Cloud Latency Optimization Using Mathematical Modelling

MSc Research Project
Cloud Computing

Douglas Sheriff Usman
Student ID: x17155924

School of Computing
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Douglas Sheriff Usman
Student ID:	x17155924
Programme:	Cloud Computing
Year:	2018
Module:	MSc Research Project
Supervisor:	Sean Heeney
Submission Due Date:	20/12/2018
Project Title:	Cloud Latency Optimization Using Mathematical Modelling
Word Count:	6849
Page Count:	44

Signature:	
Date:	18th December 2018

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Cloud Latency Optimization Using Mathematical Modelling

Douglas Sheriff Usman
x17155924

Abstract

Cloud computing has not ascended to the heights it was predicted to reach, recent network outage of Azure Cloud Service. Cloud users have been unable to tap into the infinite computing power that was promised with the birth of Cloud. This is due to the prolonged battle with cloud latency, which is the delay in communications between the cloud provider and cloud users. In this paper, we propose to explore the feasibility of mathematical modelling in mitigating cloud latency. We propose to focus on relevant methodology to actively model fuzzy logic control, implement the fuzzy logic control to mitigate cloud latency and evaluate the result from the research into this optimization process. This work is dedicated to all cloud users without specific IT skills to manage cloud services in their chosen cloud platform.
Keywords: Cloud; Networking; Fuzzy Logic Control; Fuzzy Inference System; Latency Optimisation; Mathematical Algorithms; Mathematical Modelling; Quality of Service.

1 Introduction

The term cloud computing does has series of definitions with all focused on sharing of computer resources via the Internet (Labba et al.; 2018). Cloud computing makes it possible to achieve coherence and economies of scale by sharing resources (Liu et al.; 2016). These attributes make cloud computing commercially acceptable, globally. Presently, the world of cloud computing presently, is mainly commercially oriented with business focused clouds being the expected platform for researching (Rao et al.; 2011). The eventual evolution of data centres from the traditional one server to modern-day cloud computing environment is an indication of how rapidly the field of IT is evolving. The driving forces behind this evolution are Moores law and the demand of end users with the latter not showing any sign of slowing down (Kashyap and Viradiya; 2014).

Quality of Service (QoS) is the description of the entire performance of a cloud computing service (Hans; 2018). Since cloud provides shared resources via the Internet, recent cloud environments are believed to be hitting climax with milestones about to be broken in terms of storage, computing power, cost and security. As a result, QoS has been greatly threatened in keeping up with demands. Latency which is a major problem to Internet providers is described as the delay in sending information from the provider to the end-user (Shrestha et al.; 2013). When latency occurs on the cloud, it is known as cloud latency (Lyu and Zhang; 2015). Internet being the only medium sharing resources on the cloud, this makes latency a challenge in the cloud environment.

Latency on the cloud is such a major challenge that it was highlighted as one of the very serious drawback in cloud computing courtesy of Rajkumar Buyya (Zhang et al.; 2010). Presently, demand for high performance computing resources is at an all-time high backed by the recent emergence of privately owned high performance computing (HPC) systems (Jarschel et al.; 2011). With cloud latency in the mix, the chain reaction has led to inconsistent performance from cloud services. Organization and new comers will have to make alternate arrangements (alternate compute power) to sustain future, very high performance because of the rising latency.

Amazon Web Services (AWS) resolved some of these issues by patching the kernel version and introducing auto scaling capacity. Nevertheless, latency and throughput issues persist because auto scaling capacity takes time to scale up and down leading to lag and bottleneck respectively. This research project would attempt to answer the pertinent question:

"Can we optimize cloud's latency using mathematical modelling?"

The term mathematical model is defined as describing a system utizing mathematical concepts and languages (Department of Computer Science Engineering, Jain University et al.; 2014). A mathematical model can be designed, modified and implemented on any system. Examples of good mathematical modelling software are Mathematica, Simulink, MATLAB and IBM SPSS (Sturm; 1999). By this medium, an experiment to mitigate latency issues, using mathematical modelling is very possible. (Bashar; 2014) stated that as far back as 1968, engineers were able to generate a mathematical model for an operational radio. Though this feat is regarded as obsolete technology in modern day, it is a testament to how mathematical modelling is evolving and compatible with various control apparatus. An example of utilizing mathematical model is the bee colony. The bee colony was the main inspiration that led to load balancing of project in cloud computing (L.D. and Venkata Krishna; 2013).

Fuzzy Logic Control (FLC) is a subsection of mathematical modelling, which is a logic control paradigm that is nonlinear in nature with the capability of deducing complex nonlinear connection between input and output variables (Mendel and Mouzouris; 1997). Mathematical modelling is the conversion of applied or application problem into controllable mathematical formulations whose numerical analyses provide reasonable answers and procedure to more often than not, solve problems (Chen; 2011). Mathematical model is the system that houses numerical concepts and languages. A mathematical model is created and simulated on a computing platform like MATLAB that can interface with a wide range of programming language, algorithm and matrices ((Chopra; 2012), (Lofberg; 2004) and (Selinger and Katze; 2013)). This is the reason that FLC could easily run on the mathematical model platform (Chopra; 2012). A calculation platform like MATLAB and a mathematical model like Simulink are the mathematical computer software that are most popular in terms of mathematical modelling (Entchev and Yang; 2007).

In this paper, we propose to explore the feasibility of mathematical modelling in mitigating cloud latency. We propose to focus on relevant methodology to actively model FLC, implement the FLC to mitigate cloud latency and evaluate the result from research into this optimization process. To mitigate latency, we hope to understand how to manage and detect latency thus researching into auditing, low-latency data labelling and cloud gaming would help in understanding cloud latency a lot more.

The paper is organized into 6 sections. section 2, examines related works on existing approaches and projects on managing and detecting latency as well as mathematical modelling and FLC applications. Section 3, details our Artifacts methodology. Section 4 implements this Artifact to mitigate cloud latency. Section 5 evaluates result of the Artifact and discusses the result. Section 6 concludes paper and presents perspectives in cloud computing.

2 Related Work

In this section we considered existing projects, approaches and concepts related to FLC, understanding latency and mathematical modelling. The research on Auditing Cloud Latency, Speeding up for Low-latency Data Labelling and Executing High-Demand Applications on the Cloud assisted us in understanding cloud latency in terms of managing and detecting latency. The research on mathematical modelling assisted us in understanding behaviour of the Artifact.

2.1 Auditing Cloud Latency

The term audit is the systematic screening of vouchers, statutory records, books, accounts and documents of an organization to establish its fair view (Moulder et al.; 2010). The same definition can be related to auditing on the cloud in terms of cloud latency. (Tomanek and Kencl; 2013) focused on latency and its impact on cloud computing. This was important because it ascertain how a remote data centre would perform while it is in full operation. The aim was to design tools or methods to monitor and optimize international cloud service latency. The methodology was to utilize Cloud Latency Auditing (CLAudit), a prototype planetary scale cloud latency auditing platform (Selinger and Katze; 2013). CLAudit made use of PlanetLab network to site worldwide distributed probes that continuously measure cloud service latency at various layers of the communication stack. The test was carried out on the platform using original deployment, which showed that CLAudit could detect both normal and abnormal cloud behaviours. Test results were able to pinpoint problem using data analysis thus confirming how valuable CLAudit could be to both research community and cloud service providers (Shrestha et al.; 2013).

The test concluded that with new cloud networking protocols, proposed future technologies would address the different applications latency (Touch; 2010). For now, there is none (Zhu et al.; 2013).

2.2 Speeding up for Low-latency Data Labelling

In a research by (Haas et al.; 2015), a way was discovered to increase the speed of data labelling without impeding evolution of interactive systems. There has been an increase in demand for data labelling according to (Haas et al.; 2015) but with the absence of work addressing and the presence of unpredicted latency, there is need to improve the speed. This report introduced CLAMShell, which is a system that should improve the speed of crowds to achieve consistent low latency. Studies of large crowds for labelling deployments, increased understanding of latency. Tackling every labelling-source of latency led to state-of-the-art techniques that was able to optimise current methods like active learning. CLAMShell, was evaluated in simulation and on live, which showed that the proposed techniques would provide the extent of speedup and variance reduction over current alternative strategies (Edmondson et al.; 2012).

The test concluded that, CLAMShell takes broad approach to the reduction of latency for data labelling (Haas et al.; 2015). The test also demonstrated that work could be conducted in the system that trains a single-model on the section labelled by active and passive learners (hybrid learning). The hybrid learning platform would be studied to discover if better models can be trained (Lyu and Zhang; 2015).

2.3 Executing High-Demand Applications on the Cloud

In a report by (Chen et al.; 2011), there was a burning desire to answer the question; how good is the real-timeliness of current cloud gaming systems? To answer the question, the response latency of two cloud gaming platforms ONLive and StreamMyGame, were analysed. A general methodology was proposed to measure the latency of cloud gaming systems with little or no exceptions. This methodology was applied to the two cloud gaming platforms and was observed that the OnLive platform had the most reasonable latency for real-time cloud gaming compared to the StreamMyGame which was twice that. They concluded that their methodology should help in understanding latency in PC-based cloud gaming platform and should results in further improvements in cloud latency.

A report by (Choy et al.; 2012) commend the changes that cloud computing has brought in terms of deployment of web application and services with emphasis on gaming. It also brought forward a limitation in terms of architecture of the cloud platform. The limitation is that several cloud computings vital design such as fine-grain partitioning or consolidating resources do clash with recent multimedia applications. These applications are highly sensitive to latency and most times would require unique hardware like fast memory. A study was undertaken that placed all the emphasis on on-demand gaming. This study included an extensive evaluation of the present state of cloud computing infrastructure using a case study from Amazon EC2, measurement of settings (BitTorrent), the effects of large cloud infrastructure and the probe of the impact of increasing present cloud infrastructure with servers detected close to the end-users (Shahdi-Pashaki et al.; 2018).

From extensive evaluation of current cloud infrastructure, it was proven that decent results was achieved. This results were only achieved by providing local content distribution servers with the essential hardware to aid gaming demands thereby expanding current infrastructure. This resulted in an addition of 28 percent of the end-users that could meet the required 80 milliseconds (ms) response time. (Choy et al.; 2012) trusts that by including current resources into cloud data centres, on-demand gaming could improve notably and possibly revolutionize the multi-billion-dollar video game industry.

2.4 Mathematical Modelling of Cloud Services

There were research works in mathematical modelling of cloud services, which were carried out and were believed to be the future of cloud services (Lyu and Zhang; 2015). A report by (Islam et al.; 2012) included developing an elastic model for cloud instances using mathematical modelling that considers all resource types, network bandwidth and so on that could be allocated with users permission and users could convert measured overprovisioning and underprovisioning into approximate financial results. This designed model also added cost of provisioned but did not make use resources and performance degradation costs to mimic underprovisioning. Users issue in terms of overprovisioning is the distinction between the imposed demand and imposed supply. Underprovisioning on the other hand is the measurement done from an observation of the percentage of refused requests point of view. (Islam et al.; 2012) paper concluded that by running different workloads on cloud, closely studying results and calculating geometric mean of costs, it was possible to effectively measure underprovisioning and overprovisioning metrics, respectively.

(Abdelsalam et al.; 2009) examined the mathematical correlation between Service Level Agreements (SLAs) that control most cloud applications and number of servers used in operating them. The assumuption was that the cloud is consistent, each device could operate at its own frequency with varied power consumption. SLAs also identified each clients requests by gathering the computed demand of applications executed. Examination was achieved by assuming that one job would be divided evenly over multiple servers with focused applications being web applications and web services. (Abdelsalam et al.; 2009) proposed and effectively executed a successful examination with correlation between SLAs and the number of server in operation being the highlight of the research.

A report by (Gelenbe et al.; 2012) solved the option between a local or remote cloud service in terms of the energy and QoS. This study created a mathematical formula, based on the PollaczekKhintchine formula and estimating Poisson arrival that mimics optimization issue of the load sharing between a local and a remote cloud service. Service time and power consumption (output) for a server was duly observed. This observation brought about the conclusion that by regulating the system load between local and remote cloud services the best result would be achieved (Smola and Schlkopf; 2004). Thus, they attained the best balance between energy consumption and service times.

(Mi et al.; 2010), also generated a formula that represented the multi-constraint optimisation problem, which finds best number of physical machines that makes best use of their resource while reducing power consumption. It was assumed that the Central Processing Unit (CPU) is the only resource of a physical machine (Sakellari and Loukas; 2013). Forecasting load of applications by using the Browns quadratic exponential smoothing formula and proposed a self-configuration genetic algorithm to get best reconfiguration policy (Mi et al.; 2010).

2.5 Fuzzy Logic System and its Application

An understanding of fuzzy logic system via mathematical model has been researched into and documented ((Entchev and Yang; 2007) and (Ruhit et al.; 2018)).

A research paper by (Frey et al.; 2013), looked in using fuzzy logic to regulate QoS of cloud services. The intention was to achieve desired result by utilizing merit of the cloud, which is the ability of scalability to monitor performance load metrics. The system was monitored and decision made from initial observation either to scale up or scale down the process by provision or revoke of cloud resources. This action assured the QoS and invoked the Service Level Objectives (SLO). The purpose of research was to demonstrate the normal cloud computing scaling service that ensures QoS.

The research concluded by showcasing essential parameters like request-response-time which highlighted the real-time QoS of the cloud. The use of the fuzzy control module for this project, brought about an extended QoS provisioning architecture.

This paper by (Sule et al.; 2017), looked in the reliability of the cloud platform in terms of trust and security. The project focused on creating a layer of security and trust using the FLC. The result was the creation of a Multi-Layer Trust Security Model

(MLTSM) that utilizes the fuzzy logic combination of various security mechanisms like in-direct and direct trust.

The project concluded with MLTSM being able to improve the security of deployment within cloud infrastructure in very critical area like the electrical power system and the cooling sensors. This was achieved by the result from the MLTSM that showed the on-demand verification and determination of the trust level of all proposed cloud computing platform.

2.6 Discussion

In this subsection we would discuss the above related works, highlighting research works with the strongest ideas that are related to our project proposal and comparing all the above research works with each other.

Cloud latency is essential to the operation and evolution of the cloud environment. The works of ((Haas et al.; 2015), (Chen et al.; 2011) and (Choy et al.; 2012)) examined latency in terms of heavy applications like gaming on the cloud, which requires very low latency. We were able to understand the structure and how to formulate latency. Cloud environment in practical terms is random at best. Most mathematical models cannot effectively represent cloud but the article, *Mathematical Modelling Project for Cloud Systems* best describes cloud environment and how to generate a formula for it.

The works from ((Frey et al.; 2013) and (Sule et al.; 2017)) shows that FLC can be used in the cloud platform to guarantee QoS and enforce security respectively.

Table 1 below, Table 1 below, highlights key findings from review of other research papers. This is in comparison with terms of latency, mathematical modelling, networking and content.

