Evolved Design of a Nonlinear Proportional Integral Derivative (NPID) Controller

Shubham Chopra
Portland State University

Evolved Design of a Nonlinear Proportional Integral Derivative (NPID) Controller

by

Shubham Chopra

A thesis submitted in partial fulfillment of the
requirements for the degree of

Masters of Science
in
Electrical and Computer Engineering

Portland State University
©2012

Abstract

This research presents a solution to the problem of tuning a PID controller for a nonlinear system. Many systems in industrial applications use a PID controller to control a plant or the process. Conventional PID controllers work in linear systems but are less effective when the plant or the process is nonlinear because PID controllers cannot adapt the gain parameters as needed.

In this research we design a Nonlinear PID (NPID) controller using a fuzzy logic system based on the Mamdani type Fuzzy Inference System to control three different DC motor systems. This fuzzy system is responsible for adapting the gain parameters of a conventional PID controller. This fuzzy system’s rule base was heuristically evolved using an Evolutionary Algorithm (Differential Evolution).

Our results show that a NPID controller can restore a moderately or a heavily under-damped DC motor system under consideration to a desired behavior (slightly under-damped).

Acknowledgements

I would like to take this opportunity to thank all the people who whole-heartedly supported my efforts and helped me in my endeavor to write a thesis for my Masters of Science degree in Electrical and Computer Engineering from Portland State University.

Firstly I would like to thank my professor, mentor and thesis advisor Dr. Garrison Greenwood. I will forever be grateful to him for allowing me to work as a research student under him and motivating me to do a thesis under his able guidance. Without his constant encouragement and constructive criticism this thesis would not have been possible. I would also like to thank Dr. Marek Perkowski and Dr. Christof Teuscher for agreeing to be on my thesis committee and giving me an invaluable feedback on my efforts.

I would like to thank all the staff members working in the ECE office who helped me get through all the paperwork required to present this thesis. A special thanks to my friends Prashant, Rajesh, Aniket, Siddharth, Vivek and Vishruth who helped me during my stay here at Portland State University.

Last but not the least I would like to thank my mom Dr. Neeta Chopra, dad Mr. Pradeep Chopra and my sister Priyamvada Chopra for always believing in me and my ideas and supporting me all throughout.

Contents

List of Tables

List of Figures

I. Introduction

In today's modern engineering world *Direct Current motors* are widely used in the controls industry [15]. Many industrial applications of the Direct Current motor require a *PID (Proportional Integral Derivative) controller* to control the speed or the position of the motor. Some of such systems are linear in nature and therefore a conventional PID controller would suffice but in the real world, most systems have some nonlinearity in them and thus make the system a nonlinear system. Such nonlinearities in the system cannot be dealt with using a conventional PID controller because of the controller's inability to change the gain parameters to counter the nonlinearities. The solution to this problem can be found in the form of a Nonlinear PID controller. A Nonlinear PID controller has the ability to account for the nonlinearities in the system and change the gain parameters accordingly. A Nonlinear PID controller will often have a strategy such as neural network, fuzzy logic controller, etc. to help in the tuning of the controller according to the response of the system. In our research we try to solve this problem and attempt to evolve a Nonlinear PID controller using a fuzzy logic system whose rule base is evolved using an *Evolutionary Algorithm* called the *Differential Evolution*. Figure 1 depicts a basic Nonlinear PID controller, which uses a *fuzzy logic controller* to get the offsets for the gain parameters, which help countering the nonlinearities.

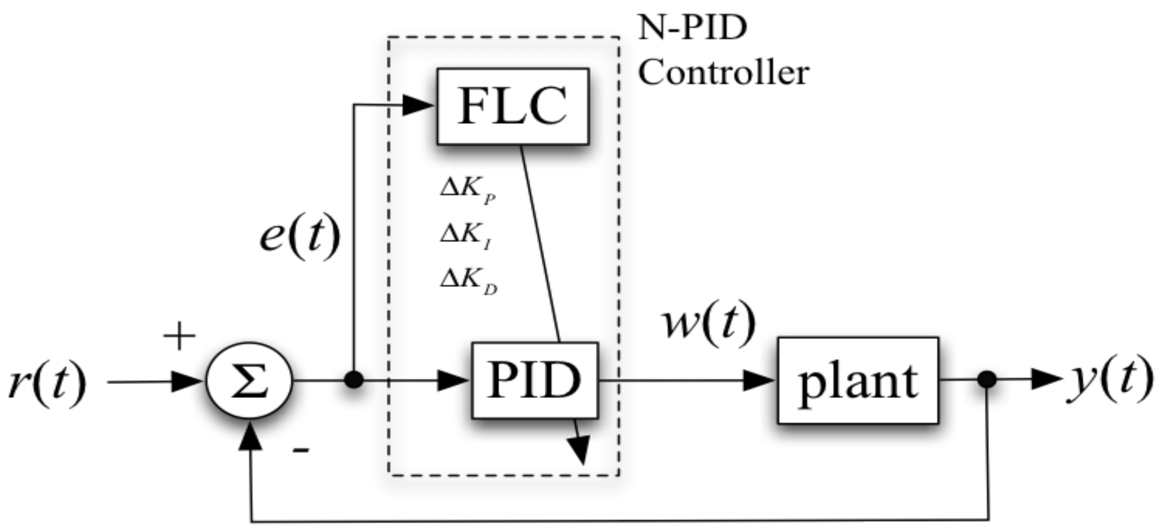


Figure 1. A basic model of a Nonlinear Proportional Integral Derivative (NPID) Controller is shown here. The Fuzzy Logic Controller (FLC) feeds the PID controller with offsets for gain parameters.

We use the DC motor as our plant, which would be controlled by the PID controller. The DC motor is loaded with three different simulated loads by changing the moment of inertia on the rotor (J) on the motor, where J is directly proportional to the load on the motor. The three different systems simulated were “slightly under-damped system”, “moderately under-damped system” and “heavily under-damped system”. We chose “slightly under-damped system” as our desired system and the other two systems as the test systems, which we attempt to restore to the desired behavior in this research.

We simulated three systems with different step responses and adapted their response with our Nonlinear PID controller. This was achieved by integrating a conventional PID controller and a fuzzy logic controller into a system to account for the nonlinearities in the system being controlled by the Nonlinear PID controller. The fuzzy logic controller has fixed output and input membership functions and an evolving rule base. We made an algorithm based on Evolutionary Algorithm to evolve the rule base of the fuzzy logic controller.

The fuzzy system used by us in our research is a Mamdani type Fuzzy Inference System that was proposed by E. H. Mamdani in 1974 [12]. Today Mamdani fuzzy system is one of the most widely used FIS in the controls industry. The Fuzzy Inference System takes in crisp values from the problem as inputs or antecedents, which are fuzzified by the input membership functions. The fuzzified inputs are evaluated using output membership functions and a good rule base provided by an experienced user or developed using a heuristic approach. This process gives us consequents for the system. The consequents are then defuzzified using a defuzzification operator such as “centroid”, “center of mass”, etc.

The rule base for our fuzzy logic controller is heuristically evolved by our algorithm, which is an Evolutionary Algorithm [17], based on the Differential Evolution strategy [10]. The rule base consists of nine rules each with two antecedents (error and rate of change of error) and has three consequents corresponding to the three offsets for the gain parameters (Kp, Ki and Kd).

An Evolutionary Algorithm is a population-based algorithm that uses a heuristic approach to optimize problems, which would be otherwise tedious to solve because of their large solution spaces. Every Evolutionary Algorithm starts off with a randomly initialized well distributed population of individuals that are unique solutions in the solution space. Each individual is operated on by a set of random or stochastic user defined operators to perturb the population and create new individuals also called as offspring for the current generation. The quality of the offspring generated in the current generation is measured using an objective function called as the fitness function. The fitness function decodes the encoded individual and merits the quality of the individual, which is used to select parents for the next generation. The algorithm iterates through this loop until a termination criterion is met.

The Differential Evolution [10] strategy used in our research was initialized with a population of a hundred individuals. Each individual consists of twenty-seven parameters corresponding to the three consequents (ΔKp, ΔKi, ΔKd offsets) for each of the nine rules. To create new offspring from parents a mutant vector is created using the “DE\Best\1” [10] strategy. The crossover operator operates on the mutant vector and the parent to create a new offspring. Fitness of the parent and the offspring is compared to select parents for the next generation. Detailed equations with the process are explained in section IV “Experiment Setup” of this thesis.

Manually tuning a PID controller has never been an easy job for the user. There exists manual tuning processes such as the Ziegler-Nichols process [11] suggested by Ziegler and Nichols for tuning a PID controller but they require tedious and repetitive observations of the step responses by the user and are not always accurate. To solve this problem we in our research attempt to devise an efficient automated tuning process for the PID controller using the step responses of the different systems controlled by the PID controller.

The step response of a system is almost always performed in the time domain, which is not easy to analyze. We could use Continuous Time Fourier Transform [16] to analyze the step responses of the system but the Continuous Time Fourier Transform equations require knowledge of the settling time and the rise time of the step response. This knowledge cannot be easily extracted from the step response in the time domain because the user more often than not does not know the starting point of a sampling period. As a solution to this problem we sample the step response over one time period and use Discrete Fourier Transform to transform the step response of the systems from the time domain to the frequency domain. This enables us to analyze the step response in the frequency domain and calculate energies of the motor in the higher and lower energy sub bands. This energy calculation in the different sub bands also serves as a point of differentiation between the three systems under consideration and thus is used to make an objective function to be used as the fitness function for our Evolutionary Algorithm.

The number of samples taken over the period of the step response was 384. We wanted the rule base in the fuzzy system to evolve for every time sample thus the algorithm was run for a total of 384 iterations before being terminated and the results are recorded. We did a total of twenty-five runs each for each test system under consideration to look for any anomalies and to achieve the best possible results. We noted that the algorithm always restored the behavior of the test systems to the desired system except for one anomaly that we encountered in one of the twenty-five runs for the moderately under-damped system. The graphs of the restored systems are shown and discussed in detail in the results section of the thesis.

The major contributions of our research are –

- The use of Evolutionary Algorithm to develop a rule base for a fuzzy logic controller, which dynamically tunes the conventional PID controller according to the nonlinearities in the system.
- The use of frequency domain characteristics to classify the system step responses.

II. Related Work

The PID controllers because of their popularity in the controls industry have been widely studied and researched. In this section we will enlist some of the different methods of tuning a PID controller that have been researched by out peers.

Tuning a PID controller manually is a tedious process and requires the expertise of highly experienced individuals who have a functional understanding of the process they are trying to control. Dr. Dong Hwa Kim of Hanbat National University, South Korea and his associates have done some work in this field and come up with various methods of tuning a PID controller.

Dr. Dong Hwa Kim and his associates in have implemented the immune algorithm and neural networks to tune a PID controller [19, 20, 21]. Immune algorithm is a well-known search and optimization algorithm that the authors of the papers used to decide the disturbance rejection for the control process and then tune the gain of the PID controller.

Another technique used by Dr. Sung-Kwun Oh Dae-Keun Lee et al was Hybrid Fuzzy Controllers [22]. Here the authors integrated a fuzzy logic controller with the PID controller to control a process. GA was used to estimate the gain parameters and scaling factors of these parameters. The member functions of the fuzzy system were not fixed and were encoded in the individual along with gain and scaling parameters for the PID controller.

Our research uses a fuzzy logic system to determine the offsets for the gain parameters instead of estimating the gain parameters. We use an EA called Differential Evolution (DE) as our algorithm to change the rule base for the fuzzy logic system and keep our membership functions fixed. Therefore our method of tuning a PID controller is novel and can be implemented to control a nonlinear process.

III. Background

1. Evolutionary Algorithm (EA)

Many real world problems are hard to solve because of the complexity of the search or solution space and possibility of an exponential number of solutions possible for a given problem. A solution or search space is defined as a hyperspace where each point in it is a unique solution for the given problem. Such real world problems, which have a large hyperspace or solution space, can be solved using a heuristic or stochastic approach. This is where Evolutionary Algorithm (EA) comes into picture. An Evolutionary Algorithm (EA) is a population-based algorithm, which uses a heuristic approach to optimize a given problem which otherwise is tedious and time consuming to solve [17]. It closely mimics the natural selection method seen in nature and uses various reproduction methods to create new offspring and maintain diversity in the population.

The basic steps in an EA are shown in figure 2. The first step of an EA is to initialize a random population of a user specified size. Each individual is encoded as a data structure, which is a set of different problem parameters. An individual is essentially a point on the solution space, which can also be called a unique solution for the problem, the EA is trying to solve. The problem parameters, which make up the individual, are nothing but different encoded values that define the individualistic traits or characteristic of the solution.

Each problem parameter in the individual is randomly initialized within the upper and lower bounds of the parameter. The use of a good random function is key to the initialization process to uniformly spread the population throughout the solution space and have a good seed population for the algorithm.

Random or stochastic reproduction operators create new offspring from a set of parents that operate upon the population. This step is an important step to effectively search the solution space. Each new offspring is an individual with a new unique solution to the problem. Various user specified recombination and mutation operators are used for this step and are a key to the success of the algorithm. A lot of research has been done in this area and each operator has its own advantages and disadvantages.

Fitness is the measure of quality of the individual. In this step the individual is decoded and applied to an objective function. The objective function is also called as the fitness function, which is a user, defined function. The fitness function returns a single value of merit, which is termed as the fitness of the solution. Fitness of the new offspring population is calculated and is used as a criterion of selection for the next generation.

The algorithm uses new offspring population from the current generation to select parents for the next generation. Fitness value of each individual is used as the figure of merit for this purpose. It is important to use a good selection technique to help maintain diversity in the population. Some examples of selection techniques are “roulette wheel selection”, “tournament selection”, etc. One of the common practices is to retain the best solution and not mutate it so the best solution is always preserved. This technique is called “elitism”.

The algorithm iterates through the loop shown in figure 2 until it is terminated. The algorithm is terminated after a suitable termination criterion such as an acceptable solution has been found or after iterating through a fixed number of generations specified by the user.

