James Duncan

Computational Science

Contents

[Introduction 2](#_Toc34842777)

[Part 1 – Euler’s Algorithm 2](#_Toc34842778)

[Values of the Simulation 3](#_Toc34842779)

[Comparison of Step Sizes (Error) 3](#_Toc34842780)

[Actual Values 3](#_Toc34842781)

[Part 2 – Creating Noise Using Box Muller Algorithm 3](#_Toc34842782)

[Why Create Noise? 3](#_Toc34842783)

[Normal Distribution 3](#_Toc34842784)

[Part 3 – Perceptron learning Algorithms 3](#_Toc34842785)

[Step Activation 4](#_Toc34842786)

[Sigmoid Activation 4](#_Toc34842787)

[Conclusion 4](#_Toc34842788)

[Appendix 4](#_Toc34842789)

[Code 4](#_Toc34842790)

[Part 1: Methods (Euler’s Algorithm, Integral Calculation, Calculation of Actual Results) 4](#_Toc34842791)

[Part 1: Coded Results 5](#_Toc34842792)

[Part 1: Coded results for Different Step Sizes 7](#_Toc34842793)

[Part 1: Actual Results Plotted 7](#_Toc34842794)

[Part 2: Methods (Box Muller Method) 8](#_Toc34842795)

[Part 2: Plots of Random Numbers 8](#_Toc34842796)

[Part 2: Add Results to X Data 9](#_Toc34842797)

[Part 2: Plot X Noise against Actual for Step 9](#_Toc34842798)

[Part 3: Step Function Perceptron 10](#_Toc34842799)

[Part 3: Step Function Perceptron Results 10](#_Toc34842800)

[Part 3: Step Function Change in Learning Rate 11](#_Toc34842801)

[Part 3: Sigmoid Function Perceptron Method 12](#_Toc34842802)

[Part 3: Sigmoid Perceptron Result 12](#_Toc34842803)

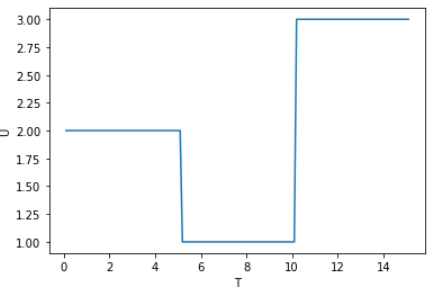
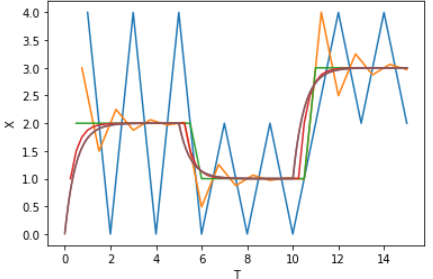
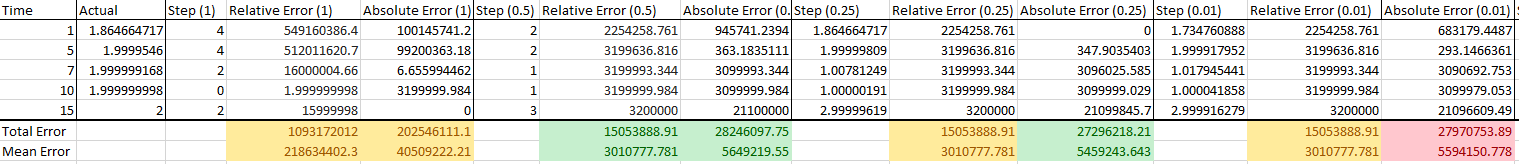
[Part 3: Effect of Learning Rates on Sigmoid Perceptron 13](#_Toc34842804)

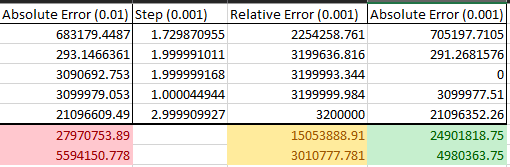
# Introduction

This report covers the simulation of a robot moving in 2D space across time. To simulate the movements of the robot, Euler’s method for solving differential equations at different time steps is implemented and compared against the actual result. This gives a dataset that can be used in further sections of the report. Noise is added to the robot to simulate a different robot moving in 2D space disrupting the signal of our robot. This noisy value is then passed to a machine learning perceptron which attempts to predict the correct position of the robot based on previous locations of the robot.

I expect the path of the robot to look something like figure 1 with the perceptron fitting the line but showing clear errors in the progress.

