

## Use Authorization

In presenting this thesis in partial fulfilment 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 \_\_\_\_\_

Parameter Identification and Controller Design for a  
one stage axial compressor system.

by

Md Fahdul Wahab Chowdhury

A thesis

submitted in partial fulfilment

of the requirements for the degree of

Master of Science in the Department of Mechanical Engineering

Idaho State University

Summer 2018

Copyright 2018 Md Fahdul Wahab Chowdhury

To the Graduate Faculty:

The members of the committee appointed to examine the thesis of Md Fahdul Wahab Chowdhury find it satisfactory and recommend that it be accepted.

---

Dr. Marco P. Schoen,  
Major Advisor

---

Dr. Kenneth W. Bosworth,  
Co-Advisor

---

Dr. Gene Stuffle,  
Graduate Faculty Representative

## **ACKNOWLEDGEMENT**

I am grateful to almighty God for the blessings that he has bestowed upon us.

I would like to convey my sincerest gratitude to my major advisor Dr. Marco P. Schoen for advice and guidelines throughout every steps of my research. I would also like to extend my sincerest thanks to Dr. Kenneth W. Bosworth for his continuous guidance and help on the computation part of this research. Their knowledge and expertise helped me to complete this thesis.

My sincerest thanks to the personal of Chinese Academy of Science who are associated with this research. From time to time, they provided valuable feedback on the progress of this research.

I would also like to thank the Chair of Mechanical Engineering Department and ISU College of Science and Engineering for providing me GTA and Career path internship that provided me the funding to complete my degree. From the very beginning of my graduate studies, Dr. Alba Perez Gracia has been supporting and guiding me all the way possible. I would like convey my sincerest gratitude to her.

Thanks to my wife who has always been beside me during this period. My sincerest thanks to my mother, siblings and friends. I also want to thank other graduate students who studied with me.

## TABLE OF CONTENT

<b>LIST OF FIGURES.....</b>	<b>viii</b>
<b>LIST OF TABLES.....</b>	<b>Ix</b>
<b>ABSTRACT.....</b>	<b>x</b>
<b>CHAPTER 1: INTRODUCTION.....</b>	<b>1</b>
1.1 Background.....	1
1.2 Problem Statement.....	7
1.3 Objective of Thesis.....	8
1.4 Thesis Outline.....	9
<b>CHAPTER 2: MOORE-GREITZER MODEL.....</b>	<b>11</b>
2.1 Development of model.....	11
2.2 Compressor Dynamics.....	13
2.3 The IGV and the Inlet.....	15
2.4 The Exit Duct.....	17
2.5 Plenum Dynamics.....	17
2.6 The Compressor Characteristics.....	19
2.7 Moore-Greitzer model.....	19
2.8 One-mode Truncated model.....	21
2.9 Optimization Problem.....	23
2.10 Potential of MG model.....	25
2.11 Shortcomings of MG model.....	26
<b>CHAPTER 3: DISCRETE TIME MG MODEL AND GENETIC ALGORITHM.....</b>	<b>27</b>
3.1 Background.....	27
3.2 Discretization of MG model.....	29
3.3 Proof of Concept.....	31
3.4 Genetic Algorithm.....	35
3.4.1 Creation of initial population .....	35
3.4.2 Creation of cost function and evaluation of each chromosome.....	38
3.4.3 Pairing.....	39
3.4.4 Mating.....	39
3.4.5 Mutation.....	40

<b>CHAPTER 4: FUZZY LOGIC CONTROLLER.....</b>	<b>41</b>
4.1 Fuzzy Logic.....	41
4.2 Fuzzy Logic Controller for Moore-Greitzer model.....	43
4.2.1 Define fuzzy linguistic variables and terms.....	44
4.2.2 Development of membership function.....	44
4.2.3 Development of Fuzzy Rules.....	46
4.2.4 Fuzzification.....	47
4.2.5 Implication.....	48
4.2.6 Aggregation.....	48
4.2.7 Defuzzification.....	48
4.3 Simulation block diagram.....	50
<b>CHAPTER 5: SIMULATION RESULTS AND DISCUSSION.....</b>	<b>52</b>
5.1 Moore-Greitzer model and characteristic curve.....	52
5.2 States of MG model.....	57
5.3 Genetic algorithm and simulation.....	59
5.4 Fuzzy logic controller.....	63
<b>CHAPTER 6: CONCLUSION AND FUTURE WORK.....</b>	<b>65</b>
6.1 Conclusion .....	65
6.2 Future work .....	67
<b>REFERENCES.....</b>	<b>68</b>
<b>APPENDIX A – Matlab <sup>TM</sup> files for simulation of Moore-Greitzer model...</b>	<b>70</b>
A-1: Calculation of the operating point for a particular throttle coefficient...	70
A-2: Characteristic equation and throttle equation.....	71
A-3: Simulation of compressor characteristic curve.....	72
A-4: Moore-Greitzer Model.....	73
A-5: Simulation of MG model using ode45.....	74
A-6: Simulation of MG model using the Runge-Kutta Method.....	75
A-7: Comparison between the simulation of Moore-Greitzer Model on discrete time and continuous time.....	77

<b>APPENDIX B – Matlab <sup>TM</sup> files for simulation of Lorenz model.....</b>	<b>78</b>
B-1: Lorenz model.....	78
B-2: Simulation of Lorenz model using ode45.....	79
B-3: Simulation of Lorenz model using the Runge-Kutta Method.....	80
B-4: Comparison between the simulation of Lorenz system using ode45 and Runge-Kutta method.....	81
<b>APPENDIX C – Matlab <sup>TM</sup> files for optimization of parameters of the compressor characteristic using Genetic Algorithm.....</b>	<b>82</b>
C-1: Genetic Algorithm.....	82
C-2: Cost function.....	84
C-3: Simulation of characteristic curve from chromosomes.....	85
C-4: Simulation of Moore-Greitzer model from chromosomes.....	86
C-5: Computing error between true simulation of states and simulation from chromosomes.....	88
C-6: Selection of Mom and Dad using roulette wheel method.....	89
C-7: Mating.....	90
C-8: Mutation.....	91
<b>APPENDIX D – Internship at INL on summer, 2017.....</b>	<b>92</b>



## LIST OF FIGURES

Figure 1.1: Basic compression system.....	2
Figure 1.2: Compressor characteristic and throttle characteristic.....	3
Figure 1.3: Rotating stall inception mechanism by Emmon.....	4
Figure 2.1: Basic compression system (MG model).....	11
Figure 2.2: Compressor blade.....	14
Figure 3.1: The four points where the derivative is evaluated for Runge-Kutta 4 <sup>th</sup> order method.....	29
Figure 3.2: Simulation of Lorenz model using ode45.....	33
Figure 3.3: Simulation of Lorenz model using Runge-Kutta method.....	33
Figure 3.4: Mean square error.....	34
Figure 4.1: Fuzzy Logic Process Flow.....	42
Figure 4.2: Diagram of Fuzzy Logic Controller.....	43
Figure 4.3: Membership function of input variable “Error”.....	45
Figure 4.4: Membership function of Output Variable.....	46
Figure 4.5: Flow of fuzzy reasoning.....	47
Figure 4.6: Fuzzy Inference system.....	49
Figure 4.7: Simulink block diagram for MG Model control.....	50
Figure 5.1: Operating point at throttle coefficient 0.21.....	52
Figure 5.2: Characteristic curve for throttle coefficient $\gamma = 0$ to 2.0 .....	53
Figure 5.3: Characteristic curve for throttle coefficient $\gamma = 0.21$ to.....	56
Figure 5.4: Simulation of states in continuous time.....	57
Figure 5.5: Simulation of states in discrete time.....	58
Figure 5.6: The error between the simulation using ode45 and Runge-Kutta method.....	59
Figure 5.7: Generation of initial population.....	60
Figure 5.8: Cost of chromosomes of initial population.....	61
Figure 5.9: Top chromosome and cost.....	61
Figure 5.10: Simulation of mass flow ( $U_e$ ) .....	63
Figure 5.11: Simulation of pressure rise ( $P_e$ ) .....	64

## LIST OF TABLES

Table 5.1: Simulated data for throttle coefficient $\gamma = 0$ to 2.0 .....	53
--	----

# Parameter Identification and Controller Design for a One Stage Axial Flow Compressor System

Thesis abstract -- Idaho State University (2018)

The research presented in this thesis details identification of the parameters of an axial flow compressor's characteristic. The parameters are dependent on the compressor's geometry and are important factors for designing a controller of such a system. In order to identify compressor specific parameters, a genetic algorithm is used for solving the optimization problem. The optimized parameters are used to design a controller for keeping the operating point of the compressor away from stall inception. A fuzzy logic controller is designed and the mass flow rate is controlled in order to operate the compressor at the desired point.

Keyword: Axial flow compressor, Stall, Genetic Algorithm, Optimization, Control, Fuzzy logic, Compressor parameters.

# **Chapter 1**

## **Introduction**

### **1.1 Background:**

The thesis represents an approach to mitigate a long standing instability problem in axial flow compressors. The instabilities known as stall and surge limits the operating range of axial flow compressors. A controller is designed to keep the operating point away from stall. Also parameters of the compressor characteristics are extracted to design an efficient controller.

Compressors are used in various applications. Some major applications are pressurization of gas and fluid for process industry, fluid transportation in pipelines, turbo jet engines, gas turbines for power generation, etc. Mainly four types of compressors are used. They are axial, reciprocating, rotary, and centrifugal. In this work, the axial flow compressor is studied.

In axial compressors, fluid flows in parallel to the axis of rotation. The working principle of axial compressors is that it accelerates the fluid flow and then converts the kinetic energy into potential energy. The fluid is accelerated by applying thrust using rotating blades and then decelerated by using stationary blades. Thus converting the kinetic energy into potential energy.

The basic compression model used for the derivation of the model is shown in Figure 1.1. According to the figure, air is fed to the compressor through the inlet and inlet guide vanes (IGV's). The air is then compressed in the compressor as it passes through the rotor and stator. The compressor is operated in a duct and the compressed air is discharged into the plenum. The plenum has a larger volume than that of the compressor and duct. The flow in a compressor is assumed as incompressible but the gas in the plenum is considered compressible. A throttle is used to control the flow through the system. The throttle is located at the exit of the plenum.

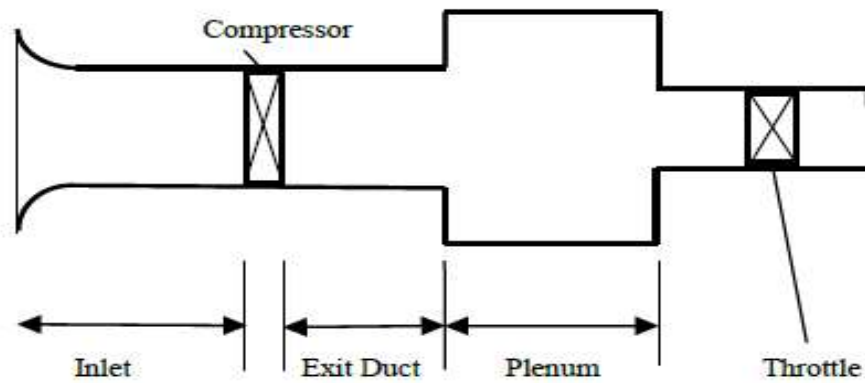


Figure 1.1: Basic Compression System

The performance of a compressor during steady and axisymmetric flow is represented by a cubic curve based on the Moore-Greitzer model, [1]. This cubic curve is known as characteristic curve. The throttle characteristic is represented by a quadratic curve. Figure 1.2 shows the typical characteristic curve and throttle curve. The horizontal axis represents mass flow rate and the vertical axis represents pressure rise. The intersection of both curves represents the operating point on steady flow condition.

This operating point indicates that the flow through the compressor and the throttle are the same and the pressure in the compressor is the same as the pressure drop through the system.

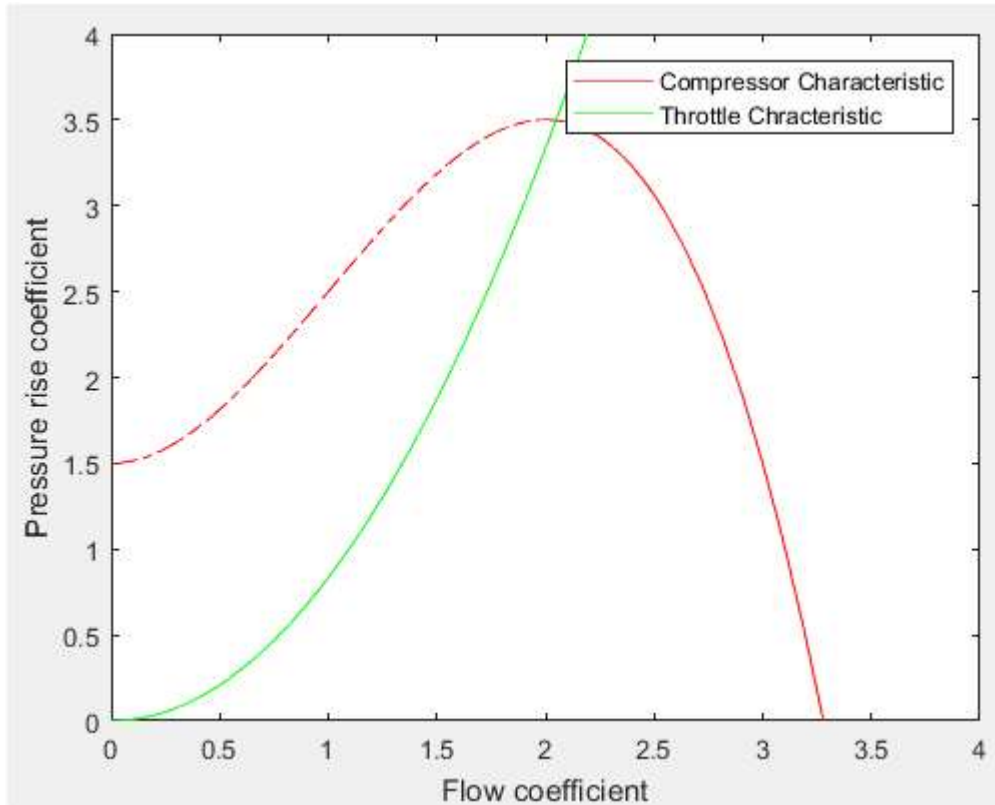


Figure 1.2: Compressor Characteristic and Throttle Characteristic

The operating point can be manipulated by controlling the mass flow rate by using the throttle. If the operating point is moved from the designed operating point to the unstable region, then two types of instability occur in the compression system. These two types of instability are known as Rotating Stall and Surge.

Rotating stall is the disruption of flow through the compressor. It happens due to one or more stall cell. The stall cell means an area of slow flow which rotates around the annulus of the compressor. The mechanism of rotating stall inception is shown in Figure 1.3.

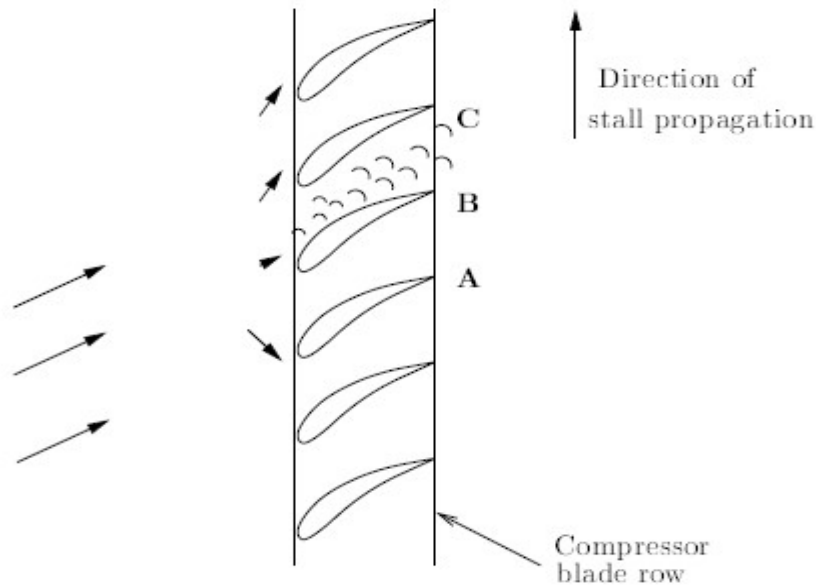


Figure 1.3: Rotating Stall inception Mechanism by Emmon [2]

The mechanism of rotating stall was described by Emmon [2]. A row of rotating blade is considered for describing the inception of rotating stall. Suppose when the fluid flow is nonuniform, a high angle of attack is produced at blade *B*. A blockage of flow is developed due to this high angle of attack between *B* and *C* which diverge the fluid to blade *A* and *C*. Hence a high angle of attack is produced on blade *A*. Thus blade *A* becomes stall after blade *B* and propagates along the direction on the blade row.

It became evident that the stall propagation and surge reduces the operating range and efficiency of an axial compressor. Since 1950, extensive research have been conducted on the flow dynamics and stall inceptions of compressors. Primarily the research was based on collecting experimental data and connecting them to the operating condition of compressors, [3] [4]. In this way, researchers were able to build a better understanding of the mechanism and inception of rotating stall.

The understanding encouraged and lead the researcher to build a mathematical model for understanding the flow mechanism and inception of stall and surge in axial compressors. In the

continuation of this effort, a theoretical model was developed by E. M. Greitzer at Massachusetts Institute of Technology in the mid-1970s. He also validated the model by collecting experimental data and comparing it with the mathematical model. The theoretical model for compression system was published in 1976 and known as the Greitzer model, [5] [6].

While Greitzer was developing the theoretical model at Massachusetts Institute of Technology, F. K. Moore was also conducting extensive research on rotating stall theory. Moore published his work in 1984 in the Journal of Engineering for Gas Turbines and Power [7].

In the summer of 1983, an arrangement was made to combine the work of Moore and Greitzer. A three weeks residence was arranged at NASA-Lewis Research Center. Their combined effort developed a set of coupled ordinary differential equation capable of describing the flow mechanism of rotating stall and surge. A report was submitted to NASA in March of 1985 detailing the development of the model. Their collaborative work is known as Moore-Greitzer Model and was published in 1986 [1]. Their model has paved the way for development of controller for mitigating the stall and surge in axial compressor.

The Moore-Greitzer model was widely accepted as a good characterization of the flow dynamics of compression system. During the 1990's, extensive research was conducted for modification and improvement of the Moore-Greitzer model. J. M. Haynes, G.J.Hendricks and A. H. Epstein modified Moor-Greitzer model by including the effect of blade row time gap, [8]. The modified model was then used to stabilize a three stage low speed axial compressor by damping circumferentially travelling low amplitude waves. The prediction of open and closed loop dynamic response using this model matched with experimental result. Paduano developed an analytical model which describes rotating stall as a traveling wave packet, [9]. Paduano validated this model by conducting an experiment where he was able to reduce the



stalling mass flow by 18%. Hendricks et al. also developed a nonlinear model by representing the flow dynamics by the unsteady Euler's equation, [10].

Based on the Hendricks model, several control strategies are developed to suppress stall and surge. H. J. Weigh and others [11] stabilized the rotating stall and surge by using feedback control on a single-stage axial compressor. They use an array of twelve air injector to stabilize the system. The control system used to measure the static pressure pattern on upstream wall and feedback the data to air injectors. D'Andrea et al. [12] also used pulsed air injector to control the onset of rotating stall. They were able to extend the stall point and thus eliminated the hysteresis loop of rotating stall.

Gravdhal and Egeland [13] developed a model based on the MG model where they have included Greitzer  $B$  parameter as another state. Thus using this new model, it became possible to design controllers for variable speed compressors. Shu Lin, Chunjie Yang, Ping Wu and Zhihuan Song [14] designed a controller to control speed and surge in variable speed axial compressor using the model developed by Gravdahl and Egeland. Shu Lin and his group designed a proportional controller to control speed and a fuzzy logic controller to control surge based on their model. Gravdahl et al. [15] also proposed a closed couple valve controller for controlling surge.

## 1.2 Problem Statement:

Controlling the fluid flow in axial compressors efficiently has been a major problem for many years. Researchers and Scientists have been working to find the complicated fluid dynamics of the compression system. Various models were developed to explain the dynamics of the fluid and the onset of stall and surge. Previously, linear models were developed to explain this phenomenon, [16] but this was not successful to completely explain the behavior of the system. To overcome this limitation of linear models, Moore and Greitzer combined their work to develop the Moore-Greitzer Model. This MG model is built on the assumption of a compressor characteristic but the parameters of the characteristic are dependent on the compressor geometry and other factors. As each compressor exhibits different characteristics, the parameters of the characteristic equation are not the same. Thus the MG model is not able to provide a compressor specific dynamics model rather it describes the general fluid dynamics of a compression system. Customizing a controller for specific compressors was not possible using the general nonlinear fluid dynamics model. In order to solve this problem, an approach is proposed in this work to extract a compressor specific compressor characteristic.

The extracted parameters are then used to design a controller for a one stage axial compressor. A fuzzy logic controller is designed to control the mass flow rate by varying the throttle of the compressor. The input into the controller is the error between the desired operating point and actual operating point. The output is the throttle coefficient. The controller is designed in a way that it adjust the throttle coefficient according to the mass flow rate of the compressor. Normally when the mass flow rate increases, the controller decreases the throttle coefficient and vice versa.

### **1.3 Objective of Thesis:**

The following is a list of objectives postulated for this thesis:

- To optimize the parameters of compressor characteristic for a particular compression system. Thus developing a technique to extract the parameters for each particular compressor in order to overcome the shortcomings of Moore-Greitzer Model which is not able to provide compressor specific characteristic. Thus making it impossible to design a controller on the basis of each compressor's characteristic.
- To utilize the optimized coefficients to design a controller for a single stage compressor model.
- To design a controller for the single stage axial compressor to keep the operating point away from the peak of the characteristic curve. Thus allowing the compressor to operate in safe region by accommodating the flow disturbance.

## **1.4 Thesis Outline:**

As the purpose of this thesis is to optimize the parameters of the characteristic of compressor of Moore-Greitzer model, at first the Moore-Greitzer model is derived and reformulated as an optimization problem. The derivation of the Moore-Greitzer model and reformulation is described in Chapter 2: Moore-Greitzer Model.

According to the reformulated MG model, the three states and compressor characteristic are simulated in discrete time using MATLAB®. The simulated data are then used as the desired output from the system. The optimization problem requires to build a cost function and initial population of chromosome for a genetic algorithm. The initial population of chromosomes is created such that each chromosome consists of four genes. These four genes are the four coefficients of the third order polynomial equation representing the compressor characteristics. After creating the initial population, a cost function is developed to find the fittest chromosomes from the population. The cost function is the error between the simulated output via genetic algorithm and the desired output. The description for optimizing the parameters using the genetic algorithm is given in Chapter 3.

The optimized parameters from the genetic algorithm is then used to design a fuzzy logic controller. The Mamdani Fuzzy Model is used in this thesis. The fuzzy input variable is the error from the designated operating point. The output variable is the throttle coefficient. The fuzzy rules are assigned such that it adjusts the throttle coefficient to keep the mass flow rate constant. Thus keeping the operating point at the designated operating point. The above fuzzy logic controller design is illustrated in Chapter 4.

The simulation of the MG model in discrete time is given in Chapter 5. Also the optimization results from the genetic algorithm and the simulation from the fuzzy logic controller are given in Chapter 5. A discussion on the results and simulation is provided.

The conclusion on the basis of the work is given in Chapter 6 along with the scope of the future work.

## Chapter 2

### Moore-Greitzer Model

#### 2.1 Development of model:

F. K. Moore and E. M. Greitzer developed a model [1] to describe the flow dynamics of axial flow compressor. The basic compression model used for the derivation of the model is shown in Figure 2.1. According to the figure, the compressor is operated in duct and the compressed air is discharged to the plenum. The plenum has a larger volume than that of the compressor and duct. The flow in compressor is assumed as incompressible but the gas in the plenum is considered compressible. A throttle is used to control the flow through the system. The throttle is located at the exit of plenum.