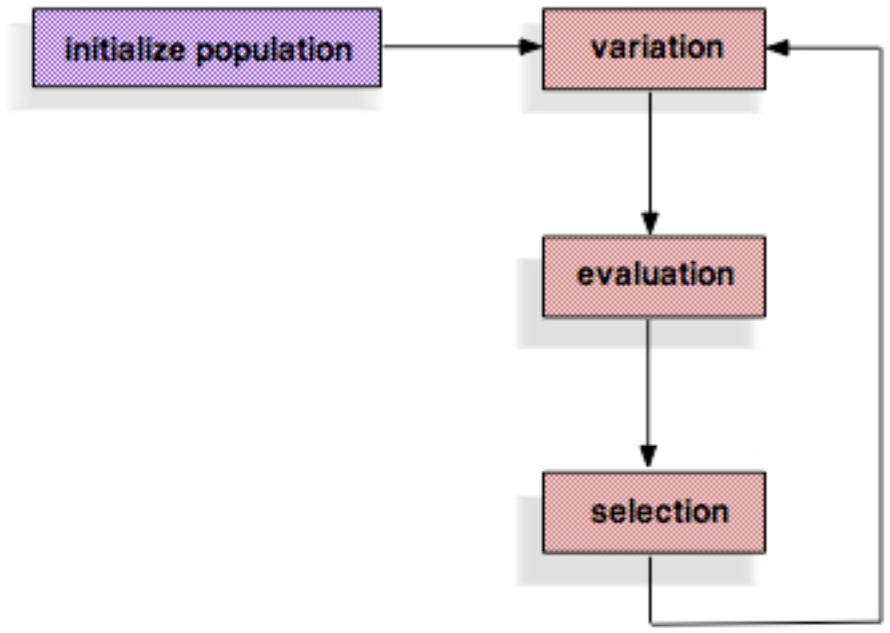


Figure 2. Basic steps of an Evolutionary Algorithm (EA) are illustrated by the flowchart shown above; Figure extracted from [1].

2. Proportional Integral Derivative (PID) Controller

A Proportional – Integral – Derivative (PID) controller [18] is a closed loop feedback controller used widely in the controls industry. A conventional PID controller is shown in figure 3.

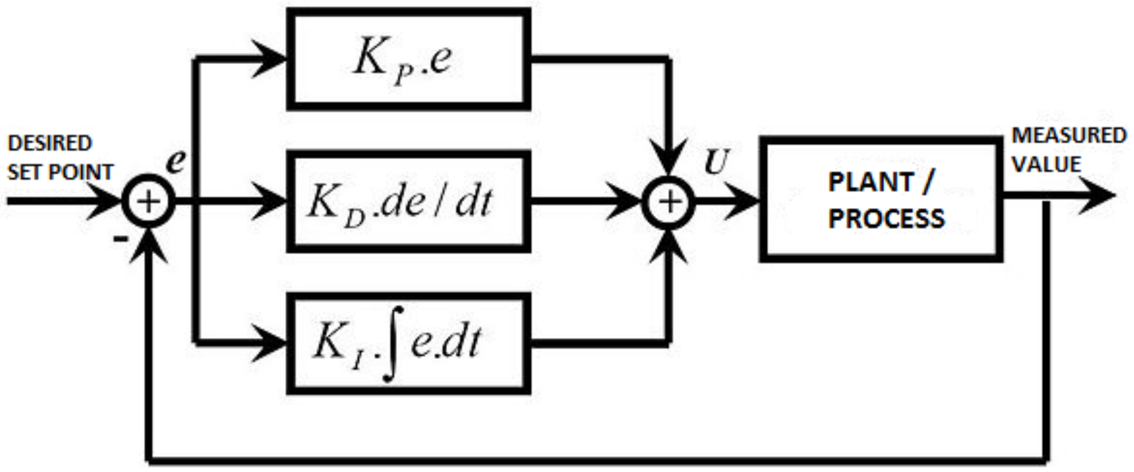


Figure 3. Block diagram of a Proportional Integral Derivative (PID) Controller is shown here. It is a representation of how the gain parameters are processed by the PID algorithm; Figure adapted from [2].

A PID controller adjusts the gain values K_p, $K_{i,}$ and K_d according to the inputs to the controller. Error (e) and rate of change of error (edot) are given as inputs to the controller. Error is the difference between the measured response and the desired response and rate of error change is the difference between two consecutive error signals. The PID controller calculates the Proportional (P), Integral (I) and Derivative (D) values using an algorithm and then an adder calculates the weighted sum of the three parameters and gives a value as an input to the plant which aims to control the plant or the process the controller is controlling such as – position of a DC motor, heating system, etc. The equation that is used to calculate the output of the controller is given below.

$$\boldsymbol{u = K_P e + K_I \int edt + K_D \frac{de}{dt}}$$

The discrete time version of the above equation is

$$\boldsymbol{u(t) = K_p e(t) + \ K_i \sum_{k=0}^{t} e(k) + K_d\big(e(t) - e(t-1)\big)}$$

Here ‘u’ is the output of the controller and ‘e’ is the error signal calculated as the difference between the measured value and the desired set point. A system controlled by a PID controller might not need all the three P, I, and D parameters to control a process successfully and can be flexibly configured in various configurations such as PI, P, PD, I, etc. as it operates on each term individually. Such a configuration is achieved by making all the unused parameters 0. Table 1 illustrates the effects of changing K parameters on a step response given to the system. While tuning K parameters manually the information from this table can be used in deciding what parameter(s) to tweak to achieve the desired result on the system step response. This process of manual tuning of the K parameters can become tedious and cumbersome because of its trial and error approach to tune the gain parameters of a PID controller.

Response	Rise Time	Overshoot	Settling Time	S-S Error
K_P	Decrease	Increase	NT	Decrease
K_I	Decrease	Increase	Increase	Eliminate
K_D	NT	Decrease	Decrease	NT

Table 1. The table tabulates the effects on a step response of changing K parameters for a conventional PID controller. This table is consulted when an experienced user is manually tuning the PID controller to achieve the desired system response. Here NT means no definite trend is noticed; Table extracted from [3].

3. Nonlinear PID Controller

Although many processes can be controlled by a conventional PID controller but for high performance with changing operating conditions and parameters we require specialized PID controllers to control the nonlinearity in the system. In the real world, many processes are nonlinear and a conventional PID controller is not capable of handling the nonlinearity in the system because of its limitation of handling only fixed K parameters. This problem can be solved using a *nonlinear PID (NPID) controller*, which uses self-tuning capabilities like fuzzy logic and neural network strategies to adapt to the changes (nonlinearities) in the system and thus being capable of offsetting the K parameters instead of keeping them fixed.

We use this capability of the nonlinear PID controller and build our system around the NPID controller. The NPID controller in our system is fed by the error and the rate of change of error signals, which go through a fuzzy logic controller to give offsets in the gain parameters of the PID controller as outputs. Our aim in this research is to identify the response of the system and establish which of the two test systems (moderately under-damped or heavily under-damped) are we dealing with and accordingly change the responses from the fuzzy logic controller, which offsets the gains of the PID controller. This will be difficult to achieve using a conventional PID controller since the system is nonlinear. Figure 4 shows the NPID controller using the fuzzy logic controller in our system.

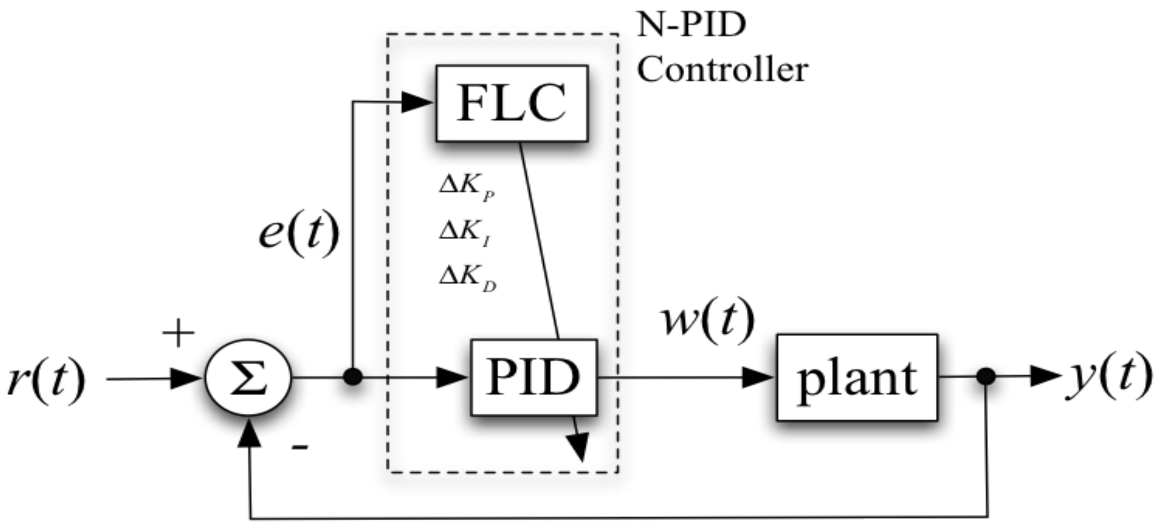


Figure 4. Nonlinear Proportional Integral Derivative (NPID) Controller. The figure illustrates how the NPID controller was configured for our system. The FLC is responsible for feeding the offsets for the gain parameters to the PID controller as shown here.

4. Discrete Fourier Transform (DFT)

Fourier Transformation on a function can be performed using *Continuous Time Fourier Transformation (CTFT)* or *Discrete Fourier Transform (DFT)*. CTFT transforms continuous time signals from time domain into frequency domain. The problem with CTFT is that we have to know the equation of the signal to perform the transformation and the equation of the signal is not always known especially if the signal is nonlinear. As a solution to this problem we take samples of the continuous time function over one period and convert the continuous time system into a discrete time function. The interval of samples is fixed and is called the sampling period. Now we have a discrete function, which essentially means that the function has finite non-zero values in a discretized form. The DFT then operates on it using the equation given below and transforms the function into frequency domain.

$$\boldsymbol{X_k = \sum_{n=0}^{N-1} x_n \,.\, e^{-i2\pi\frac{k}{N}n}}$$

In the digital world where continuous time signals cannot exist and all signals are discrete samples of the continuous time function DFT is widely used to obtain the frequency domain characteristics of the time domain signals. Other applications of DFT include signal processing, performing convolutions, etc. One of the key factors of the widespread use of DFT to analyze signals is its ability to be computed efficiently using the Fast Fourier Transform (FFT) functions available.

In our research we used the DFT to analyze the step response of the motor. The step response of the motor is in time domain, which is transformed into a response in frequency domain. This transformation lets us calculate the energy of the motor in the higher and lower frequency sub bands which are used to differentiate between the three systems (desired system, moderately under-damped system and heavily under-damped system) considered for our research. The importance of identifying the three systems from each other is critical for our research and is explained in later sections.

5. Fuzzy Logic Controller

The word fuzzy literally means “difficult to perceive clearly and understand precisely” but when the same word is used in context with a logic controller albeit “Fuzzy Logic Controller” it makes for an ingenious system widely used in the controls industry. Today’s modern engineering world poses many real world problems that cannot be stated in a classical (0 or 1) or digital (true or false) representation. The inputs and other related parameters for such problems are often ambiguous and are termed as partially true. A very good example of such a problem is the “tipping problem” [5] explained here.

For instance, if you were to go to a restaurant for a meal and after the meal you were to decide the amount of the tip for your server. You would simply take into account what the food quality was and how the server served you. Of course you could take into account a few more variables such as the ambience and other such things but for our problem let’s keep it to two variables – food quality and the service. Now you could solve this problem using two approaches –

- Linear graph to calculate the tip for the server between 5% and 25% of the bill, depending on the quality of service. The inherent problem with this solution is that you do not take into account the quality of the food. To incorporate both food quality and the quality of service you would need a 3D graph to represent the three variables and it would be a complex solution (figure 6).

- Use a fuzzy system (figure 5). The advantages of fuzzy system are that the math is much more intuitive and laid out in a simple language. You would need to come up with simple rules like “If the service is good, tip is generous” or “If the food is rancid, tip is cheap”.

The fuzzy system used here seems to be more advantageous as it is easier to comprehend and one does not have to deal with discrete variables. The fuzzy inputs are evaluated using a rule base and membership functions. The fuzzy answer then goes through a defuzzification process and a crisp output is given.

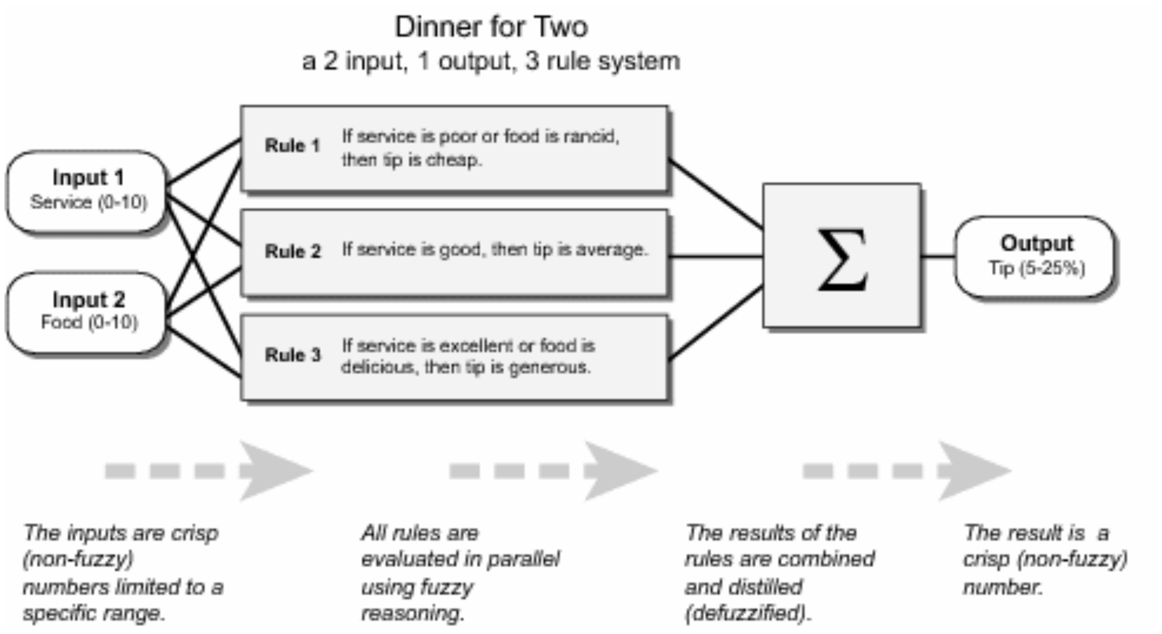

Dinner for Two
a 2 input, 1 output, 3 rule system
Input 1
Service (0-10)
Input 2
Food (0-10)
Rule 1
If service is poor or food is rancid, then tip is cheap.
Rule 2
If service is good, then tip is average.
Rule 3
If service is excellent or food is delicious, then tip is generous.
Σ
Output
Tip (5-25%)
The inputs are crisp (non-fuzzy) numbers limited to a specific range.
All rules are evaluated in parallel using fuzzy reasoning.
The results of the rules are combined and distilled (defuzzified).
The result is a crisp (non-fuzzy) number.