# Part 1 – Euler’s Algorithm



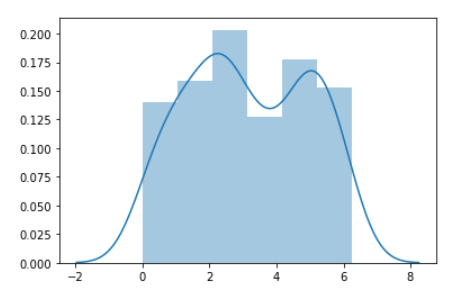
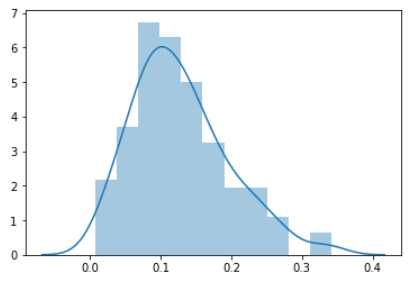
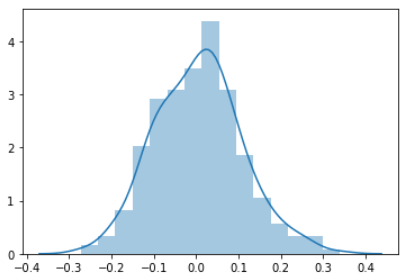
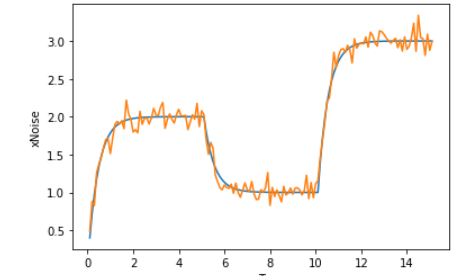


## Values of the Simulation

## Comparison of Step Sizes (Error)

## Actual Values

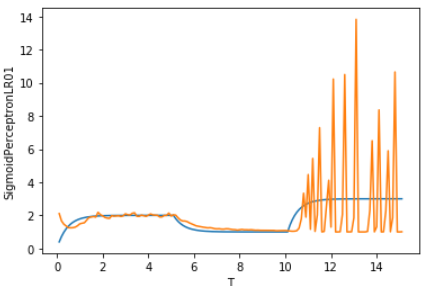
# Part 2 – Creating Noise Using Box Muller Algorithm