For clarity: A1 refers to understanding of latency; A2, measurement of cloud latency; A3, application of mathematical model; A4, networking with physical devices; A5, information presented; P1, Auditing Latency in the Cloud; P2, Speeding up for Low-latency Data Labelling; P3, Compilation of Case Studies in Cloud Gaming; P4, Mathematical modelling project for cloud systems and P5, approach Fuzzy Logic Inference system and its Application. Finally, Y refers to YES, N refers to NO and (-) refers to undetermined.

Table 1: Important elements based on other existing projects relating to Modelling

	A1	A2	A3	A4	A5
P1	Y	N	Y	-	Y
P2	Y	-	-	-	Y
P3	Y	-	Y	-	Y
P4	-	Y	Y	Y	Y
P5	-	-	Y	-	Y

Table 1 is the results of comparing all the literature or journal with each other. This was necessary to evaluate the present level of all the research materials. The table shows the elements that makes up this research project and would determine the direction of the research project.

From Table 1, some elements came close to our proposed approach. Elements such as P5 that used MATLAB to model a fuzzy logic for QoS of the cloud deployment infrastructure. The research from (Mi et al.; 2010) generated a formula using mathematical modelling to represent the multi-constraint optimization of a CPU of a physical machine.

The research work from (Haas et al.; 2015) proposed that the next phase of evolution would depend on reduction of cloud latency. The research work by P5 however predicted future work in employing real test environment in developing user friendly interface using fuzzy inference system.

2.7 Concluding Remarks on Related Works

Previous research works provided support for the idea that optimization issues could be resolved using mathematical modelling with a particular emphasis on FLC. These issues which are very complex for the traditional models can be solved with FLC and further using controlled mathematical modelling. The literature review helped us in understanding latency, which is the current challenge that plagues the cloud platform. It also, brought our attention to the very significant impact of high latency and why cloud has to overcome latency to advance to the next stage of cloud evolution.

Consequently, no suggested approach fulfills requirements to tackle all of the important elements in Table 1. These important elements included latency, mathematical model, network and content (A1, A2, A3, A4 and A5). With these limitations brought by the literature, there is a need for a feasible approach to solving cloud latency and we are going to implement in the report.

At the introduction of this section, we highlighted an Artifact that would address cloud latency. We strongly believe that through FLC, we should mitigate latency in the cloud platform. The implemented FIS, utilizes the FLC that accepts erratic wave pattern and tolerant to nonlinearities.

In section 3, we will look at the methodology of the proposed Artifact.

3 Methodology

In this section we are going to outline the proposed research approach and method to designing the Fuzzy Inference system (FIS). We will justify the choice of our proposed method, and consider FIS functional goal.

3.1 Approach

FIS is the mathematical modelling method that we employed to mitigate cloud latency issue in cloud platform. We would further explain the operational approach of this Artifact in Figure 1. Figure 1 is the operational approach of FLC that is to be attached to end user in the cloud ecosystem.

Figure 1 is the activity diagram of the FLC integrated with the cloud environment from the view of the end-user. It demonstrates how FLC integrated to the end-users cloud framework to form the FIS. End-user would access cloud network via the Internet and should automatically activate the FIS when there is a sudden prolonged downtime.

The FIS, highlighted in red, analyses the input (Fuzzifier), regulates the cloud network (Inference), bind the input by certain entities (Rules), fine tune the output (Defuzzifier) and merge the output (Linearization) (Kasabov and Song; 2002).

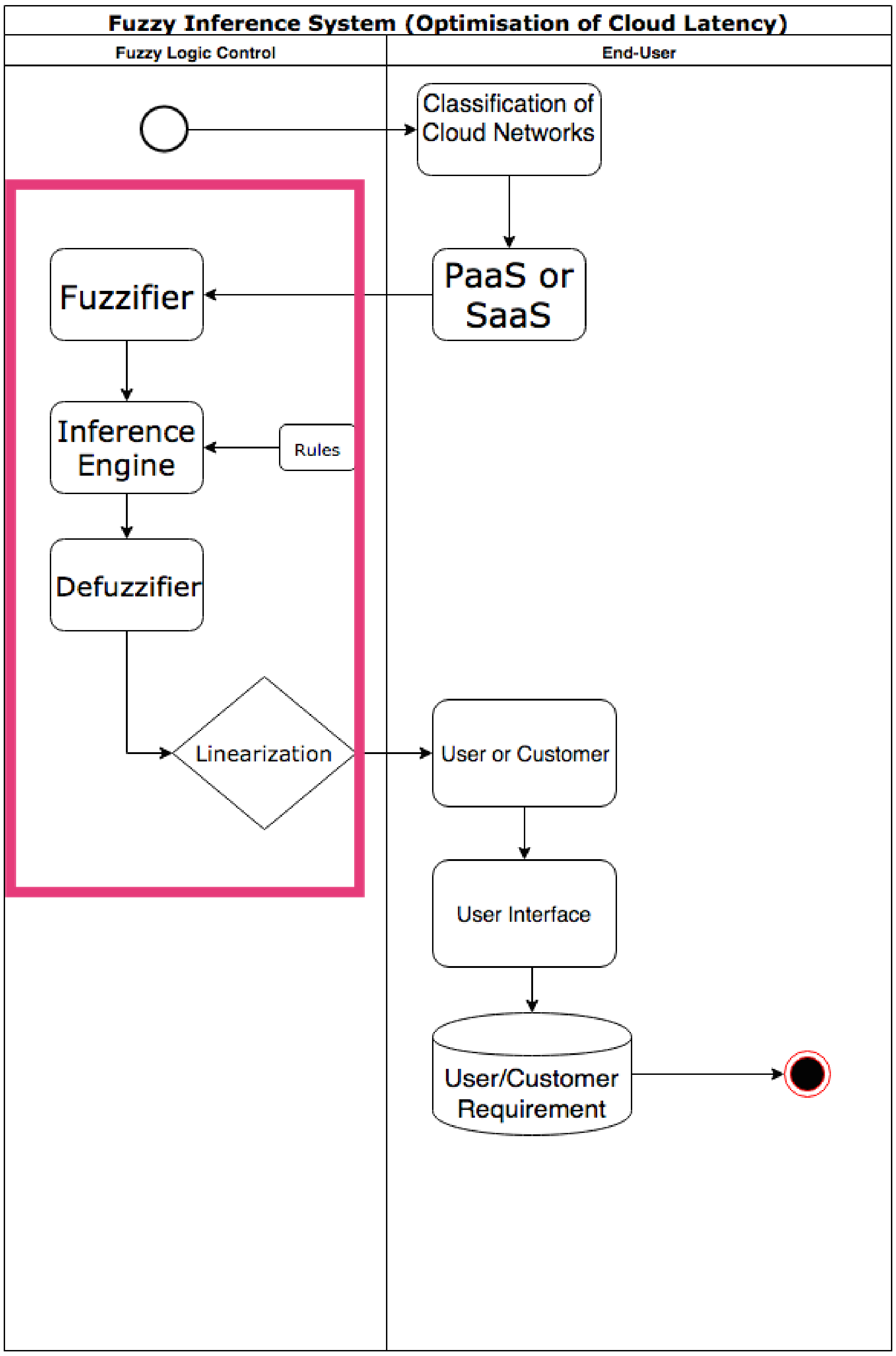


Figure 1: Activity Diagram of Operational Fuzzy Interface System.

3.2 Functional Architectural Goals

Supplying paying customers and users with seamless operation of cloud applications and services is one of the top priorities of a cloud provider. This priority is the motive behind the design of the FLC via mathematical modelling. The resulting merger between fuzzy and mathematics' yield FIS. The FIS should provide a controlled environment to reducing clouds latency by ensuring the communication time between the end-user and the cloud provider does not exceed one second per packet (less than or equal to 1 second per packet) by means of network baseline that the control system will provide (Sule et al.; 2017). As a result, the cloud platform from the customers end becomes a control system with very predictable network traffic. With the FLC ability to be taught complex algorithms, it could even assist in terms of authentication by virtue of cybernetics and neural network (Sungkap Yeo and Lee; 2011). This research project however, will be exploring solely the creation of a control system that includes the cloud environment from view of the end user.

3.3 Functional Architecture

According to (Juang et al.; 2007), FIS is a system paradigm that is based on the fuzzy set theory, fuzzy if-then rules and fuzzy reasoning. According to Mendel and Mouzouris (1997), there are two forms of FIS: Mamdani FIS and the Takagi FIS. This project employed the Mamdani FIS because of its architectural openness, ability to support multiple inputs/outputs similar to the cloud platform and its compatibility on most hardware (Hans; 2018).

The Mamdani FIS consists of the input, fuzzier, set of rules and de-fuzzification that provide sharp output Mendel and Mouzouris (1997) as seen below.

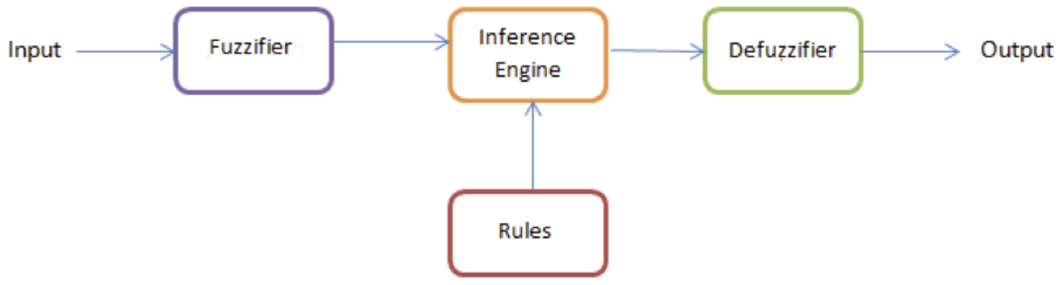


Figure 2: Fuzzy Inference System Mendel and Mouzouris (1997).

Figure 2 is a chart representation of the FIS that shows each of the parameters (fuzzifier, inference engine defuzzifier and rules) and how these relate and interact with one another. The idea of representation is to understand the operation of FIS. Two memberships could be launched at the same time but the one with maximum degree of membership would be selected. Subsequently, membership function with the minimum degree of membership would be preferred from both inputs. The resulting formula from the above action, is given below.

$$G(t) = max[min[G_1[input(i)], G_2[output(j)]$$

Above process occurs randomly, and the launched memberships would be summed up and a good value obtained from the added membership functions. The process of obtaining this good value is called de-fuzzification (Chang and Chang; 2006). Figure 3 below is an example of the FIS used to breakdown instructions for the steering angle of a car (Mukherjee and Sahoo; 2010).

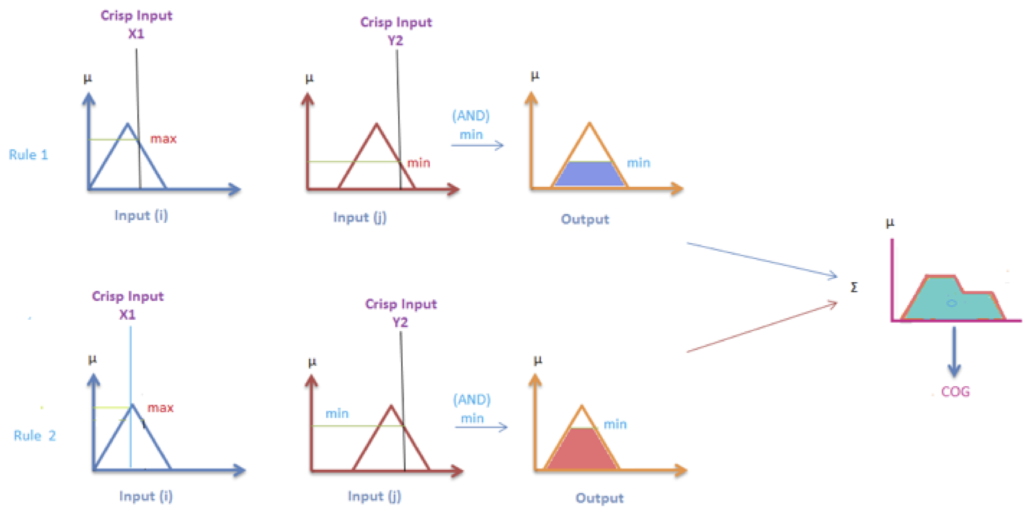


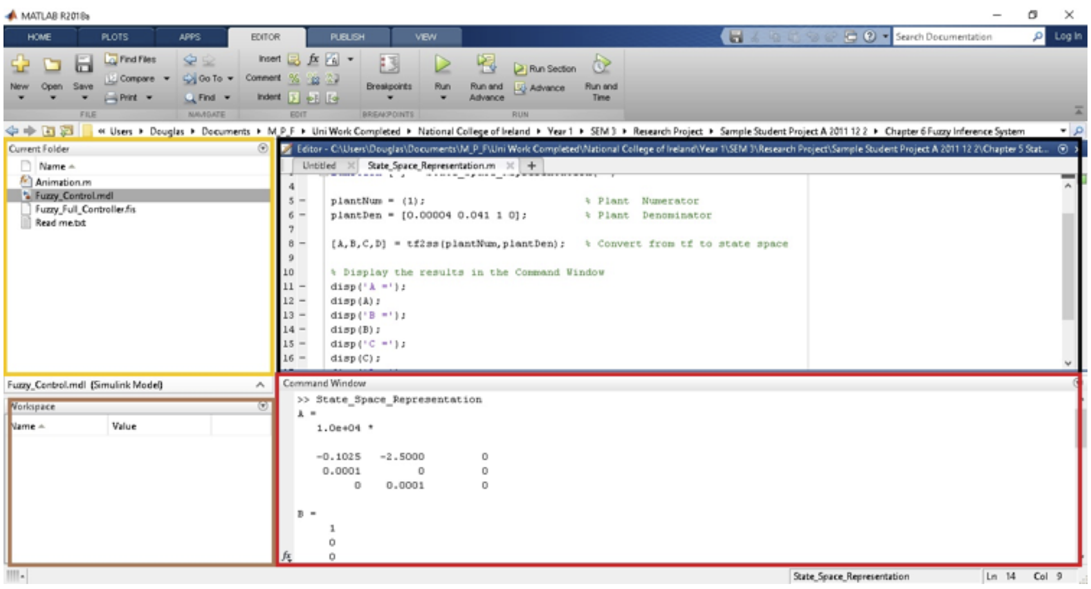
Figure 3: Example of Mamdani FIS for a Cars Steering Angle Mukherjee and Sahoo (2010).

Figure 3 is a section of the input from Figure 2, which shows that a crisp value is acquired from the sum of membership functions after de-fuzzification is completed. This is what guides the steering angle within an acceptable degree. This crisp values are also bound by rules that assist in its self-learning ability (Jang; 1993).

