. 25, 2022. [Online]. Available: https://asmedigitalcollection.asme.org/turbomachinery/article-abstract/112/1/116/433730.

[13] I. Day, "Stall inception in axial flow compressors," 1993, Accessed: Feb. 25, 2022. [Online]. Available: https://asmedigitalcollection.asme.org/turbomachinery/article-

abstract/115/1/1/418420.

[14] C. A. Mansoux, D. L. Gysling, J. D. Setiawan, and J. D. Paduano, “Distributed nonlinear modeling and stability analysis of axial compressor stall and surge,” *Proc. Am. Control Conf.*, vol. 2, no. 2, pp. 2305–2316, 1994, doi: 10.1109/acc.1994.752492.

[15] Y. Hwang and S. H. Kang, “Flow and performance calculations of axial compressor near stall margin,” *AIP Conf. Proc.*, vol. 1225, pp. 605–615, 2010, doi: 10.1063/1.3464908.

[16] I. J. Day, “Stall, surge, and 75 years of research,” *J. Turbomach.*, vol. 138, no. 1, pp. 1–16, 2016, doi: 10.1115/1.4031473.

[17] D. M. Sterbentz *et al.*, “System identification modeling and unstable behavior of the dynamics of flows within the tip region of an axial compressor blade passage,” *J. Therm. Sci.*, vol. 25, no. 2, pp. 109–116, 2016, doi: 10.1007/s11630-016-0840-4.

[18] C. Bitikofer, M. P. Schoen, J. C. Li, and F. Lin, “Characteristic Moore-Greitzer model parameter identification for a one stage axial compressor system,” *Proc. Am. Control Conf.*, pp. 164–169, 2017, doi: 10.23919/ACC.2017.7962948.

[19] E. Aung, M. P. Schoen, and J. Li, “Dynamic Characterization and Identification of Flow in a Blade,” pp. 1–10, 2019.

[20] G. Gu, A. Sparks, and S. S. Banda, “An overview of rotating stall and surge control for axial flow compressors,” *IEEE Trans. Control Syst. Technol.*, vol. 7, no. 6, pp. 639–647, 1999, doi: 10.1109/87.799664.

[21] W. M. Haddad and V. Chellaboina, “Nonlinear Dynamical Systems and Control, A Lyapunov-Based Approach,” Princeton University Press, 2009, pp. 581–587.

[22] M. Krstic, J. M. Protz, J. D. Paduano, and P. V. Kokotovic, “Backstepping designs for jet engine stall and surge control,” *Proc. IEEE Conf. Decis. Control*, vol. 3, no. December, pp. 3049–3055, 1995, doi: 10.1109/cdc.1995.478612.

[23] M. Krstic and P. V. Kokoltovic, “Lean backstepping design for a jet engine compressor model,” *Proc. Int. Conf. Control Appl. IEEE*, pp. 1047–1052, 1995.

[24] T. R. Camp and I. J. Day, “A study of spike and modal stall phenomena in a low-speed

axial compressor," *J. Turbomach.*, vol. 120, no. 3, pp. 393–401, 1998, doi: 10.1115/1.2841730.

[25] J. Storer, … N. C.-E. P. for, and undefined 1993, "An approximate analysis and prediction method for tip clearance loss in axial compressors," *asmedigitalcollection.asme.org*, 1993, Accessed: Apr. 14, 2022. [Online]. Available: https://asmedigitalcollection.asme.org/GT/proceedings-abstract/GT1993/V001T03A062/243595.

[26] C. S. Tan, I. Day, S. Morris, and A. Wadia, "Spike-type compressor stall inception, detection, and control," *Annu. Rev. Fluid Mech.*, vol. 42, pp. 275–300, 2010, doi: 10.1146/annurev-fluid-121108-145603.

[27] V. H. Garnier, A. H. Epstein, and E. M. Greitzer, "Rotating waves as a stall inception indication in axial compressors," *Proc. ASME Turbo Expo*, vol. 1, no. April 1991, 1990, doi: 10.1115/90-GT-156.

[28] M. Inoue, M. Kuroumaru, T. Iwamoto, and Y. Ando, "Detection of a rotating stall precursor in isolated axial flow compressor rotors," *J. Turbomach.*, vol. 113, no. 2, pp. 281–287, 1991, doi: 10.1115/1.2929102.