## Why Create Noise?

## Normal Distribution

## Part 3 – Perceptron learning Algorithms



## Step Activation

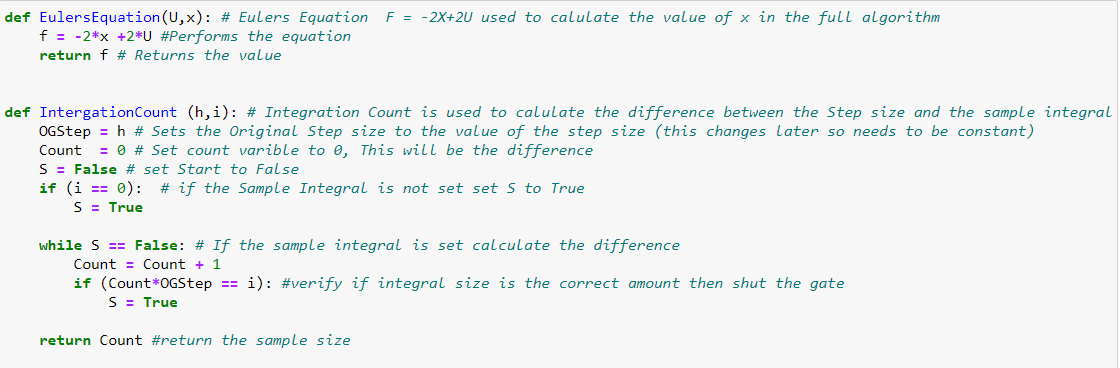
## Sigmoid Activation

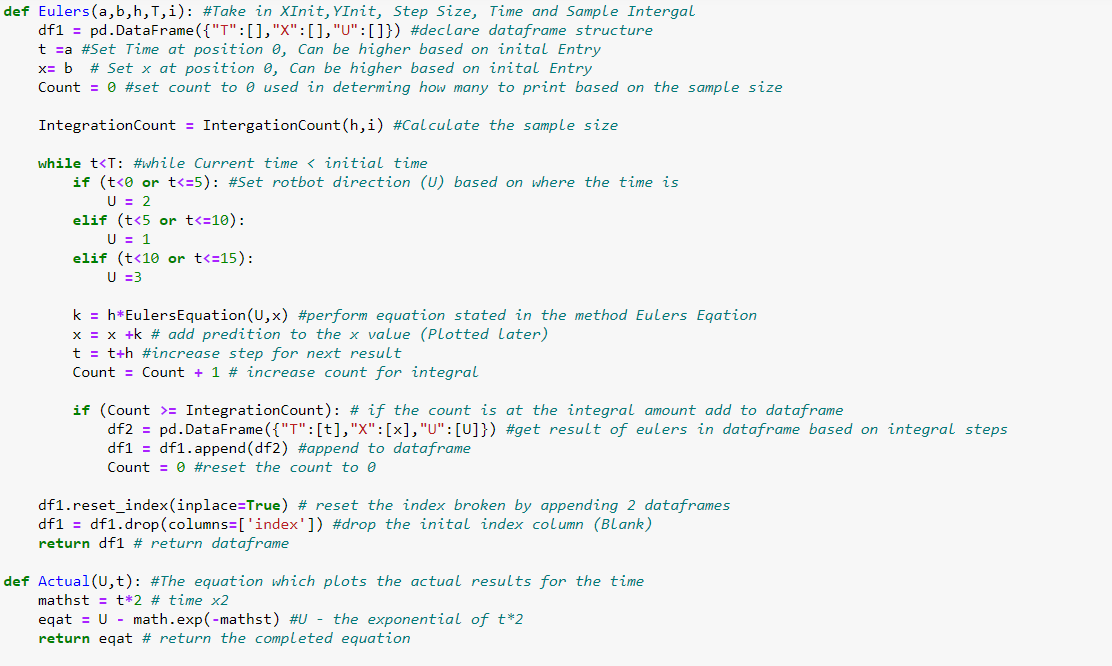
# Conclusion

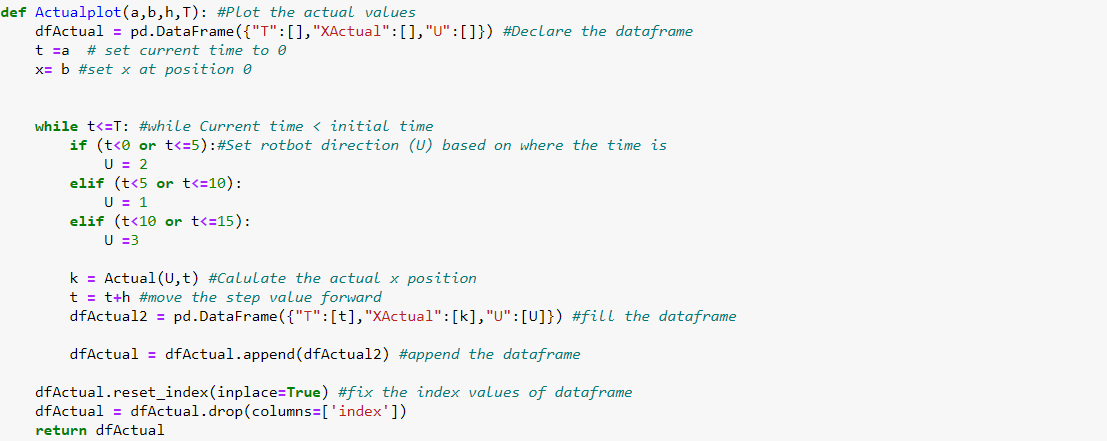
# Appendix

## Code

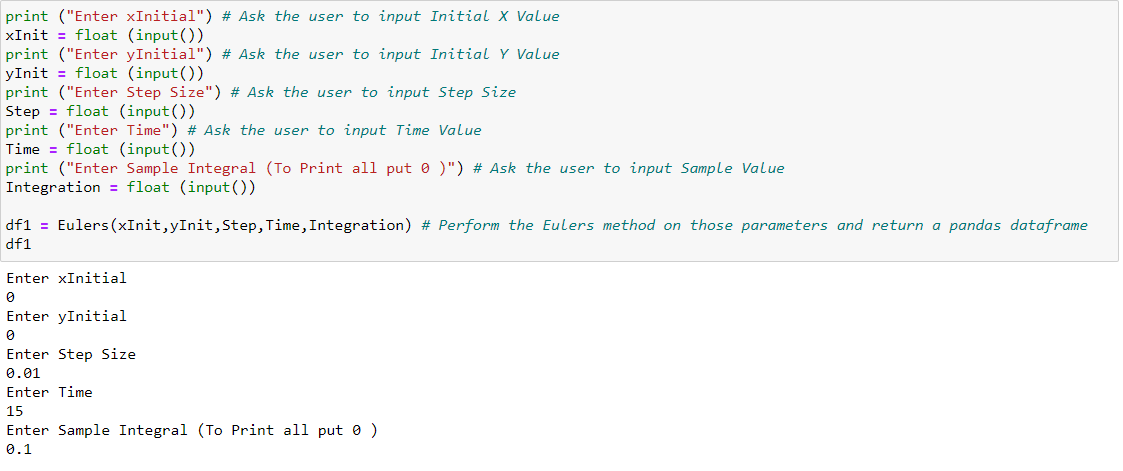
### Part 1: Methods (Euler’s Algorithm, Integral Calculation, Calculation of Actual Results)