3.4 MATLAB and its Uses

MATLAB is a numerical computing environment and a fourth-generation programming language. It was developed to allow plotting of functions, implementation of algorithms, matrix manipulation and interfacing with programs such as C, C++, Java and FORTRAN (Sungkap Yeo and Lee; 2011). Cleve Moler who was the Chairman of the Computer Science Department, University of New Mexico, developed MATLAB in 1971. He created this software for his students to calculate mathematical functions without the need to have knowledge of FORTRAN (L.D. and Venkata Krishna; 2013). In 1983 an engineer along with Steve Bangert joined Cleve Moler to rewrite MATLAB to C-language. MATLAB was adopted by control engineering in 2001 and was used to model and calculate numerical analysis of a control system (Yassein et al.; 2017). Although MATLAB is intended for numerical calculations and computing, it also has additional packages for designing various accessories. The package we would use for the fuzzy logic modelling is known as Simulink.

In Figure 4 black highlight represents editor window were the C++ script are input and executed. Red highlight is the command window that displays result or error in relation to the inputted script. Yellow highlight shows file path of the C++ script and brown highlight is the workspace that displays what function is currently operating.



```
plantNum = (1);                     % Plant  Numerator
plantDen = [0.00004 0.041 1 0];     % Plant  Denominator

[A,B,C,D] = tf2ss(plantNum,plantDen);   % Convert from tf to state space

% Display the results in the Command Window
disp('A =');
disp(A);
disp('B =');
disp(B);
disp('C =');
disp(C);
```

```
>> State_Space_Representation
A =
   1.0e+04 *

   -0.1025   -2.5000         0
    0.0001         0         0
         0    0.0001         0

B =
     1
     0
     0
```

Figure 4: The MATLAB User Interface.

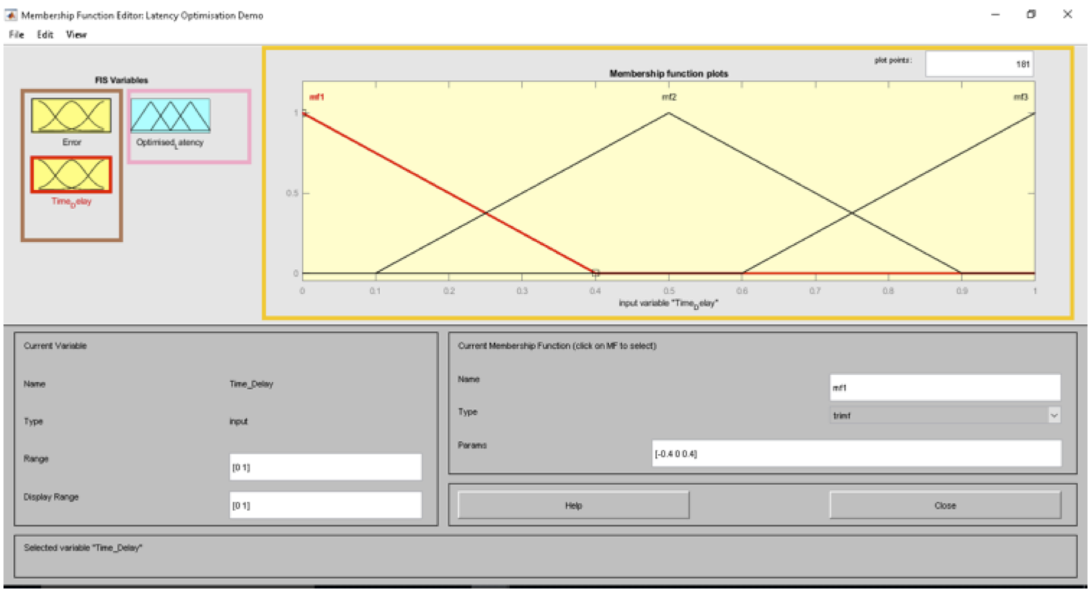
The choice of MATLAB is because of its compatibility and accessibility on regular PCs. MATLAB is well-suited because of its programmable User Interface (UI) and its self-learning algorithm that keep on improving in terms of mathematical modelling. It also has additional packages such as Simulink and mathematical computation that gives it more value in terms of modelling. Essentially, MATLAB and Simulink are easily accessible, compatible and available in most academic institutions, unlike the other options (Mukherjee and Sahoo; 2010).

3.5 Applying Fuzzy Logic

It is a popular belief that latency can be measured and if it can be measured, then it can be controlled. FLC would make the latency in the cloud platform to auto-correct itself back to the predicted and controlled value. We would model the latency as a time delay in the system, and then design the fuzzy logic by invoking required sets of linguistic fuzzy rules for the cloud platform. Currently, we assigned numerical values to these linguistic rules to represent the centre of each membership function. Then, all the inputs and output membership function set along with the rules are entered in the FIS Editor available in MATLAB.

Figure 5 is the FIS editor in the UI that configures the controller. This is the UI that is responsible for setting pattern that guides the FLC. Brown highlight is the input variables of the FIS, Pink highlight is the output variable and Yellow highlight is the editor UI for the membership function.

FIS can also be made to use solely logic rules. The minimum degree of two-launched memberships, functioning from both inputs are taken to be the output based on users selection of membership function. This is achieved with minimum degree of membership of the fuzzy controller.


Membership Function Editor: Latency Optimisation Demo
File
Edit
View
FIS Variables
Error
Time_Delay
Optimised_Latency
Membership function plots
plot points:
mf1
mf2
mf3
input variable "Time_Delay"
Help
Close

Current Variable
Name Time_Delay
Type input
Range [0 1]
Display Range [0 1]

Current Membership Function (click on MF to select)
Name mf1
Type trimf
Params [-0.4 0 0.4]
plot points: 181

Selected variable "Time_Delay"

Figure 5: The FIS Editor of MATLAB.

3.6 S-Function Level-1 Language and Code Explanation

Graphical User Interface (GUI) that was designed, simulated the control system of the cloud platform including latency and FLC. This was obtained by using MATLABs level-1 codes embedded within Simulink Function (s-function) block of Simulink. S-function language was selected based on its compatibility with s-function Application Protocol Interface (API). This API allows for interaction between MATLAB and the Simulink engine to generate a real time simulation.

S-function API Level-1 consists of nine flags with each flag representing an integer value of the s-function tasks. These tasks perform predefined functions known as "callbacks" that are operated using case expression. The table below shows various "callbacks" used in this research project (Yassein et al.; 2017).

Table 2: MATLAB Level-1 callback Methods (Yassein et al.; 2017).

Flag	"Callback"	Usage
0	mdlInitializeSizes	Initialize variables and sample time.
2	mdlUpdate	Update values with each next hit.
4	mdlGetTimeOfNextVarHit	Determine the time for the next hit.

Table 2 consists of the elements for "callback" and the ways to utilize this "callback" methods. "Callback" methods are necessary to initiate the s-function in MATLAB by tagging inputs and outputs to functions. Without "callback", it would be impossible to execute the s-function.

Coding the s-function was generated by using 3 "callback" methods: mdlInitializeSizes, mdlUpdate and mdlGetTimeOfNextVarHit as described in Table 2. The mdlInitializeSizes method was used to initialize all number of inputs, number of outputs, sample time, variables and to insert the main GUI window codes. The "callback" mdlUpdate inserts the control system and assigns inputs to the s-function block of the Simulink model. We primarily included mdlUpdate "callback" because of its ability to initiate an infinite loop which will continue to repeat itself with each operation. And finally, ”callback”, mdlGetTimeOfNextVarHit ensures the next operation occurs for every 0.1 second to include updates of the array inputs.

Full list of code is available in the configuration manual (appendix) of this research project.

In section 4, we would look at the implementation process of the FLS in a practical scenario.

4 Implementation

This section will examine the final stage of implementation of the FLC in a FIS environment.

Our first task was to design the cloud latency environment for the implementation of the FLC. The figure below is the simulation of cloud latency using MATLAB.

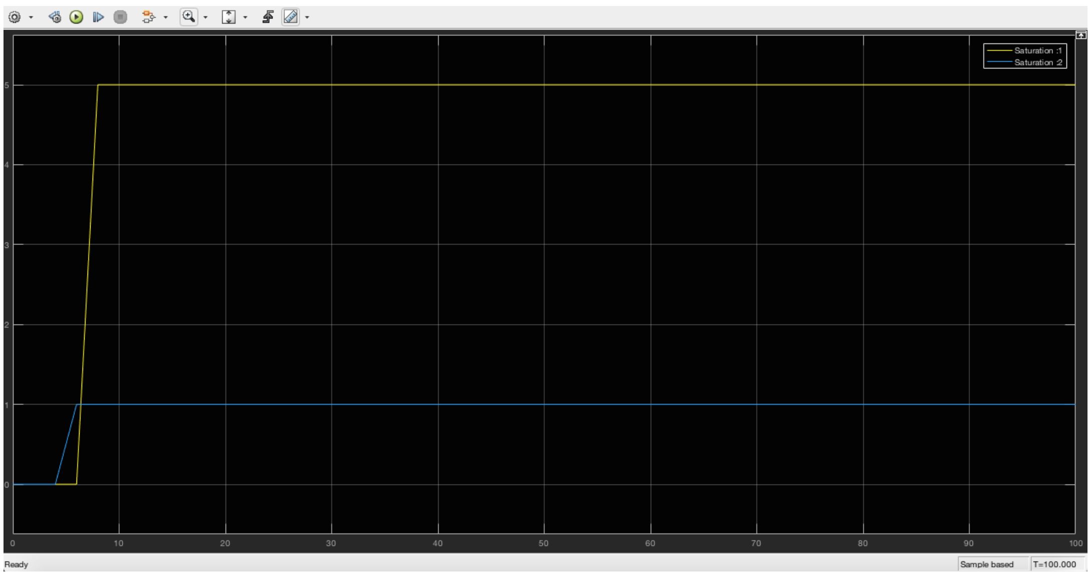


Figure 6: Cloud Latency Representation in MATLAB.

Figure 6 above is the representation of cloud latency using MATLAB. The environment consists of 2 inputs that represents the network and internal latency respectively and some limiting factors for hardware utilized. We can observe that input 1 peaks at 5 seconds and input 2 peaks at 1 second meaning the both have a combine latency of 6 seconds, which signifies that the environment is very laggy. Figure 6 is quite unrealistic in today’s standard but it helps us in understanding the impact of cloud latency on the simulated environment.

Our next course of action to implementing desired fuzzy logic on the cloud platform is to virtualize the control system model to incorporate the cloud platform and controller. We deduced that the FLC would try and correct the cloud’s latency (Figure 6) back to normal value of one second. We created 9 linguistic fuzzy rules using a logic table designed to control the cloud platform. The figure below is the summarized 9 linguistic rules that was derived for this project.

Figure 7, depicts 9 rules that governs the optimization of cloud’s latency. This was written using MATLAB ruling with the objective to bound FIS to the environment of

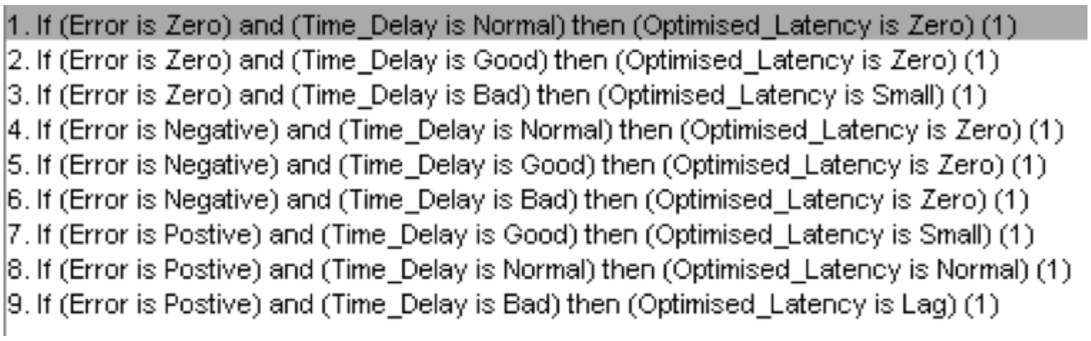

1. If (Error is Zero) and (Time_Delay is Normal) then (Optimised_Latency is Zero) (1)
2. If (Error is Zero) and (Time_Delay is Good) then (Optimised_Latency is Zero) (1)
3. If (Error is Zero) and (Time_Delay is Bad) then (Optimised_Latency is Small) (1)
4. If (Error is Negative) and (Time_Delay is Normal) then (Optimised_Latency is Zero) (1)
5. If (Error is Negative) and (Time_Delay is Good) then (Optimised_Latency is Zero) (1)
6. If (Error is Negative) and (Time_Delay is Bad) then (Optimised_Latency is Zero) (1)
7. If (Error is Postive) and (Time_Delay is Good) then (Optimised_Latency is Small) (1)
8. If (Error is Postive) and (Time_Delay is Normal) then (Optimised_Latency is Normal) (1)
9. If (Error is Postive) and (Time_Delay is Bad) then (Optimised_Latency is Lag) (1)

Figure 7: Fuzzy Linguistic Rules.

cloud computing. The rules works by assigning maybe logic to the FLC to compensate for grey areas in the cloud.

Table 3 The Fuzzy Rules.

Table 3: Fuzzy Logic Rules Generator

Error/Time Delay	Good	Normal	Bad
Negative	Zero	Zero	Zero
Zero	Zero	Zero	Small
Positive	Small	Normal	Large

Top row represents the time delay input. Left column represents the normal error input. Middle section is the output of desired optimal latency.

Table 3 is the logic table that was used to configure the rules for FIS. This table was necessary to accurately create the rules for the FLC. We assigned these linguistic rules to represent the automation of the cloud platform with all the membership function (inputs and output). Membership function was set along with the 9 rules in FIS Editor available in MATLAB. The result of this action can be observed in Figures 7, 8 and 9 below.

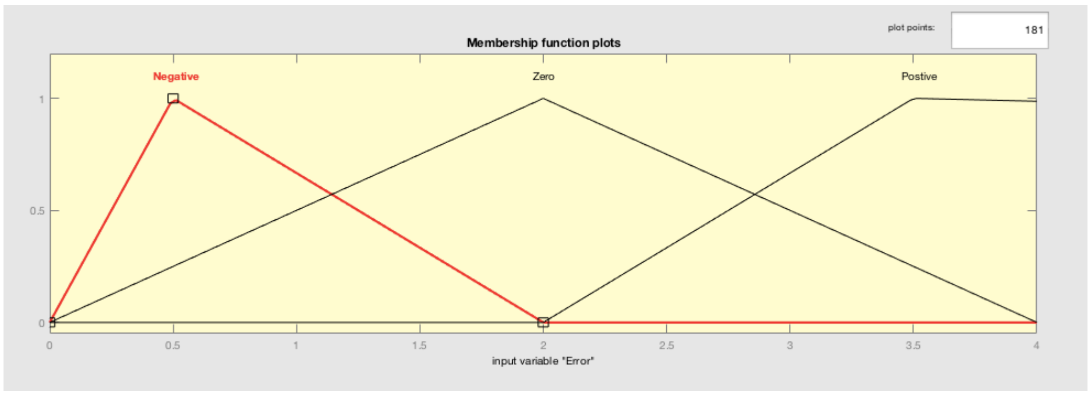


Figure 8: Input Fuzzy Sets of the Fuzzy sets for Input of the Normal Error (Error).