[29] M. M. Bright, H. Qammar, H. Vhora, and M. Schaffer, "Rotating Pip Detection and Stall Warning in High-Speed Compressors Using Structure Function," *RTO AVT Symp. Des. Priciples Methods Aircr. Gas Turbine Engines*, no. May, p. 8, 1998.

[30] N. TAHARA, M. KUROSAKI, Y. OHTA, E. OUTA, and H. SHINOHARA, "Early pre-stall investigation by sensitive stall waring measure," *Proc. IGTC03*, pp. 7–12, 2003.

[31] G. S. Heinlein, J. P. Chen, C. M. Chen, S. Dutta, and H. W. Shen, "Statistical anomaly based study of rotating stall in a transonic axial compressor stage," *Proc. ASME Turbo Expo*, vol. 2D-2017, pp. 1–13, 2017, doi: 10.1115/GT2017-64685.

[32] B. Rosner, "Percentage Points for a Generalized ESD Many-Outlier Procedure," *Technometrics*, vol. 25, no. 2, pp. 165–172, 1983.

[33] J. Li, J. Du, Z. Li, and F. Lin, "Stability enhancement with self- recirculating injection in axial flow compressor," *J. Turbomach.*, vol. 140, no. 7, pp. 1–13, 2018, doi:

10.1115/1.4039806.

[34] “Specific Gas Constant for Air.” https://web.mit.edu/calculix_v2.7/CalculiX/ccx_2.7/doc/ccx/node240.html.

[35] E. M. Greitzer, “The Stability of Pumping Systems-The 1980 Freeman Scholar Lecture,” *J. Fluids Eng.*, vol. 103, no. June, 1981.

[36] D. S. Dumble, “Advanced Kalman Filtering and Sensor Fusion.” https://www.udemy.com/course/advanced-kalman-filtering-and-sensor-fusion/learn/lecture/27760250?start=0#overview.

[37] “Relation Between KV and Torque in an Electric Motor.” https://physics.stackexchange.com/questions/215621/does-kv-in-rc-motors-rpm-volt-tell-us-anything-about-the-torque.

Appendix:

Appendix 1: Dimensional Analysis of RLS Equation:

This section performs a Matrix dimension analysis of the two representative equations used in RLS estimation.

$$\hat{\vec{\theta}}_{k+1} = \hat{\vec{\theta}}_k + \frac{\vec{P}_{k+1}\vec{\phi}^{T}_{k+1}}{1+\vec{\phi}_{k+1}\vec{P}_{k+1}\vec{\phi}^{T}_{k+1}}(\vec{y}_{k+1} - \vec{\phi}_{k+1}\hat{\vec{\theta}}_k)$$

Here, consider the terms in the parenthesis,

$\vec{\phi} \in \mathbb{R}^{1\times p}$ i.e. 1 row and p columns

$\vec{\theta} \in \mathbb{R}^{p\times 1}$

$\vec{\phi} * \vec{\theta} \in \mathbb{R}^{1\times 1}$

Here, the expression in the parenthesis represents an error in fitting the previous estimate to the new data. Similarly,

$\vec{P} \in \mathbb{R}^{p\times p}$, $\vec{\phi} * \vec{P} * \vec{\phi} \in \mathbb{R}^{1\times 1}$, $\vec{P} * \vec{\phi}^{T} \in \mathbb{R}^{p\times 1}$

Here, it is seen that the numerator of the fractional term produces a matrix similar in dimension to the parameter matrix and the denominator produces a scalar matrix.

Also, the following equation is used to update the error covariance matrix:

$$\hat{\vec{P}}_{k+1} = \hat{\vec{P}}_k - \hat{\vec{P}}_k\vec{\phi}^{T}_{k+1}\left(\frac{\vec{\phi}_{k+1}\hat{\vec{P}}_k}{1+\vec{\phi}_{k+1}\hat{\vec{P}}_k\vec{\phi}^{T}_{k+1}}\right)$$

In this case, inside the parenthesis, the denominator is a scalar matrix. For the numerator,

$\vec{\phi} * \vec{P} \in \mathbb{R}^{1\times p}$

The parenthesis term evaluates a matrix with the same dimension as the matrix ‘P’. Now,

$\vec{P} * \vec{\phi}^{T} \in \mathbb{R}^{p\times 1}$

$\vec{\phi} * \vec{P} * \vec{\phi}^{T} \in \mathbb{R}^{1\times 1}$

$P * \vec{\phi}^{T} * (\vec{\phi} * \vec{P}) \in \mathbb{R}^{p\times p}$

Hence, the two matrix equations of the RLS algorithm are dimensionally consistent.