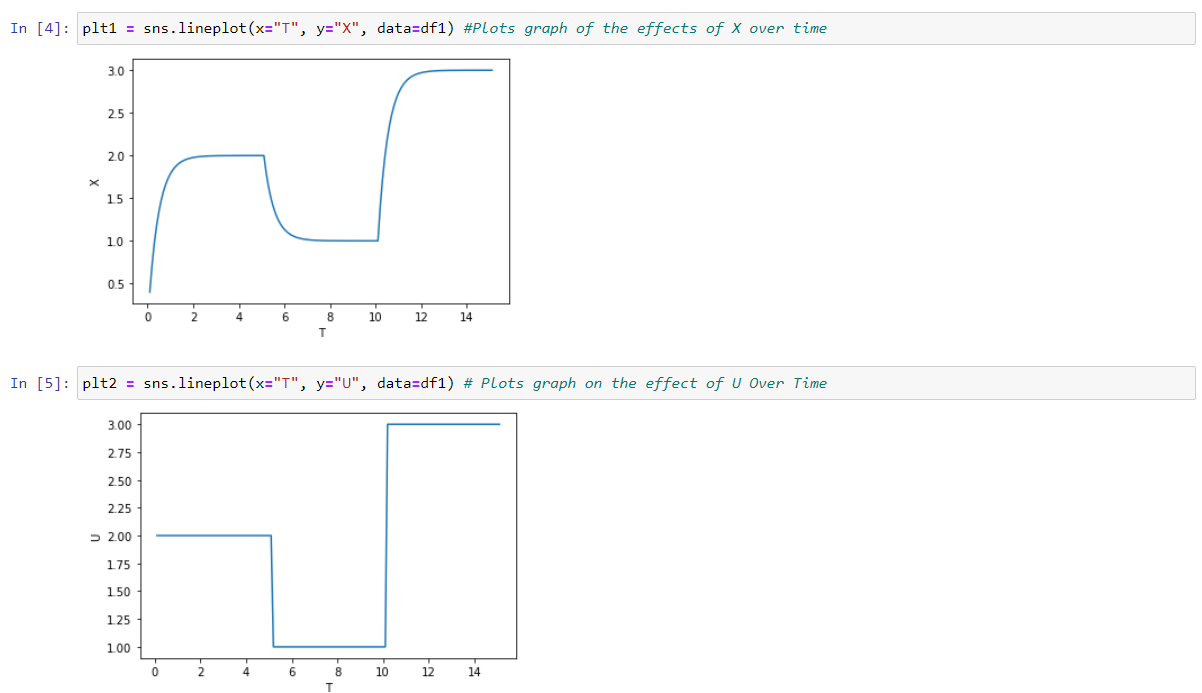


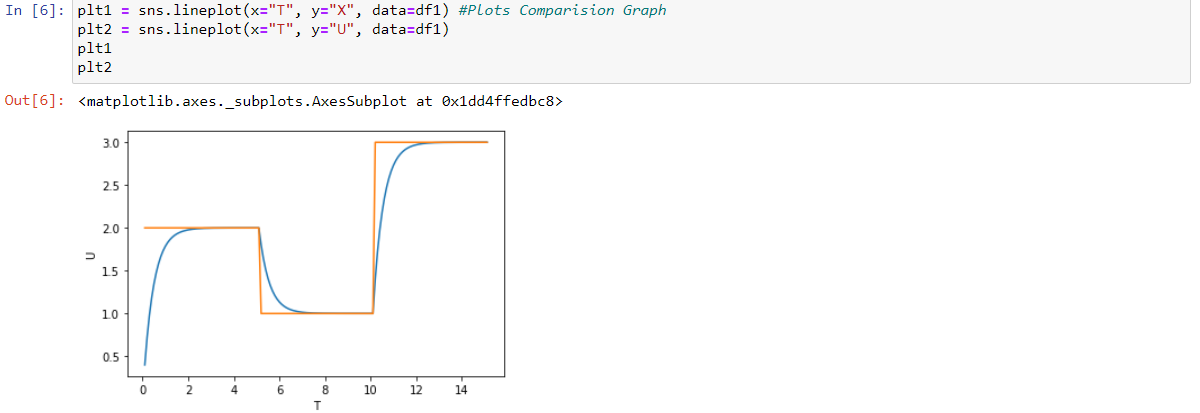


### Part 1: Coded Results



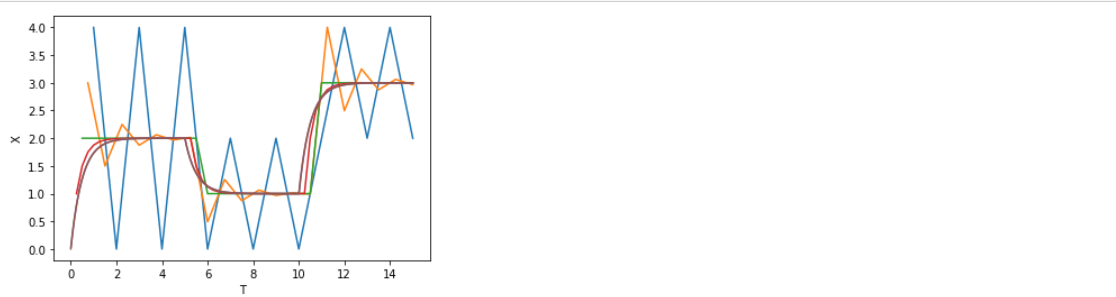