Figure 8 is the membership function plot for the network error. This is used to predict parameters that makes up network error. This is necessary for accurate design of the FLC.

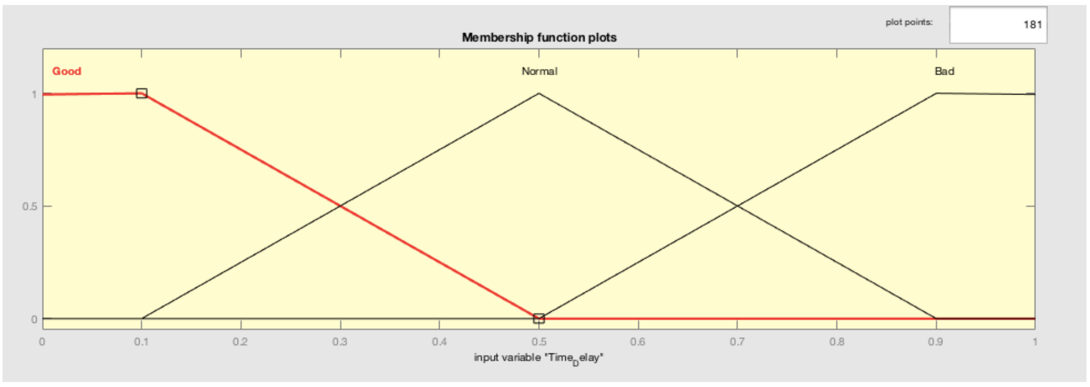


Figure 9: Input Fuzzy Sets of the Fuzzy sets for Input of the Time Delay.

Figure 9 is the membership function plot for time delay. This is used to predict the time delay. It is necessary for the accurate design of the FLC from the view of time delay.

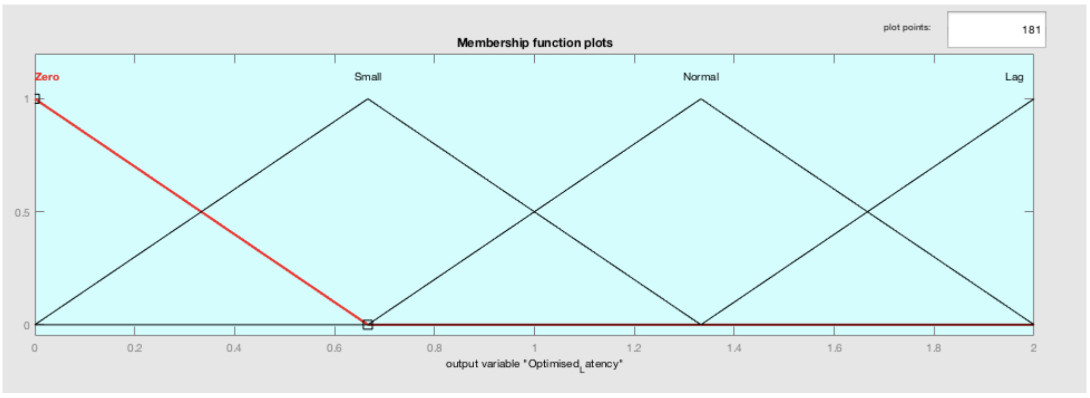


Figure 10: Output Fuzzy Sets of the Fuzzy Sets for Resulting Input of Time Delay and Normal Error.

Figure 10 is the membership function plot for the output of the FLC. This is used to predict output from the combined input parameters that makes up network error and time delay. This is necessary for accurate design of the FLC baseline.

Figure 8, Figure 9 and Figure 10 are the inputs and Output of the Fuzzy Sets. The first fuzzy set, is for the first input which is the normal error. Normal error is known as the input plus the feedback loop, which ensures signals are free from errors thus the name normal error.

The second fuzzy sets, for the input is the time delay which serves as latency in this simulation. The third fuzzy sets, for the output is the resulting end product of the test with desired optimized latency.

Finally, a Simulink model was generated to demonstrate 2 step wave inputs which would enter the fuzzy controller block as seen in Figure 8 and Figure 9. Output from the fuzzy controller that represents optimal latency was passed through a transport delay and saturation before simulation. Delay was added to emulate processing delay of signal through the router, switch or hub as well as delay in network traffic. Moreover, a saturation block was added to create some form of hardware or software limitation similar to normal operation of the cloud. Figure 11 below is the Simulink representation of the designed model.

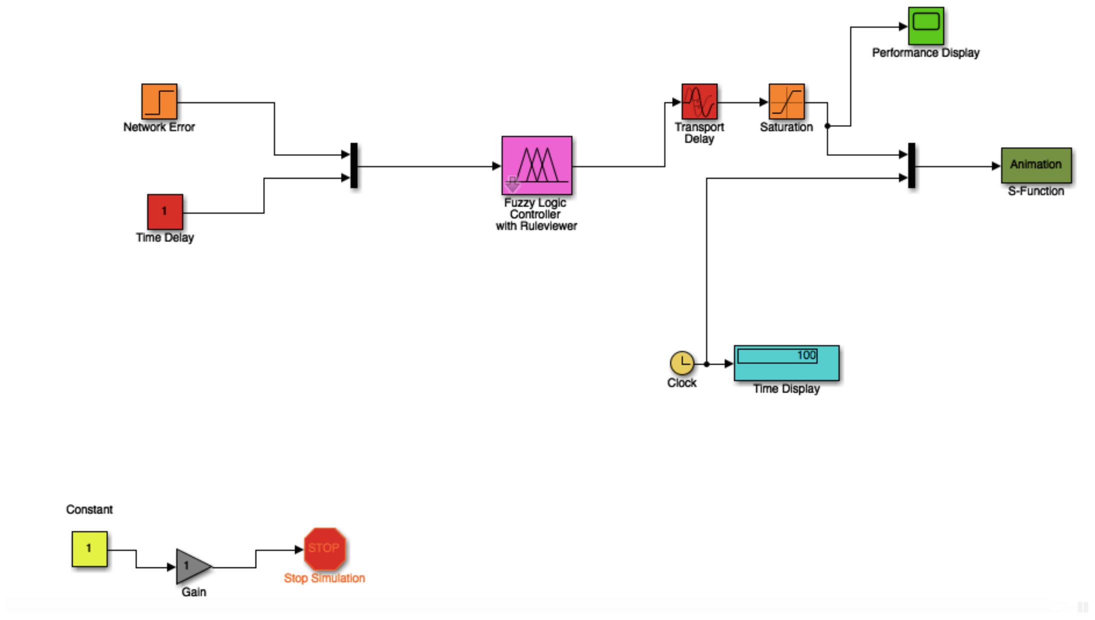


Figure 11: Simulink Model for the Fuzzy Inference System.

Figure 11 is the environment for simulating the cloud environment with the FLC. All components add to life-like model of the cloud environment with FLC. Network error and time delay are adjusted to mimic desired latency which is executed to observe response time. Response time show the reaction of the fuzzy logic to the set latency.

In section 5, we would look at the evaluation of the implemented process using live data in a practical scenario.

5 Evaluation

In this section, we will carry out test on the FIS in a practical environment and generate a comprehensive analysis of the results. Test will include the GUI, analysis of live test data and a discussion on outcome of the analysis.

5.1 Testing the GUI Simulation

The developed GUI simulation model shown in Figure 12 below is the mathematical evaluation of the fuzzy logic controller in the course of the simulation process of this project. During and after the simulation, there were no significant error related to optimization of clouds latency using the Fuzzy controller. Optimization was as we hoped and real-time algorithms worked to an okay degree with noticeable lag but nothing too serious. However, with all the good results we got, the GUI did not represent a real-life scenario of clouds latency because there were many factors we had to assume. Factors like nature of the network traffic, distance from source of cloud to the user and various authentication protocols that exist in real life cloud environment.

Figure 12 above, shows the GUI of the FIS platform. This representation helps in describing the process of the FIS when simulated. The representation is a 3-D window

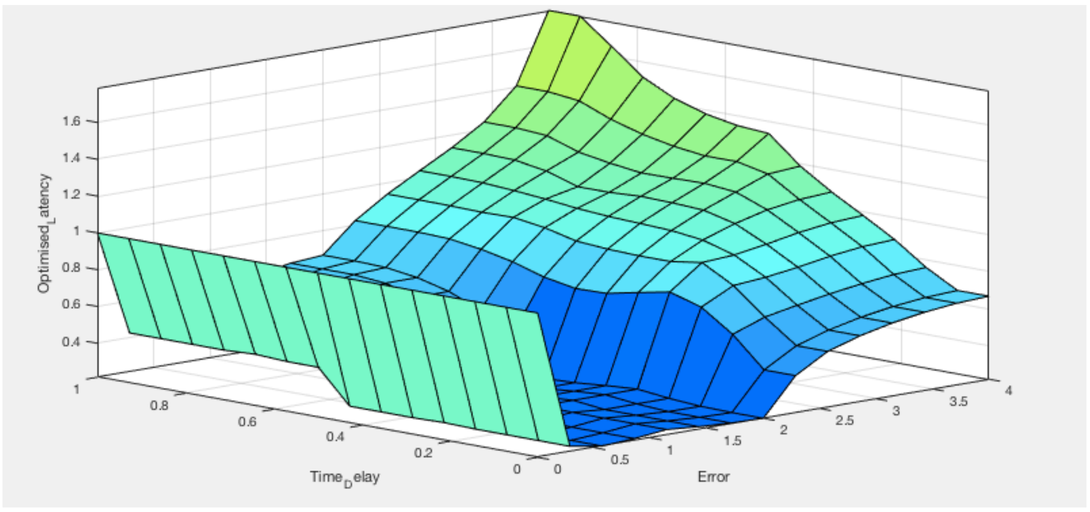


Figure 12: GUI Window for the Project.

which has 3 axes. The X-axis shows the optimization of the latency in relation to the normal error and time delay. The Y-axis shows time delay and the Z-axis shows the normal error. Both Y and Z-axes are the input with relation to output which is the X-axis. Figure 12 clearly shows that as normal error and time delay increases, optimal latency increases as well. This means that a lot of work is done in the fuzzy logic controller to maintain optimal latency (less than or equal to 1 seconds).

5.2 Analysis

In this section, we would analyse implementation of the FIS model. As explained in the previous section (implementation), we only created 9 rules. It is possible to create a fuzzy controller with more rules for complex control of the network but we opted for 9 rules for simplicity and simulation time-reduction that is only possible with 9 rules. None the less, 9 rules will still provide similar effect as the more complex rules with more time allowance.

Figure 13 is the fuzzy rules viewer for the FIS model. It is used to further show operation of the FIS at every value of inputs. First column represents executed memberships functions of the normal error input for the 9 rules. The second column represents executed memberships functions of time delay input for the 9 rules. The third and final column represents the result of both mentioned inputs that includes the added and defuzzified memberships (Frey et al.; 2013).

From Figure 13,we observe that output, optimisation latency is 0.666 when network error is 2 seconds and time delay is 0.886 seconds; are high (i.e lagging network). This is calculated by MATLAB in relation to the executed memberships of the two inputs. Inputs were both set to zero and result was 0.216 at the output. This set the scenario of some encoding latency in hardware of the cloud platform. Thus the selected input memberships from both rules are summed up to create a new set with distinct degree of membership. This is because, only single value affects time delay in the model. Centroid technique is carried out by MATLAB to achieve single crisp value. This operation occurs swiftly that all the crisp values are shown in the output as a continuous wave that oscillates between 0 and 2 as seen in Figure 13.

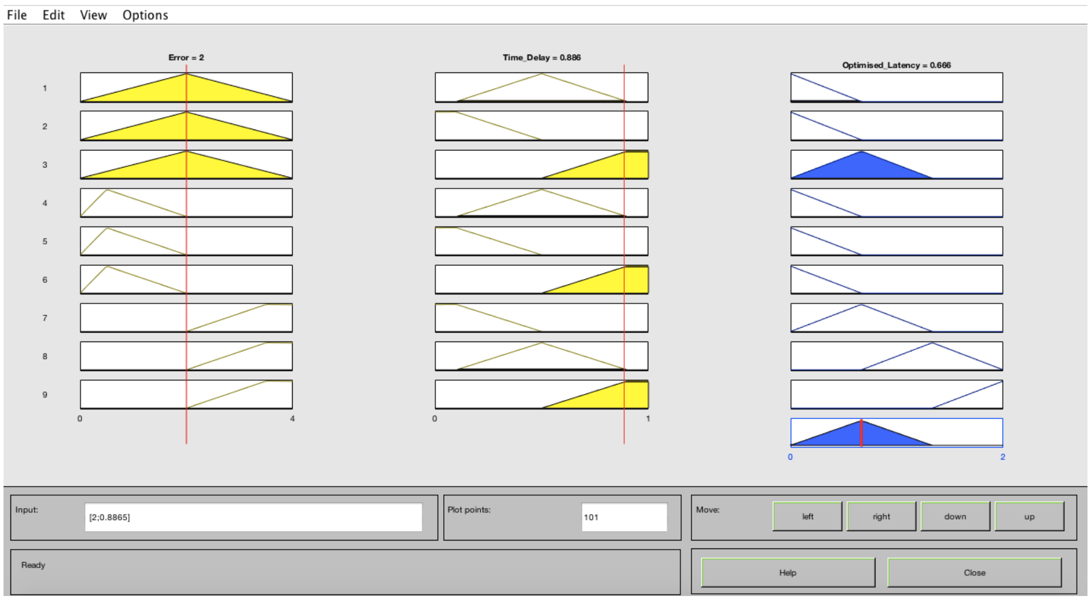


Figure 13: The Fuzzy Rules Viewer for the FIS Model.

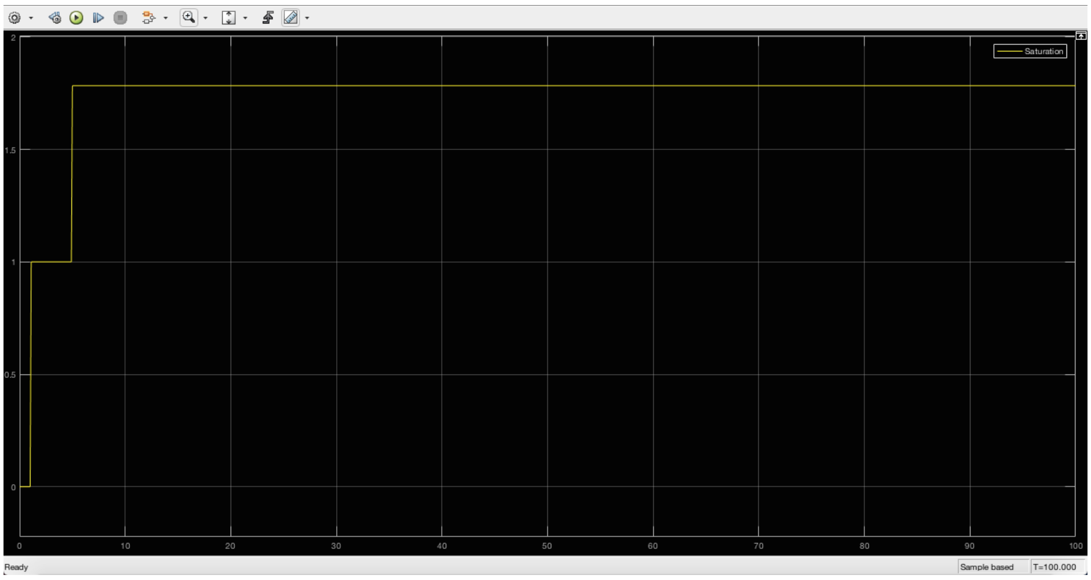


Figure 14: Fuzzy Simulink Model Output.