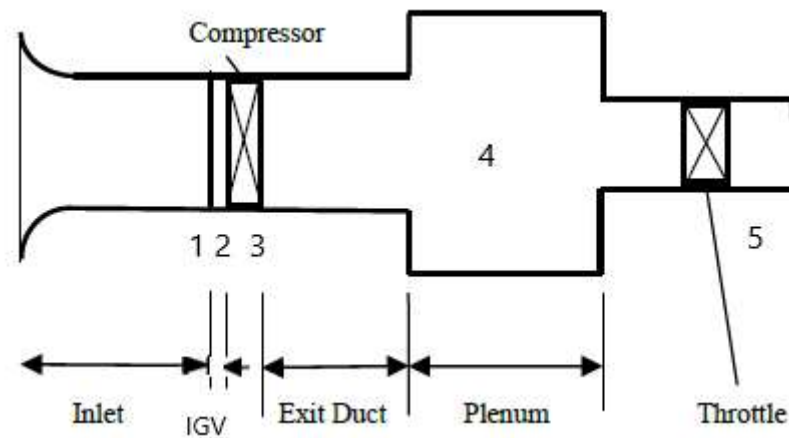


Figure 2.1: Basic Compression System (MG model)

The total pressure rise from inlet to plenum is expressed as,

$$\hat{P} \equiv \frac{P_4 - P_{Ta}}{\rho U_w^2},$$

where,

$P_4$  = Pressure at plenum,

$P_{Ta}$  = Pressure at inlet,

$\rho$  = Density of the fluid,

$U_w$  = Wheel speed at mean radius.

The total pressure rise is the accumulation of pressure rise in each control volume.

So the above equation can be expressed as,

$$\hat{P} = \frac{1}{\rho U_w^2} [(P_4 - P_3) + (P_3 - P_{T2}) + (P_{T2} - P_{T1}) + (P_{T1} - P_{Ta})] \quad (2.1)$$

where,  $(P_4 - P_3)$  is the pressure rise in exit duct.

$(P_3 - P_{T2})$  is the pressure rise in compressor.

$(P_{T2} - P_{T1})$  is the pressure rise in inlet guide vanes (IGV's)

$(P_{T1} - P_{Ta})$  is the pressure rise at inlet.

Each part of the model is developed separately and added together.

## 2.2 Compressor Dynamics:

The pressure on each blade is derived from the acceleration of the fluid flow in the blade.

Thus,

$$\Delta P = -\rho b \frac{dC}{d\tilde{t}}$$

where,  $b$  is the chord length of the blade

$C$  is the velocity of the flow in compressor passage

$\rho$  is the density of the fluid

$\tilde{t}$  is the dimensional time which can be expressed as,

$$\tilde{t} = t \left( \frac{R}{U_w} \right),$$

where,  $t$  is the non-dimensional time,

$R$  is the mean compressor radius

and  $U_w$  is the wheel speed at mean radius.

All of these quantities are described in Figure 2.2



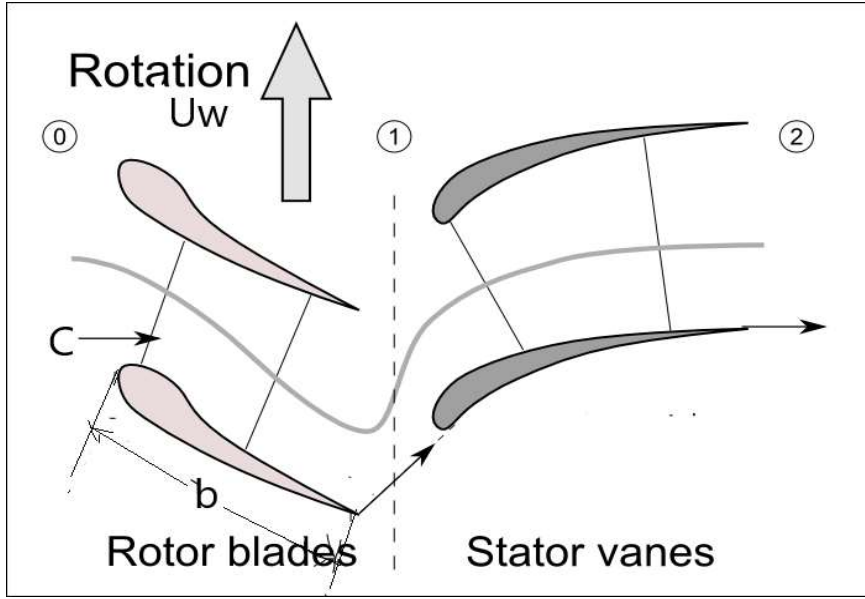


Figure 2.2: Compressor blade, [17]

If the time derivative is written in relative form then,

$$\left. \frac{dC}{d\tilde{t}} \right|_{stator} = \frac{\partial C}{\partial \tilde{t}} ,$$

$$\left. \frac{dC}{d\tilde{t}} \right|_{rotor} = \frac{\partial C}{\partial \tilde{t}} + \frac{U_w}{R} \frac{\partial C}{\partial \theta} ,$$

The pressure rise can be calculated for a single blade row as follows,

$$\frac{\Delta P}{\rho U_w^2} = F_i(\hat{U}) - \lambda_{ri} \left( \frac{\partial \hat{U}}{\partial \tilde{t}} + \frac{U_w}{R} \frac{\partial \hat{U}}{\partial \theta} \right) - \lambda_{si} \frac{\partial \hat{U}}{\partial \tilde{t}} ,$$

where,  $F_i(\hat{U})$  is a function of non-dimensional flow coefficient,  $\hat{U}$ .

$$\hat{U} = \frac{C \cos \beta}{U_w} ,$$

$$\lambda_r = \lambda_s = \frac{b}{U_w \cos \beta} ,$$

Time,  $t$  is recalled as  $t = \frac{\tilde{t}}{(R/U_w)}$ . Thus the above equation becomes

$$\frac{P_3 - P_{T_2}}{\rho U_w^2} = F_c(\hat{U}) - \Lambda \frac{\partial \hat{U}}{\partial t} - \Lambda_r \frac{\partial \hat{U}}{\partial \theta}, \quad (2.2)$$

Here,

$F_c(\hat{U}) = \sum_i F_i(\hat{U})$  is pressure rise across the compressor under clean flow condition.

$$\Lambda = \frac{U_w}{R} \sum_i (\lambda_{ri} + \lambda_{si}),$$

$$\Lambda_r = \frac{U_w}{R} \sum_i \lambda_{ri},$$

### 2.3 The IGV and the Inlet:

The inlet flow is assumed to be irrotational and incompressible. So Bernoulli's equation is applied to obtain the pressure difference across the inlet. Thus the pressure rise across the inlet is expressed in terms of the unsteady velocity potential.

$$\frac{P_{T_1} - P_{T_a}}{\rho U_w^2} = \frac{\partial \phi}{\partial t} \Big|_{x=-l_I} - \frac{\partial \phi}{\partial t} \Big|_{x=0},$$

Where,  $\phi(x, \theta, t)$  is the potential velocity of flow at inlet ( $l_I$ ).  $l_I$  is expressed as  $l_I = \frac{L_I}{R}$

where  $R$  is the mean compressor radius and  $L_I$  is the length of inlet duct.

The velocity potential can be obtained by solving the following Laplace equation:

$$\frac{\partial^2 \phi}{\partial x^2} + \left(\frac{1}{R^2}\right) \frac{\partial^2 \phi}{\partial \theta^2} = 0,$$

The boundary condition at inlet exit ( $x = 0$ ) is  $\left. \frac{\partial \phi}{\partial x} \right|_{x=0} = \hat{U}$  and at inlet entrance

$$(x = -l_I) \text{ is } \left. \frac{\partial \phi}{\partial x} \right|_{x=-l_I} = \hat{U}_0,$$

Where axial velocity at inlet entrance,  $\hat{U}_0$  is the average annular flow of the axial velocity at inlet exit,  $\hat{U}$ . The velocity potential after solving the equation is

$$\phi(x, \theta, t) = \hat{U}_0(t)x + \sum_{n=1}^N A_n(t) e^{in\theta} \frac{[e^{n(x+l_I)} + e^{-n(x+l_I)}]}{ne^{nl_I} - ne^{-nl_I}} + \text{complex conjugate}, \quad (2.3)$$

Therefore,

$$\left. \frac{\partial \phi}{\partial t} \right|_{x=0} = \sum_{n=1}^N K_n A'_n(t) e^{in\theta} + c.c., \quad (2.4)$$

$$\left. \frac{\partial \phi}{\partial t} \right|_{x=-l_I} = -l_I \hat{U}_0'(t) + \sum_{n=1}^N \tilde{K}_n A'_n(t) e^{in\theta} + c.c., \quad (2.5)$$

Where,

$$K_n = \frac{e^{nl_I} + e^{-nl_I}}{ne^{nl_I} - ne^{-nl_I}} \text{ and } \tilde{K}_n = \frac{2}{ne^{nl_I} - ne^{-nl_I}}.$$

It is assumed that no pressure loss occurs across the inlet guide vanes (IGV), therefore

$$P_{T_2} = P_{T_1}$$

## 2.4 The Exit Duct:

The pressure rise across the exit duct is expressed as,

$$\frac{P_4 - P_3}{\rho U_w^2} = -l_E \frac{\partial \hat{U}}{\partial t},$$

where,  $l_E = \frac{L_E}{R}$  ;  $L_E$  is the length of exit duct.

Now putting the pressure rise on each segment together on Equation 2.1, the following expression for total pressure rise is obtained,

$$\hat{P} = F_c(\hat{U}) - \Lambda_r \frac{\partial \hat{U}}{\partial \theta} - (\Lambda + l_E) \frac{\partial \hat{U}}{\partial t} + \left. \frac{\partial \phi}{\partial t} \right|_{x=-l_I} - \left. \frac{\partial \phi}{\partial t} \right|_{x=0}, \quad (2.6)$$

## 2.5 Plenum Dynamics:

The dynamics of the fluid flow from plenum to throttle is derived in this section. It is assumed that the fluid flow is a free jet to the plenum. The accumulation of mass flow in the plenum is calculated as the difference between the mass flow rate at the entry and exit point of the plenum.

Thus,

$$\frac{d}{dt}(m_p) = \dot{m}_c - \dot{m}_T \quad (2.7)$$

where,

$\dot{m}_c = \rho_c \tilde{U}_4 A_c$  is the mass flow entering the plenum.

$\dot{m}_T = \rho_T \tilde{U}_T A_c$  is the mass flow at throttle or exiting the plenum.

$$\dot{m}_p = \tilde{\rho}_p V_{pl}$$

Assuming that the fluid is incompressible inside the plenum,

$$\rho_c = \rho_T = \rho ,$$

Thus Equation 2.7 becomes,

$$\tilde{V}_{pl} \frac{d\tilde{\rho}_p}{dx} = \rho(\tilde{U}_4 - \tilde{U}_T) A_c , \quad (2.8)$$

If the process is isentropic inside the plenum, then

$$\frac{d\tilde{\rho}_p}{dt} = \frac{1}{\tilde{a}^2} \frac{d\tilde{P}}{dt} , \quad (2.9)$$

Where,  $\tilde{a}$  is the speed of sound.

So from Equation 2.8 and 2.9,

$$\frac{\tilde{V}_{pl}}{\tilde{a}^2} \frac{d\tilde{P}}{d\tilde{t}} = \rho(\tilde{U}_4 - \tilde{U}_T) A_c , \quad (2.10)$$

Now Greitzer's B parameter is introduced into the above equation. This equation is also

nondimensionalized by length scale  $R$  , velocity scale  $U_w$  and time scale  $\frac{R}{U_w}$  .

The Greitzer's parameter,  $B = \frac{U_w}{2wL_c}$  ,

where,

$$w = \tilde{a} \sqrt{\frac{A_c}{V_{pl} L_c}} ,$$

So, after introducing the Greitzer parameter, Equation 2.10 becomes,

$$4l_c B^2 \frac{d\hat{P}}{dt} = \hat{U}_4 - \hat{U}_T , \quad (2.11)$$

In the above equation,  $\hat{U}_4$  is same as the  $\hat{U}_0$ .  $\hat{U}_T$  is modeled as a function of the throttle coefficient and pressure drop.

$$\hat{U}_T = \sqrt{\gamma \hat{P}},$$

Where  $\gamma$  is the throttle coefficient.

## 2.6 The Compressor Characteristic $F_c(\hat{U})$ :

The compressor characteristic given by Moore and Greitzer in 1986 [1] is,

$$F_c(\hat{U}) = [\hat{P}_{co} + H[1 + \frac{3}{2}(U-1) - \frac{1}{2}(U-1)^3]], \quad (2.12)$$

The coefficient of the above polynomial equation varies for different compression system. The above equation is assumed for a particular compression system.

## 2.7 Moore-Greitzer Model:

Equation (2.6) is integrated from 0 to  $2\pi$  over  $\theta$ . Then Equation (2.4) and (2.5) are applied to express the equation in terms of mean flow  $\hat{U}_0$ .

$$\hat{P} + (\Lambda + l_E + l_I) \frac{\partial \hat{U}_0}{\partial t} = \frac{1}{2\pi} \int_0^{2\pi} F_c(\hat{U}_0 + \hat{u}_2) d\theta, \quad (2.13)$$

where,  $\hat{u}_2 = \hat{U} - \hat{U}_0 = \sum_{n=1}^N A_n(t) e^{in\theta} + c.c.$  is the nonaxisymmetric disturbance of axial velocity.

Subtracting Equation (2.13) from Equation (2.6) we have,

$$\sum_{n=1}^N (K_n - \tilde{K}_n) A'_n(t) e^{in\theta} + \Lambda_r \frac{\partial \hat{\sigma}_2}{\partial \theta} + (\Lambda + l_E) \frac{\partial \hat{\sigma}_2}{\partial t} + c.c. = F_c(\hat{U}_0 + \hat{u}_2) - \frac{1}{2\pi} \int_0^{2\pi} F_c(\hat{U}_0 + \hat{u}_2) d\theta \quad (2.14)$$

After rescaling the above equation with  $W$  for velocity and  $H$  for pressure rise, the Moore-Greitzer Model is obtained.  $2W$  is the mass flow coefficient corresponding to the maximum pressure rise coefficient and  $2H$  is the maximum pressure rise coefficient according compressor characteristic curve. Thus the final form of equation after rescaling are:

$$P'(t) = \frac{1}{4B^2Sl_c} [U_0 - U_r(P)] \quad (2.15)$$

$$U'_0(t) = \frac{S}{l_c} [-P(t) + \frac{1}{2\pi} \int_0^{2\pi} P_c(U_0 + u_2) d\theta] \quad (2.16)$$

$$\begin{aligned} & \sum_{n=1}^N (K_n - \tilde{K}_n) A'_n(t) e^{in\theta} + \Lambda_r \frac{\partial u_2}{\partial \theta} + (\Lambda + l_E) \frac{\partial u_2}{\partial t} + c.c. \\ & = SP_c(U_0 + u_2) - \frac{S}{2\pi} \int_0^{2\pi} P_c(U_0 + u_2) d\theta \end{aligned} \quad (2.17)$$

where,

$$P = \frac{\hat{P}}{H}; U_0 = \frac{\hat{U}_0}{W}; u_2 = \frac{\hat{u}_2}{W}; S = \frac{H}{W}.$$

The above Moore-Greitzer Model is derived using the following equation:

$$F_c(\hat{U}) = HP_c(U)$$

$$P_c(U) = [P_{co} + 1 + \frac{3}{2}(U - 1) - \frac{1}{2}(U - 1)^3]$$

where  $P_{co} = \frac{\hat{P}_{co}}{H}$  ; For MG Model,  $P_{co} = 1.5$

## 2.8 One-mode Truncated Models:

The first Fourier mode of the stall disturbance  $u_2(\theta, t)$  is represented as,

$$u_2(\theta, t) = R_1(t) \cos(\theta + \partial)$$

After replacing the stall disturbance by this expression, the Equation 2.15, 2.16 and 2.17 of MG model is simplified as,

$$\frac{dP}{dt} = \frac{1}{4B^2Sl_c} [U_0(t) - U_T(P)]$$

$$\frac{dU_0}{dt} = \frac{S}{l_c} [-P(t) + P_c(U_0 - 1) - \frac{3}{4}(U_0 - 1)R_1^2]$$

$$\frac{dR_1^2}{dt} = \frac{4S}{\Lambda + l_E + K_1 - \tilde{K}_1} \left\{ \frac{3}{2} [1 - (U_0 - 1)^2] \left( \frac{R_1^2}{2} \right) - \frac{3}{4} \left( \frac{R_1^2}{2} \right)^2 \right\}$$

$U_e$  and  $P_e$  are equilibrium mean flow velocity and pressure rise respectively at equilibrium point and when  $R_1 \equiv 0$ .

$$U_e = U_T(P_e)$$

$$P_e = P_c(U_e - 1)$$

Also the states of the MG model can be expressed as,

$$x_1(t) = U_0(t) - U_e$$

$$x_2(t) = P(t) - P_e$$

$$x_3(t) = R_1^2$$

Using the above equations, the one mode truncated model is expressed as,



$$\frac{dx_1}{dt} = \frac{S}{l_c} [-(x_2 + P_e) + P_c(x_1 + U_e - 1) - \frac{3}{4}(x_1 + U_e - 1)x_3] \quad (2.18)$$

$$\frac{dx_2}{dt} = \frac{1}{4B^2Sl_c} [(x_1 + U_e) - U_T(x_2 + P_e)] \quad (2.19)$$

$$\frac{dx_3}{dt} = \frac{4S}{\Lambda + l_E + K_1 - \tilde{K}_1} \left\{ \frac{3}{2} [1 - (x_1 + U_E - 1)^2] \left( \frac{x_3}{2} \right) - \frac{3}{4} \left( \frac{x_3}{2} \right)^2 \right\} \quad (2.20)$$

The above model can be further simplified by rescaling time  $t$  with  $\tau = (\frac{S}{l_c})t$ ,

$$\frac{dx_1}{d\tau} = -(x_2 + P_e) + P_c(x_1 + U_e - 1) - \frac{3}{4}(x_1 + U_e - 1)x_3, \quad (2.21)$$

$$\frac{dx_2}{d\tau} = \frac{1}{\beta^2} [(x_1 + U_e) - U_T(x_2 + P_e)], \quad (2.22)$$

$$\frac{dx_3}{d\tau} = Kl_c [4(1 - (x_1 + U_E - 1)^2)x_3 - x_3^2], \quad (2.23)$$

Where,

$$K = \frac{3}{4(\Lambda + l_E + K_1 + \tilde{K}_1)} .$$

## 2.9 Optimization Problem:

The above Moore-Greitzer model is reformulated as an optimization problem to identify the parameter of the cubic characteristic curve. The reformulation is done by assuming that  $U_e$  and  $P_e$  satisfies the following equation:

$$P_e = P_c(U_e) \quad (2.24)$$

$$U_e = U_T(P_e) \quad (2.25)$$

It is assumed that  $P_c(U_e)$  is a 3<sup>rd</sup> order polynomial.

$$P_c(U_e) = a_0 + a_1 U_e + a_2 U_e^2 + a_3 U_e^3 \quad (2.26)$$

The coefficients of the polynomial equation are identified for a given value of  $p_e$  and  $U_e$ .

Applying  $P_c(U_e)$  to  $P_c(x_1 + U_e - 1)$  results in:

$$\begin{aligned} &P_c(x_1 + U_e - 1) \\ &= a_0 + a_1(x_1 + U_e - 1) + a_2(x_1 + U_e - 1)^2 + a_3(x_1 + U_e - 1)^3 \end{aligned} \quad (2.27)$$

The throttle characteristics:

$$U_T(P_e) = \sqrt{\gamma P_e} \quad (2.28)$$

$$\text{Then, } U_T(x_2 + P_e) = \sqrt{\gamma(x_2 + P_e)} \quad (2.29)$$

Substituting Equation 2.26-2.29 into Equation 2.21-2.25:

$$\begin{aligned} \dot{x}_1 = & (-x_2 + P_e) + [a_0 + a_1(x_1 + U_e - 1) + a_2(x_1 + U_e - 1)^2 + a_3(x_1 + U_e - 1)^3 \\ & - \frac{3}{4}(x_1 + U_e - 1)x_3], \end{aligned} \quad (2.30)$$

$$\dot{x}_2 = \frac{1}{\beta^2} [(x_1 + U_e) - \sqrt{\gamma(x_2 + P_e)}], \quad (2.31)$$

$$\dot{x}_3 = K\{4[1 - (x_1 + U_e - 1)^2]x_2 - x_3^2\}, \quad (2.32)$$

$$P_c(U_e) = a_0 + a_1U_e + a_2U_e^2 + a_3U_e^3, \quad (2.33)$$

$$U_e = \sqrt{\gamma P_e}, \quad (2.34)$$

## **2.10 Potential of MG Model:**

The Moore-Greitzer Model explains the flow dynamics within an axial flow compressor. Understanding the flow dynamics of the compressor is very important when designing a controller for the system. The provision of a sufficient stall margin is considered while designing a compression system and also a controller. The post stall behavior plays another important role here. The instability of the system after the inception of stall is a great concern while running the aircraft engine. Generally the stall is nonrecoverable i.e. the only remedy when the engine enters the post stall instability condition is to shut down the engine and restart it. The Moore-Greitzer Model relates the onset of stall with the mass flow rate and pressure rise of the system. So it is possible to predict the formation of stall by analyzing the flow rate and pressure rise. Also it is possible to prevent the inception of stall by controlling the flow rate and pressure rise. Using this potential of Moore-Greitzer Model, Tommy Gravdahl and Olav Egeland, [15] have developed surge controller by backstepping and closed couple valve. Shu Lin, Chunjie Yang, Ping Wu and Zhihuan Song [14] designed a controller to control speed and surge in variable speed axial compressor. Shu Lin and his group have designed a proportional controller to control speed and a fuzzy logic controller to control surge based on this MG Model.

The MG model has opened doors for further development of the compressor dynamics. As an example, Gravdahl and Egeland [13] have developed a model based on MG model where they have included Greitzer B parameter as another state. Thus using this new model, it became possible to design controller for variable speed compressor.

### **2.11 Shortcoming of MG Model:**

One of the shortcomings of the MG model is that it assumes the compressor speed as a constant. However, that shortcoming was overcome by Gravdhal and Egeland. In present work, the speed of the compressor is assumed as a constant as well i.e. the Greitzer  $B$  parameter is assumed as constant.

Moore and Greitzer [1] developed a set of nonlinear differential equations which describes three time dependent states of a compression system. The characteristic of the compression system is assumed and is represented by a third order polynomial curve. The parameters of this third order polynomial equation was also assumed by Moore and Greitzer for a particular compressor. Each compressor has a unique characteristic and a different cubic equation. Thus a shortcoming of MG model is that it can only be used to design a controller for the general dynamics of a compressor as this characteristic curve is unknown. The controller has to be customized according to each compression system and that's why the extraction of the parameter of the polynomial equation is important.

## Chapter 3

### Discrete time MG model and Genetic Algorithm

#### 3.1 Background:

The compressor characteristics curve represents the behavior of the compressor. Each compressor shows different behavior and has different characteristic curves. From the previous chapter on the Moore-Greitzer model, the compressor characteristics and throttle characteristics are expressed by Equation 2.33 and 2.34. Both equations are represented here again.

$$P_c(U_e) = a_0 + a_1 U_e + a_2 U_e^2 + a_3 U_e^3, \quad (2.33)$$

$$U_e = \sqrt{\gamma P_e} \quad (2.34)$$

Equation 2.33 represents a third order polynomial curve and Equation 2.34 represents a quadratic curve. The intersecting point of these two curves represent the operating point ( $U_e$ ,  $P_e$ ) of the compressor at a specific throttle point. As Equation 2.34 is a function of throttle coefficient ( $\gamma$ ), it is possible to move the throttle characteristics along the compressor curve by varying the value of throttle coefficient. Thus for a different value of throttle coefficient, there is a different operating point of each compressor system.

By solving Equations 2.33 and 2.34, an operating point is obtained. As an example, for throttle coefficient of 0.5, the operating point is calculated as (0.4684, 0.4387). That means for  $\gamma = 0.5$ , the mass flow rate coefficient and pressure rise coefficient are 0.4684 and 0.4387, respectively.