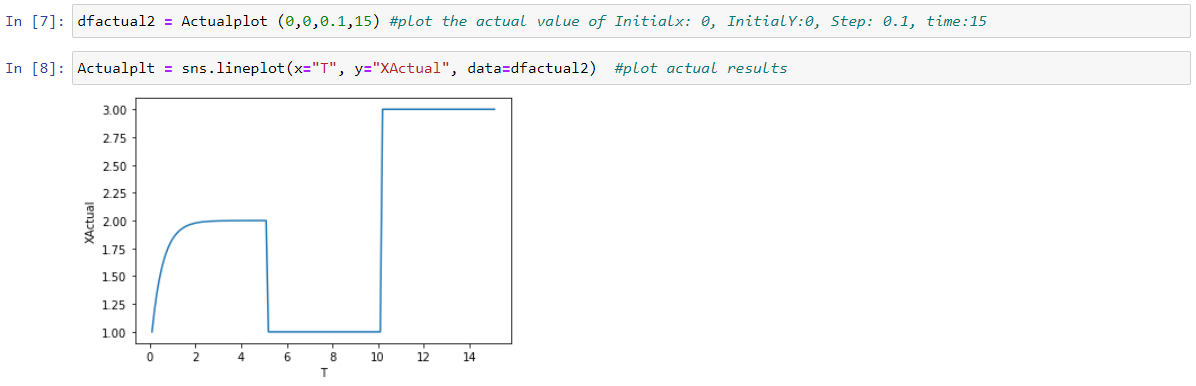


### Part 1: Coded results for Different Step Sizes

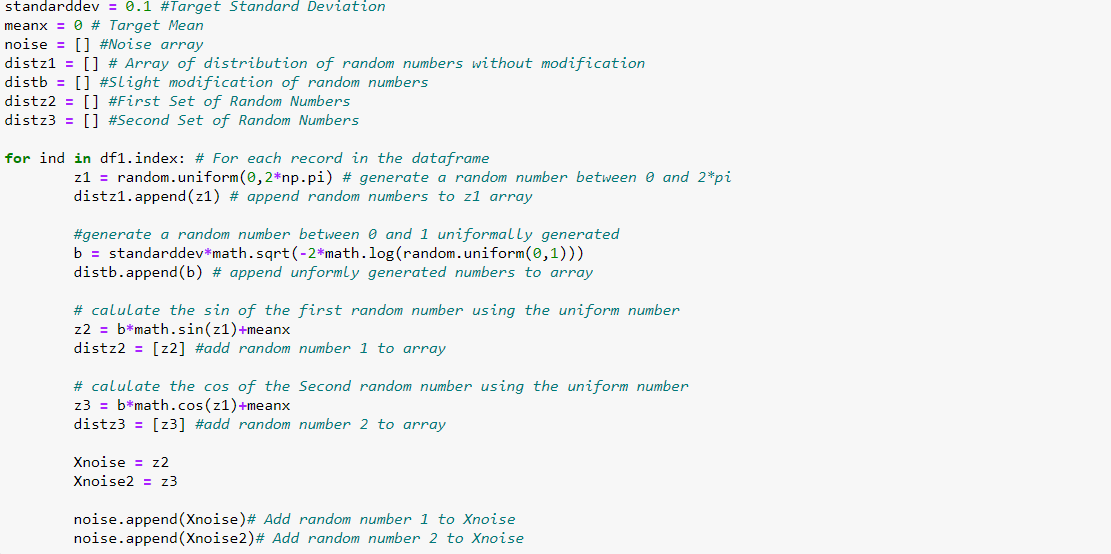




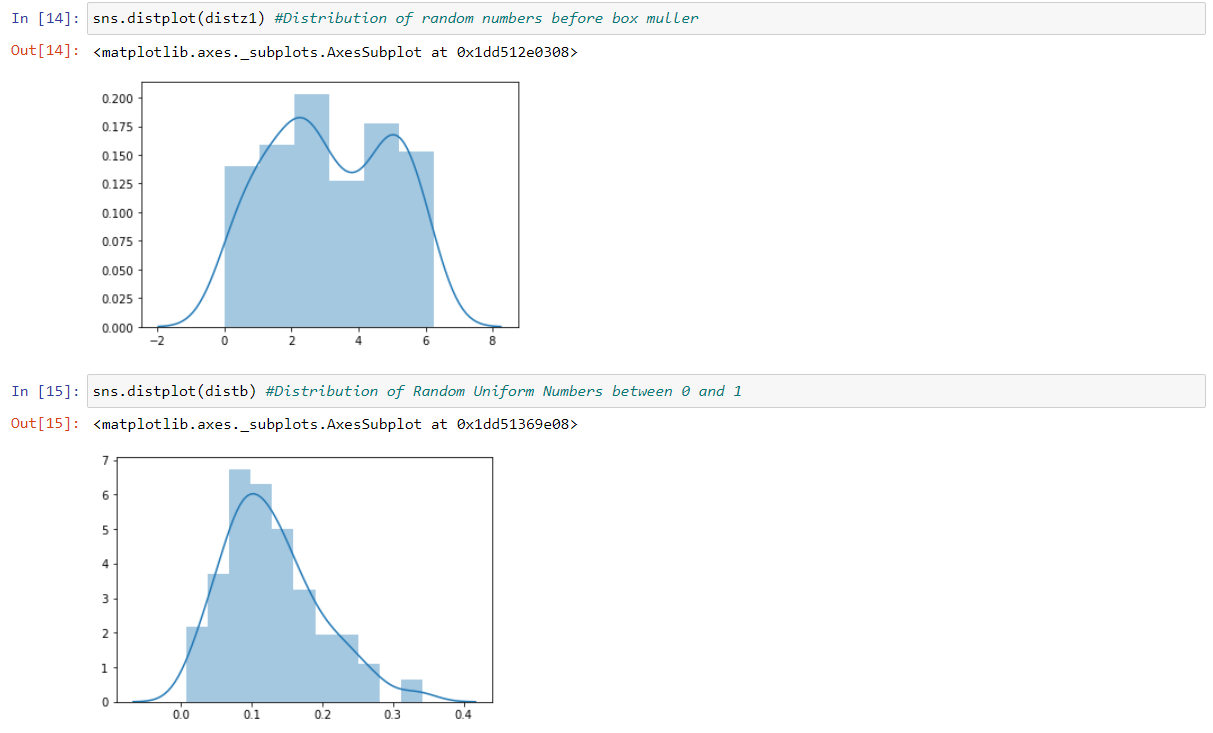