Figure 14 above represents the wave pattern of the optimal latency in respect to the fuzzy controller. This result was achieved by setting inputs to the maximum known network latency recorded (Barker and Shenoy; 2010). The input values for network error and time delay were 5 seconds and 1 seconds respectively. Result shows a steep climb to the baseline of 1 second and then a pause period of 4 seconds and finally another climb to 1.78 seconds. This action is known as response and time taken for the action to be completed is known as response time. The response time for Figure 14 was 5.02 seconds which means that it took 5.02 seconds for the FLC to enforce some normality in the system.

Using live test data from (Cooperation; 2018), we were able to test the fuzzy logic in a practical sense and arrived at the graphical result shown in Figure 15 below. Table 4 below, contains element of the average output of the fuzzy logic with the measured latency.

Table 4: The Statistical Data of the Measured Latency of Singapore to Sydney from 2017 to 2018 (Cooperation; 2018)

Month and Year	Measured Latency (ms)	Average Fuzzy Response time (ms)
October 2017	125.803	3040
November 2017	107.458	2918
December 2017	172.111	3189
January 2018	134.337	3004
February 2018	104.185	2923
March 2018	109.323	2982
April 2018	103.951	2736
May 2018	143.932	3082
June 2018	103.069	2928
July 2018	97.948	2774
August 2018	87.852	2774
September 2018	160.941	3278

Table 4 is the input (latency) and output (response time) of FIS via mathematical modelling. The mathematical model showed correlation and linear regression between the input and output (see configuration manual or appendix). Table 4 and IBM SPSS were very important in designing the graphic seen in Figure 15 below.

Figure 15 is the histogram representation of the test of FIS. We can observe performance of the FLC when subjected to real life parameters. Blue bar is the latency from the live data and the red bar is the response time of FLC. The response is how long it took FLC to adjust to the latency. Fuzzy logic does show promise in dealing with latency as observed in Figure 15.

5.3 Discussion

We started by building an Artifact to mitigate latency. We simulated latency by creating a Simulink model with network error and time delay as the input (see Figure 11). Both inputs were utilized in varying the cloud latency courtesy of statistical data of Singapore to Sydney from 2017 to 2018 (Cooperation; 2018).

We configured FLC to serve as a baseline (cut-off), we were able to limit the excess latency even when it exceeded the normal latency of 1 second to a reasonable extent.

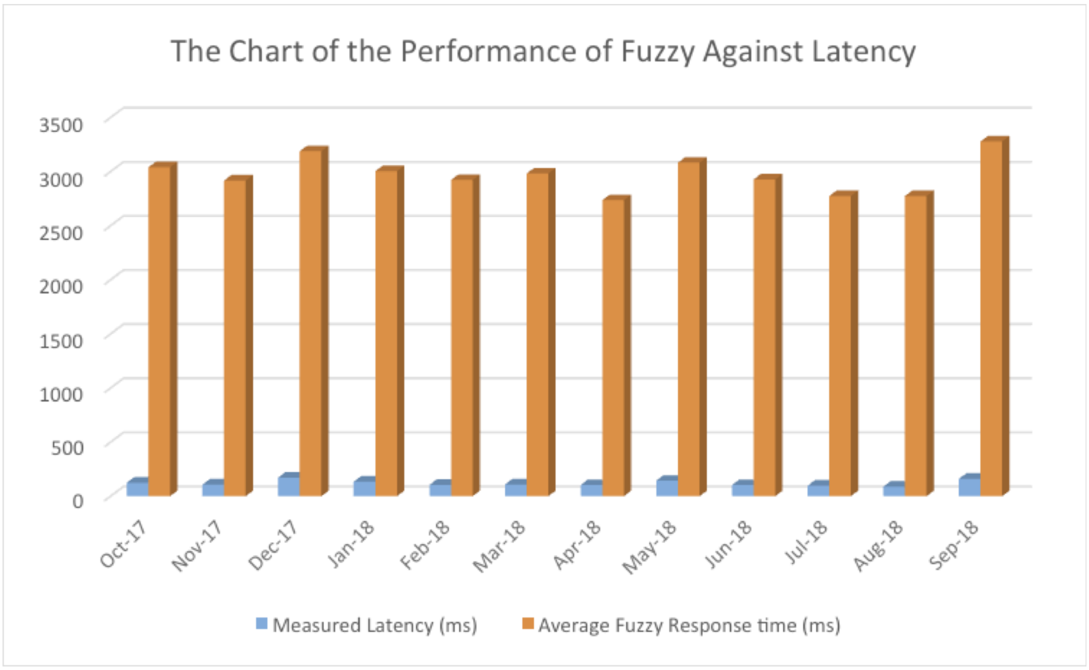


Figure 15: Graphical Representation of the FIS Test.

With initiated real life data courtesy of statistical data of Singapore to Sydney from 2017 to 2018 (Cooperation; 2018), we observed high performance in mitigating latency.

While experimenting with latency values higher than 5 seconds (6 seconds) we observed a change in the baseline from 1 seconds to 1.78 seconds. We were aware of how unrealistic the value was but we decided on this unrealistic value as a safety measure and as a way to improving FLC response time (see Figure 6). It resulted in the performance that can be observed in Table 4 and Figure 15 respectively.

Figure 14 showed the control system's tolerance to high latency (Barker and Shenoy; 2010) with a response time of 5.02 seconds. Table 4 and Figure 15 respectively, show control performance test with real life data which it handled relatively well.

The response time observed in Table 4 and Figure 15 is the time it takes for FIS to detect, shut down unessential activities and stabilize the system. It does take some time to respond as seen in Figure 14 and Figure 15. This is because it is detecting, managing essential activities, stabilizing the system and enforcing the baseline of 1 second.

Besides the increased baseline of FLC which was due to very high latency (6 seconds) and response time (inconsistent), which we can attribute to lack of a detection system for latency we can say FIS is reasonably successful in optimizing cloud latency.

In section 6, we would conclude the research project.

6 Conclusion and Perspectives

In conclusion, we needed to mitigate cloud latency, as it is a major challenge worth investing time in solving as seen in the literature review of this research report. The world of cloud computing would evolve a lot with mitigation of latency because it would boost the computing power of shared resources. We researched cloud latency by examining research papers in relation to managing and detecting latency. We then proposed an artifact (FLC) to be integrated into the cloud environment to mitigate cloud latency. This integrated environment was called FIS and we ran tests using live statistical latency data of Singapore to Sydney from 2017 to 2018 courtesy of (Cooperation; 2018).

The result shows that FLC is feasible in mitigating latency in the cloud to a reasonable extent. The FLC did take some time to complete its response to latency, which shows some of its weaknesses in detection. However, the proposed FLC could attain the best result in latency to date because of the controls robustness, learning capabilities and frankly the lack of any serious latency mitigation method.

In revised versions of FLC, the current control system can evolve to incorporate and develop its own detection system to accurately predict network latency that would make its response time quicker. Also, the FLC can be deployed on a private or hybrid cloud with the FLC acting as a network, service or application monitoring and detection tool that would be service model (application-as-a-service) or a defense against network latency from the view of the end user. Thus, in the years to come, FLC would be the ideal network latency defender that would defend customers Internet from latency and would improve the QoS of cloud.

6.1 Acknowledgments

We sincerely thank our supervisor, Sean Heeney for his valuable suggestions and ideas to the final outcome of this research project.

7 Appendix

In this section, we have all other figures and informations that were utilized to build and test the Artifact. The did not make it to the final report but the were important in their own way.

This section is broken down into 6 subsection below.

7.1 User Guide

The artifact is simple to operate but a bit complex to run effectively. In this section we would look to execute the artifact with extreme simplicity and less complexity.

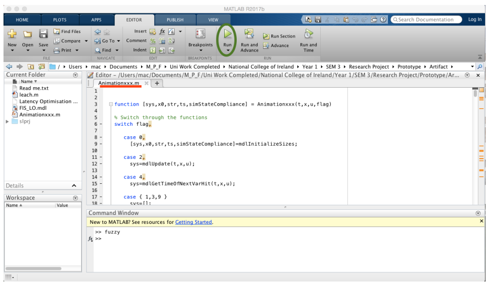
Prerequisite:

- First of we would have to ensure that the personal computer used has MATLAB 2016a or later installed. The version of MATLAB must be the full version with controller design, control system, digital processing and feedback control package installed.
- The artifact can be downloaded from a folder in our Dropbox Account (`//www.dropbox.com/sh/7cuoej66wmdn5x1/AAD_3MrE6sD1EfCHa6yRheK8a?dl=0`). The folder contains 4 files including a read me.txt file, which is important in accurately running the artifact.

To Run the Artifact

1. We start up by opening MATLAB on your PC or Mac.

2. Open the m-file. Double clicking on the m-file will bring you to the editor page see Figure 16 below. Click on the run command (green highlight). NB: In the Editor window, click run Animationxxx.m.



```
function [sys,x0,str,ts,simStateCompliance] = Animationxxx(t,x,u,flag)

% Switch through the functions
switch flag,

  case 0,
    [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes;

  case 2,
    sys=mdlUpdate(t,x,u);

  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);

  case { 1,3,9 }
    sys=[];
```

```
>> fuzzy
>>
```

Figure 16: The MATLAB Editor of Animationxxx.

3. 3- Open the **.mdl file. Double clicking on the **.mdl file would bring up the figure see Figure 17 below.

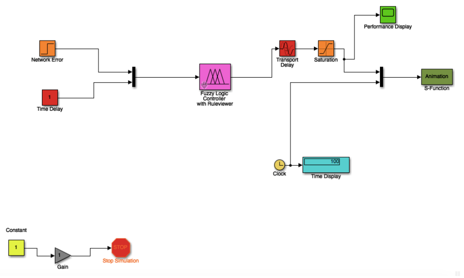
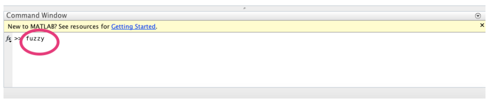


Figure 17: The Simulink Model of the FIS.

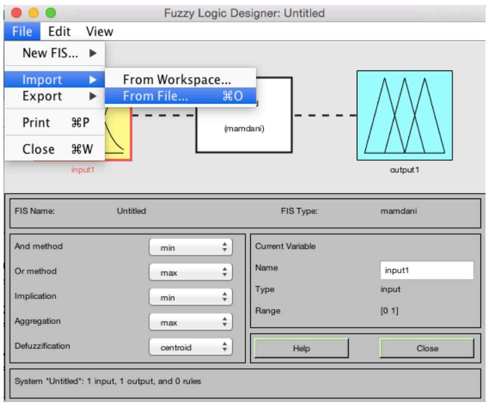
4. Type fuzzy in the MATLAB command window as seen in Figure 18 below.



```
>> fuzzy
```

Figure 18: The Command Window of MATLAB.

5. In the FIS Editor Window, from file menu to select Import and From File. As shown in Figure 19 below.
6. Select Latency Optimisation Demo.fis from Files and you will be welcomed to the controller for this artifact as seen in Figure 20 and Figure 21 below.
7. In the FIS Editor Window, from file menu to select Export to Workspace as shown in Figure 22 below.
8. Return to the *.mdl window and edit the network error (double click). See Figure 23 below.
9. 9- Finally, click run from the *.mdl window. See Figure 24 below.



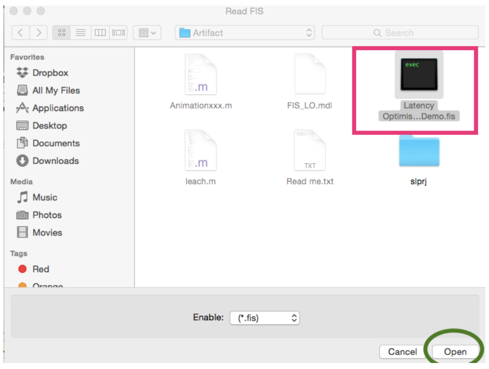
FIS Name: Untitled FIS Type: mamdani

And method min
Or method max
Implication min
Aggregation max
Defuzzification centroid

Current Variable
Name input1
Type input
Range [0 1]

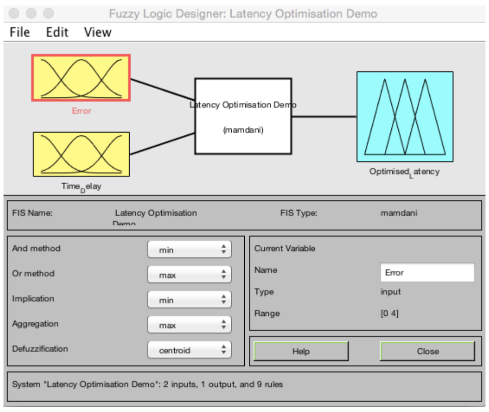
System "Untitled": 1 input, 1 output, and 0 rules

Figure 19: The FLC Editor Window 1.



Enable: (*.fis)

Figure 20: The FLC Editor Window 2.


Fuzzy Logic Designer: Latency Optimisation Demo
File Edit View
Error
Time_Delay
Latency Optimisation Demo
(mamdani)
Optimised_Latency

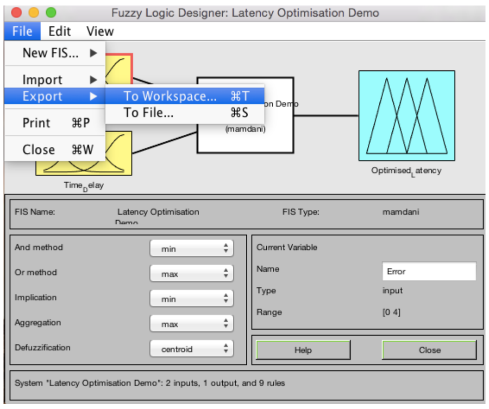
FIS Name: Latency Optimisation Demo FIS Type: mamdani

And method min
Or method max
Implication min
Aggregation max
Defuzzification centroid

Current Variable
Name Error
Type input
Range [0 4]

System "Latency Optimisation Demo": 2 inputs, 1 output, and 9 rules

Figure 21: The FLC Editor Window 3.