Figure 5. Fuzzy system for the "tipping problem". The figure shows the basic functioning of a fuzzy system. Notice the association of the inputs and the rules for the "tipping problem"; Figure extracted from [6].

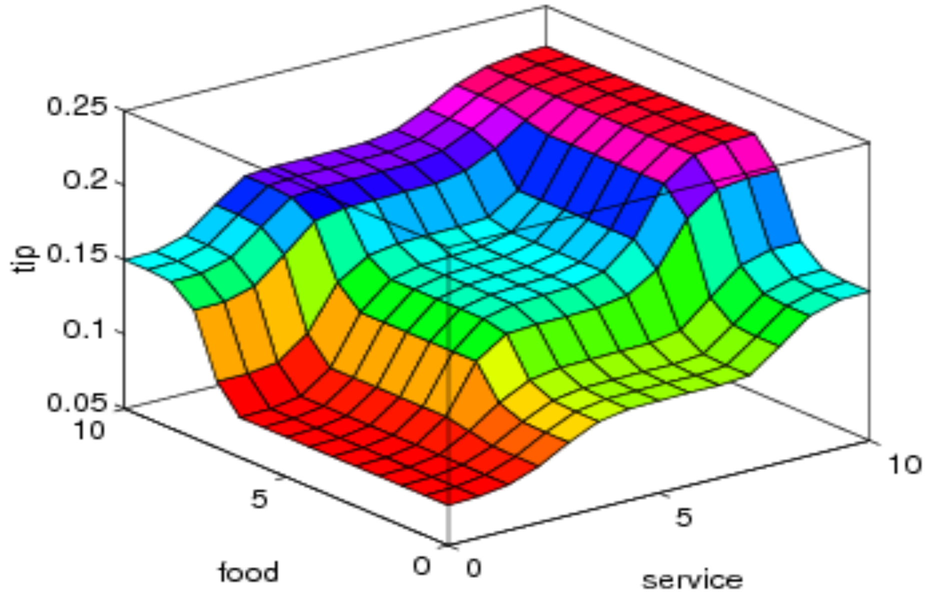


Figure 6. 3D graph for the "tipping problem". This graph represents the mathematical solution of the "tipping problem" taking quality of food and service as variables; Figure extracted from [5].

In 1974 Mamdani proposed a *Fuzzy Inference System (FIS)* [12] to control a steam engine and a boiler. This FIS has since been called a Mamdani FIS and is one of the most commonly used fuzzy system in the controls industry, which is why it also our choice of FIS in our research. A Mamdani FIS depends on a good rule base provided by an experienced user or developed by some heuristic approach. The basic steps of a fuzzy system are shown here in figure 7.

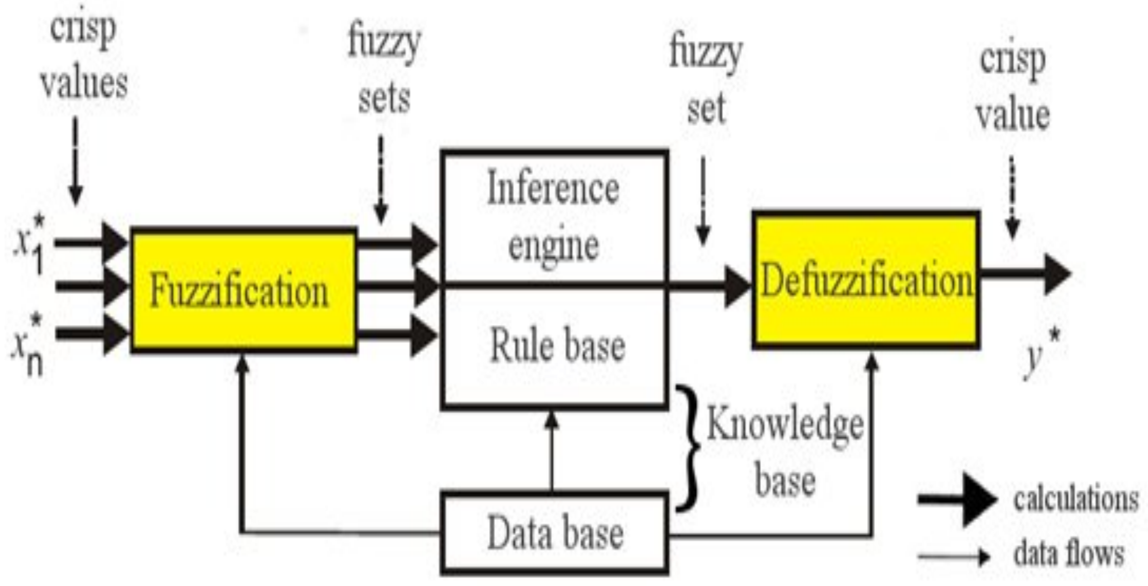


Figure 7. Figure illustrating the basic steps of a fuzzy system. A block in the diagram represents each basic step of the process. X_1… X_n are inputs to the system and Y is the output of the fuzzy system; Figure extracted from [4].

The Mamdani FIS is implemented in the matlab's fuzzy toolbox. The inputs (antecedents) entered in the fuzzy system are crisp and clear inputs, which are fuzzified by the input membership functions. Membership functions are just a representation of the inputs or outputs in common language terms. For example input membership functions for the tipping problem could be “good”, “average” and “bad”. These membership functions cover the entire range of the inputs and could be of any shape such as gaussian, trapezoidal, triangular, etc.

After the inputs have been fuzzified the fuzzy values are evaluated using the rule base that is specified by the user based on his understanding of the problem and his past experiences. These rules are written in a common easily understood language. These rules along with the output membership functions find the consequence of each rule (consequents). After adding all the weighted consequences from the rule base the FIS gets an output distribution which is then defuzzified using a defuzzification operator such as “centroid”, “center of mass”, “mean of maximum”, etc. The defuzzification process gives us the final crisp output we expect to get from a Fuzzy Logic Controller.

IV. Problem Description

A direct current (DC) motor is one of the most used motors in the industries all over the world due to its excellent speed control characteristics. This feature of the DC motors has attracted wide research attention in the engineering world on its speed and position control characteristics. The most common approach to control the position of the DC motor is using a PID controller.

The problem that lies in the system is the tuning of the PID controller. For each system using a PID controller, the PID controller has to be tuned to the right parameter values to give the desired output response and have the system perform in a desired manner. Tuning the PID controller is not an easy job, and requires a lot of manual fine-tuning to achieve the desired responses from a system.

John G. Ziegler and Nathaniel B. Nichols described one of the manual tuning methods in a paper published in 1942 [11]. The method requires setting Ki and Kd to zero and then raising the value of Kp manually till the system response starts oscillating with constant amplitude. The system is now said to have the ultimate gain (Ku). Ku and time period (Tu) of the oscillation are used to set the values of Ki and Kd. The tedious part with this method is to manually raise the value of Kp till the system starts oscillating with a constant amplitude and time period. The table, which shows the tuning process of the PID controller using the Ziegler Nichols method, is given here.

Control Type	Kp	Ki	Kd
P only	Ku/2	-	-
PI	Ku/2.2	1.2Kp/Tu	-
PID	0.6Ku	2Kp/Tu	KpTu/8

Table 2. This table tabulates the tuning formulas for the different parameters of a conventional PID controller using Ziegler – Nichols method. The equations given here have 'ultimate gain (Ku)' and 'time period (Tu)' of the system response as variables. The table can be used to tune a P, PI and PID type controllers; Table adapted from [7].

To solve this problem of tuning the PID controller we devised a novel mechanism using a fuzzy logic system with membership functions and evolved a rule base using EA. For this research we chose Differential Evolution (DE) out of the many types of EA's. Using EA would eliminate the need of manual fine-tuning and would give the desired system response, making this algorithm a better and more efficient way of tuning the PID controller to control the position of the DC motor. The algorithm and setup of the system are discussed in section V "Algorithm" and section IV "Experiment Setup" respectively.

V. Experiment Setup

The experiment was modeled on a closed loop feedback system (figure 8), which has a PID controller to mitigate the error between the desired response and the actual response.

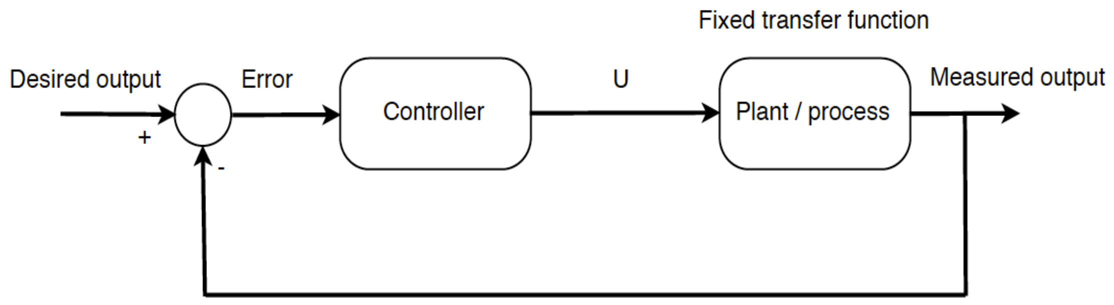


Figure 8. Closed loop feedback system. This diagram is a generic representation of a closed loop control system which uses a controller to minimize the error from the plant or a process; Figure adapted from [8].

A PID controller controls the position of a DC motor. The transfer function of the system shown below was calculated using the physical model of the system (figure 9).

$$\frac{\boldsymbol{\theta}}{\boldsymbol{V}} = \frac{\boldsymbol{K}}{\boldsymbol{s}\left((\boldsymbol{Js}+\boldsymbol{b})(\boldsymbol{Ls}+\boldsymbol{R}) + \boldsymbol{K}^2\right)}$$

Where,

J = Moment of inertia of the rotor

b = Damping ratio of the mechanical system

K = Ke = Kt = Electromotive force constant

R = Electric resistance

L = Electric inductance

V = Source Voltage

θ = Position of shaft

The three different system step responses (see figure 10a, 10b and 10c), which will be evaluated in this research are ‘slightly under-damped system step response’, ‘moderately under-damped system step response’ and ‘heavily under-damped system step response’. The system response from the slightly under-damped system is the desired system response thus making the system as our desired system. The three different system responses were achieved by varying the load on the motor. Since J is the moment of inertia component of the motor (figure 9), varying J would simulate three different loads on the motor thus giving us three different systems and three different step responses. By the research already done on DC motors it has been established that moment of inertia J is directly proportional to the load on the motor [13]. Figure 2 shows the physical model of the motor system.

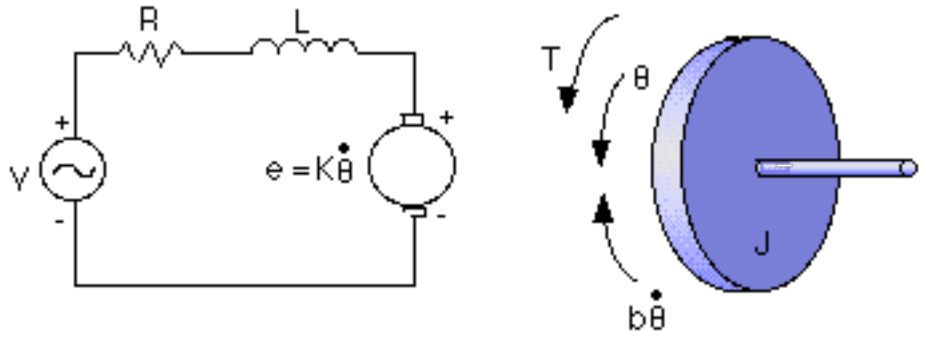


Figure 9. Physical and electrical model representations of a DC motor are shown above; Figure extracted from [9].

We use the base system (figure 9) and simulate the three different systems (three different loads) to note their initial step responses (see figure 10a, 10b and 10c). The blowups of the system step responses are also shown below (figure 10d, 10e and 10f). The controller in the loop represents a PID controller but we wanted to note the responses of the system without the PID controller so we set Kp=1, Ki=0 and Kd=0 thus effectively removing the PID controller from the loop. No disturbance signal or noise was introduced in the system. Since the aim of this research was to prove a concept we wanted to keep things simple but introducing the disturbance signal will also work with the system and should not be much different from the results we got.