The MG model is represented on the basis of the characteristics curve as the following set of equations.

$$\dot{x}_1 = (-x_2 + P_e) + P_c(x_1 + U_e - 1) - \frac{3}{4}(x_1 + U_e - 1)x_3, \quad (2.21)$$

$$\dot{x}_2 = \frac{1}{\beta^2} [(x_1 + U_e) - U_T(x_2 + P_e)], \quad (2.22)$$

$$\dot{x}_3 = K\{4[1 - (x_1 + U_e - 1)^2]x_2 - x_3^2\} \quad (2.23)$$

The above MG Model consists of three states  $x_1, x_2$  and  $x_3$ . Here

$x_1$  is the mean velocity disturbance expressed as  $x_1 \triangleq U(t) - U_e$ ,

$x_2$  is the pressure rise disturbance expressed as  $x_2 \triangleq P - P_e$ ,

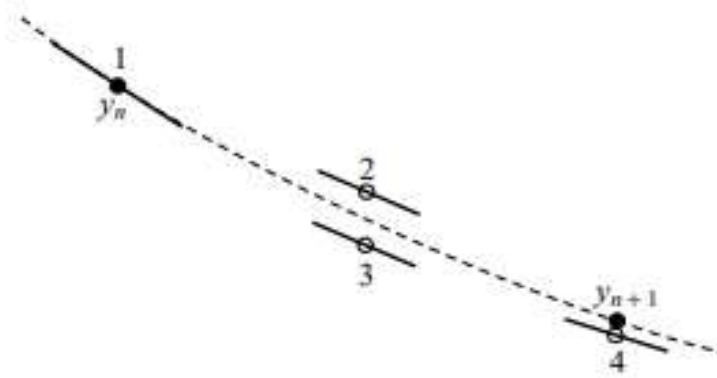
$x_3$  is the amplitude of the non-axisymmetric disturbance  $x_3 \triangleq A_1^2$ .

These states of the MG Model at operating point (0.4684, 0.4387) and throttle coefficient 0.5 are approximated in discrete time using Runge-Kutta 4<sup>th</sup> order method.

### 3.2 Discretization of MG Model:

Runge-Kutta fourth order (RK4) method is selected to solve MG Model in discrete time. Runge-Kutta 4<sup>th</sup> order method is a numerical method to solve a system of coupled ordinary differential equations. This method is equivalent to Euler's method, except whereas Euler's method considers only one point on each step, the RK4 method evaluates at four points on the step size for each iteration. Out of the four points, two points are the end points and the other two points are located on the middle of each iteration.

The four points where the derivative is evaluated at is depicted in Figure 3.1



**Figure 3.1:** The four points where the derivative is evaluated for Runge-Kutta 4<sup>th</sup> order method.

The derivative at point 1, 2, 3 and 4 are expressed as  $k_1, k_2, k_3$  and  $k_4$ , respectively. The equation used to find the derivative at these points are:

$$k_1 = f(x_n, t_n) \quad (3.1)$$

$$k_2 = f\left(x_n + h\frac{k_1}{2}, t_n + \frac{h}{2}\right) \quad (3.2)$$

$$k_3 = f\left(x_n + h\frac{k_2}{2}, t_n + \frac{h}{2}\right) \quad (3.3)$$

$$k_4 = f(x_n + hk_3, t_n + h) \quad (3.4)$$



The Moore-Greitzer (MG) Model can be expressed in the form,

$$\dot{x} = f(x, t) \quad (3.5)$$

Here  $x$  is to be approximated from the initial value  $x_0$  at initial time  $t_0$ . The standard Runge-Kutta 4<sup>th</sup> order method takes the following form:

$$x_{n+1} = x_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \quad (3.6)$$

$$t_{n+1} = t_n + h \quad (3.7)$$

$x_{n+1}$  is derived from present value  $x_n$ , step size,  $h$  and four derivative  $k_1, k_2, k_3, k_4$ .

The above method is applied to the MG model for approximating the states  $x_1, x_2$  and  $x_3$  at an operating point (0.4684, 0.4387). The initial value of  $x_1, x_2$  and  $x_3$  are assumed as 0.5, 0.5 and 2 respectively. The step size is taken as 0.0010.

### 3.3 Proof of Concept:

The concept of Runge-Kutta Method is validated by using it on a system of non-linear differential equations. The approach is to simulate the known system using Range-Kuttha 4<sup>th</sup> order method and compare it with existing result. A system of three non-linear differential equations is selected which shows the chaotic behavior of an atmospheric system. An American mathematician/meteorologist Edward Norton Lorenz invented the system while investigating the possibility of predicting weather accurately on long term basis. According to his research, there are many factors that affect long term weather prediction. A small change in any part of a system, can change the other parts of the system as well. So there are some unknown factors in the system which makes the system chaotic or unpredictable. Also this chaotic behavior of the system makes the forecasting of long term weather very difficult.

Edward Norton Lorenz built a mathematical model that depicts the way air flow in the atmosphere. The mathematical model is as follows:

$$\begin{aligned}\frac{dx}{dt} &= \sigma(y - x) \\ \frac{dy}{dt} &= x(\rho - z) - y \\ \frac{dz}{dt} &= xy - \beta z\end{aligned}$$

The model indicates rate of change of  $x$ ,  $y$  and  $z$ .  $x$  is proportional to the change of convection rate,  $y$  is proportional to the temperature variation in horizontal direction and  $z$  is proportional to the temperature variation in the vertical direction. The parameters of the model are  $\sigma$ ,  $\beta$  and  $\rho$ .

The parameters of the system not only have a numerical value but also indicate physical characteristics of the system.  $\sigma$  is known as the Prandtl number and demonstrates the viscosity

and thermal conductivity of the system. Another constant,  $\beta$  represents the dimensions of the box where the Lorenz attractor is located. Parameter,  $\rho$  indicates a control parameter which is generally difference between highest and lowest driving forces of the system. As an example, if the model is used to represent the convection flow in a fluid filled reservoir, then the difference between the temperature of the top and bottom layer is indicated by  $\rho$ .

The solution of the Lorenz's mathematical model is known Lorenz attractor. Different value of the parameter gives different behavior of the system. The value taken for the parameters of the system are:

$$\sigma = 10;$$

$$\beta = 8/3$$

$$\rho = 28$$

The Lorenz model is simulated in continuous time and discrete time. ODE45 is used to solve it on continuous time and Range- Kutta 4<sup>th</sup> order method is used to solve it on discrete time. Both simulation results are compared and it is found that they fit one another. The simulation of the states  $x$ ,  $y$  and  $z$  using ode45 and the Runge-Kutta method is shown in Figure 3.2 and 3.3.

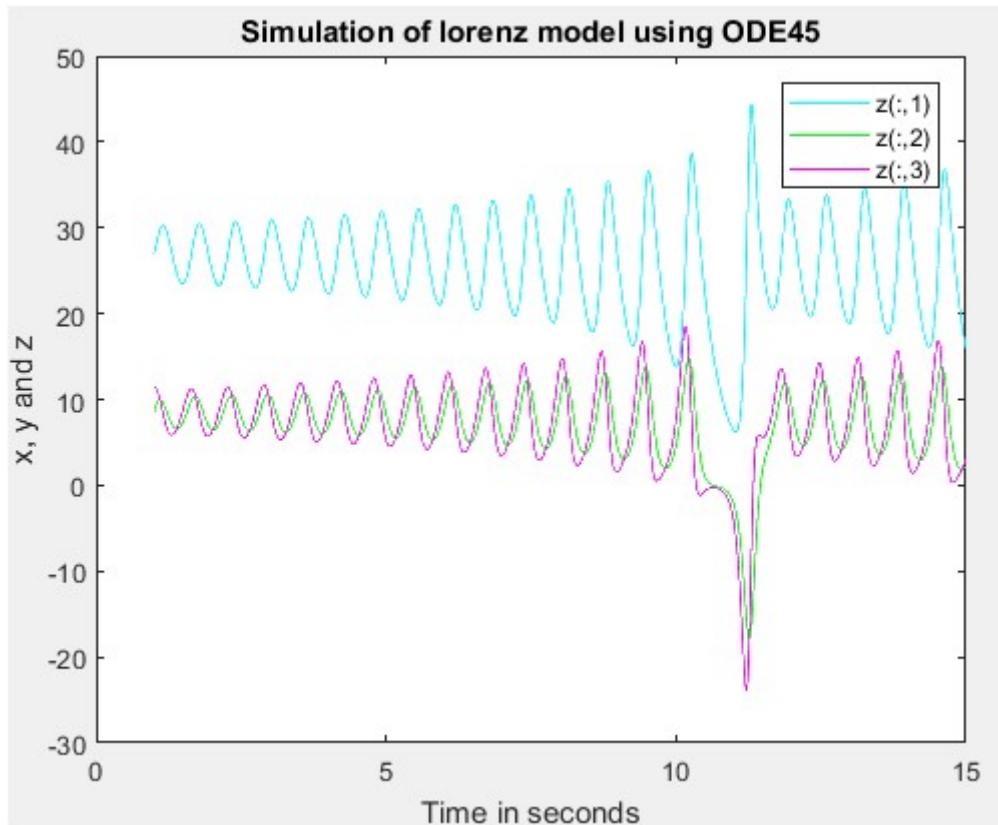


Figure 3.2: Simulation of Lorenz model using ode45.

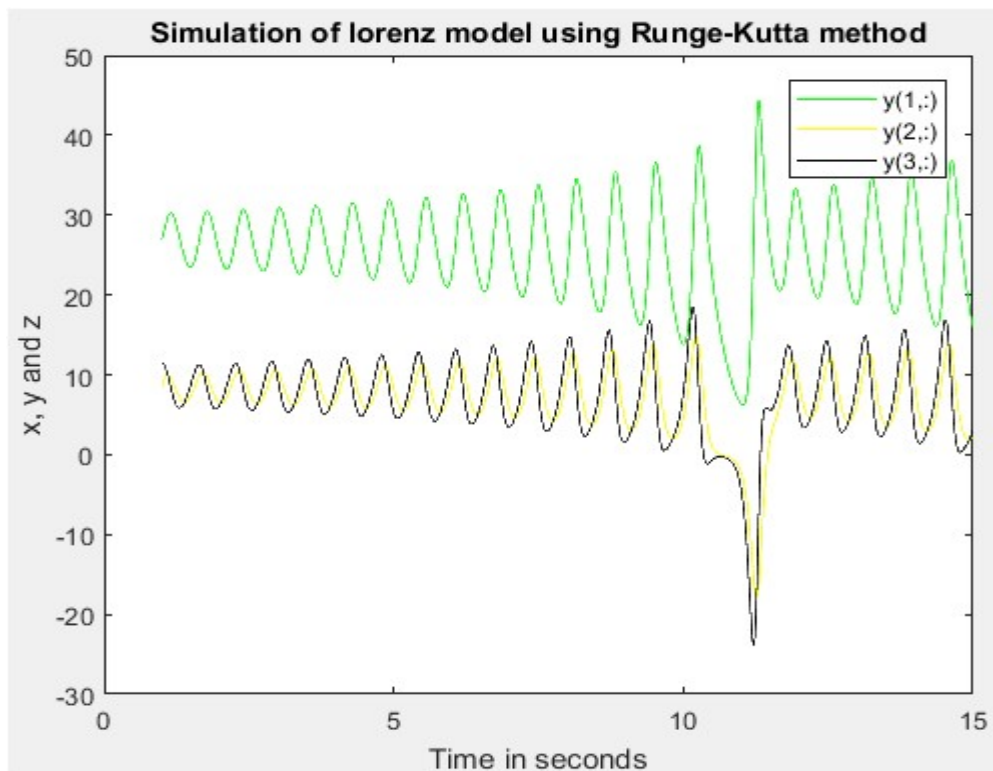


Figure 3.3: Simulation of Lorenz model using Runge-Kutta method

The error between the above two simulation of Lorenz model is shown in Figure 3.4:

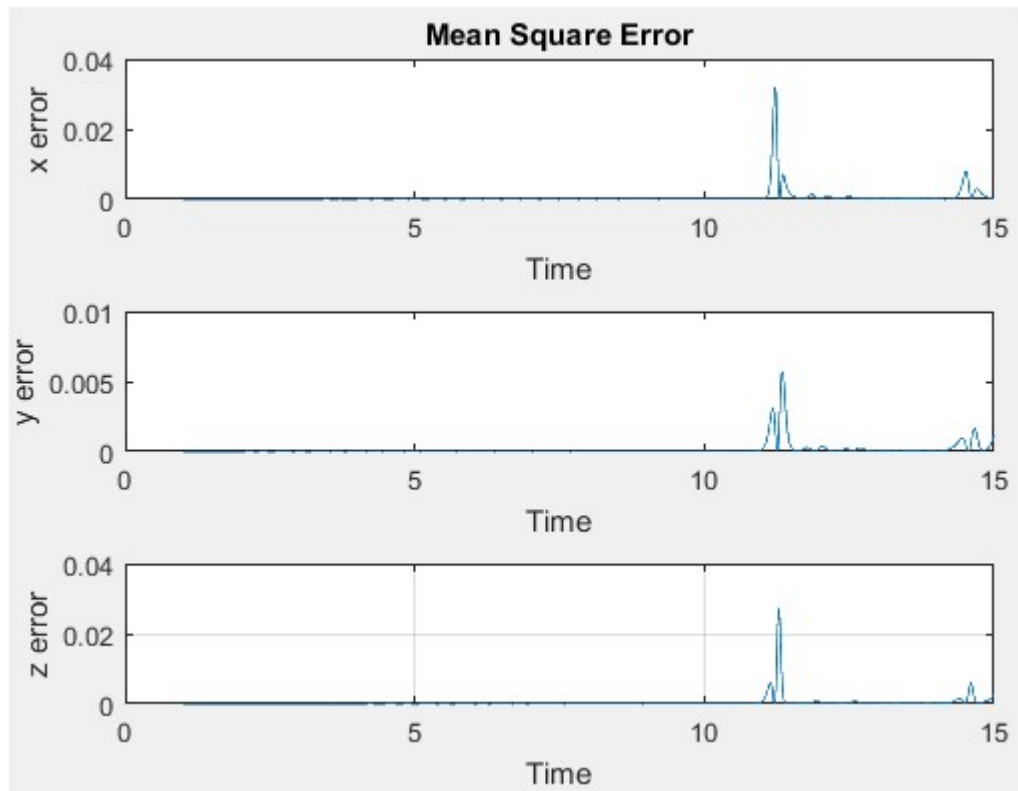


Figure 3.4: Mean square error

### 3.4 Genetic Algorithm:

Genetic Algorithm is a method to optimize constrained or unconstrained problem based on the theory of evolution of biological system. Mr. Holland and his students developed this method for doing adaptive search between 1960s and 1970s. This method is developed on the basis of the evolutionary process. In a biological system, only the fittest species survives and the children inherits the characteristics of their parents. So only the best parents can produce offspring.

As the genetic algorithm follows the evolution in a biological process, the terminology of the algorithm closely follows some terminology of biology. Some basic steps that genetic algorithm follows are:

- Creation of a fitness function.
- Creation of a population of chromosome.
- Evaluating the best chromosome based on the fitness function.
- Selecting best chromosome for mating or reproduce.
- Generating the next generation from the best chromosomes.
- Mutation among chromosomes of the new generation.
- The steps are repeated until the test converges to the requirement.

#### 3.4.1 Creation of Initial population:

A chromosome is an array of parameters. The purpose of genetic algorithm is to solve the problem and find the optimum value of the parameters. If the problem consists of  $N_{par}$  parameters, then each chromosomes is represented by  $N_{par}$  parameters array.

$$chromosome = [P_1, P_2, \dots, P_{N_{par}}],$$

For the Moore-Greitzer model, the genetic algorithm is deployed for optimizing the coefficients of the characteristic curve. As the polynomial equations consists of four coefficients ( $a_0, a_1, a_2, a_3$ ), the chromosome is an array of four elements and each element has a floating point number. Thus the chromosome of a MG model is an array of four numerical values.

$$chromosome = [a_0, a_1, a_2, a_3],$$

The initial population of the chromosomes is randomly generated. Each chromosome in the population is then evaluated based on the fitness function. The equation used for generating the initial population is

$$I_{pop} = (h_i - l_0) \times random\{N_{ipop} N_{par}\} + l_0,$$

where,

$random\{N_{ipop} N_{par}\}$  is a function that generates a matrix of size  $N_{ipop} \times N_{par}$  of uniform random number valuing between 0 and 1.

$N_{par}$  = The total number of parameters.

$N_{ipop}$  = The initial population size of each parameter.

$h_i$  = Highest value in the range of parameter

$l_0$  = Lowest value in the range of parameter

For the current optimization problem of the characteristic curve of the Moore-Greitzer model, the total number of parameters is  $N_{par} = 4$ ,

the initial population size of each parameter,  $I_{pop} = 48$ , (The algorithm is programmed in a way that the user will be able to input the population size. For illustration, this is assumed as 48)

So, the matrix size of initial population of chromosome becomes  $48 \times 4$ .

Each parameter has different range of values. The population of each parameter is generated separately. Then the population of each parameter is added together to form the total population matrix of the chromosome.

For parameter  $a_0$ , the highest number,  $h_i = 0.35$

the lowest number,  $l_0 = 0.25$

For parameter  $a_1$ , the highest number,  $h_i = 0.95$

the lowest number,  $l_0 = 0.85$

For parameter  $a_2$ , the highest number,  $h_i = 0.40$

the lowest number,  $l_0 = 0.30$

For parameter  $a_3$ , the highest number,  $h_i = -3.40$

the lowest number,  $l_0 = -3.60$



### 3.4.2 Creation of a cost function and evaluation of each chromosome:

The term “fitness” comes from the theory of evolution. It measures how fit the chromosomes are to optimize the fitness function. The fitness function is referred as cost function in this optimization algorithm. Equation 2.30-2.34 is constructed to use this optimization problem.

In order to perform the optimization problem, at first the MG model consisting of Equation 2.30 – 2.34 is simulated. This simulation provides data on characteristics curve and state response of the system. The proposed genetic algorithm finds the best value of the parameters from the initial population of chromosomes such that it fits the simulated data of characteristic curve and state response.

After the generation of the initial population of chromosomes, the fitness of each parameter of the chromosomes is evaluated and a fitness score is assigned. The fitness is evaluated on the basis of the cost function. So, in order to get the best result from the genetic algorithm, the formation of an effective cost function is crucial.

The state response and the characteristic curve is generated for each chromosome of population using the MG model. The newly generated MG model is then compared with the previously simulated true MG model. For each chromosome, the error between the two models is computed for each point. The error for each point is then summed and mean square error (MSE) is evaluated. In this way, for each chromosome, four MSE are recorded. One for the characteristic curve error and the other three for the three state response errors. Weights are assigned for each MSE and then the weighted MSE are added together to form the cost function. The cost function formed is given as Equation 3.1.

$$Cost = w_1mse_{char} + w_2mse_{x1} + w_2mse_{x2} + w_3mse_{x3} \quad (3.8)$$

### 3.4.3 Pairing:

Since each chromosome is assigned a fitness value, they are sorted from lower value to higher value. Those chromosome who has low cost is the fittest chromosome for mating. In this work, top 50% of the chromosomes are kept for mating. The selection of Mom and Dad chromosome is accomplished by a roulette wheel method, where the fittest chromosomes have the highest probability to be selected as a parent.

### 3.4.4 Mating:

There are different methods for implementing the mating operation in genetic algorithm. A combination of cross over method and extrapolation method is implemented in this work. The advantage of this method is that it inherits the Binary genetic algorithm procedure where a cross over point is randomly selected between two new parent chromosomes. The equation used for selecting the cross over point is given by Equation 3.2.

$$\alpha = \text{round}\{\text{random} \times N_{\text{var}}\}, \quad (3.9)$$

where  $N_{\text{var}} = 4$  for the MG model.

After getting the cross over point, the parent chromosome can be written as

$$\text{parent}_1 = [P_{m1} \dots P_{m\alpha} \dots P_{mN_{\text{var}}}],$$

$$\text{parent}_2 = [P_{d1} \dots P_{d\alpha} \dots P_{dN_{\text{var}}}],$$

where  $m$  and  $d$  stand for mom and dad respectively. The element at cross over point is formulated to produce a new element. This new element will appear in the new offspring.

$$P_{\text{new1}} = P_{m\alpha} - \beta(P_{m\alpha} - P_{d\alpha}),$$

$$P_{\text{new2}} = P_{d\alpha} + \beta(P_{m\alpha} - P_{d\alpha}),$$

where  $\beta$  is a random value between 0 and 1. With this new element the cross over method is carried out at the cross over point with the remaining elements. After cross over the new off springs are:

$$offspring_1 = [P_{m1}P_{m2}.....P_{new1}.....P_{dNvar}],$$

$$offspring_2 = [P_{d1}P_{d2}.....P_{new2}.....P_{mNvar}],$$

On the same way, for a different values of  $\beta$ , two more off springs can be generated from the same parents.

### 3.4.5 Mutation:

The cost function may have many local minimums. So there is a possibility that the genetic algorithm may converge quickly to a local minimum. To avoid this problem, the algorithm is forced to search other areas of the cost surface so that it does not end up on a local minimum and instead find the global minimum. In order to do that, a forced alteration on the variables of the chromosome is carried on which is known as mutation. The number of parameters to be mutated is calculated by multiplying the total number of parameter with the mutation rate. The parameter to be mutated is selected by randomly selecting the row and column of the parameter. The selected parameter is then replaced by a new random number.

## **Chapter 4**

### **Fuzzy Logic Controller Design**

#### **4.1 Fuzzy Logic:**

Fuzzy logic (FL) is a method of reasoning that resembles the human reasoning. Fuzzy logic imitates the human's decision making process which involve all intermediate possibilities between YES and NO. The YES and NO are referred as 0 and 1 in Boolean logic for TRUE and FALSE, respectively. The conventional logic block in a computer generates precise outputs like TRUE and FALSE and is not capable of producing any intermediate value between 0 and 1.

Lotfi Zadeh, a professor at University of California at Berkley, observed that human decision making is unlike than that of computers. It may contain different possibilities between YES and NO. As an example human can make decisions such as YES, Possibly YES, NO, Possibly NO, Cannot Say, etc. The fuzzy logic considers all the possibilities such as the human mind to achieve an output from the inputs.

Fuzzy logic can be implemented to control a system of various size and capabilities. As example it can be implemented from micro controller to large scale networked based controller. At present it is being implemented in automotive systems and electronic goods. Example of fuzzy logic in an automobile is the automatic gearbox, vehicle environment control and four wheel steering. In electronic goods, it is implemented on washing machines, microwave oven, vacuum cleaners, photocopiers etc.

The algorithm of a fuzzy logic system is given by the following steps:

- Define linguistic variables and terms
- Develop membership functions for variables.
- Develop base of rules.
- Fuzzification of the input crisp data into fuzzy data using membership function.
- Implication or evaluation of each rule of base of rules.
- Aggregation or combination of the output of each rule into a single fuzzy set.
- Defuzzification or converting the fuzzy output data into nonfuzzy data.

The algorithm can be represented by the Figure 4.1.

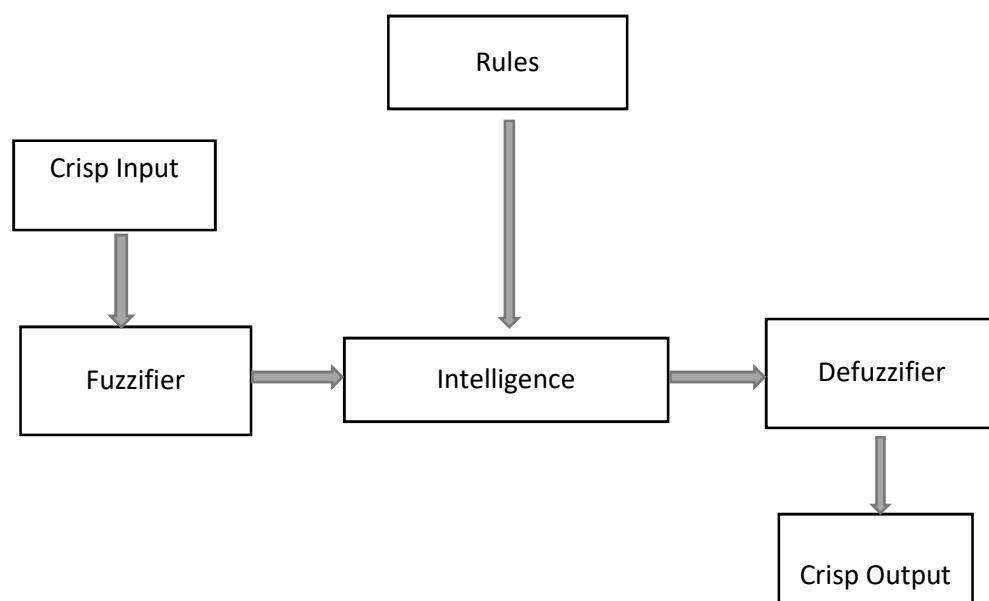


Figure 4.1: Fuzzy Logic Process Flow

## 4.2 Fuzzy Logic Controller for Moore-Greitzer Model:

The above algorithm is implemented for designing a fuzzy logic controller with the objective to control the mass flow in an axial flow compressor. The operating point is manipulated by controlling the mass flow rate in the axial compressor. The mass flow is controlled by varying the opening of throttle. Thus, a fuzzy logic controller is designed where the input is the error between the desired operating point and the actual operating point. The output is the throttle coefficient. The schematic diagram of the controller is shown in Figure 4.2.