FIS Name: Latency Optimisation Demo FIS Type: mamdani

And method	min
Or method	max
Implication	min
Aggregation	max
Defuzzification	centroid

Current Variable

Name	Error
Type	input
Range	[0 4]

System "Latency Optimisation Demo": 2 inputs, 1 output, and 9 rules

Figure 22: The FLC Editor Window 4.

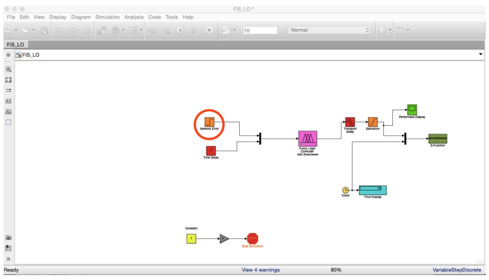


Figure 23: The Simulink Modelling Window.

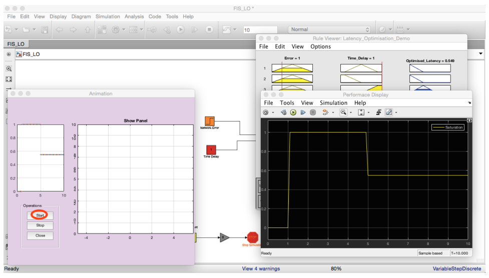


Figure 24: The Simulink Modelling Execution Window.

7.2 Fuzzy Logic Design in a Nutshell

Fuzzy Logic System or Fuzzy Inference System (FLS or FIS) is predominantly nonlinear systems that are capable of deducing complex non-linear relationships between input and output variables (Chopra; 2012). The nonlinearity property of the FIS is the most important attribute its possess in relation to modelling physical mechanism. FIS or FLS has the ability to learn the nonlinear mapping by being presented a sequence of input signal and desired response pairs, which are used in conjunction with an optimization algorithm to determine the values of the system parameters (Frey et al.; 2013). This is one of the most commonly used learning paradigms, called supervised learning. Even if the process to be modeled is non-stationary, the system can be updated to reflect the changing statistics of the process (Entchev and Yang; 2007). Unlike conventional stochastic models used to model such processes, FLSs do not make any assumptions regarding the structure of the process, nor do they invoke any kind of probabilistic distribution model, i.e., they belong to the general family of model-free, data driven, nonparametric methods (Chang and Chang; 2006).

- *Description of FLS*: FLS are a combination of intuitive and numerical systems. The system maps crisp input, x into a crisp output, y. Every FLS are associated with a set of rules known as meaningful linguistic interpretations that are all assigned to membership functions. The membership function of this interpretation is given in the formula below.

  $$Y_x : Z = [0, 1]$$

  Where Y is membership function associated with each element (x) with universal discourse within an interval [0,1] (Lofberg; 2004).

  Non-singleton fuzzification is another type of fuzzification that exists in FLS (Sule et al.; 2017). It is mostly used in cases where the training data (input) includes some kind of uncertainty (such as noise, or linguistic imprecision). Theoretically, the non-singleton fuzzifier states that the given input value is solely the correct value, however due to the presence of uncertainty the neighboring points are likely to be the correct value (L.D. and Venkata Krishna; 2013). The designer determines the shape of the membership function base on the estimated quantity of uncertainty. The ideal choice for membership function to be symmetric at point, x with the effect of noise is given below.

  $$Y_(x(x_i))e^([-(x - x_i)^2/(2^2)])$$

  When the designer is configuring the FLS it is important undergo some for of descriptive statistics for the input in order to get the best possible output (Sturm; 1999).

- *Designing of FLS for FIS*: The figures below are the 2-D GUI plot of Optimization Latency against Error and Optimization Latency against Time Delay.

  Figure 25 shows the plot of 'Optimization Latency' against 'Error' of the FIS platform. This representation is an exclusive look at FIS from the view of one input (Error). Figure 26 shows the plot of 'Optimization Latency' against 'Time Delay'

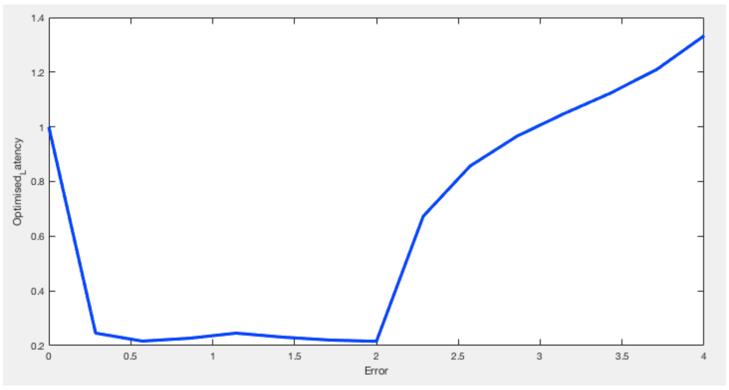


Figure 25: Optimization Latency vs Error.

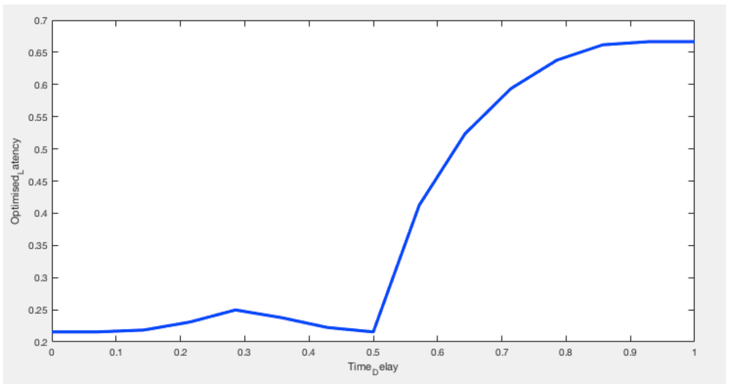


Figure 26: Optimization Latency vs Error.

of the FIS platform. This representation is an exclusive look at FIS from the view of one input (Time Delay).

7.3 Simulink Function (S- function)

In this section we would look at s-function. The s-function code helps in producing a real-time simulation environment. This involves animating the output response via DrawMode.

The full line of code used in MATLAB script is given below.

```matlab


function [sys,x0,str,ts,simStateCompliance] = Animationxxx(t,
   x,u,flag)

% Switch through the functions
switch flag,

   case 0,
     [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes;

   case 2,
     sys=mdlUpdate(t,x,u);

   case 4,
     sys=mdlGetTimeOfNextVarHit(t,x,u);

   case { 1,3,9 }
     sys=[];

   otherwise
     DAStudio.error('Simulink:blocks:unhandledFlag', num2str(
        flag));


end



function [sys,x0,str,ts,simStateCompliance]=
   mdlInitializeSizes

sizes = simsizes;

sizes.NumContStates   = 0;
sizes.NumDiscStates   = 0;
sizes.NumOutputs      = 0;
sizes.NumInputs       = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;   % at least one sample time is needed

sys = simsizes(sizes);

clear functions;


% Define Global variables
global Position_X
global Position_Y
global Input_1_Angle
global Input_2_Time


% Initializing the variables with values
Position_X = 10;
Position_Y = 100;
Input_1_Angle = 0;
Input_2_Time = 0;




global Demo
fuzzy_animinit('Animation');
Demo=findobj(0,'Name','Animation');

% Set Properties for fuzzy_animinit
set(gcf,...
    'Color',[0.9 0.8 0.9],...
    'Units','normalized',...
    'Resize','on',...
    'Toolbar','none',...
    'MenuBar','none');

set(0,'CurrentFigure',Demo);

% Create main axis for the car show and sets its properties and location.
global mainax
mainax= axes('Parent',Demo, 'XLim',[0 10],'YLim',[0 10], ...
    'Units','normalized',...
    'OuterPosition',[0.27 0.1 0.8 0.8],...  %[0 0 1 1] normalized
    'XColor','black','YColor','black', ...
    'Box','on',...
    'DrawMode','fast');
```

```

     axis equal;
     title('Show Panel');
     grid('on');


% Create a Panel for the Buttons
Buttons_Panel= uipanel('Parent',Demo,...
        'Units','normalized',...
        'Position',[0.07 0.1 0.21 0.27],...
        'Title','Operations',...
        'BackgroundColor',get(Demo,'Color'),...
        'HandleVisibility','callback',...
        'Tag','tunePanel');

% Create Start button and calls its function
    uicontrol('Parent',Buttons_Panel,...
            'Style','pushbutton',...
            'Units','normalized',...
            'Position',[0.15 .7 0.7 0.2],...
            'String','Start',...
            'SelectionHighlight','on',...
            'ButtonDownFcn', 'on',...
            'Callback',@(src,evt) start1);

% Create Stop button and calls its function
    uicontrol('Parent',Buttons_Panel,...
            'Style','pushbutton',...
            'Units','normalized',...
            'Position',[0.15 .45 0.7 0.2],...
            'String','Stop',...
            'ButtonDownFcn', 'on',...
            'Callback',@(src,evt) stop1);

% Create Close button and calls its function
    uicontrol('Parent',Buttons_Panel,...
            'Style','pushbutton',...
            'Units','normalized',...
            'Position',[0.15 0.2 0.7 0.2],...
            'String','Close',...
            'ButtonDownFcn', 'on',...
            'Callback',@(src,evt) Close_1);


% Creating second axis for the input signal.
global secondax
 secondax = axes('Parent',Demo,'Units','normalized',...
    'Position',[0.05 0.44 0.25 0.4],...  %[0 0 1 1]
```

```
       normalized
    'XColor','black','YColor','black', ...
    'Box','on',...
    'DrawMode','fast');


x0  = [];

str = [];

ts  = [-2 0];

simStateCompliance = 'UnknownSimState';



% Main window
function figNumber=fuzzy_animinit(namestr)

if (nargin == 0)
  namestr = 'Simulink Animation';
end

figNumber = findobj('Type','figure','Name',namestr)';

if isempty(figNumber),
% Initialize figure
  position=get(0,'DefaultFigurePosition');
  position(3:4)=[750 500];
  figNumber=figure( ...
      'Name',namestr, ...
      'NumberTitle','off', ...
      'BackingStore','off', ...
      'Position',position, ...
      'MenuBar', 'none');



  bottom=0.05;
  left=0.80;
  btnWid=0.15;
  btnHt=0.10;

else
    % bring figure to foreground
    figure(figNumber)
end

```

```
cla reset;
set(gca,'DrawMode','fast');
axis off;




function sys=mdlUpdate(t,x,u)
global Demo
global Position_X
global Position_Y
global mainax
global secondax


if any(get(0,'Children')==Demo),
    if strcmp(get(Demo,'Name'),'Animation'),

Input_1_Angle =     u(1);
Input_2_Time =      u(2);
% 2 = u(3);

Track = 12;



cla(mainax);                 % Clear the axis mainax


    % Corners of the car

    Reference_1_X = Position_X;
    Reference_1_Y = Position_Y;

    Point_1_X = Reference_1_X;
    Point_1_Y = Reference_1_Y;

    Point_2_X = Point_1_X;
    Point_2_Y = Point_1_Y;

    Point_3_X = Point_2_X;
    Point_3_Y = Point_2_Y;

    Point_4_X = Point_3_X;
    Point_4_Y = Point_3_Y;



```

```
 % Lines
 Line_1 = line([Point_1_X Point_2_X],[Point_1_Y Point_2_Y],'
    Parent',mainax,'linewidth',1,'color','b');
 Line_2 = line([Point_2_X Point_3_X],[Point_2_Y Point_3_Y],'
    Parent',mainax,'linewidth',1,'color','b');
 Line_3 = line([Point_3_X Point_4_X],[Point_3_Y Point_4_Y],'
    Parent',mainax, 'linewidth',1,'color','b');
 Line_4 = line([Point_4_X Point_1_X],[Point_4_Y Point_1_Y],'
    Parent',mainax, 'linewidth',1,'color','b');


% The x and y axis plot for the small axis
plot (Input_2_Time,Input_1_Angle,'parent',secondax,'LineWidth
   ',2,'Marker','.');
grid('on')
hold on
drawnow()



    end
end
sys = [];




function sys=mdlGetTimeOfNextVarHit(t,x,u)

sampleTime = 0.1;     %  Next Hit after 0.1 seconds
sys = t + sampleTime;


% Stop and Close GUI
function Close_1()
set_param ('FIS_LO/Gain','Gain','1')

close (gcf)


% Start the simulation
function start1()
global mainax
global secondax
set_param ('FIS_LO/Gain','Gain','1')

cla (mainax);
cla(secondax);
```

```
cla (mainax);
cla(secondax);
cla (mainax);
cla(secondax);

set_param ('FIS_LO/Gain','Gain','0')
sim('FIS_LO');

% Stop the simulation
function stop1()
set_param ('FIS_LO/Gain','Gain','1')
  cla (mainax);
```

The lines of codes listed above were used to create the real time simulation of FIS. Each function was linked to the "callback" that started and updated the inputs and output respectively.

7.4 Linear Regression of Latency against Fuzzy Control

In this section, we have all the tables that initiated the correlation between latency and fuzzy logic control. This evaluation were done using IBM SPSS software.

The table result of testing the FLC with live data are given in the figures below. The data used for this test was courtesy of Live Data (`https://enterprise.verizon.com/terms/latency/`).

The tables and figures below represents the framework that constitute the Linear Regression model.

Table 5: The Elements of the Descriptive Statistics of Measured latency against Average Control time.

	Mean	Std. Deviation	N
Measured Latency (ms)	120.90917	26.593225	12
Average Fuzzy Control (response time,ms)	2969.00	165.096	12

Table 5 above, represents the descriptive part of the elements and houses the most common descriptive statistics such as the mean, standard deviation, dependent variable (measured latency) and the independent variable (average fuzzy control). The result from the view of the standard deviation shows that most of the variables from both the independent and dependent are closely spread (low standard deviation values) in relation to their normal values.

Table 6 above, is the pearson correlation table for this variables. It shows the correlation between the dependent variable (measured latency) and the independent variable (average fuzzy control).

The term correlation comes from co which mean together and relation which means connection (Lofberg; 2004). The technique correlation is used to know whether variables that are studied are correlated and it is also to test their strength of association. In correlation, the output (r) must be equal to +1 or -1 (r=+1 or r=-1) to be significant and the p-value should be less or equal to 0.05. r=+1 means that the variables correlated all increase together while r=-1 means when one of the variable increases, then the other

Table 6: The Elements That Makes Up The Correlations Between The Measured latency and The Average Control time.

		Latency (ms)	Avg. FLC (ms)
Pearson Correlation	Measured Latency (ms)	1.000	.910
	Average Fuzzy Control (ms)	.910	1.000
Sig. (1-tailed)	Measured Latency (ms)	.	.000
	Average Fuzzy Control (ms)	.000	.
N	Measured Latency (ms)	12	12
	Average Fuzzy Control (ms)	12	12

decreases. The above significance (r=+1 or r=-1) means there is a relationship while if the output is greater than +1, less than -1 or greater than 0.05 then there is no relationship meaning that there is no correlation. Therefore, the variables used in this research project were significantly correlated.