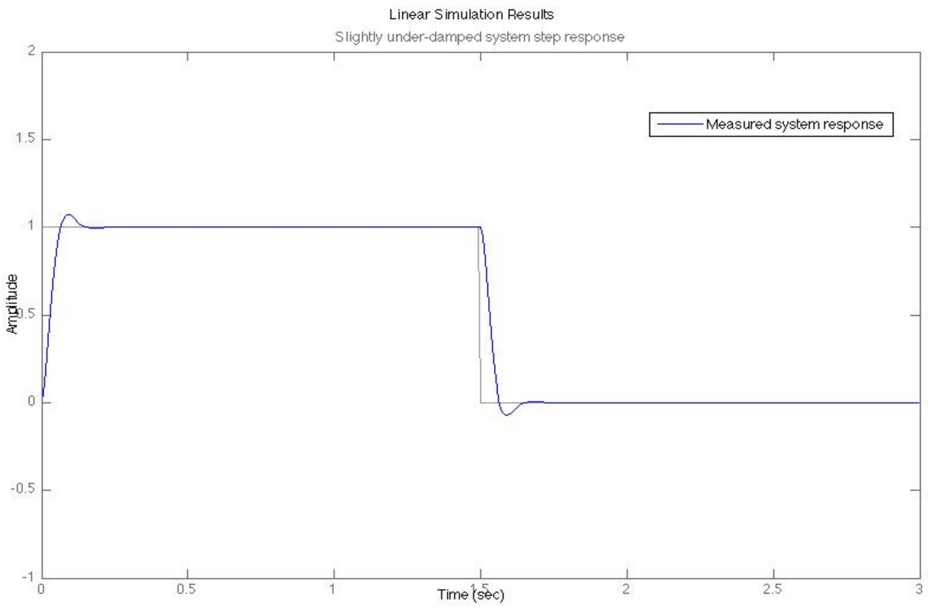


Figure 10a. Figure illustrates the system step response if the slightly under-damped (desired). Notice the low peak overshoot and smaller settling time.

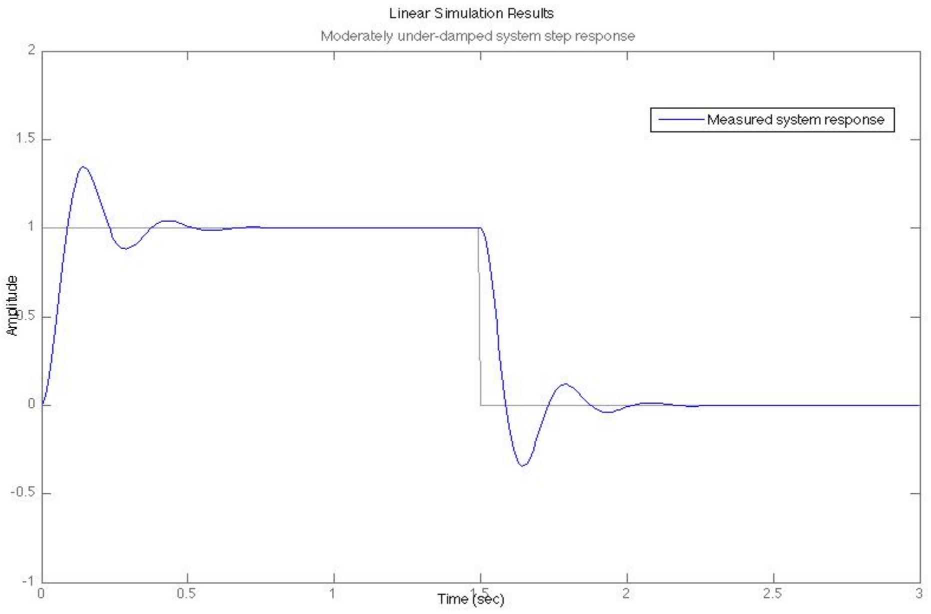


Figure 10b. Figure illustrates the system step response of the moderately under-damped system. Notice the higher peak overshoot and longer settling time.

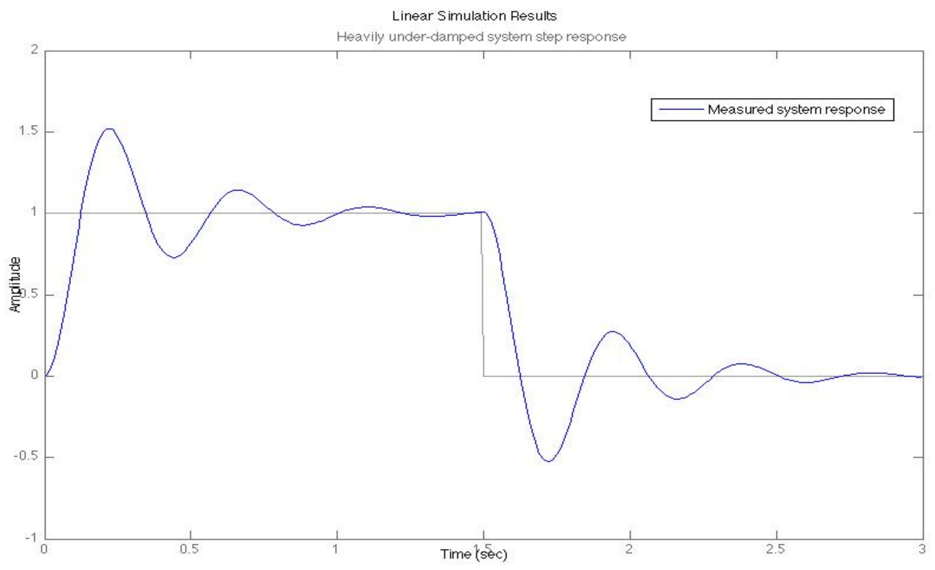


Figure 10c. Figure illustrates the system step response of the heavily under-damped system. Notice the very high peak overshoot and the longest settling time.

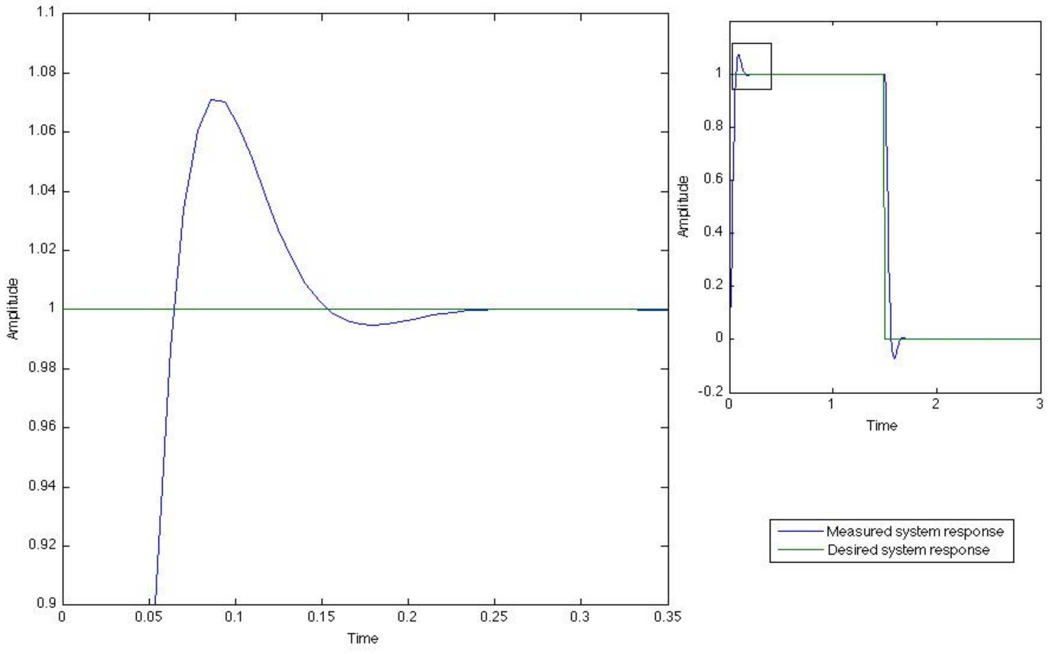


Figure 10d. A blowup of the slightly under-damped system step response is shown here. The peak overshoot and settling time is better visualized here.

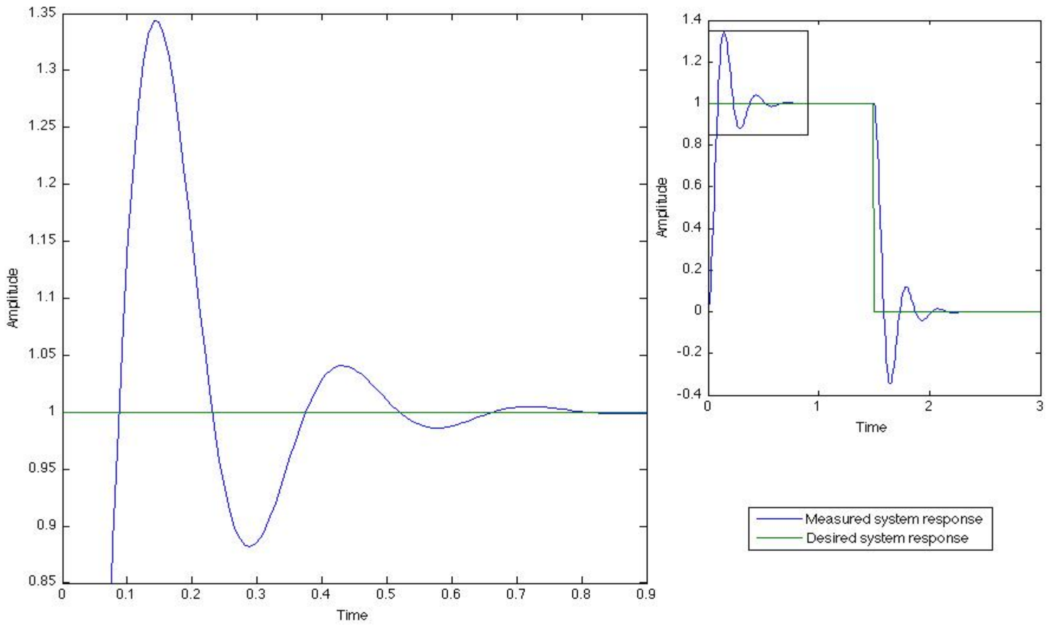


Figure 10e. A blowup of the moderately under-damped system step response. The peak overshoot is clearly higher than the desired system. The higher the peak overshoot the worse the system performance.

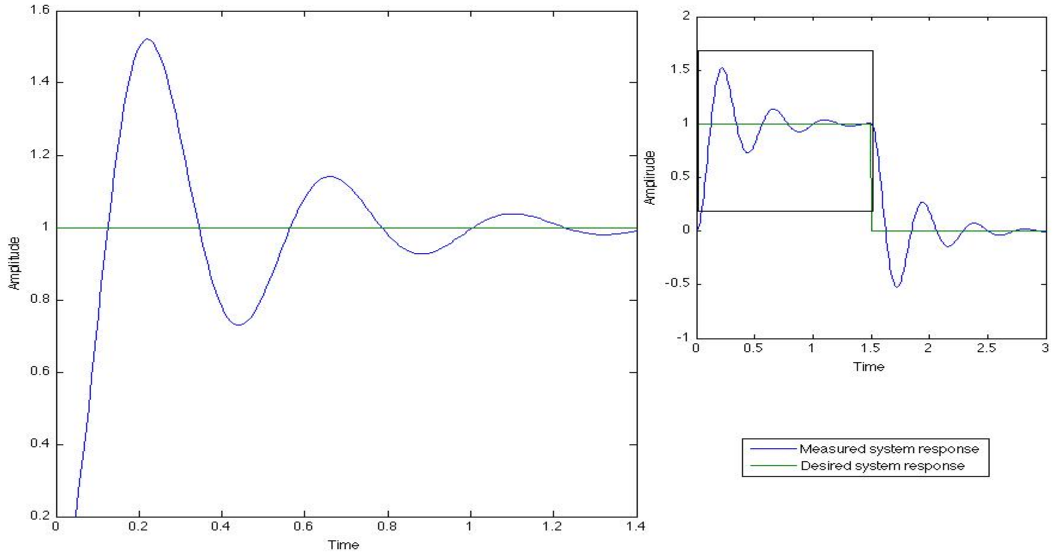


Figure 10f. A blowup of the heavily under-damped system step response. The peak overshoot for this system is the highest and hence this is the worst performing system out of the three systems under consideration.

To manually tune the PID controller we need to know the settling time and the rise time of the step response, which is difficult to compute in the time domain, as it is difficult to determine the starting point of the step response. However if we transform the step response in the frequency domain using the DFT function we will no longer have the problem of determining the starting point of the step response since the DFT will transform the function into a frequency domain function over one time period which will be easier to analyze in terms of the higher and lower frequency sub bands.

To eliminate the problem of determining the starting point of the step response and to transform the function into time domain we take the system step response of each system and find the DFT of the response. By taking the DFT of the system we were able to distinguish between the three responses in terms of the energy sub bands in the higher and lower frequencies of the system. The other benefit of checking the energy of the system is that we can use the difference between the energy of the systems as a merit of fitness for a given solution. This is essentially to say that the energy of the systems in the frequency sub bands could be used as a fitness function for our EA. After carefully studying the stem plots generated from the DFT of the systems we were able to single out an energy band, which was most different in the three system responses (figure 11a, 11b and 11c). A comparison of the three stem plots is shown in figure 11d. The three systems we used have different frequency components in the systems step responses that are clearly differentiated the systems. Each lobe in the plot corresponds to a frequency sub-band of the system. We after carefully analyzing the frequency responses of the systems using the stem plots identified eleven points (25 – 35) to successfully differentiate between the step responses of the systems. We then calculate the energy of that band using Parseval's theorem [14] and arrive at a value of energy for the desired response. This energy value is used as the fitness value in the algorithm.

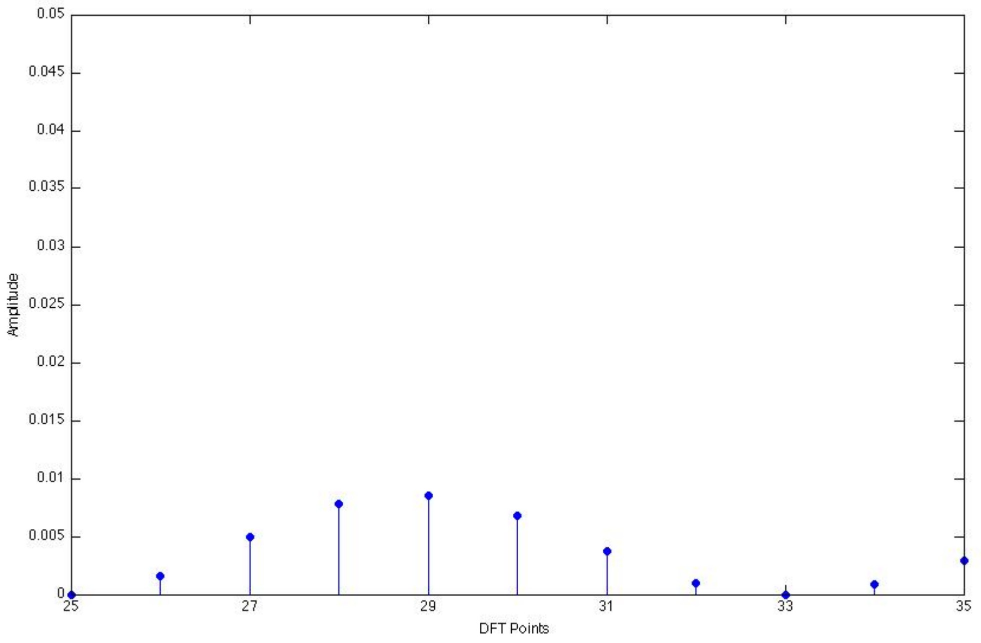


Figure 11a. Stem plot for point 25 – 35 for the slightly under-damped system. The smaller amplitudes of the stems are an indication of the presence of low frequency elements in the chosen frequency sub-band.