Appendix 2: Python Implementation of Recursive Least Squares (RLS) Algorithm

```
8
9   #Shishir Khanal
10  #Class to implement Autoregressive Model Identification using Recursive least Squares
11  #This class doesn't work if y[0...n] =0 where n = number of desired eigenvalues
12      #This is because the np.linalg.inv() in the batch least squates evaluation results in singluarity
13  import numpy as np
14  import random as random
15  from matplotlib import pyplot
16
17  class Autoregressivemodel:
18      def __init__(self, data, numberofeigenvalues, rowlength):
19          self.data = data
20          self.rowlength = rowlength
21          #the notes start indexing from 1 we start from 0, test for consistency
22          self.numberofeigenvalues = numberofeigenvalues
23          #self.modelparameters = np.matrix()
24          self.previousP = np.matrix(np.zeros((self.numberofeigenvalues, self.numberofeigenvalues)))
25          self.P = np.matrix(np.zeros((self.numberofeigenvalues, self.numberofeigenvalues)))
26          self.previoustheta = np.matrix(np.zeros((self.numberofeigenvalues, 1)))
27          self.theta = np.matrix(np.zeros((self.numberofeigenvalues, 1)))
28          self.phi = np.matrix(np.zeros((1, self.numberofeigenvalues)))
29
30  #-----------Functions to evaluate BLS---------------------------------------
31      def slicearray(self,array, start,end):
32          capitalphi_i = array[start:end]
33          return capitalphi_i[::-1]
34
35      def evaluateY(self, capitalphi, finalindex):
36          Y = capitalphi[:,0]
37          Y = Y[1:]
38          Y = np.append(Y,self.data[finalindex])
39          return Y[:,None]
40
41      def populatematrix(self):
42          capitalphi = np.zeros((self.rowlength-1, self.numberofeigenvalues))
43          # -1 to correct matrix indexing
44          #capitalphi matrix only needs 'matrixsize - 1' elements
45              #This is taken care by range function 'stop' index is not included
46          for i in range(self.numberofeigenvalues, self.rowlength):
47              index = (i - self.numberofeigenvalues)
48              capitalphi[index,:] = self.slicearray(self.data,index,i)
49          return capitalphi
50
51      def batchleastsquares(self, initialdata):
52          capitalphi = self.populatematrix()
53          product = np.matmul(capitalphi.transpose(), capitalphi)
54          self.previousP = np.linalg.inv(product)
55          Y = self.evaluateY(capitalphi,self.rowlength - 1)
56          self.previoustheta = np.matmul(self.previousP, np.matmul(capitalphi.transpose(), Y))
57          return
58  #-------------end Functions to evaluate BLS---------------------------------
59
```