Table 7: The Elements of the Variables Entered/Removed(a)

Model	Variables Entered	Variables Removed	Method
1	Average Fuzzy Control (response time,ms)[b]	.	Enter

a. Dependent Variable: Measured Latency (ms) b. All requested variables entered.

From Table 7 above, we can observe that this model has only one-type model. This model shows that all the independent variable is accounted for. This also shows that the variables entered contains one independent predictors.

Table 8: Model Summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.910a	.829	.812	11.543506

a. Predictors: (Constant), Average Fuzzy Control (response time,ms)

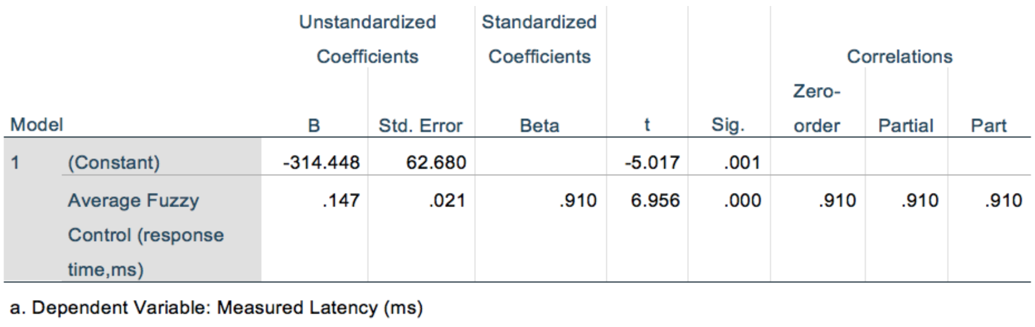
From Table 8 above, we can observe that this model is a single model. This model has its value of r as 0.910 which is significantly correlated (r is less than 1) and has just one model. This model shows that all the independent variables are account for. The value for R Square is 0.829 which is 83 percent of the variances and shows that the dependent variables can be accounted for the independent variables.

Table 9: The Table of ANOVA(a)

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	6446.671	1	6446.671	48.379	.000b
	Residual	1332.525	10	133.253		
	Total	7779.196	11			

a. Dependent Variable: Measured Latency (ms) b. Predictors: (Constant), Average Fuzzy Control (response time,ms)

From Table 9 above, we have the ANOVA table that is testing sequentially the statistical significant of the model. From the model we can see that the F-value is 48.379 with 1 and 10 degree of freedom and it is statistically significant with 0.000 which is less than 0.001.



Model		Unstandardized Coefficients		Standardized Coefficients			Correlations		
		B	Std. Error	Beta	t	Sig.	Zero-order	Partial	Part
1	(Constant)	-314.448	62.680		-5.017	.001			
	Average Fuzzy Control (response time,ms)	.147	.021	.910	6.956	.000	.910	.910	.910

a. Dependent Variable: Measured Latency (ms)

Figure 27: The Snapshot of The Table of Coefficients(a).

From Figure 27, we have the intercept (constant) as -314.448 with the other unstandardized betas. From the standardized and correlation column, we can see a mixture of negative and positive value which shows the complexity in this regression model. The model has sig values that are 0.001 meaning it is statistically significant.

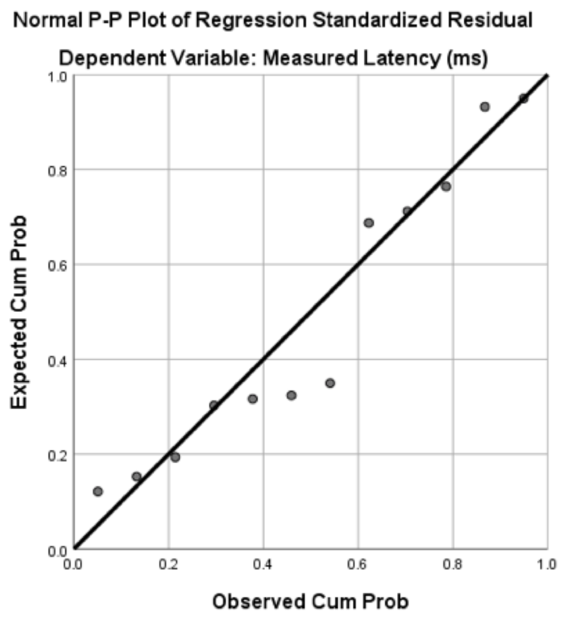


Figure 28: P-P Plot Representation of Linear Regression.

Figure 28 is the linear regression model of the research project values. Figure 28 shows the location of the project values in respect to linearity. Figure 29 is the scatterplot which represents the statistical significants of the model because it does not exceed -2 and +2 respectively.

7.5 Project Plan

This section consists of the project plan for the duration of this research project. The plan consists of the task done for this project , a chart to show the time spent on each task and a Gantt chart that gives a general idea of the duration of the project.

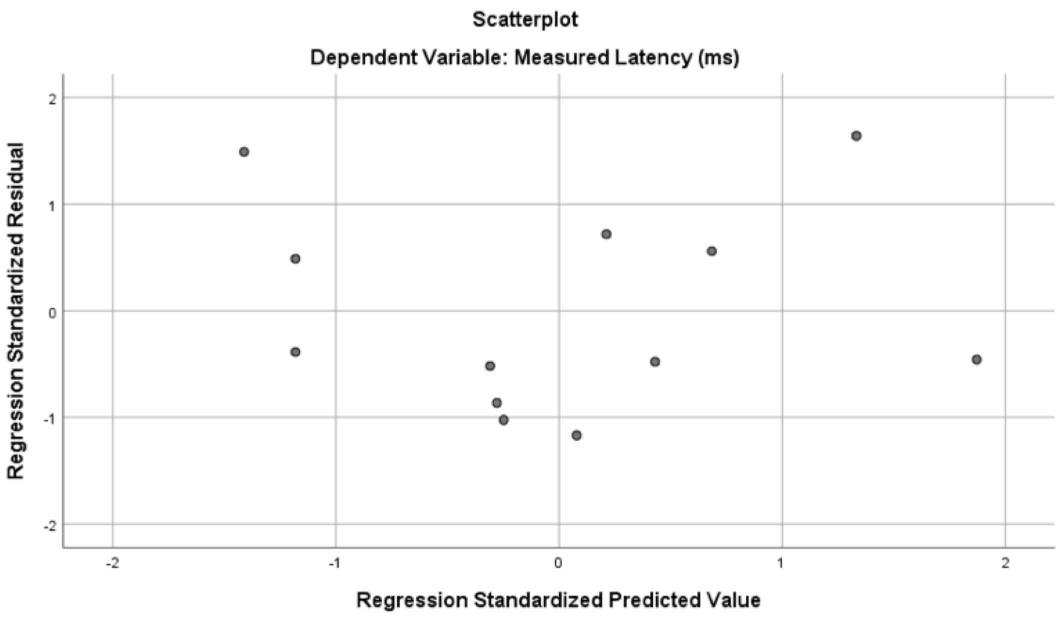


Figure 29: Scatterplot of the Linear Regression.

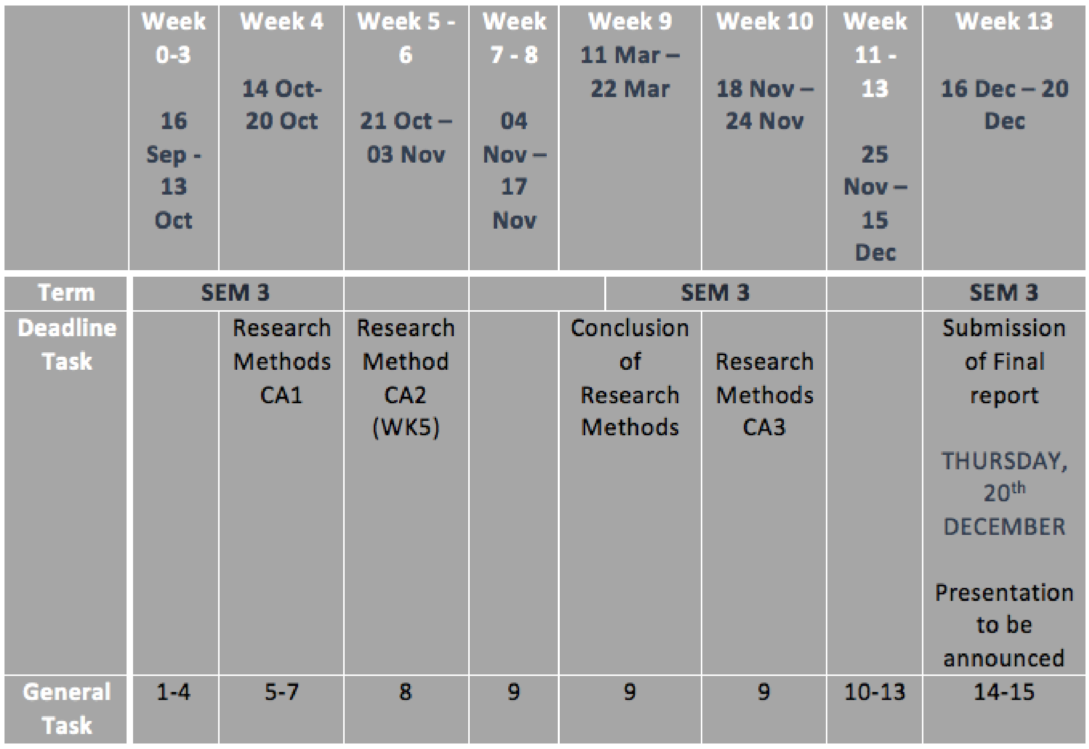
Table 10: The Table of Tasks Carried Out During the Research Project Life Cycle

Task Number	Description
1	Get information about the Latency in Cloud
2	Research into journals that supports cloud latency as a major issue
3	Identify the essential literature for the research project literature review
4	Research into the various available Mathematical Models
5	Formulate the parameters that would be required for the mathematical model
6	Compare all found mathematical model in terms of complexity and feasibility
7	Design and configure the rules of fuzzy logic control via mathematical modelling
8	MATLAB system design and investigation into other forms of controller design
9	MATLAB S-function design
10	Test uncertainties (saturation and transport delay) on the fuzzy logic control
11	Complete all MATLAB design and Live Data testing
12	Mathematical evaluation of tested data
13	Completed the first draft of the report
14	Completed the final draft of the report
15	Creating presentation slides and submitting the final report

Table 10 is the task number and description of each task. The purpose of the table is to understand what each task number stands for.

Figure 30 is a chart of the research project task execution for semester 3. This included how long each task took to complete and how we balanced it with our other module (Research Methods).

Figure 31 is a Gantt chart of the research project task execution for semester 3. This included a more detailed look into the progression of each task.



	Week 0-3 16 Sep - 13 Oct	Week 4 14 Oct- 20 Oct	Week 5 - 6 21 Oct – 03 Nov	Week 7 - 8 04 Nov – 17 Nov	Week 9 11 Mar – 22 Mar	Week 10 18 Nov – 24 Nov	Week 11 - 13 25 Nov – 15 Dec	Week 13 16 Dec – 20 Dec
Term	SEM 3				SEM 3			SEM 3
Deadline Task		Research Methods CA1	Research Method CA2 (WK5)		Conclusion of Research Methods	Research Methods CA3		Submission of Final report THURSDAY, 20th DECEMBER Presentation to be announced
General Task	1-4	5-7	8	9	9	9	10-13	14-15

Figure 30: A chart of the allocated task along with the expected completion date.

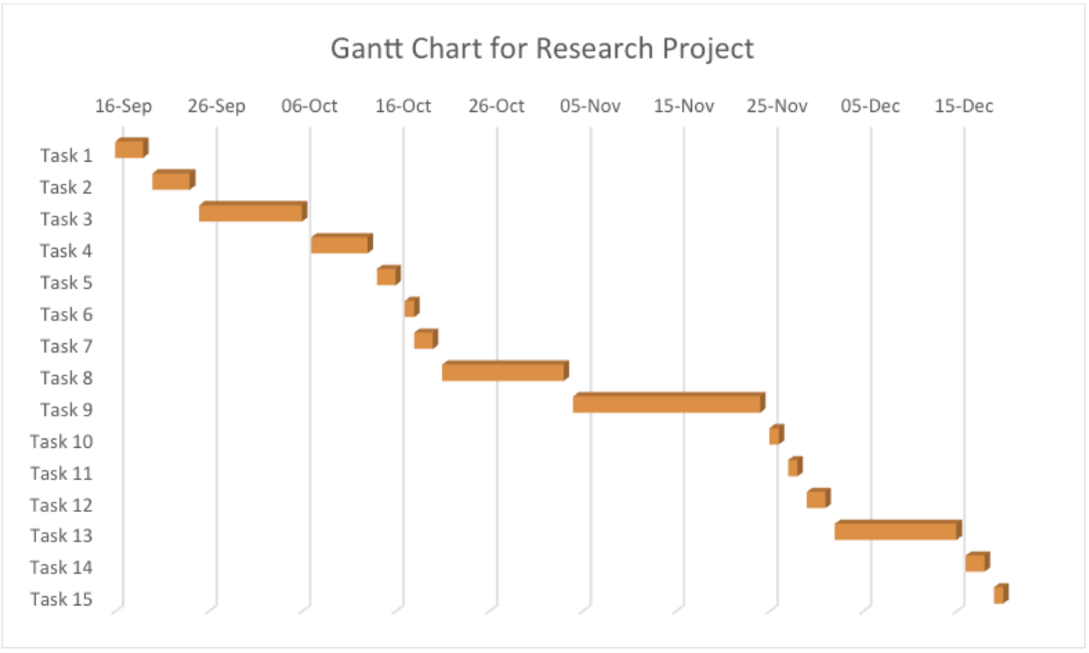


Figure 31: Gantt Chart of The Research Project.

7.6 Glossary of Terms

- Actuating signal: It is the signal that drives the controller. If this signal is the difference between the input and output, it is called the error.
- Analogue-to-digital converter: It is a device that converts analogue signals to digital signal.
- Block diagram: It is a representation of the interconnection of subsystem that makes up a system.
- Bode diagram: It is a sinusoidal frequency response plot, where the magnitude response is plotted separately from the phase response. The magnitude plot is dB against log W, and the phase versus log W.
- Break frequency: It is a frequency where the Bode magnitude plot change slope.
- Breakaway point: It is a point on the real axis of the s-plane where the root locus leaves the real axis and enters the complex plane.
- Break-in point: It is a point on the real axis of the s-plane where the root locus enters the real axis from the complex plane.
- C-language: It is a high level programming language used for system programming.
- Characteristic equation: It is the equation formed by settling the characteristic polynomial to zero.
- Characteristic polynomial: It is the denominator of a transfer function. Simply it is the unforced differential equation, where the differential operators are replaced by s.
- Closed-loop system: It is a system that monitors its output and corrects for disturbances. It is characterized by feedback path from the output.
- Compensation: it is the addition of a transfer function in the forward or feedback path for the purpose of improving the steady-state performance of a control system.
- Compensator: It is a subsystem inserted into the forward or feedback path for the purpose of improving the steady-state error.
- Controllability: It is a property of a system that were an input can be found to take every state variable from a desired initial state to a desired final state in finite time.
- Controlled variable: it is the output of a plant that the system is controlling for the purpose of desired transient response, stability and steady-state error characteristics.
- Controller: It is the subsystem that generates the input to the plant.
- Critically damped response: It is the step response of a second-order system with a given natural frequency that is characterized by no overshoot and a rise time that is faster than any possible over damped response with the same natural frequency.