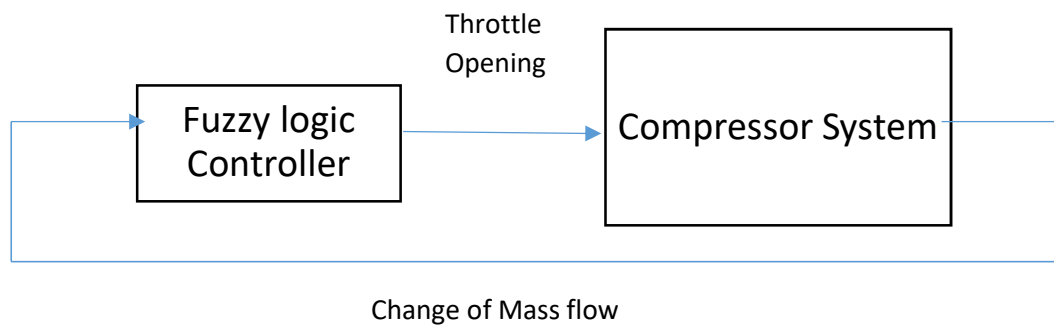


Figure 4.2: Diagram of Fuzzy Logic Controller

#### 4.2.1 Define Fuzzy linguistic variables and terms:

The input into the fuzzy logic controller for the Moore-Greitzer model is the error between the desired mass flow and the actual mass flow. So, the input variable is ‘Error’ and it consists of linguistic variables. The linguistic variables are terms in simple word. The linguistic set of input variable “Error” is expressed as,

$$\text{Error} = \{\text{Positive, Zero, Negative}\},$$

where Positive means actual mass flow is higher than the desired mass flow, zeros means the desired mass flow and the actual mass flow are the same, Negative means the actual mass flow is lower than the desired mass flow.

The output of the controller is the throttle opening. So the output variable is throttle opening. The linguistic set of throttle opening is,

$$\text{Throttle opening} = \{\text{Close, Partially Open, Open}\}.$$

#### 4.2.2 Development of membership function:

A membership function is a curve that assigns a membership value to each input value. The degree of membership or membership value ranges from 0 to 1. This membership value represents how closely the input crisp value belongs to a fuzzy set. As an example, if  $x$  is the element of a universe of discourse  $X$ , then the fuzzy set can be represented as,

$$A = \{x, \mu_A(x) \mid x \in X\}, \quad (4.1)$$

where,  $\mu_A$  is the membership function of  $x$  in the fuzzy set  $A$ . As  $x$  is the element of  $X$ , the membership function maps each element of  $X$  in the fuzzy set.

Equation 4.1 can be used to develop membership function of the input variable of the Moore-Greitzer Model. The universe of discourse  $X$  is the input Variable “ERROR” and  $x$  is the

elements of  $X$  i.e. the linguistic set of input variable - Positive, Zero, Negative. The membership function  $\mu_A$  for each linguistic term is mapped from 0 to 1.

The MATLAB® fuzzy logic toolbox has 11 types of built in membership function. These built in membership functions are formed from several basic function:

- Linear piece-wise function.
- Gaussian distribution function
- Sigmoid curve
- Cubic and quadratic polynomial curve

Thus there is a wide range of membership functions to choose from. It is also possible to create own membership functions. It is not necessary to use a complicated membership function for a good fuzzy inference system. The simplest membership function such as the triangular or trapezoidal membership function could work very well.

In this work, triangular membership functions are selected due to the advantage of simplicity. The name of the triangular membership function is ‘trimf’ and is represented by a triangle formed by connecting of three points.

The membership function for the input variable ERROR is represented in Figure 4.3

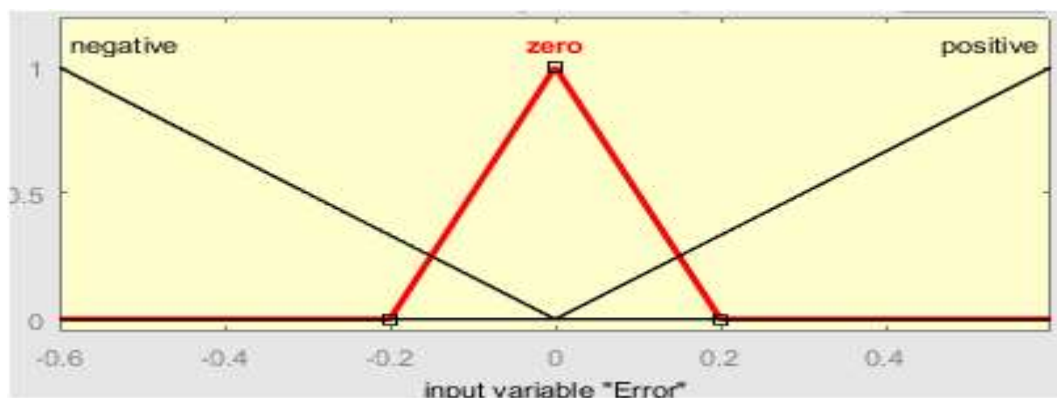


Figure 4.3: Membership function of input variable “Error”



Similarly, the membership functions for the output variable “Throttle Opening” are formed. The range of the output variable is selected from 0 to 2. Figure 4.4 represents the membership function of the output variable.

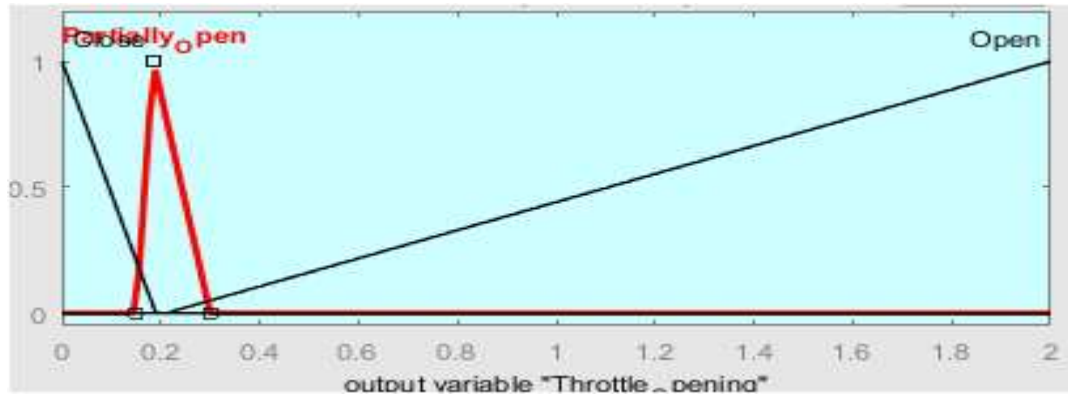


Figure 4.4: Membership function of the output variable

#### 4.2.3 Development of Fuzzy Rules:

The *if-then* rules statement is used to construct the fuzzy rules. The form of the *if-then* rule is,

*If x is A then y is B,*

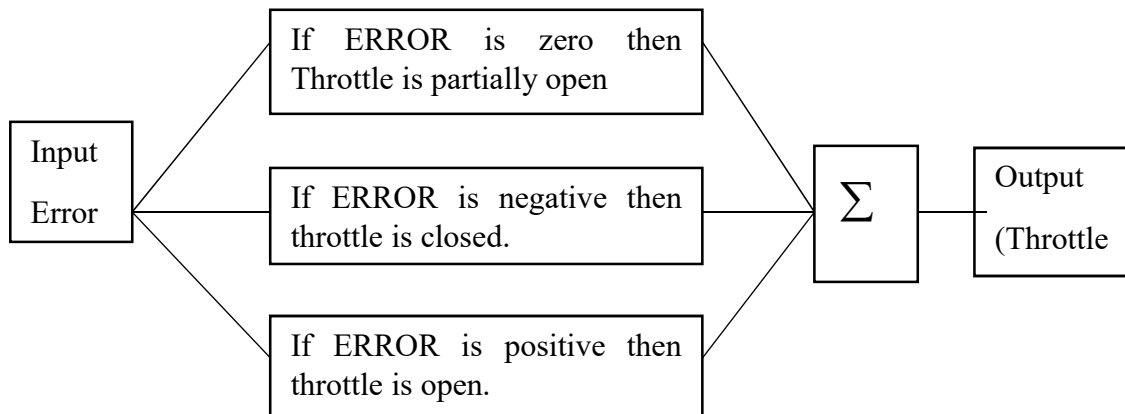
where A and B are fuzzy sets for input variable  $X$  and output variable  $Y$ . The first part '*If x is A*' is called antecedent and the second part '*Y is B*' is called consequent.

In the Moore-Greizer model, the  $X$  represents the input variable “ERROR” and the  $Y$  represents the output variable “Throttle.” Thus the base of rules constructed for this present controller can be represented as:

- If ERROR is zero then Throttle is partially open
- If ERROR is negative then throttle is closed.
- If ERROR is positive then throttle is open.

In order to implement the rules, at first the input needs to be fuzzified. The flow of information in the fuzzy inference system can be represented by Figure 4.5.

A system of one input, one output and three rules



The inputs are non-fuzzy number within a range

The rules are evaluated using fuzzy reasoning

The results are aggregated and defuzzified

output is non-fuzzy

Figure 4.5: Flow of fuzzy reasoning

#### 4.2.4 Fuzzification:

The membership functions for each linguistic term converts the crisp input into a fuzzy input. This process is called fuzzification. Fuzzification assigns a value between 0 and 1 for each crisp value. As an example, in the present work the linguistic term “Positive” ranges from 0 to 0.6. Any value in this range corresponds to a number in membership function. Suppose 0.6 has a degree of membership 1 while 0 has degree of membership 0. Any value between 0 and 0.6 will have a corresponding fuzzy value between 0 and 1.

#### **4.2.5 Implication:**

After fuzzification, the evaluation of each rule is performed. This evaluation is known as implication. The implication process consists of weighing each rule separately. Each rule can be weighted by a value between 0 and 1. One rule is given less effect over other rules by assigning the weight to the rules. 1 is the maximum weight so one rule can be made less effective by assigning a weight lower than 1 to that rule. In this work, all rules have same weight 1.

#### **4.2.6 Aggregation:**

Each rule is evaluated separately and the output for each rule is a fuzzy set. The aggregation method is to combine all the fuzzy sets into one fuzzy set. There are three built in method for performing the aggregation:

- Sum
- Max
- Probor

In the present work, the Max aggregation method is used.

#### **4.2.7 Defuzzification:**

The output of the result after aggregation is a range of output values. These output values are a fuzzy set and need to be defuzzified in order to get a single output or crisp output value. This process to get a single output from a fuzzy set is known as fuzzification process. There are five built in fuzzification method in fuzzy logic toolbox. The centroid calculation is the most popular fuzzification method. The other fuzzification method are smallest of maximum, largest of maximum, middle o maximum and bisector.

After following all the steps as listed above, the final throttle output for an input error -0.0622 is found as 0.185 as shown in Figure 4.6.

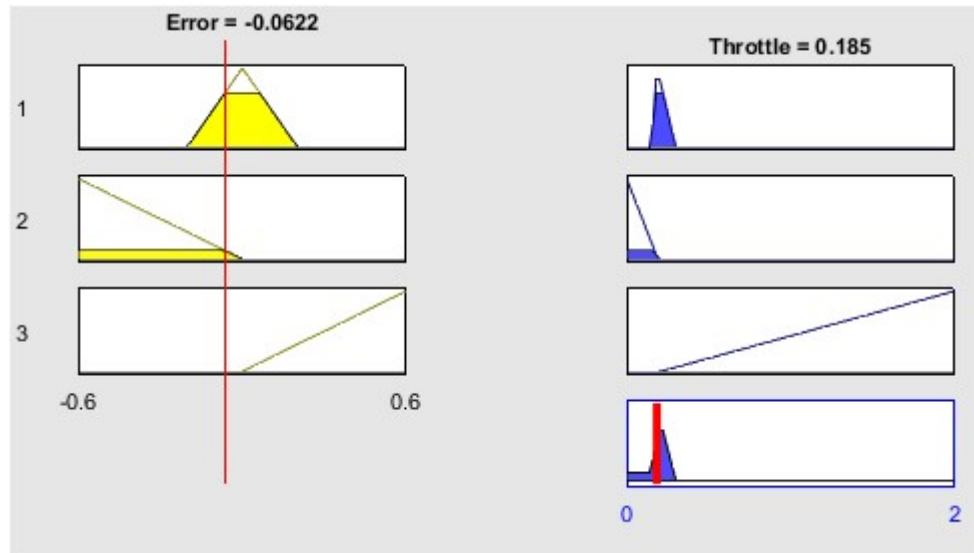


Figure 4.6: Fuzzy Inference system

The controller is designed in a way that when there is no error then the throttle coefficient is 0.21. When the error is positive, i.e. the operating mass flow is greater than the designated mass flow, then the throttle opening is reduced to reduce the mass flow rate and vice versa.

### 4.3 Simulation block diagram:

The fuzzy logic controller is implemented in a Simulink® model to control the mass flow rate. A MATLAB® function block is used to evaluate the Moore-Greitzer compressor characteristics. The output of this block is the operating point. The operating point is represented by the mass flow rate and pressure rise coefficient. The controller is designed to keep the mass flow rate at 0.3270 and the pressure rise coefficient at 0.5093. The error between designated operating point and the actual operating point is measured. The error is then given as input to the controller. The controller changes the throttle coefficient correspondingly to reduce the error.

The complete Simulink® block diagram is given is Figure 4.7

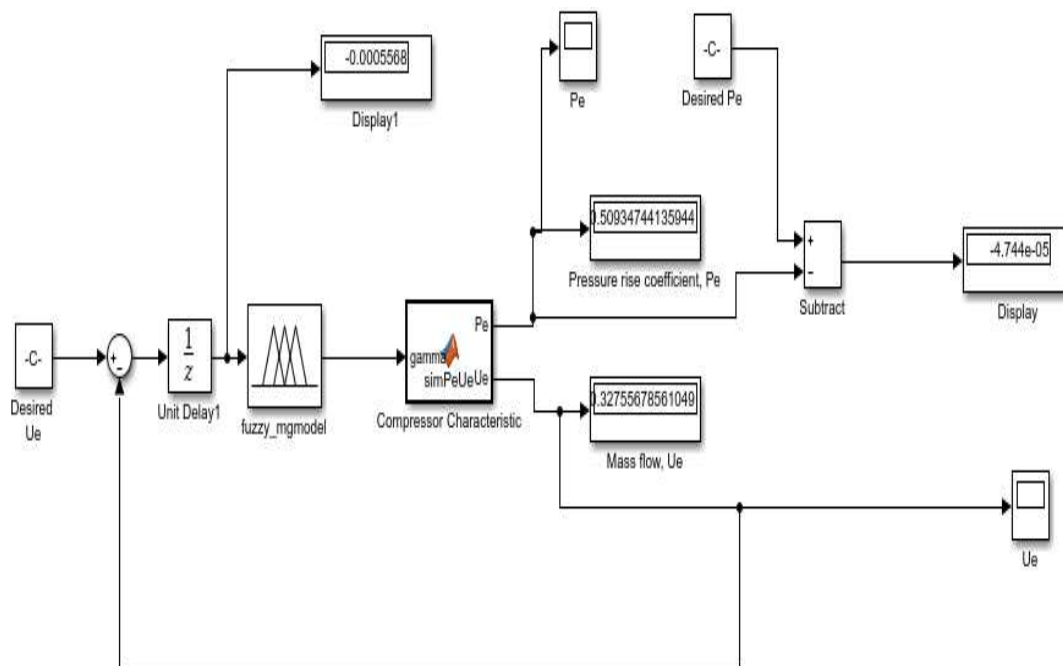


Figure 4.7: Simulink block diagram for MG Model control

As it can be seen from Figure 4.7, a constant block is used to specify the desired operating point i.e. the desired mass flow rate. A sum block is used to find the error between the desired

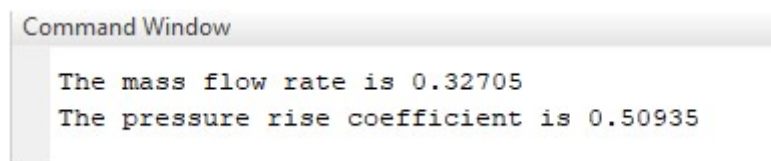
mass flow and the actual mass flow. The error is fed into a block which is linked to the fuzzy logic controller. The fuzzy logic controller takes the input and executes the necessary action according to the rules constructed for this controller. The output of this block is the throttle coefficient which is fed to the matlab function block. The matlab function block is linked to the matlab function 'simPeUe' which computes the corresponding mass flow rate and pressure rise coefficient. The feedback of the mass flow rate is given into the sum block and the error is computed. The process continues until the stable operating condition is achieved.

## Chapter 5

### Simulation Results and Discussion

#### 5.1 Moore-Greitzer model characteristic curve:

The characteristic curve is represented by a third order polynomial equation. Equation 2.33 represents the compression characteristic curve. The throttle characteristic is represented by a quadratic equation and the equation is Equation 2.34. The intersecting point of the two curve is known as the operating point. The intersecting point is found by solving the above two equations for a particular throttle coefficient. Each throttle coefficient produces a different operating point. For a throttle coefficient 0.21, the equations are solved and it is found that the mass flow and pressure rise coefficient are 0.3271 and 0.5093 respectively. The Matlab® program is shown in appendix 1 and the output is shown in Figure 5.1.



```
Command Window
The mass flow rate is 0.32705
The pressure rise coefficient is 0.50935
```

Figure 5.1: Operating point at throttle coefficient 0.21

Using the similar method, the Equation 2.33 and 2.34 are solved again and simulated for a range of value of throttle coefficient. The range for the value for the throttle coefficient is from 0 to 2.0. The simulation result within this range is shown in Figure 5.2.

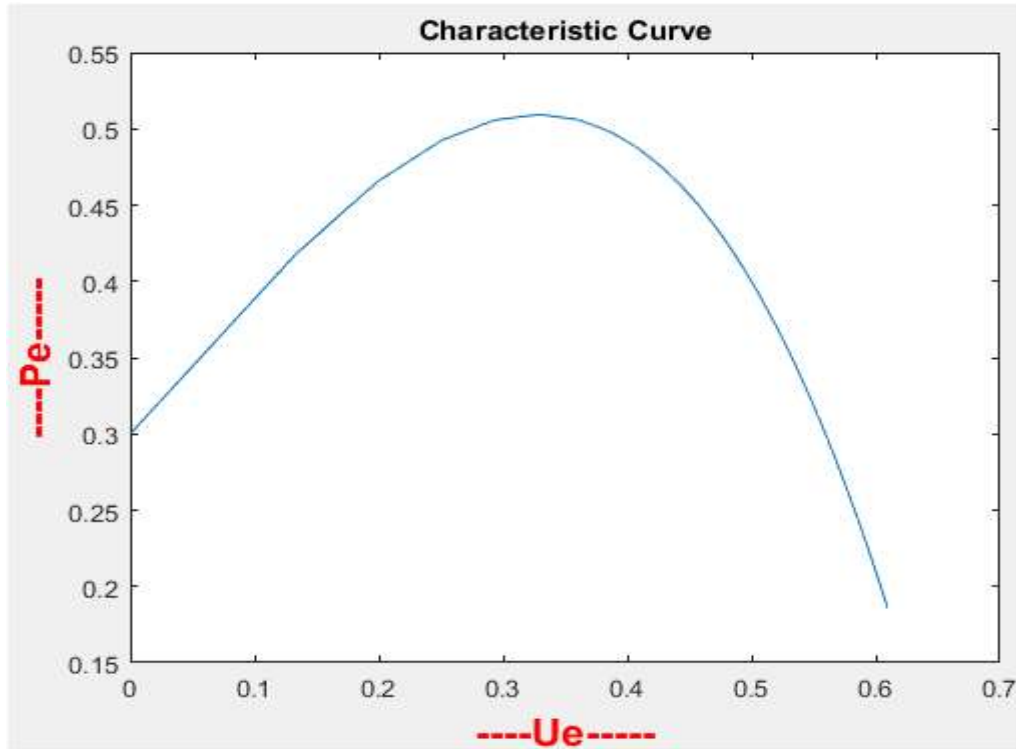


Figure 5.2: Characteristic curve for throttle coefficient  $\gamma = 0$  to 2.0

It can be seen from Figure 5.2 that the maximum pressure rise is 0.5093 i.e. at this point the slope of the characteristic curve is zero. The left side of this point is known as unstable region and the right side is known as the stable region. Our aim is to operate the compressor on the stable region. The throttle coefficient and mass flow rate with respect to the maximum pressure rise are 0.21 and 0.3270 respectively. The simulated data are expressed as Table 5.1.

Table 5.1: Simulated data for throttle coefficient  $\gamma = 0$  to 2.0

Throttle coefficient	Mass Flow ( $U_e$ )	Pressure rise ( $P_e$ )
0	0	0.3
0.042553191	0.133360476	0.417947828
0.085106383	0.199016441	0.465388526
0.127659574	0.250742954	0.492497402



0.170212766	0.293415153	0.505793113
0.212765957	0.329197472	0.509343554
0.255319149	0.35949227	0.506169189
0.29787234	0.385347349	0.498510787
0.340425532	0.407581874	0.487986257
0.382978723	0.426844863	0.475735396
0.425531915	0.443653039	0.462545841
0.468085106	0.458419345	0.448953175
0.510638298	0.471475268	0.435315816
0.553191489	0.483088355	0.421869031
0.595744681	0.493475873	0.408763089
0.638297872	0.502815412	0.396089896
0.680851064	0.511253108	0.383901494
0.723404255	0.518910021	0.372222872
0.765957447	0.525887095	0.361060836
0.808510638	0.532269039	0.350410144
0.85106383	0.538127375	0.340257758
0.893617021	0.543522842	0.33058578
0.936170213	0.548507312	0.321373472
0.978723404	0.553125316	0.312598651
1.021276596	0.557415278	0.304238629
1.063829787	0.561410505	0.29627085
1.106382979	0.565140006	0.288673301
1.14893617	0.568629147	0.281424778

1.191489362	0.571900203	0.274505046
1.234042553	0.574972807	0.267894918
1.276595745	0.577864326	0.261576291
1.319148936	0.580590179	0.255532144
1.361702128	0.583164095	0.249746516
1.404255319	0.58559834	0.244204463
1.446808511	0.587903904	0.238892015
1.489361702	0.590090661	0.23379612
1.531914894	0.592167507	0.228904594
1.574468085	0.594142478	0.224206059
1.617021277	0.596022849	0.219689896
1.659574468	0.597815222	0.21534619
1.70212766	0.599525602	0.211165681
1.744680851	0.601159463	0.207139718
1.787234043	0.602721804	0.203260213
1.829787234	0.604217201	0.199519605
1.872340426	0.605649848	0.195910815
1.914893617	0.607023599	0.192427217
1.957446809	0.608341997	0.189062601
2	0.609608312	0.185811147

The stable region of the characteristic curve is simulated again in Figure 5.3. The range of throttle coefficient is from 0.21 to 2.0.