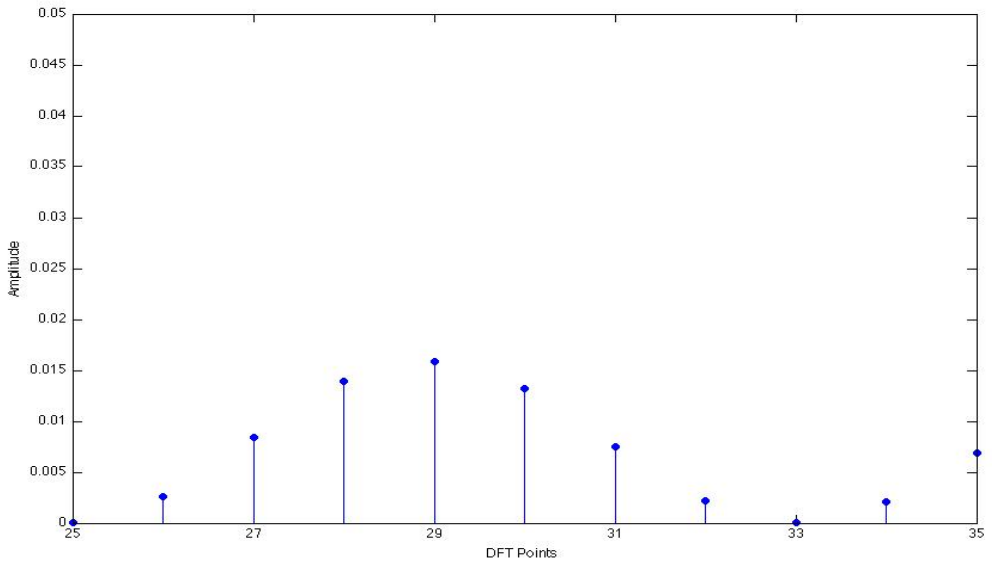


Figure 11b. Stem plot for point 25 – 35 for the moderately under-damped system. The presence of higher frequency elements here correspond to a higher peak overshoot in the step response of the system.

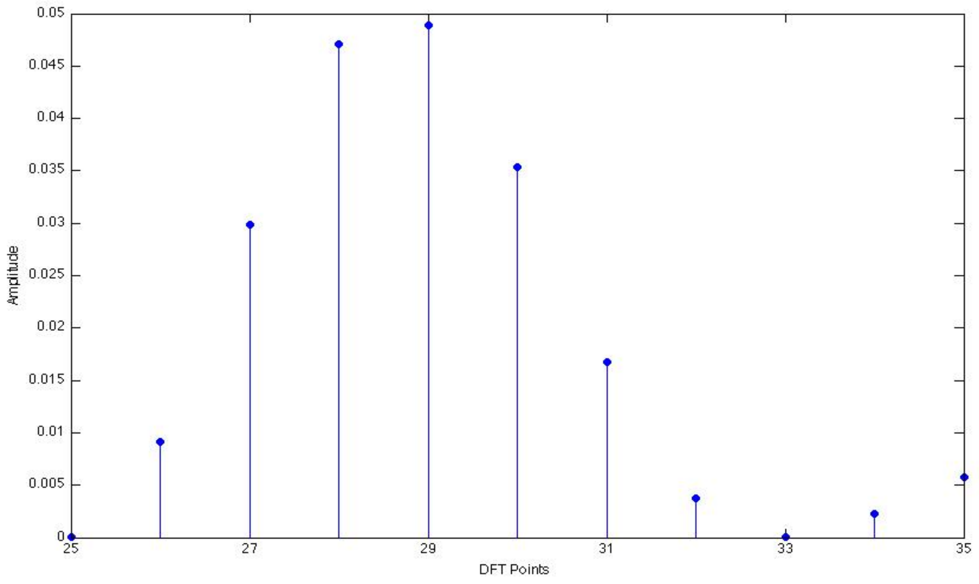


Figure 11c. Stem plot for point 25 – 35 for the heavily under-damped system. The high amplitudes correspond to the high energy found in the frequency sub-band due to the highest peak overshoot.

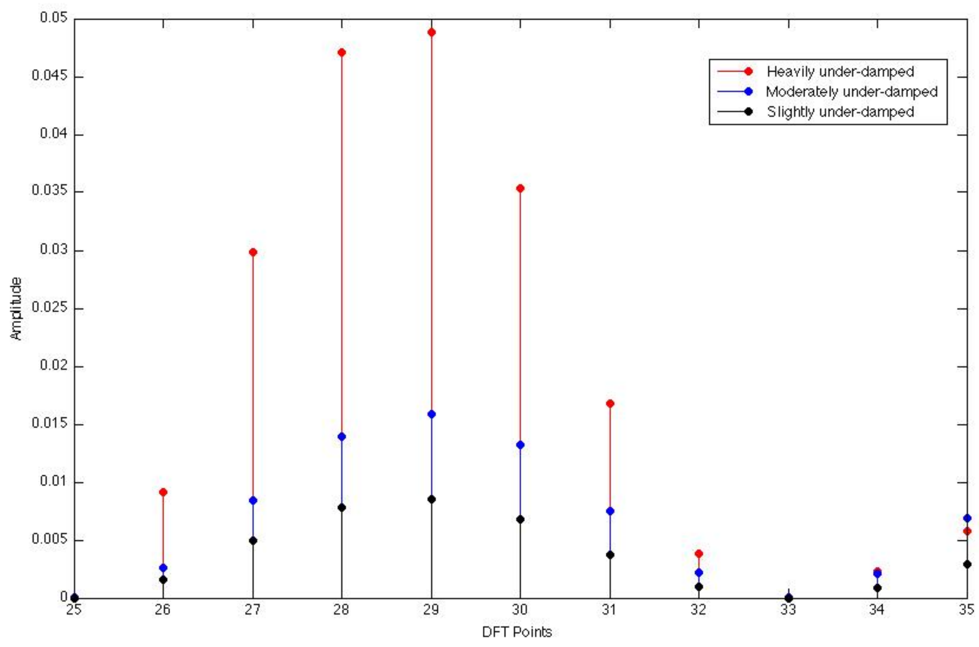


Figure 11d. Comparison stem plot of point 25 – 35 for the three different systems. The difference between the three systems can be clearly noticed here. The system's peak overshoot can be directly related to the energy in the frequency sub-band of the system. This difference was used to classify the systems.

Now we have the three system step responses of which one is the desired system response and the other two are the test system responses, which have to be restored to the desired system step response and we also know the energy values they should have for the specific energy band. We are going to design two fuzzy logic controllers, one for each test system to restore them to the desired system behavior. The fuzzy logic controller that we will be designing is shown in figure 12.

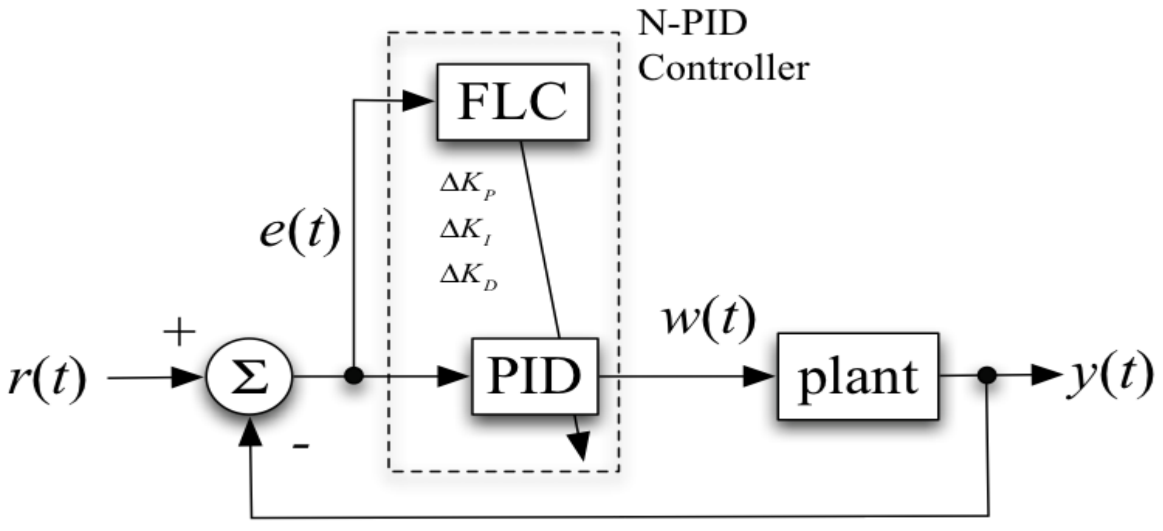
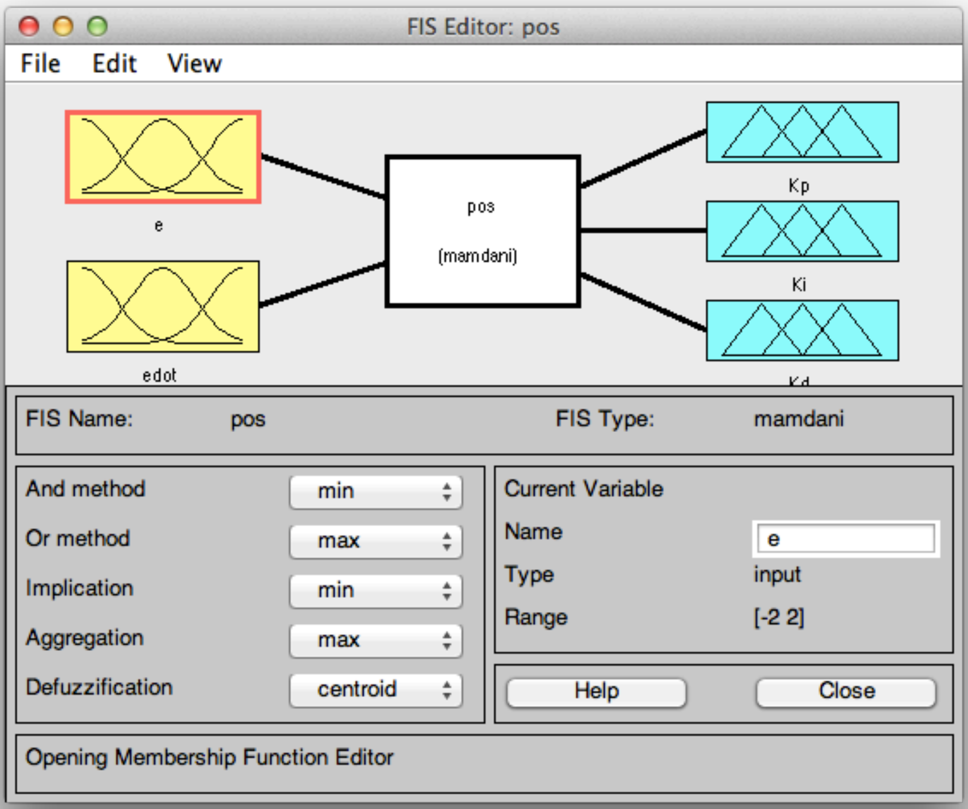


Figure 12. Block diagram illustrating the use of a Fuzzy logic controller to design the NPID controller system.

We designed a fuzzy logic controller with fixed membership functions but an evolving rule base to feed us ∆Kp, ∆Ki, ∆Kd offsets (figure 12). The fuzzy logic controller we use is a Mamdani type fuzzy system from the Matlab's fuzzy logic toolbox (figure 13). We have three membership functions (negative, no change and positive) for error (e) and rate of change of error (edot) signals. Error is calculated as the difference between the actual response and desired response at a single time step. Rate of change of error is the difference between error at the present time step and error at the previous time step. These e and edot values are fed to the fuzzy system, which calculates the ∆Kp, ∆Ki, ∆Kd offsets using the five membership functions, defined in the system (negative large, negative small, no change, positive small and positive large). Each ∆K offset has its own set of membership functions, which makes the fuzzy system have three output values for every input of e (error) and edot (rate of change of error).