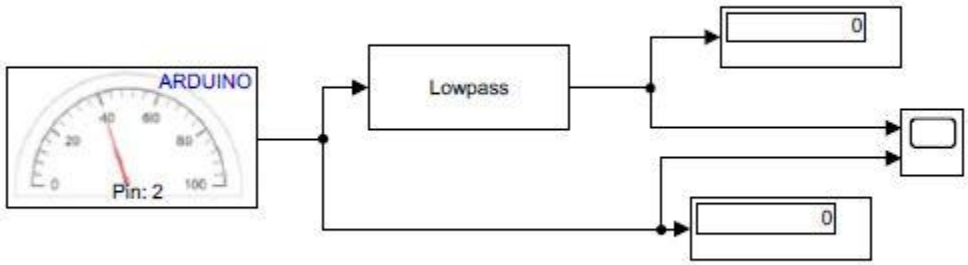
```
60      #--------------Functions to aid RLS-----------------------------------------
61          def buildphi(self, index):
62              return self.slicearray(self.data,(index - self.numberofeigenvalues),index)
63
64          def evaluateP(self):
65              phi = np.asmatrix(self.phi)
66              num = np.matmul(phi,self.previousP)
67              den = 1 + np.matmul(phi, np.matmul(self.previousP,phi.transpose()))
68              #den is a scalar matrix this could cause some problems
69              dividend = np.divide(num, den)
70
71              #[None,:] is to get the appropriate matrix product
72              #Otherwise, matmul evaluates dot product for 1D arrays by default
73              secondterm = np.matmul(self.previousP, np.matmul(phi.transpose(), dividend))
74
75              P = np.subtract(self.previousP,secondterm)
76              return P
77
78          def evaluatetheta(self, currentdata):
79              parenthesisterm = currentdata - np.matmul(self.phi,self.previoustheta)
80              num = np.matmul(self.P, self.phi[:,None])
81              den = 1 + np.matmul(self.phi, np.matmul(self.P, self.phi[:,None]))
82              dividend = np.divide(num, den)
83              secondterm = dividend*parenthesisterm
84              theta = np.add(self.previoustheta, secondterm)
85              return theta
86      #------------end Functions to aid RLS-----------------------------------------
87
88      #------------RLS implementation-----------------------------------------------
89          def evaluatemodel(self):
90              modelparameters = np.matrix(np.zeros((self.numberofeigenvalues, (len(self.data)-self.rowlength+1))))
91              self.batchleastsquares(self.data[:self.rowlength])
92              modelparameters[:,0] = self.previoustheta
93              for i in range(self.rowlength, len(self.data)):
94                  self.phi = self.buildphi(i)
95                  self.P = self.evaluateP()
96                  self.theta = self.evaluatetheta(self.data[i])
97                  modelparameters[:,i - self.rowlength +1] = self.theta
98                  self.previousP = self.P
99                  self.previoustheta = self.theta
100             return modelparameters
101     #------------end RLS implementation-------------------------------------------
102
103     def main():
104         y = np.array(np.zeros(10000))
105         yp = np.array(np.zeros(10000))
106         #e = random.uniform(0,0.025)
107         y[0] = 10
108         y[1] = 15
109         rowlength = 100#rowlength should be >># of eigenvalues
110         for i in range(2, len(y)):
111             #s^2 + 2*s*(1/2) + (1/2)^2
112             y[i] = -0.15*y[i-1]-0.92*y[i-2]+random.uniform(0,0.025)
113         pyplot.plot(y[0:50])
```

```
114         pyplot.plot(y[0:50], label = 'y_Original')
115         classar = Autoregressivemodel(y,5,rowlength)
116         modelparams = classar.evaluatemodel()
117         yp[0] = 10
118         yp[1] = 15
119         param = modelparams[:,9900]
120         for i in range(1, len(yp)):
121             #s^2 + 2*s*(1/2) + (1/2)^2
122             #yp[i] = param[0]*yp[i-1]+param[1]*yp[i-2]+random.uniform(0,0.025)
123             yp[i] = param[0]*yp[i-1]+param[1]*yp[i-2]+param[2]*yp[i-3]+param[3]*yp[i-4]+param[4]*yp[i-5]+random.uniform(0,0.025)
124         pyplot.plot(yp[0:50], label = 'y_Estimated')
125
126         pyplot.title("Comparison of Original Data with Data from Estimated AR Model")
127         pyplot.xlabel("Discrete Time Events->")
128         pyplot.ylabel("y->")
129         pyplot.legend()
130         # pyplot.plot(modelparams)
131         # print(modelparams[:,994])
132
133     if __name__ == "__main__":
134         main();
```

Appendix 3: Python Implementation of Generalized Extreme Studentized Deviate Test (GESDT)