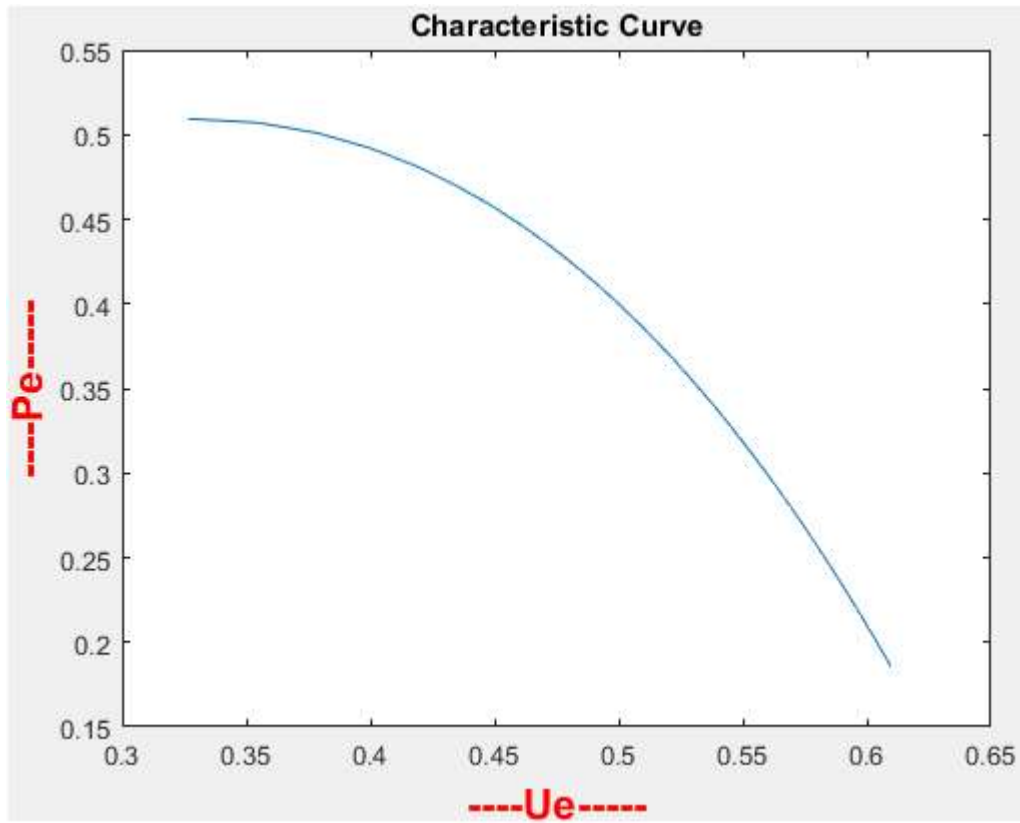


Figure 5.3: Characteristic curve for throttle coefficient  $\gamma = 0.21$  to  $2.0$

## 5.2 States of the Moore-Greitzer Model:

The three-coupled differential equation of Moore-Greitzer model is solved and simulated in Matlab®. The ode45 method and Runge-Kutta Method are used for simulation in continuous and discrete time. The simulation using ode45 is shown in Figure 5.5.

The states of the MG model are  $x_1, x_2$  and  $x_3$ . Here,

$x_1$  is the mean velocity disturbance expressed as  $x_1 \triangleq U(t) - U_e$ ,

$x_2$  is the pressure rise disturbance expressed as  $x_2 \triangleq P - P_e$ ,

$x_3$  is the amplitude of the non-axisymmetric disturbance  $x_3 \triangleq A_1^2$ .

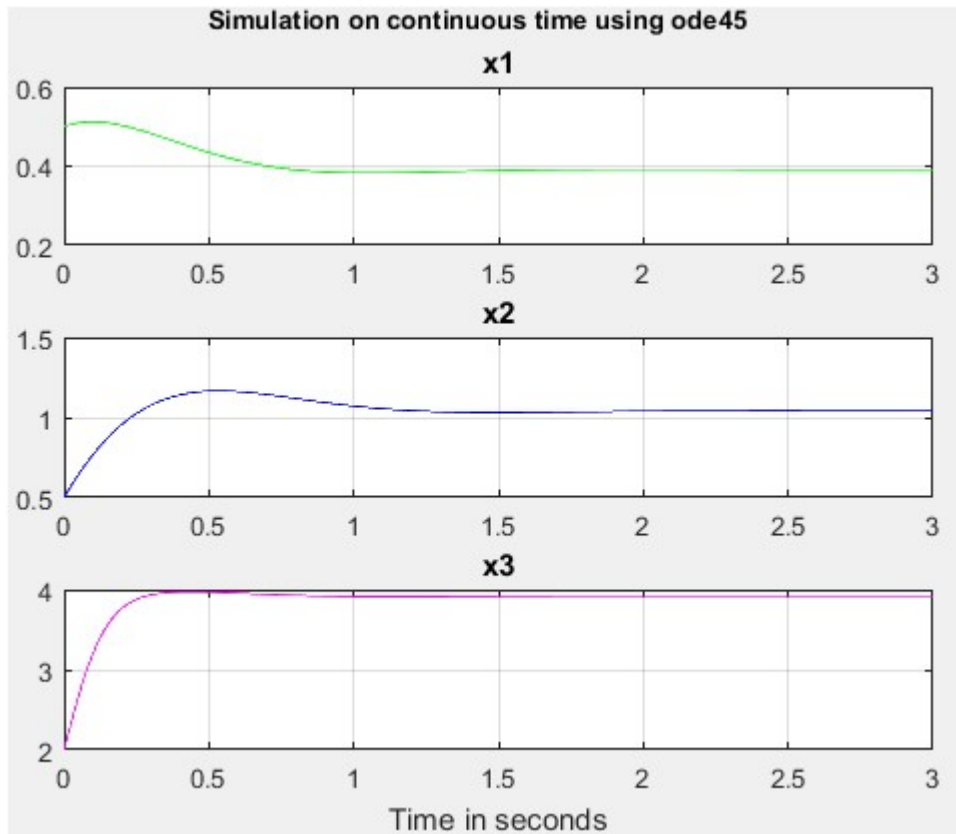


Figure 5.4: Simulation of states in continuous time

Similarly, the states are simulated using the Runge-Kutta method in Figure 5.6.

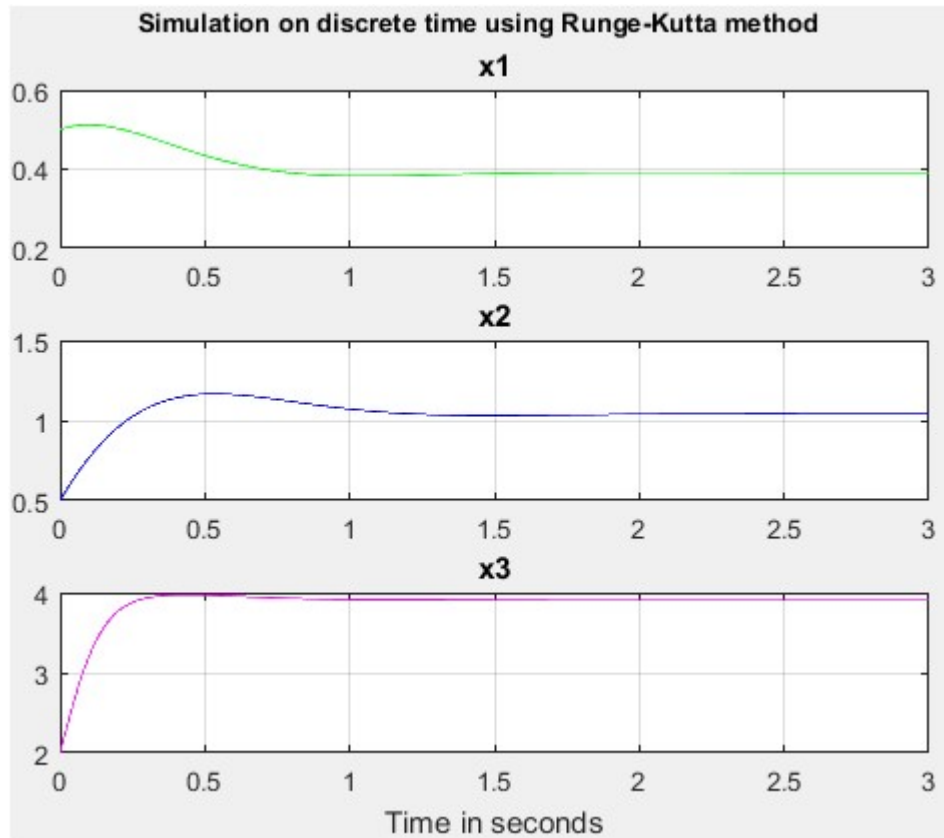


Figure 5.5: Simulation of states in discrete time

The error between the simulation using ode45 and Runge-Kutta method is shown in Figure 5.7.

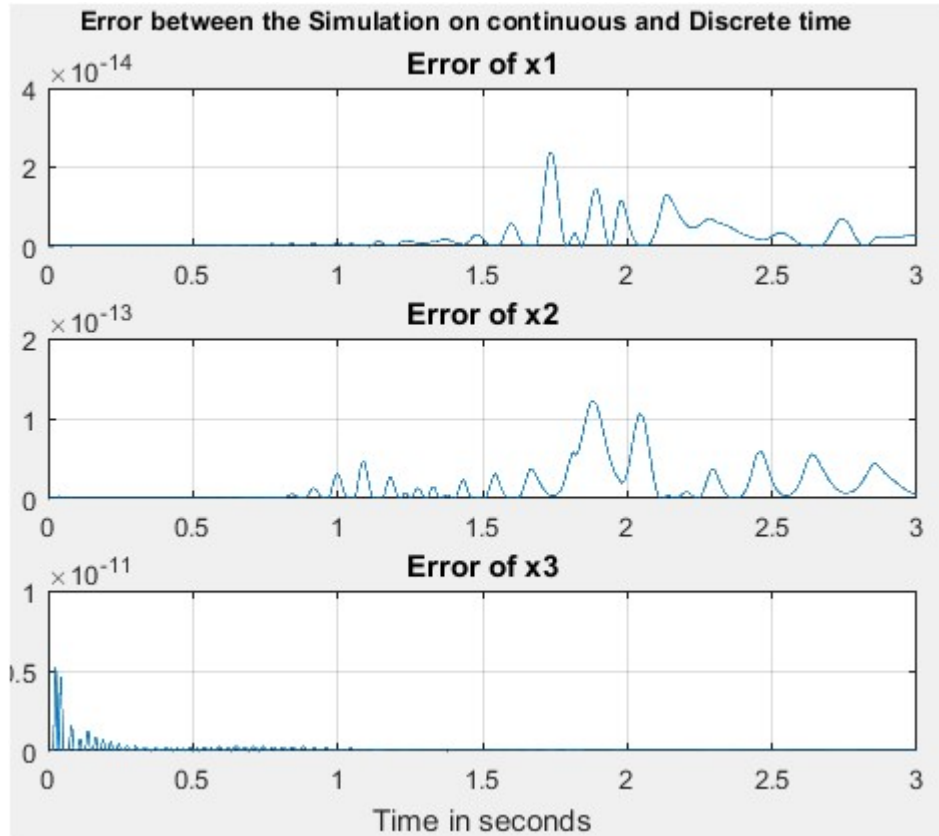


Figure 5.6: The error between the simulation using ode45 and Runge-Kutta method.

### 5.3 Genetic Algorithm results and simulation:

Using genetic algorithm, at first initial population of the chromosome is created. The created program allows user to specify the initial population size. As an example, for initial population size 98, a random population for each gene of a chromosome is generated within range. Thus for four gene of a chromosome, the total population size becomes a matrix of size  $98 \times 4$ . The simulation of the initial population is shown in Figure 5.8 for each parameter separately.

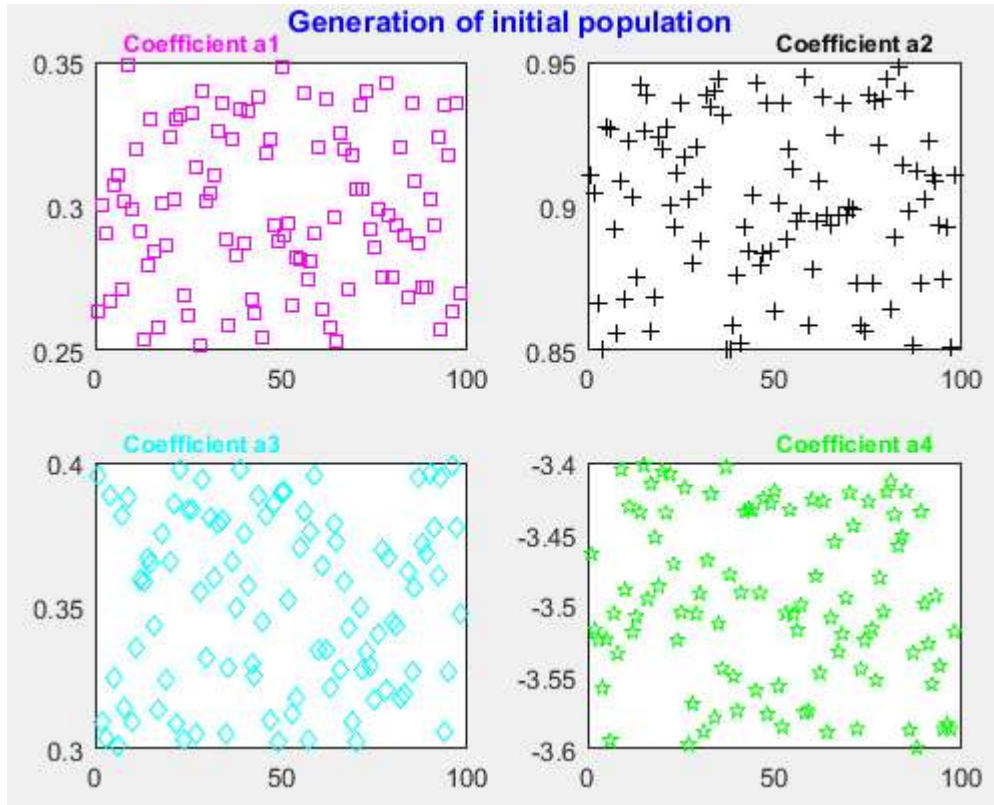


Figure 5.7: Generation of initial population

The cost of each chromosome is evaluated based on the cost function. The program for evaluating the cost is given in Appendix 3.

The mean cost and minimum cost of the initial population is 0.0214 and 0.0194 respectively.

The cost of each chromosome of initial population is also represented in Figure 5.9.

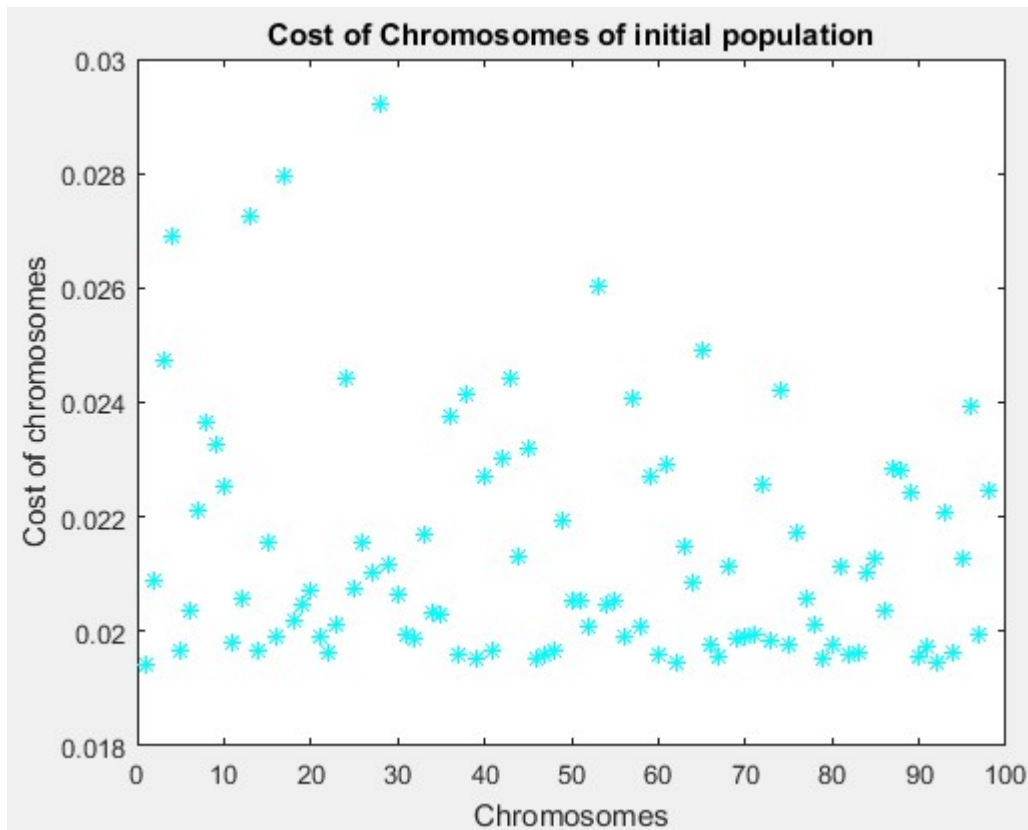


Figure 5.8: Cost of chromosomes of initial population.

As the cost of each chromosome is evaluated, they are sorted from lower cost to higher cost.

The best chromosomes are selected by pairing (Appendix 3), mating (Appendix 4) and mutation (Appendix 5) according to the process described in chapter 3. These steps are repeated for 20 iterations. Finally at the end of 20<sup>th</sup> generation, the best chromosome produces cost of 0.0194. The top chromosome and its associated cost in Matlab® output is represented in Figure 5.10.

```
TopChrom =
    0.2637    0.9107    0.3958   -3.4638

ans =
    0.0194
```

Figure 5.9: Top chromosome and cost



Thus the best coefficients of compressor characteristic are:

$$a_1 = 0.2637$$

$$a_2 = 0.9107$$

$$a_3 = 0.3958$$

$$a_4 = -3.4638$$

#### 5.4 Fuzzy logic Controller:

The fuzzy logic controller is designed to control the mass flow rate in order to keep the operating point within the stall margin. The controller is designed to manipulate the throttle opening to reduce or increase the mass flow rate. It is found from the compressor characteristic simulation that the throttle coefficient 0.21 produces mass flow rate 0.3270 and pressure rise 0.5093. The controller is designed to keep the operating point at (0.3270, 0.5093). The simulation of the output is represented in Figure 5.10 and 5.11.

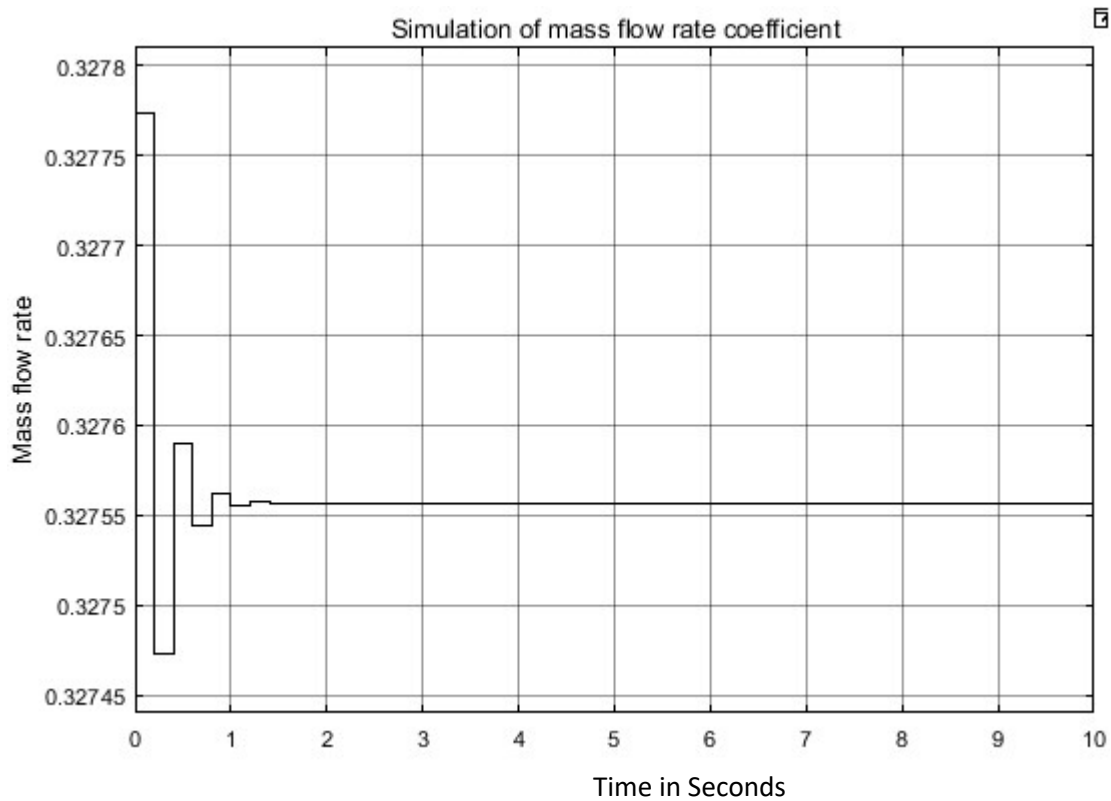


Figure 5.10: Simulation of mass flow ( $U_e$ )

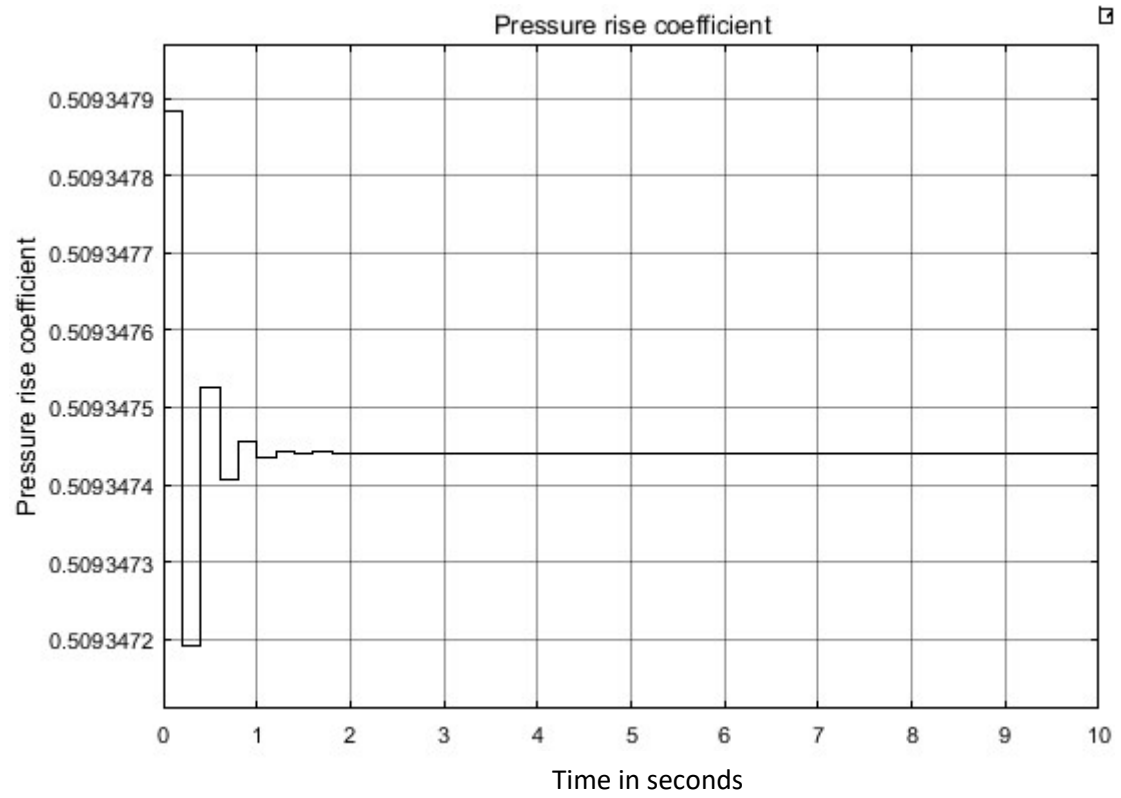


Figure 5.11: Simulation of pressure rise coefficient ( $P_e$ )

## **Chapter 6**

### **Conclusion and future work**

#### **6.1 Conclusion:**

In this thesis, the parameters of an axial flow compressor characteristic is optimized. The optimized parameters are then used to design a fuzzy logic controller to keep the operating point within the stall margin. At first, different methods are investigated in order to solve the Moore-Greitzer model in discrete time. Discretization of the MG model was not possible through  $z$ -transform because the system is nonlinear. Euler's method is implemented and satisfactory results are found. As Euler's method only considers the first and last point within a step, this method does not always provide precise results. From inspiration of Euler's method success and also considering its shortcomings, the Runge-Kutta 4<sup>th</sup> order method is used for solving and simulating the nonlinear system. The simulated compressor characteristics and three states of the MG model are used as the desired or the true simulation.