- Damping ratio: It is the ratio of the exponential decay frequency to the natural frequency.
- Digital-to-analogue converter: It is a device that converts digital signals to analogue signals.
- Disturbance: An unwanted signal that corrupts the input or output of a plant.
- Dominant poles: It is the pole that predominantly generates the transient response.
- Error: It is difference between the input and output of a system.
- Feedback: It is a path that a signal uses to flow back to a previous signal in the forward path so that it could be added or subtracted.
- Fortran: It is an all-purpose programming language that is suited for scientific and numerical calculation.
- Gain: It is the ratio of output to input, usually used to describe the amplification in the steady state of the magnitude of sinusoidal inputs.
- Gain margin: It is the amount of additional open-loop gain, expressed in decibels (dB), required at 180 degree of phase shift to make the closed-loop system unstable.
- Instability: It is the characteristic of a system defined by a natural response that grows without bounds as time approaches infinity.
- Laplace transformation: It is a transformation that transforms linear differential equations into algebraic expressions. This transformation is very useful for modelling and designing a control system.
- Linear system: It is a system possessing the properties of superposition and homogeneity.
- Observability: It is the property by which the state variables can be estimated from knowledge of the input, u(t) and output, y(t).
- Open-loop system: It is a system that does not monitor its output or correct disturbances.
- Peak time: It is the time required for the under damped step response to reach the first or peak.
- Per cent overshoot, Percent OS: It is the amount that the under damped step response overshoot the steady state, or final, value at the peak time, expressed as a percentage of the steady-state value.
- Phase margin: It is the amount of additional open-loop phase shift required at unit gain to make the closed-loop system unstable.
- Poles: It is the roots of the characteristic equation in the denominator that are common to the numerator of the transfer function.

- Rise time, Tr: It is the time required for the step response to go from 0.1 of the final value to 0.9 of the final value.
- Root locus: It is the locus of closed-loop poles as a system parameter is varied.
- Routh-Hurwitz criterion: It is a method for determining how many roots of a polynomial in s are in the right half of the s-plane, the left half of the s-plane, and on the imaginary axis.
- Sensitivity: It is the fractional change in a system characteristic for a frictional change in a system parameter.
- Settling time, Ts: It is the amount of time required for the step response to reach and stay within 2 percent of the steady-state value.
- Stability: it is that characteristic of a system defined by a natural response that decays to zero as time approaches infinity.
- State-space representation: it is a mathematical model for a system that consists of simultaneous, first-order differential equations and an output equation.
- Steady state error response: it is the difference between the input and output of a system after the natural response has decayed to zero.
- Subsystem: it is a system that is a portion of a larger system.
- Summing junction: it is a block diagram symbol that shows the algebraic summation of 2 or more signals.
- Transfer function: It is the ratio of the Laplace transform of the output of a system to the Laplace transform of the input.
- Under damped response: It is the step response of a second-order system that is characterized by overshoot.
- VLSI microchip: It is a set of electronic circuits on one small plate of a semiconductor material
- Z-transformation: It is a transformation related to the Laplace transformation that is used for the representation, analysis and design of sampled signals and systems.
- Zero-order sample-and-hold: It is a device that yields a staircase approximation to the analogue signal.
- Zeros: This is the root of the numerator that is common to the characteristic equation in the denominator of the transfer function.

References

Abdelsalam, H. S., Maly, K., Mukkamala, R., Zubair, M. and Kaminsky, D. (2009). Analysis of Energy Efficiency in Clouds, IEEE, pp. 416–421.
URL: *http://ieeexplore.ieee.org/document/5359623/*

Barker, S. K. and Shenoy, P. (2010). Empirical Evaluation of Latency-sensitive Application Performance in the Cloud, *Proceedings of the First Annual ACM SIGMM Conference on Multimedia Systems*, MMSys '10, ACM, New York, NY, USA, pp. 35–46.
URL: *http://doi.acm.org/10.1145/1730836.1730842*

Bashar, A. (2014). Modeling and Simulation Frameworks for Cloud Computing Environment: A Critical Evaluation, p. 6.

Chang, F.-J. and Chang, Y.-T. (2006). Adaptive neuro-fuzzy inference system for prediction of water level in reservoir, *Advances in Water Resources* **29**(1): 1–10.
URL: *http://www.sciencedirect.com/science/article/pii/S0309170805001338*

Chen, K.-T., Chang, Y.-C., Tseng, P.-H., Huang, C.-Y. and Lei, C.-L. (2011). Measuring the Latency of Cloud Gaming Systems, *Proceedings of the 19th ACM International Conference on Multimedia*, MM '11, ACM, New York, NY, USA, pp. 1269–1272.
URL: *http://doi.acm.org/10.1145/2072298.2071991*

Chen, Y. (2011). Mathematical Modelling of End-to-End Packet Delay in Multi-hop Wireless Networks and their Applications to QoS Provisioning, p. 159.

Chopra, P. (2012). Applications Of Fuzzy Logic in Cloud Computing: A Review, **6**(11): 4.

Choy, S., Wong, B., Simon, G. and Rosenberg, C. (2012). The brewing storm in cloud gaming: A measurement study on cloud to end-user latency, IEEE, pp. 1–6.
URL: *http://ieeexplore.ieee.org/document/6404024/*

Cooperation, V. (2018). Latency Statistics.
URL: *https://enterprise.verizon.com/terms/latency/*

Department of Computer Science Engineering, Jain University, Ngenzi, A., R, D. S. and Suchithrar, D. (2014). "Applying mathematical models in cloud computing: A survey", *IOSR Journal of Computer Engineering* **16**(5): 36–46.
URL: *http://www.iosrjournals.org/iosr-jce/papers/Vol16-issue5/Version-2/F016523646.pdf*

Edmondson, J., Gokhale, A. and Schmidt, D. (2012). Approximation Techniques for Maintaining Real-Time Deployments Informed by User-Provided Dataflows within a Cloud, *2012 IEEE 31st Symposium on Reliable Distributed Systems*, pp. 372–377.

Entchev, E. and Yang, L. (2007). Application of adaptive neuro-fuzzy inference system techniques and artificial neural networks to predict solid oxide fuel cell performance in residential microgeneration installation, *Journal of Power Sources* **170**(1): 122–129.
URL: *http://linkinghub.elsevier.com/retrieve/pii/S0378775307007252*

Frey, S., Luthje, C., Huwwa, V. and Reich, C. (2013). Fuzzy Controled QoS for Scalable Cloud Computing Services, *CLOUD COMPUTING* p. 6.

Gelenbe, E., Lent, R. and Douratsos, M. (2012). Choosing a Local or Remote Cloud, IEEE, pp. 25–30.
URL: *http://ieeexplore.ieee.org/document/6472455/*

Haas, D., Wang, J., Wu, E. and Franklin, M. J. (2015). CLAMShell: Speeding Up Crowds for Low-latency Data Labeling, *Proc. VLDB Endow.* **9**(4): 372–383.
URL: *http://dx.doi.org/10.14778/2856318.2856331*

Hans, R. (2018). *QoS-aware Cloud Infrastructure Provisioning in Heterogeneous Environments*, Ph.D. Thesis, Technische Universitt, Darmstadt.
URL: *http://tuprints.ulb.tu-darmstadt.de/7665/*

Islam, S., Lee, K., Fekete, A. and Liu, A. (2012). How a consumer can measure elasticity for cloud platforms, ACM Press, p. 85.
URL: *http://dl.acm.org/citation.cfm?doid=2188286.2188301*

Jang, J.-R. (1993). ANFIS: adaptive-network-based fuzzy inference system, *IEEE Transactions on Systems, Man, and Cybernetics* **23**(3): 665–685.

Jarschel, M., Schlosser, D., Scheuring, S. and Hofeld, T. (2011). An Evaluation of QoE in Cloud Gaming Based on Subjective Tests, *2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pp. 330–335.

Juang, Y.-S., Lin, S.-S. and Kao, H.-P. (2007). Design and implementation of a fuzzy inference system for supporting customer requirements, *Expert Systems with Applications* **32**(3): 868–878.
URL: *http://www.sciencedirect.com/science/article/pii/S0957417406000431*

Kasabov, N. K. and Song, Q. (2002). DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction, *IEEE Transactions on Fuzzy Systems* **10**(2): 144–154.

Kashyap, D. and Viradiya, J. (2014). A Survey Of Various Load Balancing Algorithms In Cloud Computing, **3**(11): 5.

Labba, C., Bellamine Ben Saoud, N. and Dugdale, J. (2018). A predictive approach for the efficient distribution of agent-based systems on a hybrid-cloud, *Future Generation Computer Systems* **86**: 750–764.
URL: *http://www.sciencedirect.com/science/article/pii/S0167739X17324329*

L.D., D. B. and Venkata Krishna, P. (2013). Honey bee behavior inspired load balancing of tasks in cloud computing environments, *Applied Soft Computing* **13**(5): 2292–2303.
URL: *http://linkinghub.elsevier.com/retrieve/pii/S1568494613000446*

Liu, Q., Cai, W., Shen, J., Fu, Z., Liu, X. and Linge, N. (2016). A speculative approach to spatial-temporal efficiency with multi-objective optimization in a heterogeneous cloud environment: A speculative approach to spatial-temporal efficiency with multi-objective optimization in a heterogeneous cloud environment, *Security and Communication Networks* **9**(17): 4002–4012.
URL: *http://doi.wiley.com/10.1002/sec.1582*

Lofberg, J. (2004). YALMIP : a toolbox for modeling and optimization in MATLAB, IEEE, pp. 284–289.
URL: *http://ieeexplore.ieee.org/document/1393890/*

Lyu, M. R. and Zhang, L. . (2015). Guest Editorial: Recommendation Techniques for Services Computing and Cloud Computing, *IEEE Transactions on Services Computing* **8**(3): 422–424.

Mendel, J. M. and Mouzouris, G. C. (1997). Designing fuzzy logic systems, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* **44**(11): 885–895.

Mi, H., Wang, H., Yin, G., Zhou, Y., Shi, D. and Yuan, L. (2010). Online Self-Reconfiguration with Performance Guarantee for Energy-Efficient Large-Scale Cloud Computing Data Centers, IEEE, pp. 514–521.
URL: *http://ieeexplore.ieee.org/document/5557272/*

Moulder, S., Sheridan, T., Dulai, A. and Rossini, G. (2010). Deep Learning in the Cloud with Neural Networking, p. 24.

Mukherjee, K. and Sahoo, G. (2010). Development of Mathematical Model for Market-Oriented Cloud Computing, *International Journal of Computer Applications* **9**(11): 19–24.
URL: *http://www.ijcaonline.org/volume9/number11/pxc3871927.pdf*

Rao, B. T., Sridevi, N. V., Reddy, V. K. and Reddy, L. S. S. (2011). Performance Issues of Heterogeneous Hadoop Clusters in Cloud Computing, p. 6.

Ruhit, S. K., Vijaya, A. and Kotia, S. S. (2018). Simulation of rotation and scaling algorithm for numerically modelled structures, *IOP Conference Series: Materials Science and Engineering* **402**(1): 012050.
URL: *http://stacks.iop.org/1757-899X/402/i=1/a=012050*

Sakellari, G. and Loukas, G. (2013). A survey of mathematical models, simulation approaches and testbeds used for research in cloud computing, *Simulation Modelling Practice and Theory* **39**: 92–103.
URL: *http://linkinghub.elsevier.com/retrieve/pii/S1569190X13000658*

Selinger, C. and Katze, M. G. (2013). Mathematical models of viral latency, *Current Opinion in Virology* **3**(4): 402–407.
URL: *http://www.sciencedirect.com/science/article/pii/S1879625713001090*

Shahdi-Pashaki, S., Teymourian, E. and Tavakkoli-Moghaddam, R. (2018). New approach based on group technology for the consolidation problem in cloud computing-mathematical model and genetic algorithm, *Computational and Applied Mathematics* **37**(1): 693–718.
URL: *https://doi.org/10.1007/s40314-016-0362-4*

Shrestha, P. L., Hempel, M., Sharif, H. and Chen, H.-H. (2013). Modeling Latency and Reliability of Hybrid Technology Networking, *IEEE Sensors Journal* **13**(10): 3616–3624.
URL: *http://ieeexplore.ieee.org/document/6563136/*

Smola, A. J. and Schlkopf, B. (2004). A tutorial on support vector regression, *Statistics and Computing* **14**(3): 199–222.
URL: *http://link.springer.com/10.1023/B:STCO.0000035301.49549.88*

Sturm, J. F. (1999). Using SeDuMi 1.02, A Matlab toolbox for optimization over symmetric cones, *Optimization Methods and Software* **11**(1-4): 625–653.
URL: *http://www.tandfonline.com/doi/abs/10.1080/10556789908805766*

Sule, M., Li, M., Taylor, G. and Onime, C. (2017). Fuzzy logic approach to modelling trust in cloud computing lt, *IET Cyber-Physical Systems: Theory Applications* **2**(2): 84–89.

Sungkap Yeo and Lee, H.-H. S. (2011). Using Mathematical Modeling in Provisioning a Heterogeneous Cloud Computing Environment, *Computer* **44**(8): 55–62.
URL: *http://ieeexplore.ieee.org/document/5740825/*

Tomanek, O. and Kencl, L. (2013). CLAudit: Planetary-scale cloud latency auditing platform, IEEE, pp. 138–146.
URL: *http://ieeexplore.ieee.org/document/6710568/*

Touch, J. D. (2010). MIRAGE: A Model for Latency in Communication, p. 18.

Yassein, M. B., Aljawarneh, S. and Alodibat, S. (2017). A mathematical model of integrated cloud computing and WSNs for emergency systems, *2017 International Conference on Engineering MIS (ICEMIS)*, pp. 1–12.

Zhang, Q., Cheng, L. and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges, *Journal of Internet Services and Applications* **1**(1): 7–18.
URL: *http://www.springerlink.com/index/10.1007/s13174-010-0007-6*

Zhu, J., Zheng, Z., Zhou, Y. and Lyu, M. R. (2013). Scaling Service-Oriented Applications into Geo-distributed Clouds, *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, pp. 335–340.