### Part 1: Actual Results Plotted

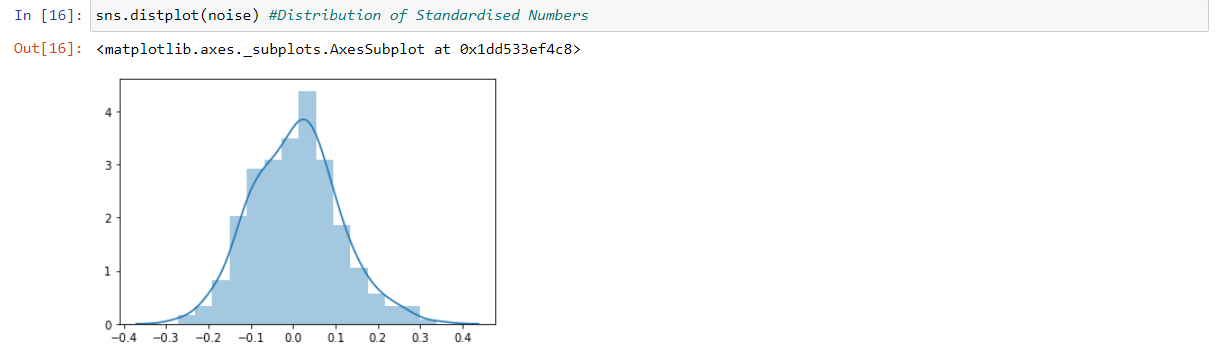


### Part 2: Methods (Box Muller Method)

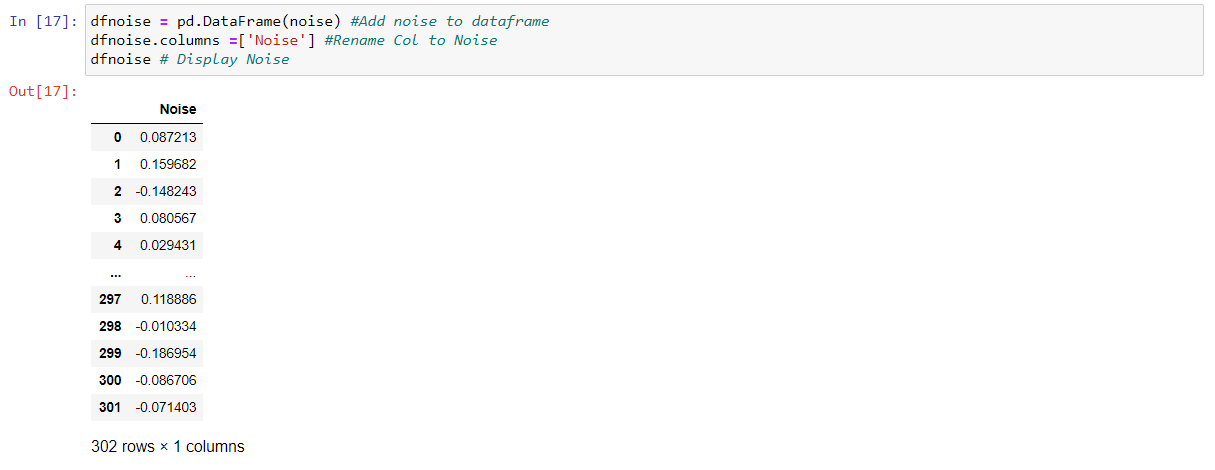


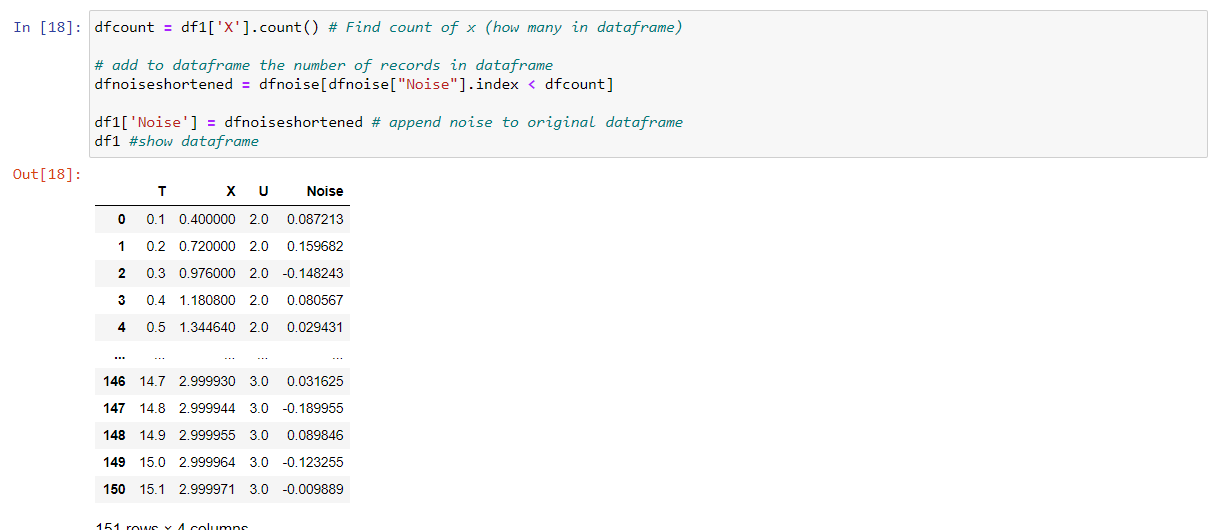