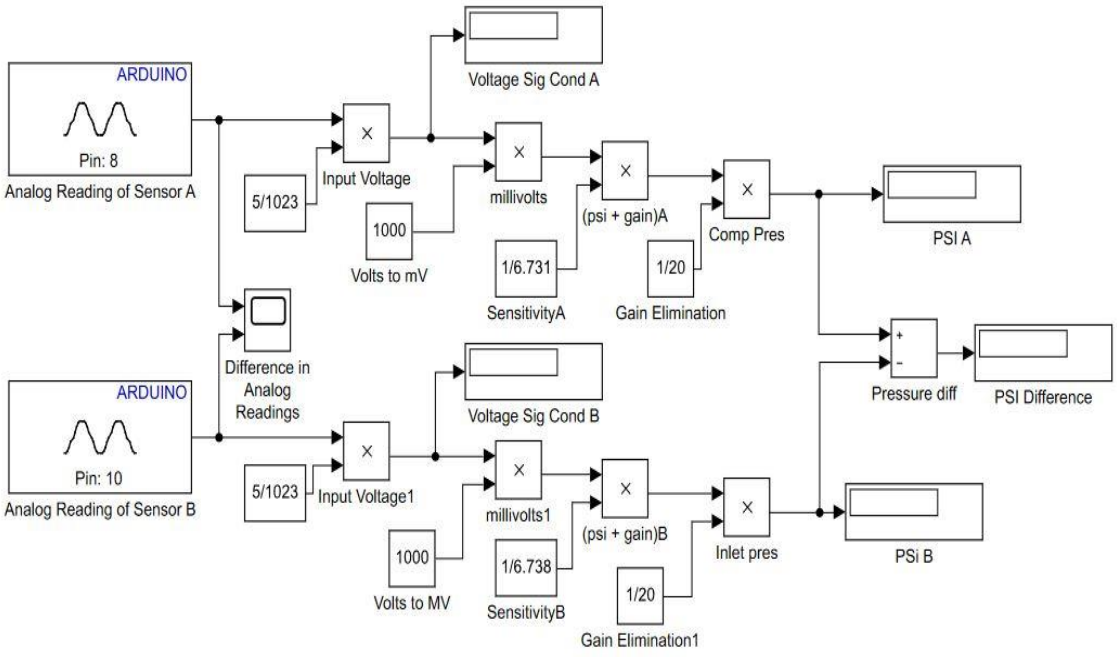
```python
#Shishir Khanal
#Python class to implement Generalized Extreme Studentized Deviate Test(GESDT)
import numpy as np
import scipy.stats as stats

class GESDT:
    def __init__(self, data, outlierupperbound, significancelevel):
        self.alpha = significancelevel
        self.data = data
        self.r = outlierupperbound
        self.R = np.zeros(self.r)
        self.lamda = np.zeros(self.r)

#------------Function to evaluate test statistics-----------------------------

    def computevariance(self):
       mean = np.mean(self.data)
       variance = self.data - mean

       maxvariance = np.amax(variance)
       maxvarianceindex = np.where(variance == maxvariance)
       return (maxvariance, maxvarianceindex[0][0])

    def evaluateR(self, index):
            maxvariance, maxvarianceindex = self.computevariance()
            stddeviation = np.std(self.data)
            self.R[index] = maxvariance/stddeviation
            return (maxvariance, maxvarianceindex)

#------------end Function to evaluate test statistics-------------------------

#------------Function to evaluate critical values-----------------------------

    def evaluatelamda(self, index):
        #i = 1 becuase while i is iterating by 1 size is decreasing by 1
        n = len(self.data)
        tdistribution = stats.t.ppf(1 - self.alpha/(2*n), n - 2 )
        num = (n - 1)*np.sqrt(np.square(tdistribution))
        den = np.sqrt(n) * np.sqrt(n - 2 + np.square(tdistribution))
        self.lamda[index] = num/den
        return
#------------end Function to evaluate critical values-------------------------

#------------Function to evaluate GESDT---------------------------------------

    def findoutliernumber(self):
        for i in range(0, self.r):
            (maxvariance, maxvarianceindex) = self.evaluateR(i)
            self.evaluatelamda(i)
            self.data = np.delete(self.data,maxvarianceindex)

        outlier = 0

        for i in range(0, self.r):
            if self.R[i] > self.lamda[i]:
                outlier += 1

        return outlier
#------------end Function to evaluate GESDT-----------------------------------

def main():
    y = np.random.random(100)
    # x= np.arrange(len(y))
    y[14] = 9
    y[83] = 10
    y[44] = 14
    y[99] = 20
    y[5] = 1.6
    #data = []
    out = GESDT(y,5,0.05)
    points = out.findoutliernumber()
    print(points)
    #print(points)

if __name__ == "__main__":
    main();
```

Appendix 4: Tachometer Block in Simulink to compare the RPM output with and without the smoother