FIS Name: pos

FIS Type: mamdani

And method: min

Or method: max

Implication: min

Aggregation: max

Defuzzification: centroid

Current Variable

Name: e

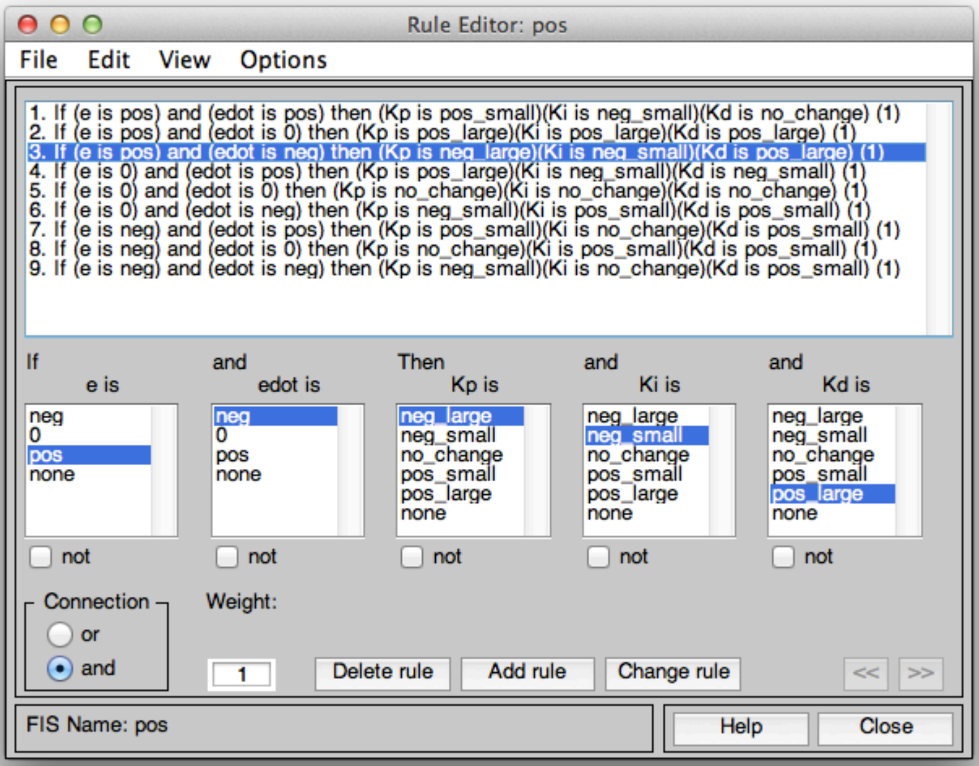
Type: input

Range: [-2 2]

Opening Membership Function Editor

Figure 13. A screenshot of Mamdani type fuzzy system implemented using the fuzzy logic toolbox in Matlab for our research. Here 'e' and 'edot' are inputs which are fed to the rule base which then gives the three outputs 'Kp', Ki' and 'Kd'.

Since there are three membership functions for the inputs (e and edot) there are nine possible combinations of the inputs thus there are nine rules to comprehensively cover all possibilities for each output. Therefore the fuzzy system has nine rules which each have two antecedents and three consequents (figure 14).



```
1. If (e is pos) and (edot is pos) then (Kp is pos_small)(Ki is neg_small)(Kd is no_change) (1)
2. If (e is pos) and (edot is 0) then (Kp is pos_large)(Ki is pos_large)(Kd is pos_large) (1)
3. If (e is pos) and (edot is neg) then (Kp is neg_large)(Ki is neg_small)(Kd is pos_large) (1)
4. If (e is 0) and (edot is pos) then (Kp is pos_large)(Ki is neg_small)(Kd is neg_small) (1)
5. If (e is 0) and (edot is 0) then (Kp is no_change)(Ki is no_change)(Kd is no_change) (1)
6. If (e is 0) and (edot is neg) then (Kp is neg_small)(Ki is pos_small)(Kd is pos_small) (1)
7. If (e is neg) and (edot is pos) then (Kp is pos_small)(Ki is no_change)(Kd is pos_small) (1)
8. If (e is neg) and (edot is 0) then (Kp is no_change)(Ki is pos_small)(Kd is pos_small) (1)
9. If (e is neg) and (edot is neg) then (Kp is neg_small)(Ki is no_change)(Kd is pos_small) (1)
```

If e is	and edot is	Then Kp is	and Ki is	and Kd is
neg	[✓ neg]	[✓ neg_large]	neg_large	neg_large
0	0	neg_small	[✓ neg_small]	neg_small
[✓ pos]	pos	no_change	no_change	no_change
none	none	pos_small	pos_small	pos_small
		pos_large	pos_large	[✓ pos_large]
		none	none	none
☐ not	☐ not	☐ not	☐ not	☐ not

Connection: ○ or [✓ and]

Weight: 1

FIS Name: pos

Figure 14. A Sample rule from the rule base for the fuzzy logic controller is illustrated here. The rule is read as "If e is pos and edot is neg then Kp is neg_large and Ki is neg_small and Kd is pos_large".

Our choice of EA was Differential Evolution algorithm (DE). DE is a very efficient EA, which is used in many engineering and related fields for solving optimization problems over a fitness landscape. The description of the DE we used is given below.

The individual we made consisted of twenty-seven problem parameters since there are three outputs (∆Kp, ∆Ki, ∆Kd) and each output has nine rules in the rule base of the fuzzy system (figure 15a). After encoding the individual we randomly generated a population of a hundred individuals using the equation (1) below.

$$\boldsymbol{x_i^j = x_{\min}^j + rand(0,1)\,.\,(x_{\max}^j - x_{\min}^j)}$$

Where i=1:100 (individuals), and j=1:27 (problem parameters).

Kp	Kp	Kp	…	Kp	Ki	Ki	Ki	…	Ki	Kd	Kd	Kd	…	Kd
1	2	3	…	9	1	2	3	…	9	1	2	3	…	9

Figure 15a. Configuration of an individual. Here the numbers signify the rule number in the rule base, for example rule number 2 would have its Kp, Ki and Kd as the 2nd, 11th and 20th element of the individual.

3	1	3	2	4	3	3	1	4	4	3	5	4	5	3	3	2	1	4	2	5	3	5	4	1	4	3

Figure 15b. Sample individual consisting 27 parameters. The elements of the individual signify the membership function of the consequents with 1 being neg_large and 5 being pos_large.

To create new offspring from the parent population we create a mutant vector. There are five different commonly used ways of mutating the population but we choose “DE/best/1” [10] which involves randomly selecting three different parents and then employing the equation (2).

$$\boldsymbol{V}_{i,G} = \boldsymbol{X}_{r_1^i,G} + \boldsymbol{F} \,.\left(\boldsymbol{X}_{r_2^i,G} - \boldsymbol{X}_{r_3^i,G}\right)$$

Where F is a user-defined constant, which is set to 0.24 in our algorithm. The value of F was chosen keeping in mind that the parameters should not go out of bounds. Since the maximum value of $\left(\boldsymbol{X}_{r_2^i,G} - \boldsymbol{X}_{r_3^i,G}\right)$ is four and four multiplied by 0.24 is 0.96 which makes sure that the parameter elements of the mutant vector do not go out of bounds.

Crossover operation is done using the equation (3).

$$\boldsymbol{u}_{i,G}^{j} = \begin{Bmatrix} \boldsymbol{v}_{i,G}^{j}\,, \boldsymbol{if}\ \left(\boldsymbol{rand}_j[\mathbf{0}\,,\mathbf{1})\ \leq \boldsymbol{CR}\right) \\ \boldsymbol{x}_{i,G}^{j}\,, \qquad\qquad \boldsymbol{otherwise} \end{Bmatrix}$$

Where, CR is a user specified constant within [0,1). We chose CR = 0.5 to implement binomial crossover. If any element of the offspring exceeds the upper and lower bounds of the gene we randomly reinitialize that element to within the bounds.

Fitness of the newly created offspring population is evaluated using the fitness function. ∆Kp, ∆Ki, ∆Kd values are calculated using the fuzzy logic controller which are added to the respective K parameters given from the PID controller in each generation. The fitness of each parent is compared with its offspring and the better-fit individual is introduced in the next population as parent for the next generation.

An example process of creating an offspring from a set of parents is summarized below.

First a population of 100 individuals is randomly initialized using equation 1. Three parents are then randomly selected.

Parent 1 (randomly selected from the population)

1	3	1	2	2

Parent 2 (randomly selected from the population)

3	1	1	3	2

Parent 3 (randomly selected from the population)

1	2	3	2	3

A mutant vector is formulated using the three parents and F=0.6 using equation 2. F was chosen to be 0.6 for calculation purposes in this example.

2	2	0	3	1

Now since the third element is out of bounds we will have to reinitialize it to within bounds so we randomly select the element from within the bounds. This gives us our final mutant vector, which would be used in the crossover operation.

Mutant vector (after initialization)

2	3	2	3	1

Crossover operation performed between parent 1 and the mutant vector at 50% probability. The crossover operation is performed using equation 3.

2	3	1	2	2

Offspring created from the parents

2	3	1	2	2

The DE parameters chosen for our experiment were –

- Population size = 100
- Mutation by “DE/rand/1” strategy
- Crossover rate = 0.5
- Fitness function = $|E_{system} – E_{desired}|$; Energy is calculated in the frequency sub-band of 25th harmonic to the 35th harmonic
- Selection strategy = comparison between parent and offspring
- Number of generations = 384

VI. Algorithm for the EA

```
Kp=1; Ki=0; Kd=0
Initialize a population of 100 individuals, each of which has an encoded rule base
for the fuzzy system
Start For loop, Generation = 0 : 384
    Simulate the system
    Calculate e and edot
    Using DE algorithm perform the crossover and mutation operations
    Start For loop i = 1
        Get ∆Kp, ∆Ki and ∆Kd for the offspring
        Simulate the system using the ∆K offsets
        Calculate the energy of the system
        Get ∆Kp, ∆Ki and ∆Kd for the parent
        Simulate the system using the ∆K offsets
        Calculate the energy of the system
        Compare the fitness of the parent to its offspring
        Keep the better-fit individual for the next generation
        i=i + 1
    End For loop
    Find the best-fit individual for the current generation
    Get ∆Kp, ∆Ki and ∆Kd
    Add the offsets to respective K parameters
```

```
        Generation = Generation + 1
    End For loop
```

VII. Results

Using the code we wrote, we were successfully able to restore the behavior of the two test systems (moderately under-damped and heavily under-damped) to the behavior of the desired system. We conducted twenty-five runs for each system to be sure of our results and find the best result. The graphs of the step responses for the three systems – desired system, moderately under-damped system and heavily under-damped system are shown in figure 16a, 16b and 16c.

The first step in the process was to generate step responses of the three systems (figure 16a, 16b and 16c), we then proceeded to get the DFT of the three step responses and generate stem plots for the first fifty points in the DFT. This step was aimed at getting some information on the frequency response of the systems and gets a clear differentiation between the three responses. The stem plots for the three systems with first fifty points plotted are shown in figure 17a, 17b and 17c.

The graphs of the step responses that are illustrated below each show the effects of varying the load on the DC motor. The varying load simulated by the code we wrote simulates three system step responses (slightly under-damped, moderately under-damped and heavily under-damped). The overshoot peaks and settling time of the response tells us about the damping of the system. More under-damping of the system would result in a longer settling time and higher peak overshoot.

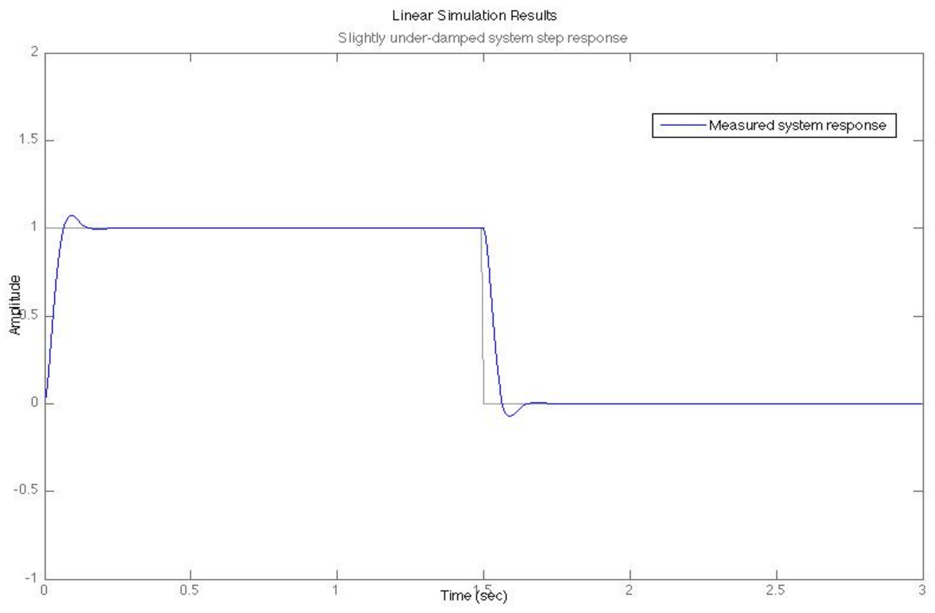


Figure 16a. Figure illustrates the step response of the slightly under-damped (desired) system.

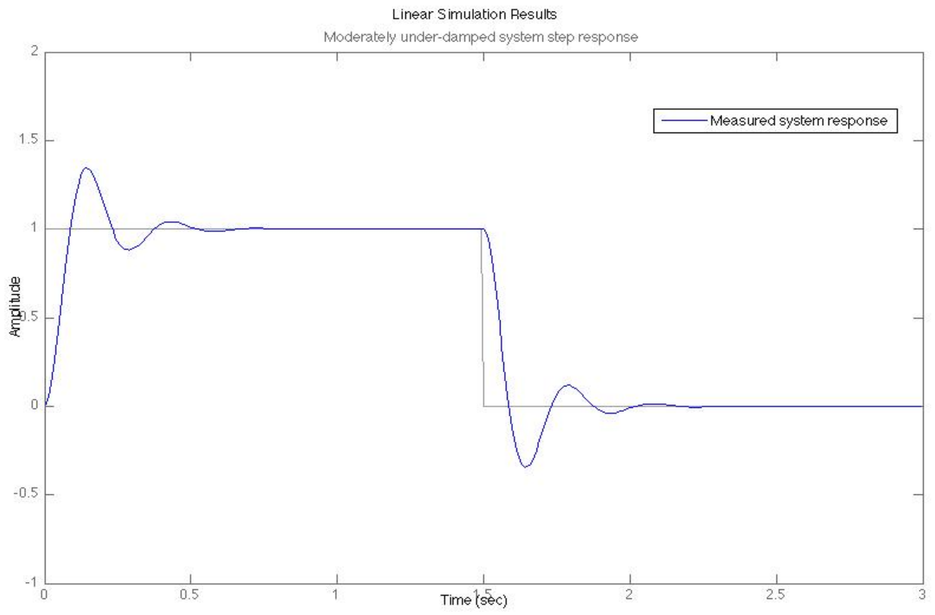


Figure 16b. Figure illustrates the step response of the moderately under-damped system.