A cost function is developed in order to optimize the parameters of the compressor characteristics using genetic algorithm. The cost function is the weighted sum of the error between the true simulated data and data generated by the chromosomes. The initial population of chromosomes is generated and the cost of each chromosome is evaluated. The best chromosomes were selected for pairing using roulette wheel method. After selecting the Mom and Dad chromosomes, mating and mutation were carried out. This process is repeated for 10 generation and finally the best chromosome is found. The four genes of this best chromosome is the four parameters of the compressor characteristics.

A Mamdani fuzzy logic controller is designed in such a way that changes the throttle opening in response to the mass flow change. The mass flow rate is measured and is given feedback to

the reference input value. The error between this two values is used as the input to the fuzzy logic controller. The simulation of the mass flow rate and pressure rise is observed. The mass flow rate and pressure rise became stable at the desired operating point.

## 6.2 Future work:

The future work on this research topic includes the implementation of the designed fuzzy logic controller in the real compressor at the Chinese Academy of Science. The controller can be implemented in a Raspberry PI and also the programming can be done in Python according to the internship at Idaho National Laboratory [Appendix D]. The Greitzer's  $B$  parameter is considered constant in this research i.e. the speed of the compressor is considered constant. The research can be extended for a variable speed compressor. The Greitzer's  $B$  parameter is proportional to the speed of the compressor. Thus considering a variable speed compressor, gives the flexibility of changing the  $B$  parameter to control stall and surge. As Moore-Greitzer [18] concluded that low values of Greitzer's  $B$  parameter facilitates rotating stall and high value of  $B$  facilitates surge, considering  $B$  parameter as variable can be very useful while designing a controller.

At first the characteristic coefficients of the compressor are assumed. More research can be conducted to see the effect of these coefficients on the controller's performance. As an example, the second coefficient of the characteristic curve is considered as zero in the assumption of the nature of the characteristic curve of the MG model. This value makes the characteristic curve to go up just after crossing the Y-axis. In this paper, the second coefficient is assumed to be 0.90. The performance of the controller can be investigated for these two different values of the second coefficient.

## References:

- [1] Moore, F. K., and Greitzer E. M., 1986, "A Theory of Post-Stall Transients in Axial Compression Systems: Part I-Development of Equations," *Journal of Engineering for Gas Turbines and Power*, 108:68-76, 1986.
- [2] Emmons, H.W., C.E. Pearson and H.P. Grant (1955). Compressor surge and stall propagation. *Transactions of the ASME* 77, 455-469.
- [3] Iura, T., and Rannie, W. D., 1953, "Observations of Propagating Stall in Axial-Flow Compressor," California Institute of Technology, Pasadena, CA.
- [4] Marble, F. E., 1953, "Propagation of Stall in a Compressor Blade Row," *Journal of the Aeronautical Sciences*, 22(8), pp. 541-554.
- [5] Greitzer, E. M., 1976, "Surge and Rotating Stall in Axial Flow Compressors, Part I: Theoretical Compression System Model," *ASME J. Eng. for Power*, 98(2), pp. 190-198.
- [6] Greitzer, E. M., 1976, "Surge and Rotating Stall in Axial Flow Compressors, Part II: Experimental Results and Comparison with Theory," *ASME J. Eng. for Power*, 98(2), pp. 199-217.
- [7] Moore, F. K., 1984, "A Theory of Rotating Stall of Multistage Axial Compressor: Part I, II, and III," *Journal of Engineering for Gas Turbines and Power*, 106(2), pp.313-336.
- [8] Haynes, J. M., Hendricks, G. J., and Epstein, A. H., 1994, "Active Stabilization of Rotating Stall in a Three-Stage Axial Compressor," *ASME Journal of Turbomachinery*, 116, pp. 226-239.
- [9] Paduano, J. D. e. a., 1994, "Modeling for Control of Rotating Stall," *Automatica*, 30(9), pp. 1357-1373.
- [10] Hendricks, G. J., Sabnis, J. S., and Feulner, M. R., 1997, "Analysis of Instability

- Inception in High-Speed Multistage Axial-Flow Compressors," ASME Journal of Turbomachinery, 119, pp. 714-722.
- [11] Weigl, H. J., Paduano, J. D., and Frechette, L. G., 1998, "Active Stabilization of Rotating Stall and Surge in a Transonic Single-Stage Axial Compressor," ASME Journal of Turbomachinery, 120(4), pp. 625-636.
  - [12] D'Andrea, R., Behnken, R. L., and Murray, R. M., 1997, "Active Control of an Axial Flow Compressor via Pulsed Air Injection," ASME Journal of Turbomachinery, 119, pp. 742-752.
  - [13] Gravdahl, J.T., Egeland, O., "A Moore-Greitzer Model with spool dynamics," Proceeding of the 36<sup>th</sup> IEEE Conference on Decision and Control, pp. 0191-2216, 1997.
  - [14] Shu Lin, Chunjie Yang, Ping Wu, Zhihuan Song, "Fuzzy logic surge control in variable speed axial compressors," 10<sup>th</sup> IEEE International Conference on Control and Automation (IICA), pp. 1948-3449, 2013.
  - [15] Gravdahl, J. T., Egeland, O., "Compressor surge control using closed-coupled valve and backstepping," Proceeding of the 1997 American Control Conference, 1997-June.
  - [16] H.W. Emmons, C.E. Pearson, H.P. Grant, "Compressor surge and stall propagation," Trans. ASME, 79, pp. 455-469. 1955.
  - [17] [http://lcas.otaski.org/index.php?title=File:Compressor\\_blades.png](http://lcas.otaski.org/index.php?title=File:Compressor_blades.png).Date: 4/4/2018
  - [18] Moore, F. K., and Greitzer E. M., 1986, "A Theory of Post-Stall Transients in Axial Compression Systems: Part II-application," Journal of Engineering for Gas Turbines and Power, 108:231-239, 1986.



## **APPENDIX A – Matlab <sup>TM</sup> files for simulation of Moore-Greitzer model**

### **A-1: Calculation of the operating point for a particular throttle coefficient**

```
%% Computation of operating point for a particular throttle coefficient
%% Polynomial equations
%
% <include>polynomial.m</include>
%
function [Pe,Ue]= solvePeUe (gamma)
% clear;
% clc;
%% Coefficient and paramters
global a0 a1 a2 a3
a0 = 0.30;
a1 = 0.90;
a2 = .35;
a3 = -3.5;
gammalow = 0.5;
gammahigh = 2;
N = 96;%length(CHROMOSOMES);
steps = (gammahigh-gammalow)/(N-1);
solution = zeros(2,1);
gamma = 0.21;
%% Solving m file polynomial of Characteristic Curve
fun = @(y)polynomial(y,gamma);
x0 = [1;1];
solution = fsolve(fun,x0);
Pe = solution(1);
Ue = solution(2);
% Display result
X = ['The mass flow rate is ', num2str(Ue)];
Y = ['The pressure rise coefficient is ', num2str(Pe)];
disp(X);
disp(Y);
```

## **A-2: Characteristic equation and throttle equation**

```
% Polynomial and quadratic equations

function F = polynomial(y,gamma)
global a0 a1 a2 a3
F1 = -y(1)+a0+a1.*y(2)+a2.*(y(2)^2)+a3.*(y(2)^3);
F2 = -y(2)+sqrt(gamma.*y(1));
F = [F1;F2];
End
```

### **A-3: Simulation of compressor characteristic curve**

```
%% Simulation of characteristic curve for Moore-Grietzner Model
%% Polynomial equations
%
% <include>polynomial.m</include>
%
function [Pe,Ue]= PeUesolve()
% clear;
% clc;
%% Coefficient and paramters
global a0 a1 a2 a3
a0 = 0.30;
a1 = 0.90;
a2 = .35;
a3 = -3.5;
gammalow = 0.21;
gammahigh = 2;
N = 48;%length(CHROMOSOMES);
steps = (gammahigh-gammalow)/(N-1);
%N = floor((gammahigh-gammalow)/steps)+1;
gamma = gammalow:steps:gammahigh;
% steps = 0.010;
% N = floor((gammahigh-gammalow)/steps)+1;
% gamma = gammalow:steps:gammahigh;
solution = zeros(2,N);
Pe = zeros(1,N);
Ue = zeros(1,N);
%% Simulation of Characteristic Curve
for i = 1:N;
fun = @(y)polynomial(y,gamma(i));
x0 = [1;1];
options = optimoptions('fsolve','Display','Off');
solution = fsolve(fun,x0,options);
Pe(i) = solution(1);
Ue(i) = solution(2);
% disp('Pe = ');
% disp(Pe);
% disp('Ue = ');
% disp(Ue);
end

figure(1)
plot(Ue,Pe)
title('Characteristic Curve');
xlabel('----Ue----','color','r','Fontweight','bold','FontSize',16);
ylabel('----Pe----','color','r','Fontweight','bold','FontSize',16);
grid on
save('data','Pe','Ue')
xlswrite('The characteristic data.xls',gamma,1,'A2')
xlswrite('The characteristic data.xls',Ue,1,'B2')
xlswrite('The characteristic data.xls',Pe,1,'C2')
```

#### **A-4: Moore-Greitzer Model**

```
function xdot = mgmodel(t,y)
global k gamma beta Pe Ue a0 a1 a2 a3
a0 = .30;
a1 = .90;
a2 = .35;
a3 = -3.5;

xdot = [(-y(2)+Pe)+ [a0 + a1*(y(1)+Ue-1) + a2*(y(1)+Ue-1).^2 + a3*(y(1)+Ue-1).^3] - (3/4)*(y(1)+Ue-1)*y(3);
        (1/beta^2)* ((y(1)+Ue)-sqrt(gamma*(y(2)+Pe))) ;
        k*(4*(1-(y(1)+Ue-1).^2)*y(3)-y(3).^2)];
% gamma + gamma*.10*sin(5*t)];
% xdot = [xdot1;xdot2;xdot3];
end
```

### **A-5: Simulation of MG model using ode45**

```
%% Simulation of MG Model on continuous time
function [t,z]= continuous_time()
clear;
clc;
gamma=1.8;
y0 = [0.5;0.5;2];
tfinal = 3;
tspan = 0:0.00010:3;%[1,tfinal];
%tspan = [0,tfinal];
opts = odeset('reltol',1e-6);
[t,z]=ode45(@mgmodel,tspan,y0,opts);
figure(3);

subplot(3,1,1);
plot(t,z(:,1),'g');
title('x1');
grid on;

subplot(3,1,2);
plot(t,z(:,2),'b');
title('x2');
grid on;

subplot(3,1,3);
plot(t,z(:,3),'m');
title('x3');
grid on;

ha = axes('Position',[0 0 1 1],'Xlim',[0 1],'Ylim',[0
1], 'Box','off','Visible','off','Units','normalized', 'clipping' , 'off');
text(0.5, 1,'\bf Simulation on continuous
time','HorizontalAlignment','center','VerticalAlignment','top');

end
```

## **A-6: Simulation of MG model using the Runge-Kutta Method**

```
%% Simulation of states of Moore-Greitzer Model on Discrete time
% Pe and Ue are solved. The values of Pe and Ue are used to simulate x1,x2
% and x3 for a value pf gamma within range.
```

```
function y = discrete_time()
%% Initial condition
global beta k gamma Pe Ue
clc;
clear;
%% Parameter known from compressor geometry
H = .25;
W = .25;
S = H/W;
B = .15;
beta = 2*B*S; % Stagger angle of the rotor blade

R = 250; % Radius of the duct in the compressor
lid = 2.9; % Length of the inlet duct
ratio = R/lid;
led = 2.37; % Length of the exit duct
timelag_r = .8; % time lag for the rotor
timelag_s = .48; %time lag for the stator
timelag = timelag_r + timelag_s; % total time lag
k1 = (exp(ratio)+exp(-ratio))/(exp(ratio)-exp(-ratio));
k1tilda = 2/(exp(ratio)-exp(-ratio));
k = 3/4*(timelag+led+k1+k1tilda);

%% Generating a random value of gamma within the range of (gammalow,
gammahigh)
gammalow = 0.5;
gammahigh = 2.0;
%gamma = (gammahigh-gammalow).*rand(1)+gammalow;
gamma = 1.8;

%% Generating value of Pe and Ue from the random value of gamma
poly = @(y)polynomial(y,gamma);
x0 = [1;1];
solv = fsolve(poly,x0);
Pe = solv(1);
Ue = solv(2);
%[Pe, Ue] = PeUesolve();
y1 = [0.5;0.5;2];
%% Time and step size
t0 = 0;
Tstop = 3;
Ts = 0.00010;
T = Tstop-t0;
N = floor(T/Ts)+1;
t = zeros(N,1);
y = zeros(length(y1),N);
y(:,1)= y1;
t(1) =1;
h = Ts;

%% for loop

for i=1:N-1
    k1 = mgmodel (t(i) , y(:,i) );
```

```

        k2 = mgmodel(t(i) + 0.5*h, y(:,i) + 0.5*h*k1);
        k3 = mgmodel(t(i) + 0.5*h, y(:,i) + 0.5*h*k2);
        k4 = mgmodel(t(i) + h, y(:,i) + h*k3);
        y(:,i+1) = y(:,i) + (h/6)*(k1+2*k2+2*k3+k4);
    end
    i = 0:Ts:Tstop;
    figure(7)
    subplot(3,1,1);
    plot(i,y(1,:), 'g');
    title('x1');
    grid on;

    subplot(3,1,2);
    plot(i,y(2,:), 'b');
    title('x2');
    grid on;

    subplot(3,1,3);
    plot(i,y(3,:), 'm');
    title('x3');
    grid on;

    ha = axes('Position',[0 0 1 1], 'Xlim',[0 1], 'Ylim',[0
1], 'Box','off', 'Visible','off', 'Units','normalized', 'clipping' , 'off');
    text(0.5, 1, '\bf Simulation on discrete
time', 'HorizontalAlignment','center', 'VerticalAlignment', 'top');

end

```

## **A-7: Comparison between the simulation of Moore-Greitzer Model on discrete time and continuous time.**

```
%% Comparison between the simulation of Moore_Greitzer Model on discrete
time and continuous time.
% The Moore_Greitzer Model is simulated on discrete time using Runge-Kutta
method and on continuous time using ODE45.
%% Moore-Greitzer Model
% Moore-Greitzer Model is represented by
%
% <include>mgmodel.m</include>
%
%% Code for simulation of Moore-Greitzer Model on discrete time
% Simulation using RangeKutta method
%
% <include>discrete_time.m</include>
%
%% Simulation of Moore-Greitzer Model model on discrete time
y= discrete_time();

%% Simulation on continuous time using ODE45
[t,z]= continuous_time();

%% Codes for simulation on continuous time
%
% <include>continuous_time.m</include>
%
%% Mean square error
errorx1 = z(:,1)-y(1,:);
errorx2 = z(:,2)-y(2,:);
errorx3 = z(:,3)-y(3,:);
sq_errorx1 = errorx1.^2;
sq_errorx2 = errorx2.^2;
sq_errorx3 = errorx3.^2;

figure(8)
subplot(3,1,1);
plot(t,sq_errorx1);
title('Error of x1')
grid on
subplot(3,1,2);
plot(t,sq_errorx2);
title('Error of x2')
grid on
subplot(3,1,3);
plot(t,sq_errorx3);
title('Error of x3')
grid on
ha = axes('Position',[0 0 1 1],'Xlim',[0 1],'Ylim',[0
1],'Box','off','Visible','off','Units','normalized','clipping','off');
text(0.5, 1,'\bf Error between the Simulation on continuous and Discrete
time','HorizontalAlignment','center','VerticalAlignment','top');
```



## **APPENDIX B – Matlab <sup>TM</sup> files for simulation of Lorenz model**

### **B-1: Lorenz model**

```
function ydot = lorenz(t,y)
% lorenz attractor rhs
global sigma rho beta
ydot=zeros(3,1);
ydot(1) = -beta*y(1) +y(2).*y(3);
ydot(2) = -sigma*(y(2) - y(3));
ydot(3) = -y(2).*y(1) + rho*y(2) -y(3);
return
```

## **B-2: Simulation of Lorenz model using ode45**

```
function [t,z]= lorenz_driver()
clear;
clc;
% lorenz driver
global sigma rho beta
sigma = 10;
rho = 28;
beta = 8/3;
zeta = sqrt(beta*(rho-1));
yc = [rho-1;zeta;zeta];

y0 = yc + [0;0;3];
tfinal = 20;
tspan = 1:0.0010:20;%[1,tfinal];
%tspan = [1,tfinal];
opts = odeset('reltol',1e-6);
[t,z]=ode45(@lorenz,tspan,y0,opts);
% y0n = y0+ .001*randn(size(y0));
% [tn,yn]=ode45(@lorenz,tspan,y0n,opts);
% figure(1)
% plot3(z(:,1),z(:,2),z(:,3),'r')
figure(2)
plot(t,z(:,1),'c',t,z(:,2),'g',t,z(:,3),'m')
title('Simulation on continuous time using ODE45');
legend('z(:,1)', 'z(:,2)', 'z(:,3)');
%hold on
End
```

### **B-3: Simulation of Lorenz model using the Runge-Kutta Method**

```
% Simulation on discrete time using Runge-Kutta Method
function y = RK4()
clc;
clear
%% Initial condition
global sigma rho beta
sigma = 10;
rho = 28;
beta = 8/3;
zeta = sqrt(beta*(rho-1));
yc = [rho-1;zeta;zeta];
y1 = yc + [0;0;3];
%% Time and step size
t0 = 1;
Tstop = 20;
Ts = 0.0010;
T = Tstop-t0;
%%
N = floor(T/Ts)+1;
t = zeros(N,1);
y = zeros(length(y1),N);
y(:,1)= y1;
t(1) =1;
h = Ts;

%% for loop

for i=1:N-1
    k1 = lorenz (t(i) , y(:,i) );
    k2 = lorenz(t(i) + 0.5*h, y(:,i) + 0.5*h*k1);
    k3 = lorenz(t(i) + 0.5*h, y(:,i) + 0.5*h*k2);
    k4 = lorenz(t(i) + h, y(:,i) + h*k3);
    y(:,i+1) = y(:,i) + (h/6)*(k1+2*k2+2*k3+k4);
end
i = 1:Ts:Tstop;
% figure(5)
% plot3(y(1,:), y(2,:),y(3,:))
figure(6)
plot(i,y(1,:), 'g', i, y(2,:), 'y', i, y(3,:), 'k');
title('Simulation on discrete time using Runge-Kutta method');
legend('y(1,:)', 'y(2,:)', 'y(3,:)');

end
```

#### **B-4: Comparison between the simulation of lorenz system using ode45 and Runge-Kutta Method**

```
%% Comparison between the simulation of lorenz system on discrete time and
continuous time.
% Lorenze system consists of three differential equations. It is
% non-linear, deterministic and three dimensional.
% The system is simulated on discrete time using Runge-Kutta method and on
continuous time using ODE45.
%% Lorenz system
% The differential equation of lorenz system
%
% <include>lorenz.m</include>
%
%% Code for simulation of lorenz model on discrete time
% Simulation using RangeKutta method
%
% <include>RK4.m</include>
%
%% Simulation of lorenz model on discrete time
y= RK4();

%% Simulation on continuous time using ODE45
[t,z]= lorenz_driver();

%% Codes for simulation on continuous time
%
% <include>lorenz_driver.m</include>
%
%% Mean square error
error = z-y';
sqerror = error.^2;
figure(8)
plot(t,sqerror)
```

## **APPENDIX C – Matlab <sup>TM</sup> files for optimization of parameters of the compressor characteristic using Genetic Algorithm**

### **C-1: Genetic Algorithm**

```
% ***** GPid_con.m *****  
%  
%                               Version 1.0  
%                               June 25, 2000  
%                               Dr. Marco P. Schoen  
%                               Modification: Md Fahdul Wahab Chowdhury  
%  
% _____  
  
% Toolbox for genetic programming. Code for continues  
% genetic algorithm.  
%clear;clc  
  
% Define Variables:  
maxiterations=input('Maximum Number of iterations: ');  
ipopsize=input('Population Size of Generation 0: ');  
popsize=input('Population Size for Generations 1 - end: ');  
%popsize=popsize*ipopsize;  
keep=input('Number of Chromosomes kept for mating: ');  
%keep=keep*popsize;  
pars=input('Total Number of parameters in a chromosome: ');  
mutaterate=input('Mutation rate: ');  
% hi=input('High end of parameter value: '); %for all parameters  
% lo=input('Low end of parameter value: '); %for all parameters  
op=1;%input('Probability options: 1. Top-Buttom 2.Random 3. Weigh-Rand. :  
'');  
  
% Create the initial population, evaluate costs, and sort  
CHROMOSOMES = zeros(ipopsize,pars);  
hi = [0.35 0.95 0.40 -3.40];  
lo = [0.25 0.85 0.30 -3.60];  
for i = 1:pars  
    CHROMOSOMES(:,i)= (hi(i)-lo(i))*(rand(ipopsize,1))+lo(i);  
    xlswrite('chromosomes.xls',CHROMOSOMES,1,'A2');  
    figure(9)  
    subplot(2,2,1);  
    plot(CHROMOSOMES(:,1),'ms');  
    title('Coefficient  
a1','FontSize',9,'color','m','HorizontalAlignment','Right')  
  
    subplot(2,2,2);  
    plot(CHROMOSOMES(:,2),'k+');  
    title('Coefficient  
a2','FontSize',9,'color','k','HorizontalAlignment','Left')  
  
    subplot(2,2,3);  
    plot(CHROMOSOMES(:,3),'cd');  
    title('Coefficient  
a3','FontSize',9,'color','c','HorizontalAlignment','Right')  
  
    subplot(2,2,4);  
    plot(CHROMOSOMES(:,4),'gp');  
    title('Coefficient  
a4','FontSize',9,'color','g','HorizontalAlignment','Left')
```

```

    ha = axes('Position',[0 0 1 1],'Xlim',[0 1],'Ylim',[0
1], 'Box','on','Visible','off','Units','normalized','clipping','off');
    text(0.5, 1, '\bf Generation of initial population
','FontSize',11,'Color','b','HorizontalAlignment','center','VerticalAlignme
nt','top');

end

cost_initial=costfunction(CHROMOSOMES);
xlswrite('chromosomes',cost_initial,1,'E2');
figure(2)
plot(cost_initial,'c*');
xlabel('Chromosomes');
ylabel('Cost of chromosomes');
title('Cost of Chromosomes of initial population');
%CHROMOSOMES = chromosomes();

% CHROMOSOMES will be a matrix of random numbers within hi - lo

% Loop:
gen=0;quit=0;
h = waitbar(0,'Please wait...');
while (gen<maxiterations & (~quit))
    gen=gen+1;
    cost=costfunction(CHROMOSOMES);
    New=[cost,CHROMOSOMES];
    xlswrite('chromosomes',New,2,'A2');
    New2=sortrows(New,[1]); cost=New2(:,1);CHROMOSOMES=New2(:,2:pars+1);
    mincost(gen)=min(cost);
    meancost(gen)=mean(cost);
    stdcost(gen)=std(cost);
    % Pairing,Mating, and Mutation
    %CHROMOSOMES=New2(1:popsi,2:3);cost=New2(1:popsi,1);
    [Mom,Dad]=pairing(CHROMOSOMES,cost,keep,popsi,op);
    CHROMOSOMES=matecon(Mom,Dad,CHROMOSOMES,keep,popsi,pars);
    CHROMOSOMES=mutatecon(CHROMOSOMES,mutaterate,popsi,pars,hi,lo);
    % Check for Conversions
    % if mincost(gen)< ... and/or meancost(gen) < ... and or stdcost(gen)<
    % ... quit=1
    waitbar(gen/maxiterations);
    figure(10)
plot(gen,meancost(gen),'b*');
title('Mean Cost');
hold on
end;
close(h);

TopChrom=CHROMOSOMES(1,:);
min(cost)
figure(3)
plot(cost,'c*');
xlabel('Chromosomes');
ylabel('Cost of chromosomes');
title('Cost of Chromosomes of last generation');