### Part 2: Plots of Random Numbers

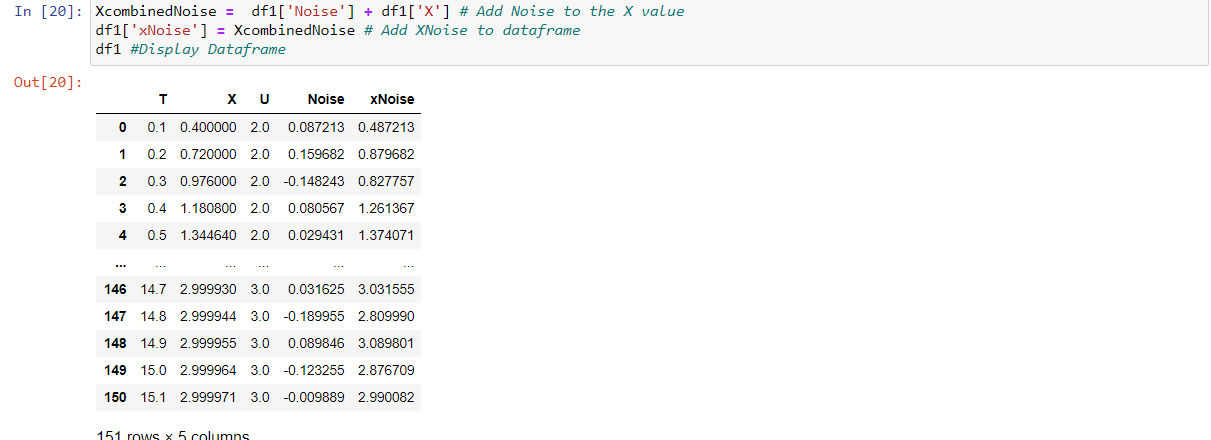




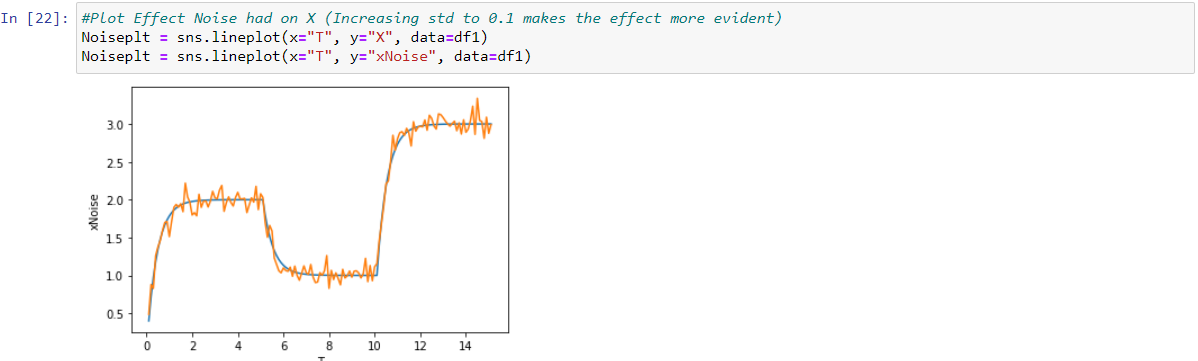
### Part 2: Add Results to X Data