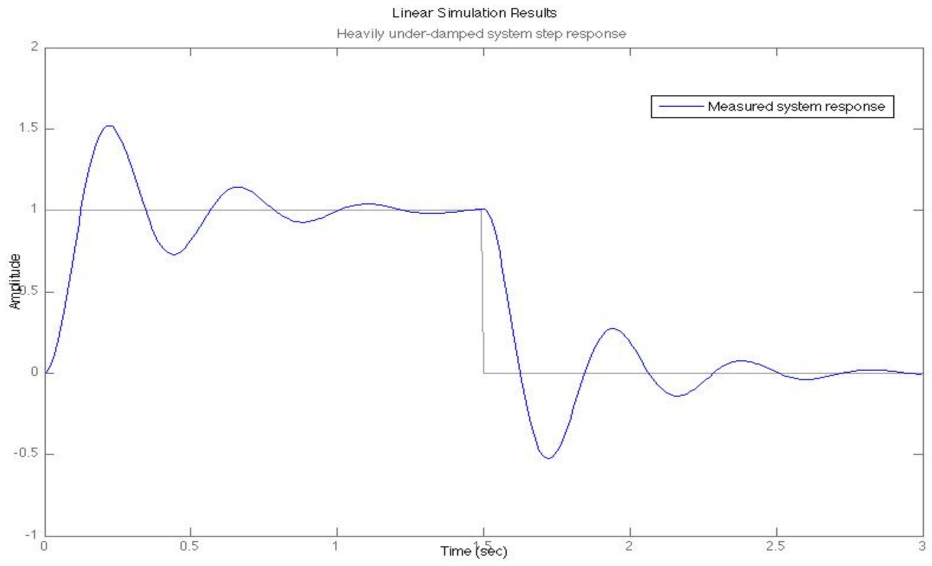


Figure 16c. Figure illustrates the step response of the Heavily under-damped system.

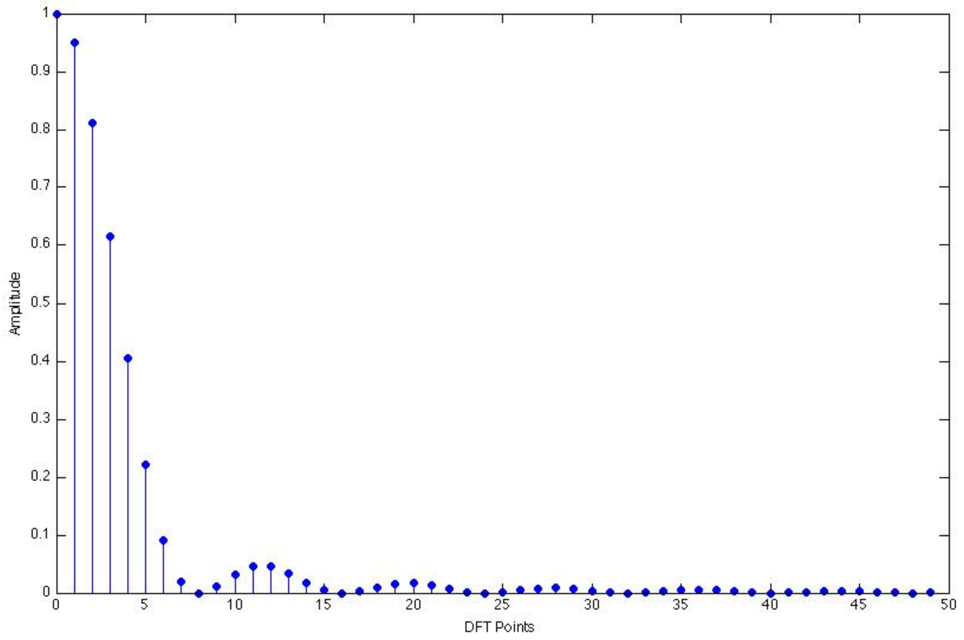


Figure 17a. The graph is the DFT of the slightly under-damped (desired) system (50 points). The graph indicates the presence of low frequency components due to the shorter peak overshoot and settling time of the system's step response.

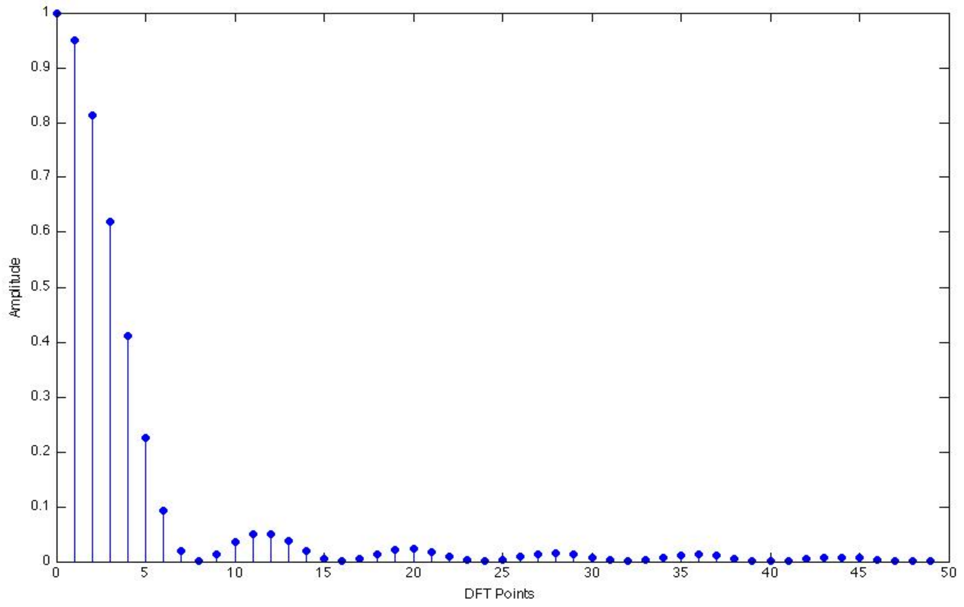


Figure 17b. The graph is a DFT of the moderately under-damped system (50 points). The stem plot here is more wavy indicating that the system has a higher peak overshoot and settling time.

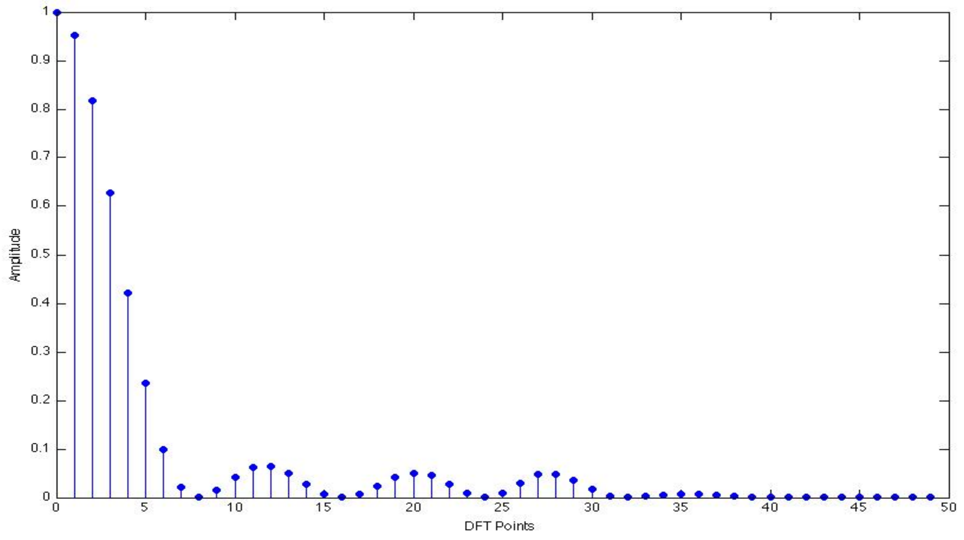


Figure 17c. The graph is a DFT of the heavily under-damped system (50 points). Higher frequency components of the step response indicating the highest peak overshoot and settling time are clearly visible from this stem plot.

Each lobe in the stem plots 17a, 17b and 17c is a representation of a frequency sub-band of the respective system. On careful analysis of the plots (17a, 17b and 17c) we decided to choose points 25 to 35 for the calculation of the energy values for each system since they are the most distinct looking frequency sub-band in the plots. We use this energy value as our fitness function to evaluate the fitness of each individual and ultimately get the optimal solution with the help of the EA. Figure 18 shows the comparison stem plots for the points 25 to 35 of the three systems. The graph clearly shows a distinction between the frequencies of the responses of the three systems

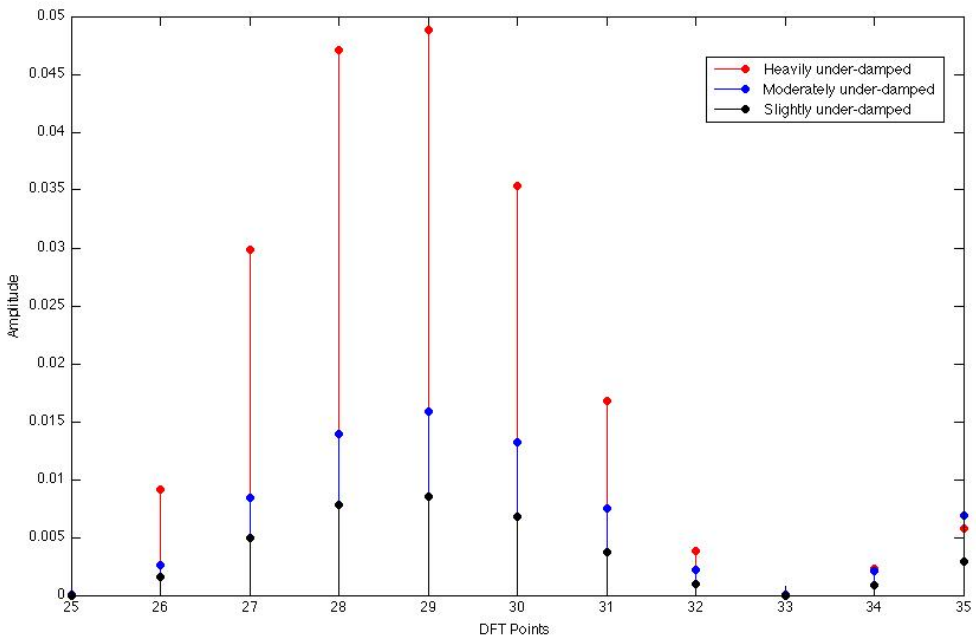


Figure 18. 11-point DFT plot for the three systems under consideration. The difference between the different frequency components of the three systems is clearly visible here. The higher the peak overshoots the higher the energy in the frequency sub-band.

We started off with the three systems and used the algorithm with each test system to restore it to the desired system. The algorithm was run for 384 generations and then terminated. After running the algorithm we found that the test systems were restored to the desired system (figure 19b and 19c). The step response of the desired system is also shown below for reference (figure 19a). The final population consisted of individuals with similar fitness levels and did not differ a lot from each other. This tells us that the solution space has plateaus of individuals of similar fitness levels instead of peaks. What was interesting was that after each run the K parameters were different but the energy of the system remained the same. This tells us that a system can be restored in more than one ways and will almost always have different K parameters but a similar if not the same energy value. This validates our choice of choosing energy as our choice of fitness function and demonstrates that different systems can be differentiated on the basis of their energy values.

While looking at the plots, one might point out the difference between the rise times of the restored systems in comparison with the desired system. This is because of our research effort was primarily based on restoring the peak overshoot of the step response and not necessarily the rise time of the system's step response. Minimizing the peak overshoot makes sure that the output signal follows the target signal as closely as possible and the error signal is minimized.

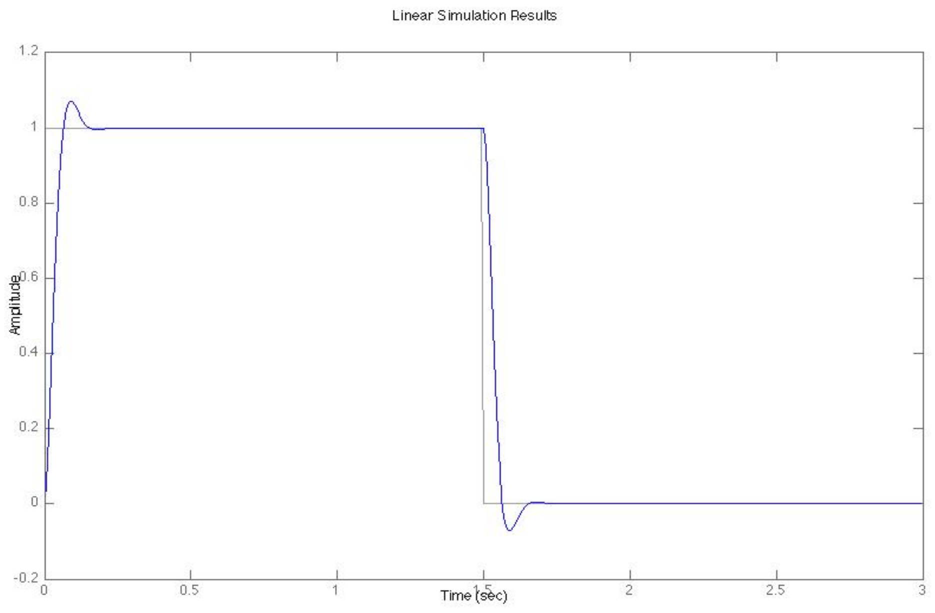


Figure 19a. The figure is the step response of the slightly under-damped system (desired system).