```

## **C-2: Cost function**

```
function cost = costfunction(CHROMOSOMES)
%% calling function to evalauate the desired curve
[Pe,Ue]= PeUesolve();
desired_curve = [Pe Ue];
%% Calling function to simulate curve using the random value of Chromosome
within high and low value
[Pm, ma] =solvece(CHROMOSOMES);
simulated_curve = [Pm, ma];
%% Simulation of the states of MG Model on discrete time using Runge-Kutta
Method
error_state = state_error(CHROMOSOMES);
x1_error = error_state(1,:);
x2_error = error_state(2,:);
x3_error = error_state(3,:);

%% Pre-allocating the size of error
[row,col] = size(simulated_curve);
error = zeros(size(simulated_curve));
mean_sq_error = zeros(row,1);
cost=zeros(row,1);
%error = simulated_curve- desired_curve;
%% Creating loop to evaluate the error
for i = 1:row
    error(i,:) = simulated_curve(i,:)- desired_curve;
    error_sq = error.^2;
    mean_sq_error(i,:) = sum(error_sq(i,:))/length(error_sq);
    cost(i,:) = 10*mean_sq_error(i,:)+ x1_error(i,:)+
x2_error(i,:)+x3_error(i,:);
end
end
```

### **C-3: Simulation of characteristic curve from chromosomes**

```
%% Simulation of characteristic curve for Moore-Grietzner Model
%% Polynomial equations
%
% <include>polynomial.m</include>
%
function [Pm,Um]= solvece(CHROMOSOMES)
% clear;
% clc;
%% Coefficient and paramters
a0 = CHROMOSOMES(:,1);
a1 = CHROMOSOMES(:,2);
a2 = CHROMOSOMES(:,3);
a3 = CHROMOSOMES(:,4);
[row, col] = size(CHROMOSOMES);
gammalow = 0.5;
gammahigh = 2.0;
%steps = 0.010;
N = 48;%length(CHROMOSOMES);
steps = (gammahigh-gammalow)/(N-1);
%N = floor((gammahigh-gammalow)/steps)+1;
gamma = gammalow:steps:gammahigh;
% N = floor((gammahigh-gammalow)/steps)+1;
% gamma = gammalow:steps:gammahigh;
%solution = zeros(2,N);
Pm = zeros(row,N);
Um = zeros(row,N);
%% Simulation of Characteristic Curve
for i = 1:N;
    for j = 1:row
        fun = @(y)characteristic(y,gamma(i),CHROMOSOMES(j,:));
        x0 = [1;1];
        options = optimoptions('fsolve','Display','off');
        solution = fsolve(fun,x0,options);
        Pm(j,i) = solution(1);
        Um(j,i) = solution(2);
    end
    desired_curve = [Pm, Um];
    z = size(desired_curve);
end
end
```



## **C-4: Simulation of Moore-Greitzer model from chromosomes**

```
function [y] = RK4_chrom(CHROMOSOMES)
%% Initial condition
global beta k gamma Pe Ue
% global sigma rho
% sigma = 10;
% rho = 28;
%%% Parameter known from compressor geometry
H = .25;
W = .25;
S = H/W;
B = .15;
beta = 2*B*S; % Stagger angle of the rotor blade

R = 250; % Radius of the duct in the compressor
lid = 2.9; % Length of the inlet duct
ratio = R/lid;
led = 2.37; % Length of the exit duct
timelag_r = .8; % time lag for the rotor
timelag_s = .48; %time lag for the stator
timelag = timelag_r + timelag_s; % total time lag
k1 = (exp(ratio)+exp(-ratio))/(exp(ratio)-exp(-ratio));
k1tilda = 2/(exp(ratio)-exp(-ratio));
k = 3/4*(timelag+led+k1+k1tilda);

%% Generating a random value of gamma within the range of (gammalow,
gammahigh)
gammalow = 1.8-1.8*0.3;
gammahigh = 1.8+1.8*0.3;
gamma = 0.5;%(gammahigh-gammalow).*rand(1)+gammalow;

%% Generating value of Pe and Ue from the random value of gamma
poly = @(y)polynomial(y,gamma);
x0 = [1;1];
solv = fsolve(poly,x0);
Pe = solv(1);
Ue = solv(2);
y1 = [0.5;0.5;2];
%% Time and step size
t0 = 1;
Tstop = 3;
Ts = 0.0010;
T = Tstop-t0;
N = floor(Tstop/Ts)+1;
t = zeros(N,1);
y = zeros(length(y1),N);
y(:,1)= y1;
t(1) =1;
h = Ts;

%% for loop

a0 = CHROMOSOMES(:,1);
a1 = CHROMOSOMES(:,2);
a2 = CHROMOSOMES(:,3);
a3 = CHROMOSOMES(:,4);
[row,col] = size(CHROMOSOMES);
for j = 1:row
for i=1:N-1
```

```

        k1 = mgmodel_chrom (t(i)          , y(:,i) ,CHROMOSOMES(j,:))
);
        k2 = mgmodel_chrom(t(i) + 0.5*h, y(:,i) + 0.5*h*k1,
CHROMOSOMES(j,:));
        k3 = mgmodel_chrom(t(i) + 0.5*h, y(:,i) + 0.5*h*k2,
CHROMOSOMES(j,:));
        k4 = mgmodel_chrom(t(i) +      h, y(:,i) +
h*k3,CHROMOSOMES(j,:));
        y(:,i+1,j) = y(:,i) + (h/6)*(k1+2*k2+2*k3+k4);
end
end
i = 0:Ts:Tstop;
figure(7)
subplot(3,1,1);
plot(i,y(1,:,1), 'g');
title('x1');
grid on;

subplot(3,1,2);
plot(i,y(2,:,2), 'b');
title('x2');
grid on;

subplot(3,1,3);
plot(i,y(3,:,3), 'm');
title('x3');
grid on;

```

### **C-5: Computing error between true simulation of states and simulation from chromosomes.**

```
function mean_sq_error = state_error(CHROMOSOMES)
%% simulated states from chromosomes
x = RK4_chrom(CHROMOSOMES);
x1 = x(1,:,:);
x2 = x(2,:,:);
x3 = x(3,:,:);

%% desired state model
y = RK4();
y1 = y(1,:);
y2 = y(2,:);
y3 = y(3,:);

%% finding error
[row,col,page] = size(x);
%mean_sq_error = zeros(row,1,page);
for i = 1:page
x1_error(1,:,i) = x(1,:,i)-y1;
x2_error(1,:,i) = x(2,:,i)-y2;
x3_error(1,:,i) = x(3,:,i)-y3;
% Error for state x1
error_sq_x1 = x1_error.^2;
mean_sq_error_x1(i) = sum(error_sq_x1(1,:,i))/length(error_sq_x1);
% Error for state x2
error_sq_x2 = x2_error.^2;
mean_sq_error_x2(i) = sum(error_sq_x2(1,:,i))/length(error_sq_x2);
% Error for state x3
error_sq_x3 = x3_error.^2;
mean_sq_error_x3(i) = sum(error_sq_x3(1,:,i))/length(error_sq_x3);
mean_sq_error = [mean_sq_error_x1; mean_sq_error_x2; mean_sq_error_x3];

end
```

## **C-6: Selection of Mom and Dad using roulette wheel method**

```
function [Mom,Dad]=pairing (CHROMOSOMES, cost, keep, popsize, op)
% Based on the probability option op, one of the following
% three pairing criterias is used:
% 1. Top-Down, 2. Random, 3. Weighted Random
% for the selection of the parents of the next generation

replacements=(popsize-keep)/2;denum=0;
if op==1
    for r=1:replacements
        denum=denum+r;
    end;
    for n=1:replacements
        probn(n)=n/denum;
    end;
end;
if op==2
    %need to write code
end;
if op==3
    %need to write code
end;

%Cumulative Probabilities
cum=0;odds=zeros(1,replacements);
for i=1:replacements
    cum=probn(i)+cum;
    odds(1,i)=cum;
end;
%Roll dice for Parents
pick1=rand(1,replacements); %vector of random # for Mom
pick2=rand(1,replacements); %vector of random # for Dad

Mom=zeros(1,replacements);Dad=Mom;
for i=1:replacements
    for j=2:replacements
        if (pick1(i)<odds(j) & pick1(i)>odds(j-1))
            Mom(i)=j;
        end;
        if Mom(i)==0
            Mom(i)=1;
        end;
        if (pick2(i)<odds(j) & pick2(i)>odds(j-1))
            Dad(i)=j;
        end;
        if Dad(i)==0
            Dad(i)=1;
        end;
    end;
end;
```

### **C-7: Mating**

```
function CHROMOSOMES=matecon(Mom,Dad,CHROMOSOMES,keep,popsize,pars)
% Code for continuous GP mating. Selects a crossover point
% ceil rounds to next higher integer. Row index contains
% first offspring, row index +1 contains second offspring
% Mom-vector containing row numbers of first parent
% Dad-vector containing row numbers of second parent

CHROMOSOMES%to test chromosomes on the screen during development phase
replace=(popsize-keep)/2;
for ic=1:replace
    alpha=ceil(rand*pars);i=2*(ic-1)+1;
    beta=rand(1);
    CHROMOSOMES(keep+i,alpha)=CHROMOSOMES(Mom(ic),alpha)-
    beta*(CHROMOSOMES(Mom(ic),alpha)-CHROMOSOMES(Dad(ic),alpha));

    CHROMOSOMES(keep+i+1,alpha)=CHROMOSOMES(Dad(ic),alpha)+beta*(CHROMOSOMES(Mo
    m(ic),alpha)-CHROMOSOMES(Dad(ic),alpha));

end;
```

## **C-8: Mutation**

```
Function CHROMOSOMES=mutatecon(CHROMOSOMES,mutaterate,popsize,pars,hi,lo)
% Inside a loop iterating over the number of mutations, a random
% parameter in the population is selected and replaced by a new
% random parameter

nmu=ceil(popsize*pars*mutaterate);
for i=1:nmu
    row=ceil((popsize-1)*rand)+1;
    col=ceil(pars*rand);
    if col == 1
        CHROMOSOMES(row,col)=(hi(1)-lo(1))*rand+lo(1);
    elseif col == 2
        CHROMOSOMES(row,col)=(hi(2)-lo(2))*rand+lo(2);
    else
        CHROMOSOMES(row,col)=(hi(3)-lo(3))*rand+lo(3);
    end
end;
end;
```

## **APPENDIX D**

### **Report on Internship at Idaho National Laboratory on summer, 2017**



---

## Report on Internship from May 22, 2017 to August 11, 2017

Submitted to

**Dr. Rob Hovsapien**

Manager - Power and Energy System

Idaho National Laboratory

Prepared by

Md Fahdul Wahab Chowdhury

Master of Science in Mechanical Engineering

Idaho State University, Pocatello, Idaho.



## **Abstract**

Research, tests and projects are implemented during the period of internship in Idaho National Laboratory (INL). Ongoing research on Micro Grid Management System (MGMS) and Plug-in Electric Vehicle (PEV) are studied. Several tests are carried out in collaboration with National Renewable Energy Laboratory (NREL) on MGMS. Real Time Digital Simulation (RTDS) platform is used to do a test like adjustment of electrolyzer power according to the faults simulation from INL grid model.

A project on installation of one hundred raspberry PIs is completed. The operating system on Raspberry PIs are installed, the networking of the Raspberry PIs was completed by making Ethernet cable and networking those to VILLAS node and Real Time Digital Simulation (RTDS). This project is going to be used to do some tests on Plug-in Electric Vehicles (PEV). Each Raspberry PI is capable of sending and receiving some data through VILLAS node. The VILLAS node then MUX and DEMUX the data and send them to RTDS and OPAL-RT. The RTDS and OPAL-RT are used for simulation. On the other hand a research on the optimization of the cost of an electrolyzer for a Hydrogen Re-fueling station is carried on. Sequential least squares programming algorithm (SLSQP) optimizer on python is implemented to do the optimization.

## **Acknowledgement**

This report is prepared as a part of an internship at Idaho National Laboratory. It has been big opportunity for me to engage and learn some cutting edge research of the world. I would like to specially thank Mr. Anudeep Medam and Mr. Rahul Kadavil for their continuous support during the internship program. I would also like to thank Dr. Manish Mohanpurkar and Dr. Fernando G. Dias for their guidance. From the beginning to the end, the person who has always been very supportive and friendly is Mr. Cliff Laughmiller. I would like to convey my gratitude and thanks to him. Most importantly, I would like to thank Dr. Rob Hosvagian for giving me the opportunity to work in his team.

It would not have been possible to work in INL as an intern without the continuous effort of Ms. Sandy Shea, Management Assistant at office of Research of Idaho State University. My sincere thanks and gratitude to her.

At last, I would like to thank my advisor Dr. Marco P. Schoen, Professor of Department of Mechanical Engineering, Idaho State University for supporting and encouraging me all the way.

## Table of Contents:

<b>Abstract</b>	<b>i</b>
<b>Acknowledgement</b>	<b>ii</b>
<b>List of Tables</b>	<b>v</b>
<b>List of Figures</b>	<b>v</b>
<b>Chapter 1: Optimization of an electrolyzer</b>	<b>1</b>
1.1 Introduction	1
1.2 Technical Description	1
1.3 Calculation	2
1.3.1 Electricity cost to produce hydrogen	2
1.3.2 Demand Calculation	3
1.4 Optimization	7
1.4.1 Cost optimization equation for electrolyzer	7
1.4.2 Optimization Result and Discussion	9
<b>Chapter 2: Installation of Raspberry PIs</b>	<b>11</b>
2.1 Introduction	11
2.2 Procedure	12
2.2.1 Installation of the operating system on the Raspberry PI	12
2.2.2 Set up the Raspberry PI	13
2.2.3 Assigning a static IP address	13
2.2.4 Arranging Raspberry PI on Shelves	13
2.2.5 Making Network Cable	14
2.2.6 Connecting all the Raspberry PI to the network	15
2.2.7 Testing by sending and receiving some data	17

2.3	Conclusion	18
	<b>REFERENCES</b>	<b>19</b>
	<b>APPENDIX: Python code for optimization</b>	<b>20</b>

## **List of Tables**

Table 1– Calorific Power of Different fuels (25°C and 1 atm)	2
Table 2– Energy density of different fuels	3
Table 3: Demand of hydrogen fuel on peak time	3
Table 4: Demand of hydrogen fuel on off-pick time	4
Table 5: Specification of Raspberry PI 3 model B	7

## **List of Figures**

Figure 1: Electrolysis Process	1
Figure 2: Optimization Result of Electrolyzer	6
Figure 3: Ethernet Cable(Order of Wires)	9
Figure 4: Arrangement of Raspberry PI on Shelf (1)	9
Figure 5: Arrangement of Raspberry PI on Shelf (2)	10
Figure 6: A front view of the Raspberry PIs	10
Figure 7: client.py	11
Figure 8: server.py	11

# Chapter 1: Optimization of an electrolyzer

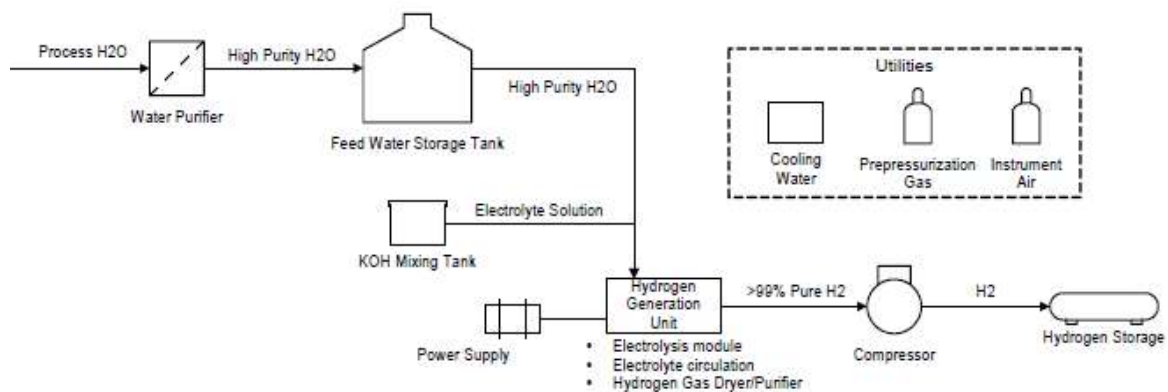
## 1.1 Introduction:

This report provides a summary on the study of the electrolytic hydrogen production technologies, and a cost optimization of the processes. A cost analysis is carried out to determine the effects of electricity price on hydrogen gas production through electrolysis. A specific electrolyzer model is selected and its energy requirement is used to calculate the demand of the electricity. In this study no capital, maintenance or operating costs are taken into consideration.

## 1.2 Technical Description:

Hydrogen is produced via electrolysis by passing electricity through two electrodes in water. The water molecule is split and produces oxygen at the anode and hydrogen at the cathode.

In Figure 1, a very general electrolysis process is depicted. The process and the equipment may vary according to the method used for electrolysis. As an example water purification and electrolytic solution may vary from system to system.



**Figure 1: Electrolysis Process.** <sup>[1]</sup>

In the Figure 1, a compressor and Hydrogen storage tank are added but in the analysis the produced hydrogen is directly fed to the vehicle. The utilities generally used for the process are electricity for electrolysis, cooling water etc.

### 1.3 Calculation:

#### 1.3.1 Electricity cost to produce hydrogen:

At present electrolyzer efficiencies, electricity costs should be between 4 and 5.5 cents per KWh to keep the hydrogen production cost lower than \$3.00/kg<sup>[1]</sup>.

In this study, Norsk Atmospheric Type No.5040 (5150 Amp DC) is considered as an electrolyzer.<sup>[1]</sup>

Electricity required to produce 1 kg hydrogen = 53.5 KWh/kg

Cost of electricity for KWh on peak time = \$0.055/KWh

Cost of electricity for KWh on off-peak time = \$0.04/KWh

Electricity cost for producing 1 kg of hydrogen on peak time,

$$CH2_{peak} = 53.5 \frac{KWh}{Kg} \times 0.055 \frac{\$}{Kwh}$$

$$= \$2.94/kg$$

Electricity cost for producing 1 kg of hydrogen on off-peak time,

$$CH2_{offpeak} = 53.5 \frac{KWh}{Kg} \times 0.04 \frac{\$}{Kwh}$$

$$= \$2.14/kg$$

### 1.3.2 Demand Calculation <sup>[1]</sup>:

The efficiency of fuel cell is almost twice than the efficiency of internal combustion engine as can be seen from Table 1. That's why Fuel cell vehicle can go twice the distance of the gasoline vehicle can go by 1 gallon <sup>[2]</sup>.

**Table 1– Calorific Power of Different fuels (25°C and 1 atm)**

Fuel	HHV	LHV
Hydrogen	141.86 MJ/kg	119.93 MJ/kg
Methane	55.53 MJ/kg	50.02 MJ/kg
Propane	50.36 MJ/kg	45.60 MJ/kg
Gasoline	47.50 MJ/kg	44.50 MJ/kg
Gas-oil	44.80 MJ/kg	42.50 MJ/kg
Methanol	19.96 MJ/kg	18.05 MJ/kg

SOURCE: BEJAN, A. Advanced Engineering Thermodynamics. New York: Wiley. (1988).

One of the disadvantages of working with hydrogen is that on ambient condition, it has very low energy density as can be seen from Table 2. That's why it needs to be compressed to liquid state and stored at low temperature. So most part of the operation related to hydrogen has an important efficiency drop due to compression and transportation.



**Table 2– Energy density of different fuels**

<b>Fuel</b>	<b>Specific Energy</b>	<b>Energy Density</b>
Hydrogen (ambient condition)	141.86 MJ/kg	2.8 Wh/L
Hydrogen at 690 bar, 15°C	141.86 MJ/kg	1250.0 Wh/L
Liquid Hydrogen at -240 °C	141.86 MJ/kg	2358.6 Wh/L
Methane at 1 bar, 15 °C	55.60 MJ/kg	10.5 Wh/L
Natural Gas	53.60 MJ/kg	10.1 Wh/L
Gasoline	47.50 MJ/kg	9500.0 Wh/L

Source: <https://energy.gov/sites/prod/files/2014/03/f12/fcm01r0.pdf> (2001).

The driving range of a fuel cell vehicle is typically around 300 miles on one tank of hydrogen.

The maximum capacity of a tank is normally 5 kg of compressed hydrogen gas.

While determining the number of cars to be served, it is assumed that a car typically requires 200 kg of hydrogen to travel 12,000 miles in a year with a rate of 60 miles/kg of hydrogen.

The assumed demand of Hydrogen is depicted in Table 1:

**Table 3: Demand of hydrogen fuel on peak time**

Start time	End time	Demand of Hydrogen in (kg)	Power required to produce 1 kg $H_2$ in KWh/kg	Demand of Energy in KW
Peak time				
08:01:00	09:00:00	30	53.5	1605
09:01:00	10:00:00	30	53.5	1605
10:01:00	11:00:00	30	53.5	1605
11:01:00	12:00:00	30	53.5	1605
12:01:00	13:00:00	30	53.5	1605
13:01:00	14:00:00	30	53.5	1605
14:01:00	15:00:00	40	53.5	2140
15:01:00	16:00:00	40	53.5	2140
16:01:00	17:00:00	40	53.5	2140
	Total	300		16050

**Table 4: Demand of hydrogen fuel on off-pick time**

Start time	End time	Demand of Hydrogen in (kg)	Power required to produce 1 kg $H_2$ in KWh/kg	Demand of Energy in KW
Off-peak time				
17:01:00	18:00:00	20	53.5	1070
18:01:00	19:00:00	20	53.5	1070
19:01:00	20:00:00	20	53.5	1070
20:01:00	21:00:00	20	53.5	1070
21:01:00	22:00:00	20	53.5	1070
22:01:00	23:00:00	20	53.5	1070
23:01:00	00:00:00	20	53.5	1070
00:01:00	01:00:00	20	53.5	1070
01:01:00	02:00:00	20	53.5	1070
02:01:00	03:00:00	20	53.5	1070
03:01:00	04:00:00	20	53.5	1070
04:01:00	05:00:00	20	53.5	1070
05:01:00	06:00:00	20	53.5	1070
06:01:00	07:00:00	20	53.5	1070
07:01:00	08:00:00	20	53.5	1070
Total		300		16050

## 1.4 Optimization:

### 1.4.1 Cost optimization equation for the electrolyzer:

An attempt is made to optimize the cost of re-fuelling hydrogen station according to the demand on peak time and off-peak time. An optimization program in python is developed to minimize the cost function under some constraints.

The cost function given to minimize is:

$$Cost = [CH2_{peak} \times Hpr_{peak}] + [CH2_{offpeak} \times Hpr_{offpeak}]$$

Where,

- $CH2_{peak}$  = Cost of hydrogen production on peak time = \$2.94/kg
- $CH2_{offpeak}$  = Cost of hydrogen production on off – peak time = \$2.14/kg
- $Hpr_{peak}$  = Amount of hydrogen to be produced in peak time(in kg)
- $Hpr_{offpeak}$  = Amount of hydrogen to be produced in off – peak time(in kg)
- $H_{rate}$  = Rate at which hydrogen is produced = 43.59 kg/hr
- $SOC_i$  = Initial State of Charge (SOC<sub>i</sub>)
- $SOC_p$  = Predicted/Required SOC
- $CF$  = Conversion factor from SOC (in %) to Power (in KW)
- $\eta$  = Efficiency of Electrolyzer = 73%
- $k$  = SOC (in %) to hydrogen (in kgs) = 3.93
- $P_{ref}$  = Demanded/ Predicted Power
- $t_1$  = Start time (On peak)
- $t_2$  = End time(On peak)
- $t_3$  = Start time (Off peak)
- $t_4$  = End time(Off peak)

The constrained are:

i) Constraint1:  $10 \leq SOC_i \leq 95$ .