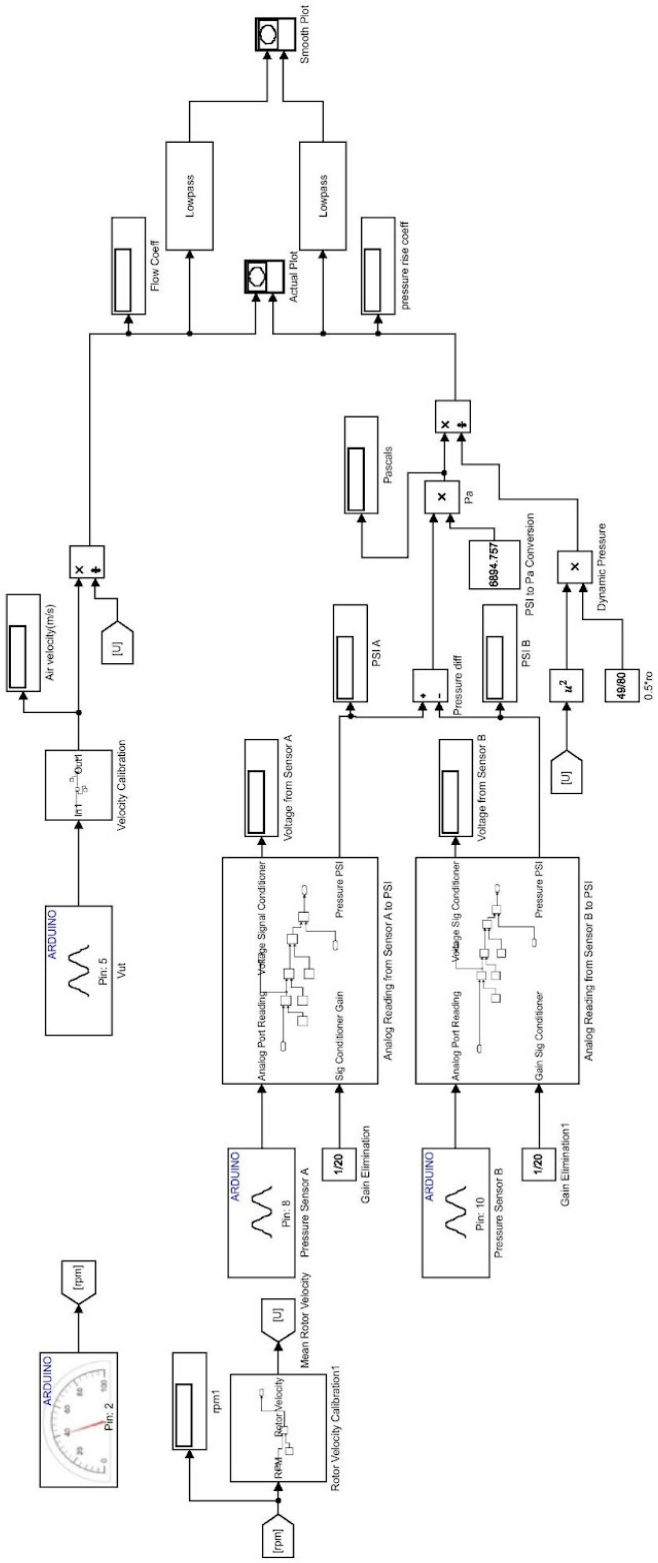


Blocks to Measure RPM of the Signal with and without LPF Smoother

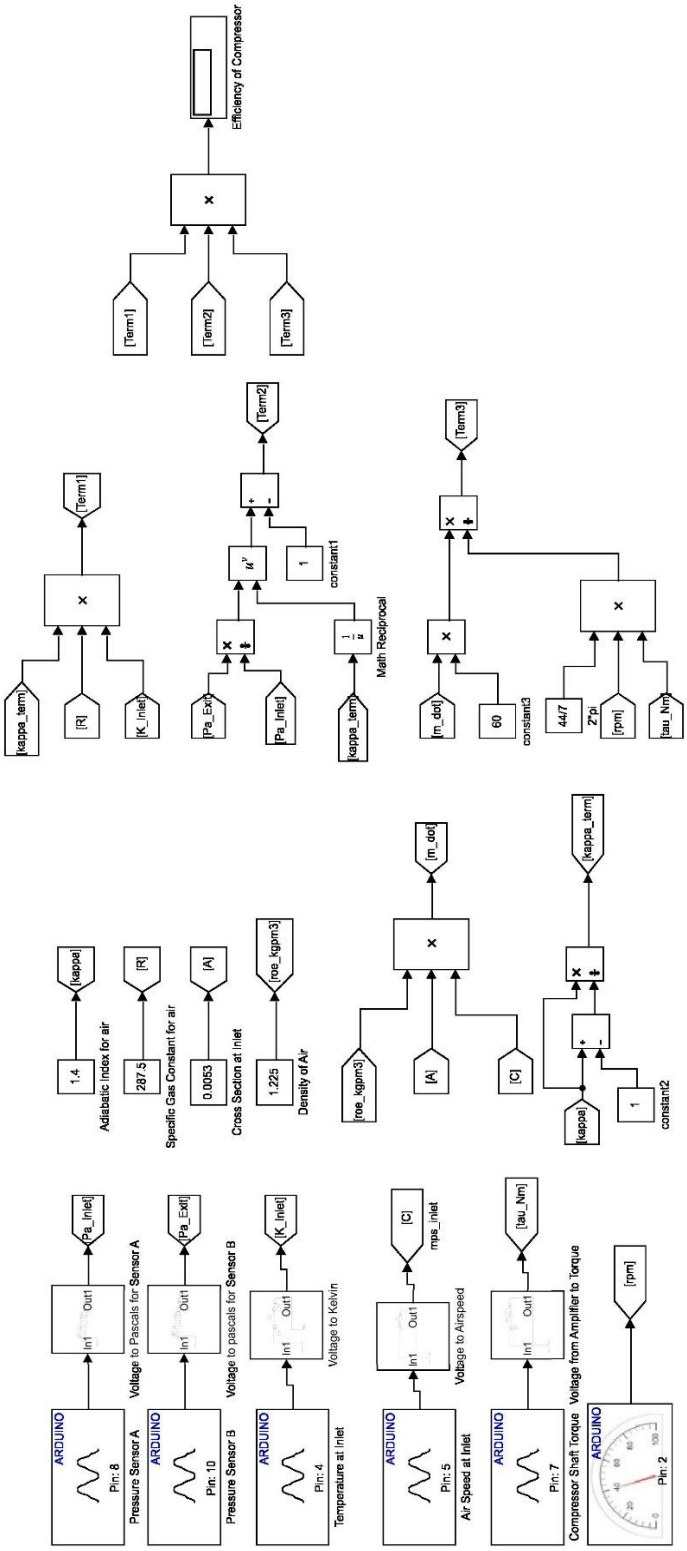
Appendix 5: Simulink implementation of Differential Pressure Measurement



Appendix 6: Simulink implementation of Compressor Characteristic Plot



Appendix 7: Simulink implementation of the Compressor Efficiency Measurement



Appendix 8: Evaluation of Theoretical and Experimental Value of Gain of Op-Amp TLV2471AIP

Circuit Design: Passive Non-Inverting Amplifier

$R_f = 459\ \Omega$

$R_i = 9.7\ \Omega$

Theoretical Gain = $(1 + R_f/R_i) \cong 48.32$

For Experimental Gain, the circuit is supplied with a known input voltage and the corresponding output voltage was measured and the gain was evaluated.

Table A-8.1: Evaluation of the Theoretical Gain of the Op-Amp Module

V_{in} (in Volts)	**V_{out} (in Volts)**	**Gain = V_{out}/V_{in}**
0.00	0.05	inf
0.01	0.464	46.4
0.02	0.954	47.7
0.03	1.445	48.167
0.04	1.935	48.375
0.05	2.422	48.44
0.06	2.912	48.537
0.07	3.401	48.585
0.08	3.83	47.875
0.09	4.32	48
0.1	4.81	48.1

Eliminating the outlier for the first measurement (for gain of inf) and averaging the rest of the gain values. **Experimental Gain:** $\frac{480.179}{10}$ = **48.02**

Appendix 9: Arduino Code to Operate the Compressor RPM using a Potentiometer

CompressorCode

```
//Shishir Khanal
//Original Code by Doug Jackson
//Script to change the rpm of the SS compresssor motor using a potentiometer(plus emergency stop)
//-----------------------------------------------

#include <Servo.h>

Servo cServo;             //create servo object for compressor
const int cPotPin = 0;    //attach compressor potentiometer to pin A0
int cServoVal;            //create variable to store compressor potentiometer value

int StopButtonPin = 2;    //attach emrgency stop button signal lead to pin 2

void setup() {
  cServo.attach(10);      //set yellow wire for compressor controller to pin 10

  pinMode(StopButtonPin, INPUT);  //declare switch as input
  attachInterrupt(digitalPinToInterrupt(StopButtonPin), EmergencyStop, LOW); //interrupts when emergency button pressed
}

void loop() {

  cServoVal = analogRead(cPotPin);                //read compressor potentiometer
  cServoVal = map(cServoVal, 0, 1023, 0, 179);    //scale potentiometer value (179 from previous group's info)
  cServo.write(cServoVal);                        //set compressor
}

void EmergencyStop() {
  cServo.write(0);        //set compressor speed to zero (while button is pressed only)
}
```