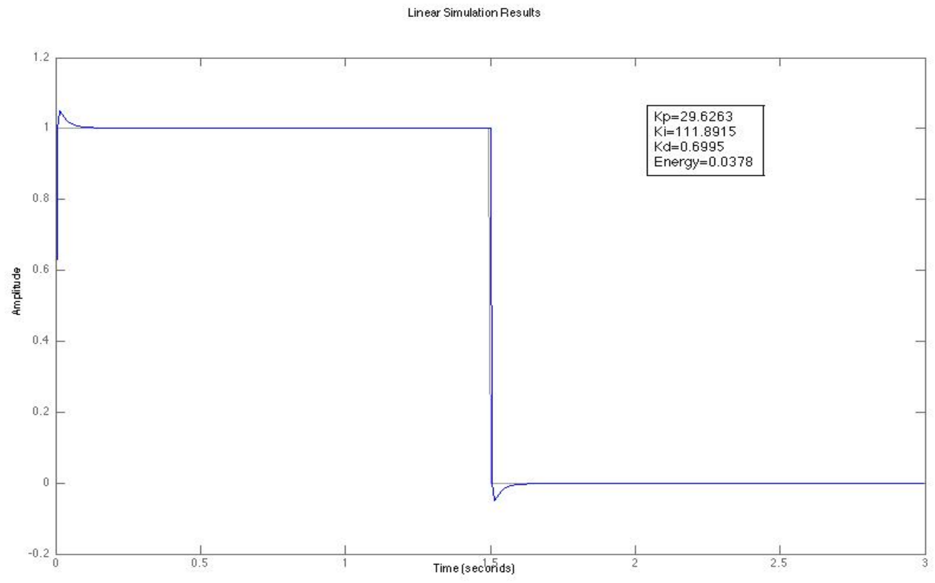


Figure 19b. Moderately under-damped system restored to the desired system. Notice the peak overshoot of the system has been reduced in the process of restoration. The minimized peak over shoot will ensure a similar performance to the desired system.

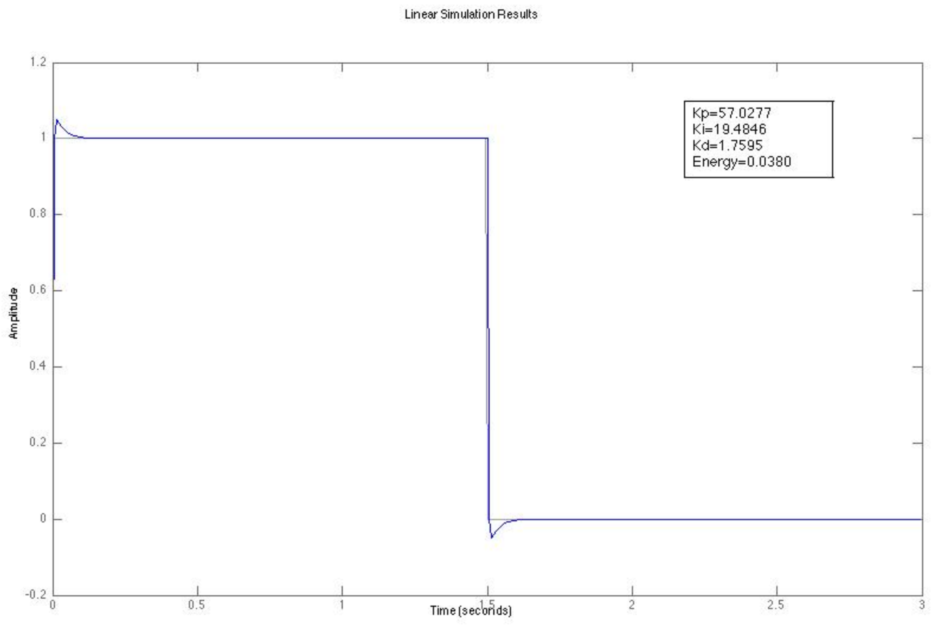


Figure 19c. Heavily under-damped system restored to the desired system. Notice the peak overshoot of the system has been reduced in the process of restoration. The minimized peak over shoot will ensure a similar performance to the desired system.

Figure 20 shows the evolution of the solution for the problem. The graph is a plot of the fitness value of the best-fit individual with respect to the generation. Here the lower the fitness value of the individual the better fit the individual so the aim of the algorithm was to achieve a fitness value of close to 0. The plot shows us that the algorithm was able to find a suitable solution with a similar energy level as that of the desired system in just a few generations.

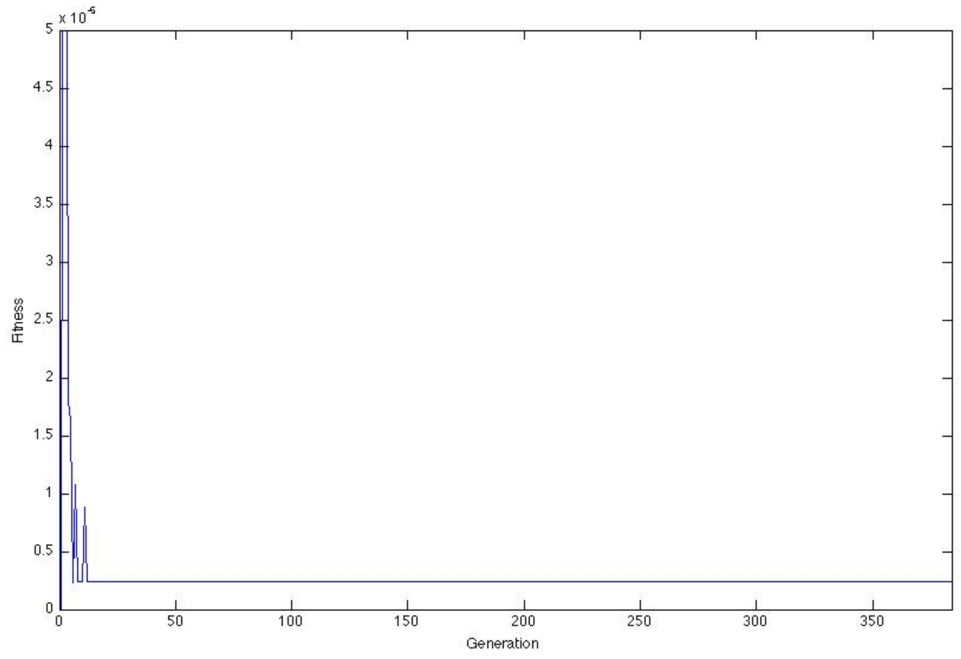


Figure 20. Fitness v/s generation plot for a typical run. The graph shows that an optimal solution was found in the first 20 generations of the run. Here solution having a fitness value closest to 0 is considered to be optimal.

While trying to restore the test systems to the desired system using our algorithm we also got some results, which had the same energy values, but the restored system response did not compare to the system response of the desired system (see figure 20). This result shows us that a test system while being restored can assume different gain parameter values but still have a similar energy value in the frequency sub bands.

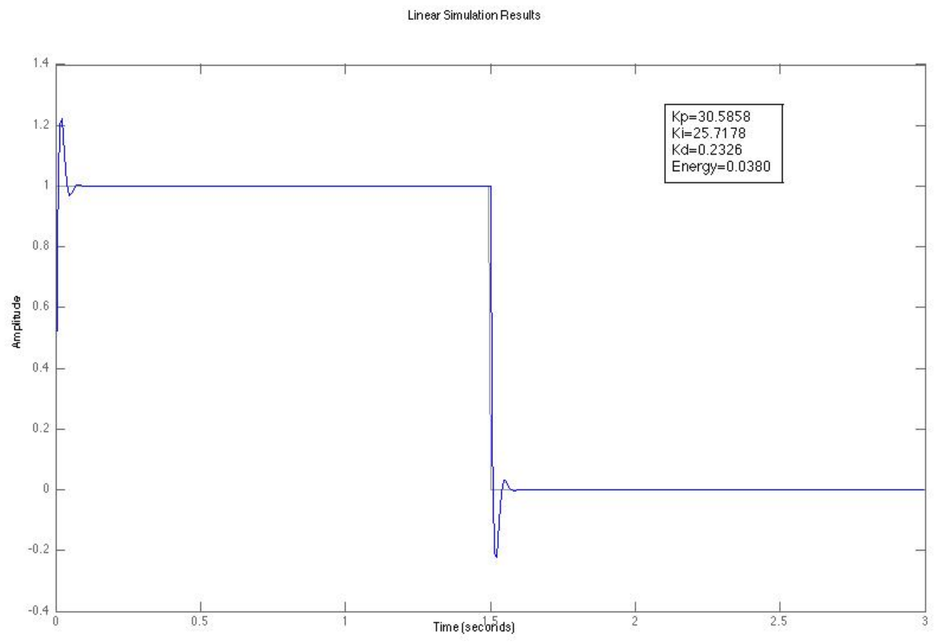


Figure 21. Moderately under-damped system restored to the same energy value. The figure shows that the peak overshoot for the system was not reduced but an energy level similar to the desired system was achieved making this solution an anomaly.

VIII. Conclusion

- We were able to show a heavily or moderately under-damped system can be restored to a slightly under-damped system's behavior by using a NPID controller.
- We were able to use the DFT effectively to calculate energy values of the systems under consideration to classify behaviors.
- We observed anomalies (i.e. same energy as desired system but radically different behavior). These were unexpected and suggest that our fitness function will require further analysis.

IX. Contributions

Contributions of our research work are listed below.

1. Use of Evolutionary Algorithm (EA) to implement an alternative method for tuning a PID controller that can be implemented on an embedded system. We used an EA to evolve the rule base for the fuzzy logic controller used as a part of the NPID controller.
2. The use of DFT to find the energy of the step responses of the system. DFT was used to transform the step response of the system from the time domain to the frequency domain, which helped in classifying the different test systems under consideration. It also was used as a measure of fitness of the evolved solutions.

X. Future Work

This research was done to present a proof of concept and there is a possibility of more work that can be done to improve the algorithm made by us. A few areas where the research work could be expanded are –

1. Research can be done to explore the possibility of evolving the system while it is online. Our algorithm requires the system to be offline while being tuned to the right gain parameters by the NPID controller.
2. We can expand the application of the algorithm by modifying the algorithm to adapt with dynamically changing unanticipated loads. In our research we used three different anticipated loads on the DC motor system that yielded three different but anticipated step responses.
3. If we refer back to the figures 17a, 17b and 17c, which are stem plots for the frequency response of the systems we will notice a big difference between the shapes and magnitudes of the stem plots between point 8 and 30. However when we compute the energy of these systems using Parseval's theorem the energy values are not very different. We could spend some more time analyzing the frequency response of the systems and formulate a more efficient and effective way of calculating the energy of the system.

XI. References

[1] IEEE Computational Intelligence Society. *Introduction to Evolutionary Computation* [Online]. Available: http://web.cecs.pdx.edu/~greenwd/EC.html

[2] Wikipedia. *Frog Lab Spring 2007* [Online]. Available: http://ediacaran.mech.northwestern.edu/neuromech/index.php/Frog_Lab_Spring_2007

[3] Carnegie Mellon (1997, August 26). *Control Tutorials for Matlab: PID Tutorial* [Online]. Available: http://www.engin.umich.edu/group/ctm/PID/PID.html

[4] J. Smolova and M. Pech (2010, January 12). *Fuzzy Expert System for Evaluating of Bargaining Power of Customers in SC* [Online]. Available: http://www.cvis.cz/eng/hlavni.php?stranka=novinky/clanek.php&id=68

[5] Mathworks. *An Introductory Example: Fuzzy Versus Nonfuzzy Logic* [Online]. Available: http://www.mathworks.com/help/toolbox/fuzzy/fp130.html

[6] Mathworks. *Overview of Fuzzy Inference Process* [Online]. Available: http://www.mathworks.com/help/toolbox/fuzzy/fp351dup8.html

[7] Wikipedia. *Ziegler – Nichols method* [Online]. Available: http://en.wikipedia.org/wiki/Ziegler%E2%80%93Nichols_method

[8] Carnegie Mellon (1997, August 24). *Control Tutorials for Matlab, Example: PID design method for the DC motor position* [Online]. Available: http://www.engin.umich.edu/group/ctm/examples/motor2/PID2.html

[9] Carnegie Mellon (1997, August 24). *Control Tutorials for Matlab, Example: Modelling DC motor position* [Online]. Available: http://www.engin.umich.edu/group/ctm/examples/motor2/motor.html

[10] K. Price and R. Storn. *Differential Evolution* [Online]. Available: http://www.icsi.berkeley.edu/~storn/code.html

[11] J. G. Zeigler and N. B. Nichols. "Optimum Settings for Automatic Controllers". *Transactions of the American Society of Mechanical Engineers*, pp. 64:759-768, 1942.

[12] E. H. Mamdani, "Application of fuzzy algorithms for control of simple dynamic plant". *Electrical Engineers, Proceedings of the Institution of*, pp. 12:1585-1588, 1974

[13] R. Rojas. Free University of Berlin, Berlin, Germany. *Models for DC Motors* [Online]. Available: http://robocup.mi.fu-berlin.de/buch/motors.pdf

[14] Wikipedia. *Parseval's Theorem* [Online]. Available: http://en.wikipedia.org/wiki/Parseval%27s_theorem

[15] R. Condit (2004). Microchip technologies Inc. *Brushed DC motor fundamentals* [Online]. Available: http://ww1.microchip.com/downloads/en/appnotes/00905a.pdf

[16] Bracewell, R. N. (2000), *The Fourier Transform and Its Applications* (3rd ed.), Boston: McGraw-Hill.

[17] Bäck, T. (1996), *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford Univ. Press.

[18] Bennett, Stuart (1993). *A history of control engineering, 1930-1955*. IET. pp. 48

[19] Dong Hwa Kim; Won Pyo Hong, Jae Ho Hwang and Seung Hack Lee, *"Intelligent tuning of the 2-DOF PID controller on the DCS for steam temperature control of thermal power plant", Industrial and Commerical Power Systems Technical Conference, 2002. 2002 IEEE*, vol., no., pp.1-13, 8-8 May 2002

[20] Dong Hwa Kim and Jae Hoon Cho, *"Robust tuning for disturbance rejection of PID controller using evolutionary algorithm", Fuzzy Information, 2004. Processing NAFIPS '04. Annual Meeting of the IEEE*, vol.1, no., pp. 248- 253 Vol.1, 27-30 June 2004

[21] Dong Hwa Kim and Kyu Young Lee, *"Neural networks control by immune network algorithm based auto-weight function tuning", Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on*, vol.2, no., pp.1469-1474, 2002

[22] Oh S.K., Lee D.K., Pedrycz W. and Kim D.H. (2004), "*The Genetic Design of Hybrid Fuzzy Controllers", Cybernetics and systems, Proceeding of the,* vol. 35, no. 4, pp. 333-361 June 2004.