(Initial State of Charge should be between 10% and 95 %.)

ii) Constraint2:  $SOC_p = \frac{Pref}{\eta \times CF}$

(Predicted State of charge is depended on predicted power requirement and conversion factor from SOC to power)

iii) Constraint3:  $8.0 \leq t_1, t_2 \leq 17.0$

(The time range of peak time is between 8.00 AM and 5.00 PM)

iv) Constraint4:  $17.0 < t_3, t_4 \leq 23.59$

v) Constraint5:  $0.0 \leq t_5, t_6 < 8.0$

(The time range of off-peak time is divided into two parts. One part is between 5.01 PM and 11.59 PM and another part is between 0.00 AM and 8.00 AM)

vi) Constraint 6a:  $Hpr_{peak} = k \times (SOC_p - SOC_i)$

(Hydrogen production on peak time is equivalent to the difference between predicted SOC and initial SOC. For converting the SOC to required amount of hydrogen in kg, it is multiplied by a conversion factor 'k')

vii) Constraint 6b:  $Hpr_{offpeak} = k \times (95 - SOC_i)$

(The predicted demand of hydrogen on off peak time is fixed as it is 95% of SOC. So the production of hydrogen in that period is dependent on initial State of Charge and the conversion factor)

viii) Constraint 6c:  $H_{rate} \times (t2 - t1) = Hpr_{peak}$

Constraint 6d:  $H_{rate} \times (t4 - t3) = Hpr_{offpeak}$

(The hydrogen production rate is 43.59 kg/hr for the assumed electrolyzer. So the total hydrogen production in a given time is calculated by multiplying the production rate with the duration of the time)

#### 1.4.2 Optimization result and discussion:

The cost function and constraints are studied and analyzed. After analyzing, the cost function and constraints are modified but still some more modification is needed to produce best optimization result.

Using the constraint, the cost function is optimized by python programming. Sequential least squares programming algorithm (SLSQP) optimizer is used to minimize the cost function. Even though the program successfully terminated the optimization, more work is needed on the electrolyzer optimization modelling. The cost function consists of two parts. The first part evaluate the cost on peak time and the second part evaluate the cost on off peak time. Both part are supposed to generate the cost according to the demand of power at their respective time. According to the constraint given, hydrogen production on peak time is evaluated based on the demand on that time but the calculation of the hydrogen production on off peak time is not effected by demand on that time because no constraint is assigned to do it.

The conversion factor from state of charge to power is also playing an important role during the optimization. Little work on these issues like conversion factors and constraints can make the optimization more efficient. The optimization result from the python optimization program is depicted in Figure 3.

```

===== RESTART: C:\Users\CHOWMW\Documents\electrolyzer.optimize_demand.py =====
Optimization terminated successfully.      (Exit mode 0)
      Current function value: 1471.4804892648594
      Iterations: 8
      Function evaluations: 40
      Gradient evaluations: 4
[  8.00000000e+00  1.70000000e+01  1.00000000e-01  3.62181361e+01
  3.01605198e+02  2.73252900e+02  2.54700000e+01  1.02214325e+02]
1524.0
t1 = 8.0
t2 = 17.0
t3 = 0.1
t4 = 36.2181361449
Hprpeak = 301.605198389
Hpr_offpeak = 273.2529

```

Figure 2: Optimization Result of Electrolyzer

The optimization result shows the cost of producing 301 kg hydrogen on peak time and 273.25 kg hydrogen on off-peak time. The demand of hydrogen on peak time is given as 300 kg. So it produces 301.60 kg hydrogen for that period at a conversion rate 215 (SOC to power). More work is needed to find the exact conversion rate as it makes a big difference for the cost optimization. No demand is taken into consideration while calculating the hydrogen production on off-peak time. It is assumed that the hydrogen produced on off-peak time considered the initial value of the optimization algorithm and constraints given for that period. The end time of the off-peak time range does not stay within limit. So more research is needed to solve this inconsistency.

## Chapter 2: Installation of Raspberry PIs

### 2.1 Introduction:

The Raspberry PI is a credit card sized little computer which is capable of doing many things like Desktop computer. The specification of Raspberry PI 3 mode B is given as following:

**Table 5: Specification of Raspberry PI 3 model B**

Introduction Date	2/29/2019
SoC	BCM2837
CPU	Quad Cortex A53 @ 1.2GHz
Instruction Set	ARMv8-A
GPU	400MHz VideoCore IV
RAM	1GB SDRAM
Storage	Micro-SD
Ethernet	10/100
Wireless	802.11n/Bluetooth 4.0
Video Output	HDMI / Composite
Audio Output	HDMI / Headphone
GPIO	40

Source: <http://hackaday.com/2016/02/28/introducing-the-raspberry-pi-3/>

Due to its size and capability, it can be used in many electronics projects. For this reason, in order to perform some research on electric vehicle charging, a platform with hundreds of raspberry PI is installed. Each Raspberry PI is connected to Real Time Digital Simulation (RTDS) system through VILLAS node and capable of sending and receiving data like voltage,



frequency etc. This installation process is carried out in some steps. In this chapter, all the steps that are taken are discussed with elaboration.

## **2.2 Procedure:**

The installation process is carried out in some steps. The steps are:

- i. Installation of the operating system on the Raspberry PI.
- ii. Set up of the Raspberry PI
- iii. Assigning a static IP address.
- iv. Putting Raspberry PIs on Shelves.
- v. Making network cable.
- vi. Connecting all the Raspberry PIs to the network.
- vii. Testing by sending and receiving some data.

### **2.2.1 Installation of the operating system on the Raspberry PI:**

- The first step is to download the operating system for the raspberry PI. The official operating system is Raspbian. The official image of Raspbian is downloaded on a Desktop computer from Raspberry PI website<sup>[5]</sup>
- A micro SD card is inserted into card reader. The card reader is then connected to the computer.
- An image writing tool is needed to install the image of Raspbian to the SD card. In this case, Win32 Disk imager is downloaded and used to write the ‘.img’ file into the SD card.

### **2.2.2 Set up the Raspberry PIs:**

After installation of the operating system, the Raspberry PI is ready to set up. The following steps are taken to set up the Raspberry PI<sup>[6]</sup>:

- The microSD card is inserted into the card slot of the Raspberry PI.
- A USB keyboard and a USB mouse are plugged into the USB ports.
- One end of a HDMI cable is connected to the Raspberry PI and other end is connected to the monitor.
- Power supply is connected to the Raspberry PI. When the power supply is plugged into the power outlet the Raspberry PI is turned on and booted up.

### **2.2.3 Assigning a static IP address:**

When the setup is complete, each Raspberry PI is assigned a static IP address starting from 141.221.118.128 to 141.221.118.227. The specifications are as following:

IP address: 141.221.118.128 to 141.221.118.227

Mask: 255.255.255.0

Gateway: 141.221.118.65

DNS server: 141.221.118.69

### **2.2.4 Arranging raspberry PI on shelves:**

The Raspberry PIs are put and glued on the shelves. Each Shelf contains 20 Raspberry PI. Five PIs are connected to one charger. So four chargers are used for twenty Raspberry PIs in one shelves. Each Raspberry PIs and chargers are adhered to the shelf using sticky VELCRO sticky back tape.

### 2.2.5 Making network cables:

One hundred Ethernet cables are made for one hundred Raspberry PI. The following things are used<sup>[7]</sup>:

- Bulk Ethernet cable category 5e or CAT-5e
- Bulk RJ45 Connectors for CAT-5e
- RJ-45 Crimping tool
- An Ethernet cable tester.

The bulk Ethernet cable is cut into one hundred PIs. The following steps are taken to make the cable:

- The plastic sheath is cut into about 1 inch from the end of the cable.
- The wires are untwisted and straightened. The wires are then organised according to the figure 4: Ethernet cable (order of wires)

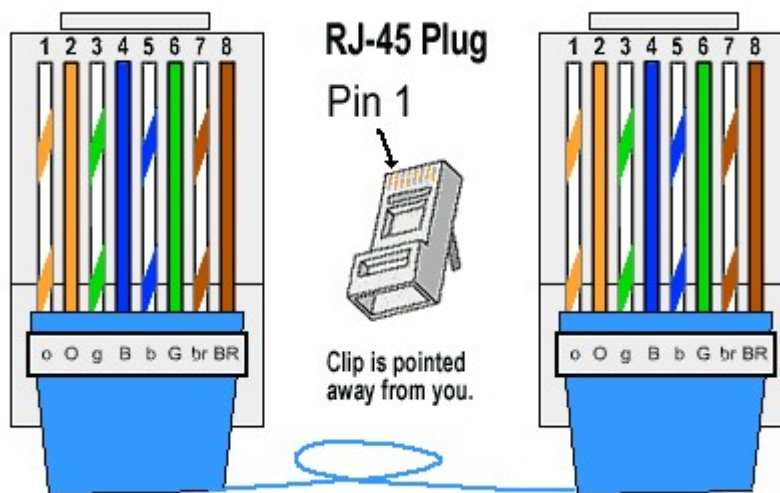


Figure 3: Ethernet cable (order of wires)<sup>[7]</sup>

- All the colored wires are then pushed into the connector RJ-45 according to the Figure 3.

- The connector is placed into the Ethernet crimper and the handles are cinched down. After removing the connector from the crimper, it is ready to use.
- The steps are repeated for the other end and both ends are tested using Ethernet cable tester.

### **2.2.6 Connecting all the Raspberry PIs to the network:**

Each end of the Ethernet cable is connected to one Raspberry PI. The other end is connected to the network. These connections are made to do some simulation on Real Time Digital Simulation (RTDS) and OPAL RT. VILLAS node act as a gateway between them.

Below Figure 5, 6 and 7 shows the networked Raspberry PIs.



Figure 4: Arrangement of Raspberry PI on Shelf (1)

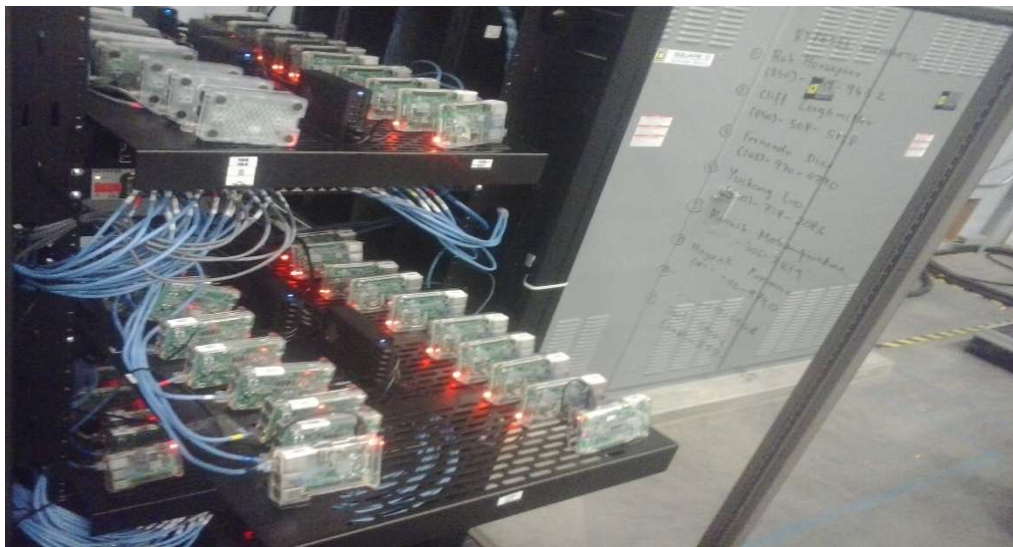


Figure 5: Arrangement of Raspberry PIs in shelf (2)

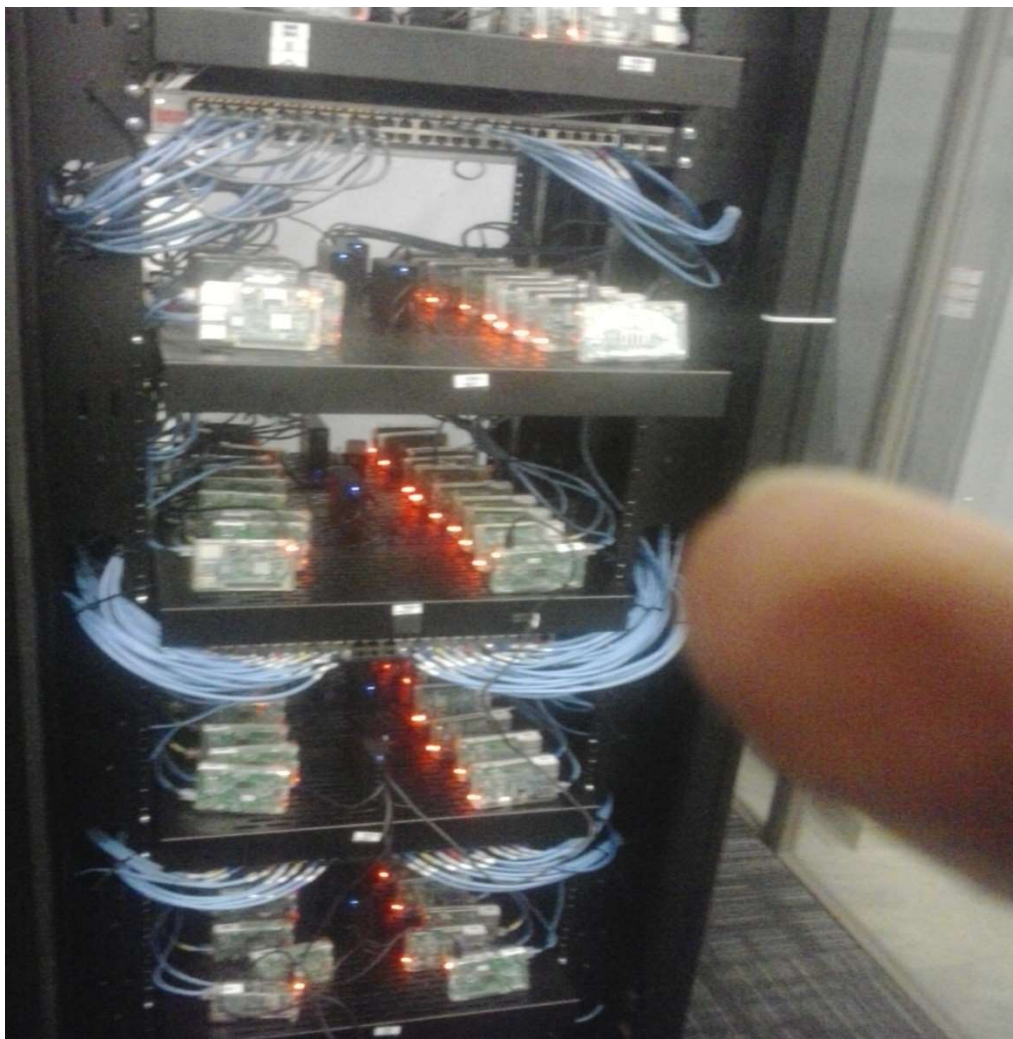


Figure 6: A front view of the Raspberry PIs

### 2.2.7 Testing by sending and receiving some data:

A python program is developed for sending and receiving a float number. A float number is sent from one of the Raspberry PI to VILLAS node and tested successfully.

The python program used for sending and receiving data is:

The 'client.py' file depicted in Figure 8 is used to send data.

```
1 import socket, Struct
2 import time
3
4 host = '141.221.118.93'
5 port = 11000
6 Float = 0.4
7 while True:
8     mysocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
9     mysocket.bind("141.221.118.93", 5000)
10    Float +=1
11
12    print('sending the data')
13    print(Float, struct.pack(">f", Float))
14
15    # Send data
16    mysocket.sendto(bytearray(struct.pack(">f"Float)), (host, port))
17
18    # Receive data
19    print("Receiving data")
20    data, server = mysocket.recvfrom(1024)
21    print(data)
22    time.sleep(1)
23
```

Figure 7: client.py

The server.py file in Figure 9 is used to receive data



```

1 import socket
2
3 host = '141.221.118.93' #socket.gethostname()
4 port = 8000
5
6 mySocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7 mySocket.bind((host,port))
8
9 mySocket.listen(5)
10 print('Initalizing Connection')
11 while True:
12     print('Finding Connection')
13     connection, addr = mySocket.accept()
14     print('Got Connection from', addr)
15     Conn.send(str.encode('Thank you for connecting'))
16 connection.close()
17
18
19

```

Figure 8: server.py

### 2.3 Discussion:

A test is performed by sending a float number from a raspberry PI having IP address 141.221.118.160 to the VILLAS node having IP address 141.221.118.93. Using the program client.py and server.py, another Test is going to be completed for both receiving and sending data. After that more test will be carried out for simulation on RTDS system and OPAL-RT.

## REFERENCES:

- [1] Johanna Ivy, “Summary of Electrolytic Hydrogen production”, *Milestone Completion Report*, National Renewable Energy Laboratory, September 2004.
- [2] Abdulla Rahila, Rupert Gammonb, Neil Browna “Dispatchable Hydrogen Production at the Forecourt for Electricity Grid Balancing”, *International Conference – Alternative and Renewable Energy Quest*, AREQ 2017, 1-3 February 2017, Spain.
- [3] M. B. Stevens , M. W. Fowler , A. Elkamel & S. Elhedhli “Macro-level optimized deployment of an electrolyser-based hydrogen refuelling infrastructure with demand growth”, *Engineering Optimization*, Vol. 40, No. 10, October 2008, 955–967.
- [4] “Current (2009) State-of-the-Art Hydrogen Production Cost Estimate Using Water Electrolysis”, National Renewable Energy Laboratory.
- [5] <https://www.raspberrypi.org/downloads/raspbian/>
- [6] <https://www.raspberrypi.org/documentation/installation/installing-images/README.md>
- [7] <http://www.groundcontrol.com/galileo/ch5-ethernet.htm>



## APPENDIX: Python code for optimization

```

                                electrolyzer.optimize_demand.py
##-----Optimization of a Electrolyzer-----##
##-----Md Chowdhury-----##
##-----Idaho National Laboratory-----##

import numpy as np
from numpy import vectorize
from scipy.optimize import minimize
import datetime as dt

import xlrd
import pandas as pd

# Parameters
#Hpr = Constraint Amount of hydrozen to be produced
peak_start = "08:00:00"
peak_end = "17:00:00"
peak_start_time = dt.datetime.strptime(peak_start,'%H:%M:%S')
peak_end_time = dt.datetime.strptime(peak_end,'%H:%M:%S')
off_peak_start = "17:00:00"
off_peak_end = "23:59:00"

Hrate = 2.83 #Rate at which hydrogen is produced( kg/hr)
CH2peak = 2.94 #Cost of hydrozen production on peak in $/kg
CH2offpeak = 2.14 # Cost of hydrozen production off peak in $/kg
CF = 1000
eff = 0.73
k = 3.93

# Reading the demand data from Excel file
workbook = xlrd.open_workbook("demand.xlsx")
worksheet = workbook.sheet_by_name("Sheet1")
Demand = worksheet.cell(4,4).value
Pref = Demand
for row in range(worksheet.nrows):
    demand = worksheet.cell_value(row,2)
    #print(worksheet.cell_value(row,2))
#Pref = 100
#Soci = 30

# Objective function of the electrolyzer
def objective(ch):
    t1 = ch[0]
    t2 = ch[1]
    t3 = ch[2]
    t4 = ch[3]
    # t5 = ch[4]
    # t6 = ch[5]
    Hprpeak = ch[4]
    Hproffpeak = ch[5]
    soci = ch[6]
    socp = ch[7]
    #Pref = ch[10]

    #Cost = CH2peak*Hrate*(t2-t1)+ CH2offpeak*Hrate*(t4-t3)+ CH2offpeak*Hrate*(t6-t5)
    Cost = CH2peak*Hprpeak+ CH2offpeak*Hproffpeak #+ CH2offpeak*Hproffpeak
    return Cost

# time constraint (8.0<=t1, t2 <=17.0)

```

Page 1

electrolyzer.optimize\_demand.py

```
def constraint1(ch):
    return ch[0] -8.0

def constraint2(ch):
    return ch[0] -8.0

def constraint3(ch):
    return 17.0-ch[0]

def constraint4(ch):
    return 17.0-ch[1]

def constraint5(ch):
    return 17.0-ch[1]

def constraint6(ch):
    return ch[1]-8.00

def constraint7(ch):
    return ch[1]-ch[0]

# time constraint (0.0<t3, t4<=8.0)

def constraint8(ch):
    return ch[2]-0.1

def constraint9(ch):
    return ch[2]-0.1

def constraint10(ch):
    return 8.0-ch[2]

def constraint11(ch):
    return 8.00-ch[3]

def constraint12(ch):
    return ch[3]-0.1

def constraint13(ch):
    return ch[3]-ch[2]

#time constraint (17.0<t3, t4<=23.59)

#def constraint14(ch):
#    return ch[2]-17.0

#def constraint15(ch):
#    return 24.0-ch[2]

#def constraint16(ch):
#    return ch[3]-17.0

#def constraint17(ch):
#    return 23.59-ch[3]

#def constraint18(ch):
#    return ch[3]-ch[2]

# constraint for initial soc_i (10<=soc_i<=95)

def constraint19(ch):
```

Page 2

```

electrolyzer.optimize_demand.py

    return ch[6]-10

def constraint20(ch):
    return 95 - ch[6]

# constraint for Pref

#def constraint21(ch):
#    return ch[10]-10000

#def constraint22(ch):
#    return 100- ch[10]

# constraint for SOCP [SOCp = Pref/eff*CF]

def constraint23(ch):
    return ch[7] - 16050/(eff*CF)

# constraint for SOCP [Hprpeak = k*(SOCp-SOCi)]

def constraint24(ch):
    return ch[4] - k*(ch[7]-ch[6])

def constraint25(ch):
    return ch[5] - k*(95-ch[6])

# Constraint for Hprpeak = Hrate*(t2-t1)
def constraint26(ch):
    return (Hrate*(ch[1]-ch[0])- ch[6])

# Constraint for Hproffpeak = Hrate*(t4-t3)
def constraint27(ch):
    return (Hrate*(ch[3]-ch[2])- ch[7])

#Initial guess
n= 8
ch0 = np.zeros(n)
#ch0 = [0,0,0,0,0,0]
ch0[0] = 8.0
ch0[1] = 16.0
ch0[2] = 1.0
ch0[3] = 7.0
#ch0[4] = 17.0
#ch0[5] = 23.0
ch0[4] = 300.0
ch0[5] = 300.0
ch0[6] = 10
ch0[7] = 30
#ch0[10] = 100

# Time constraint
con1 = {'type': 'eq', 'fun': constraint1}
con2 = {'type': 'ineq', 'fun': constraint2}
con3 = {'type': 'ineq', 'fun': constraint3}
con4 = {'type': 'eq', 'fun': constraint4}
con5 = {'type': 'ineq', 'fun': constraint5}
con6 = {'type': 'ineq', 'fun': constraint6}
con7 = {'type': 'ineq', 'fun': constraint7}
con8 = {'type': 'ineq', 'fun': constraint8}

```

Page 3

```

electrolyzer.optimize_demand.py
con9 = {'type': 'eq', 'fun': constraint9}
con10 = {'type': 'ineq', 'fun': constraint10}
con11 = {'type': 'ineq', 'fun': constraint11}
con12 = {'type': 'ineq', 'fun': constraint12}
con13 = {'type': 'ineq', 'fun': constraint13}
#con14 = {'type': 'ineq', 'fun': constraint14}
#con15 = {'type': 'ineq', 'fun': constraint15}
#con16 = {'type': 'ineq', 'fun': constraint16}
#con17 = {'type': 'ineq', 'fun': constraint17}
#con18 = {'type': 'ineq', 'fun': constraint18}
con19 = {'type': 'ineq', 'fun': constraint19}
con20 = {'type': 'ineq', 'fun': constraint20}
#con21 = {'type': 'ineq', 'fun': constraint21}
#con22 = {'type': 'ineq', 'fun': constraint22}
con23 = {'type': 'eq', 'fun': constraint23}
con24 = {'type': 'eq', 'fun': constraint24}
con25 = {'type': 'eq', 'fun': constraint25}
con26 = {'type': 'eq', 'fun': constraint26}
con27 = {'type': 'eq', 'fun': constraint27}
cons = ([con1,con2,con3,con4,con5,con6,con7,con8,con9,con10,con11,con12,con13,
con19,
con20,con23,con24,con25,con26,con27])#,con14,con15,con16,con17,con18,con22,con23])
solution = minimize(objective,ch0,method = 'SLSQP',constraints = cons, options=
{'disp':True})

#print('Final SSE objective: ' + str(objective(ch0)))

#print('Initial objective: ' + str(objective(x)))

print(solution.x)
#print(objective(ch0))
print('t1 = ' + str(solution.x[0]))
print('t2 = ' + str(solution.x[1]))
print('t3 = ' + str(solution.x[2]))
print('t4 = ' + str(solution.x[3]))
#print('Hprpeak = ' + str(solution.x[4]))
#print('Hpr_offpeak = ' + str(solution.x[5]))

```