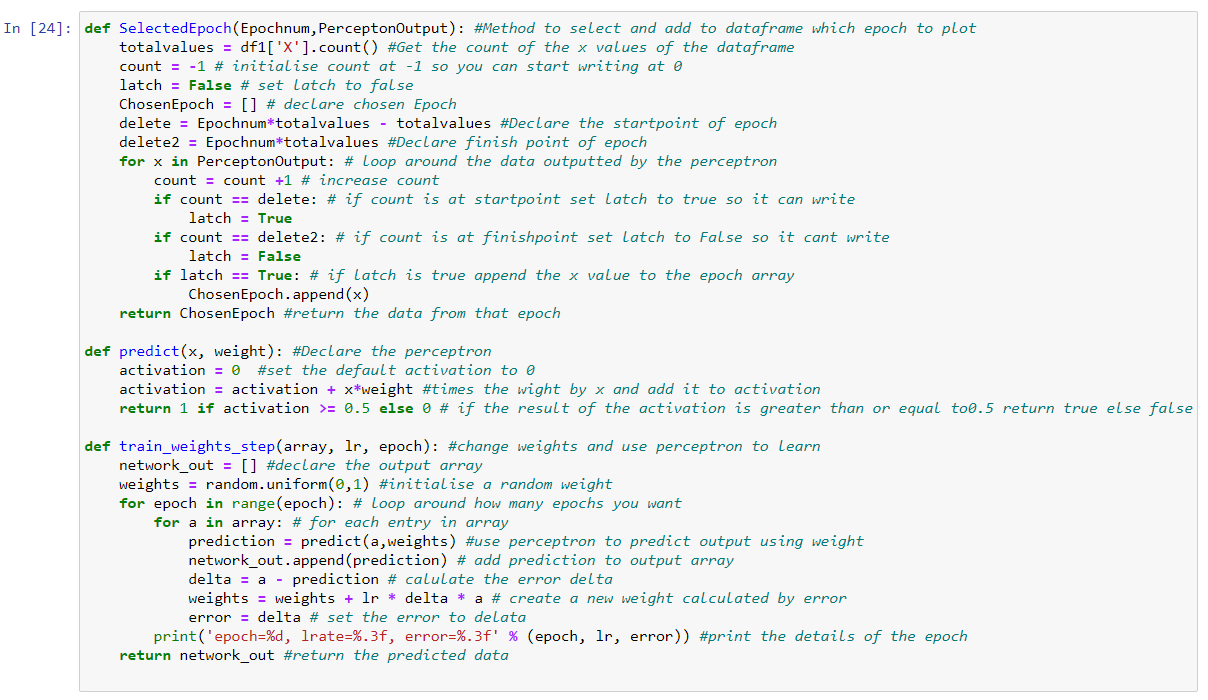




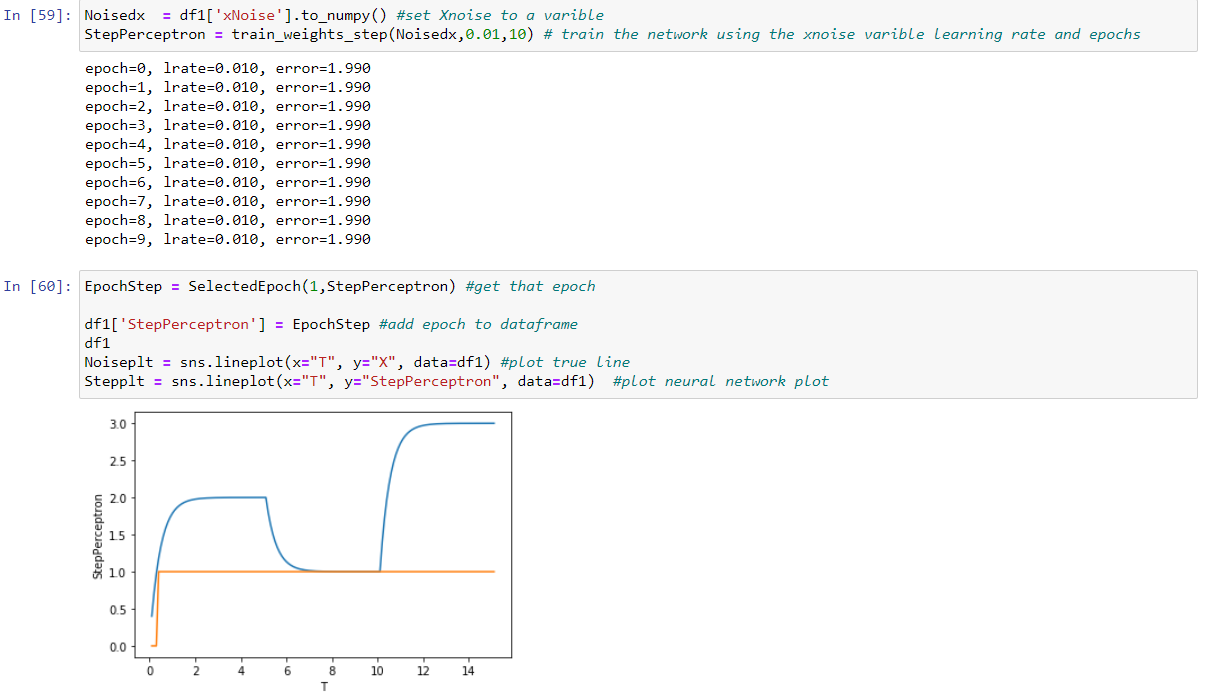
### Part 2: Plot X Noise against Actual for Step



### Part 3: Step Function Perceptron

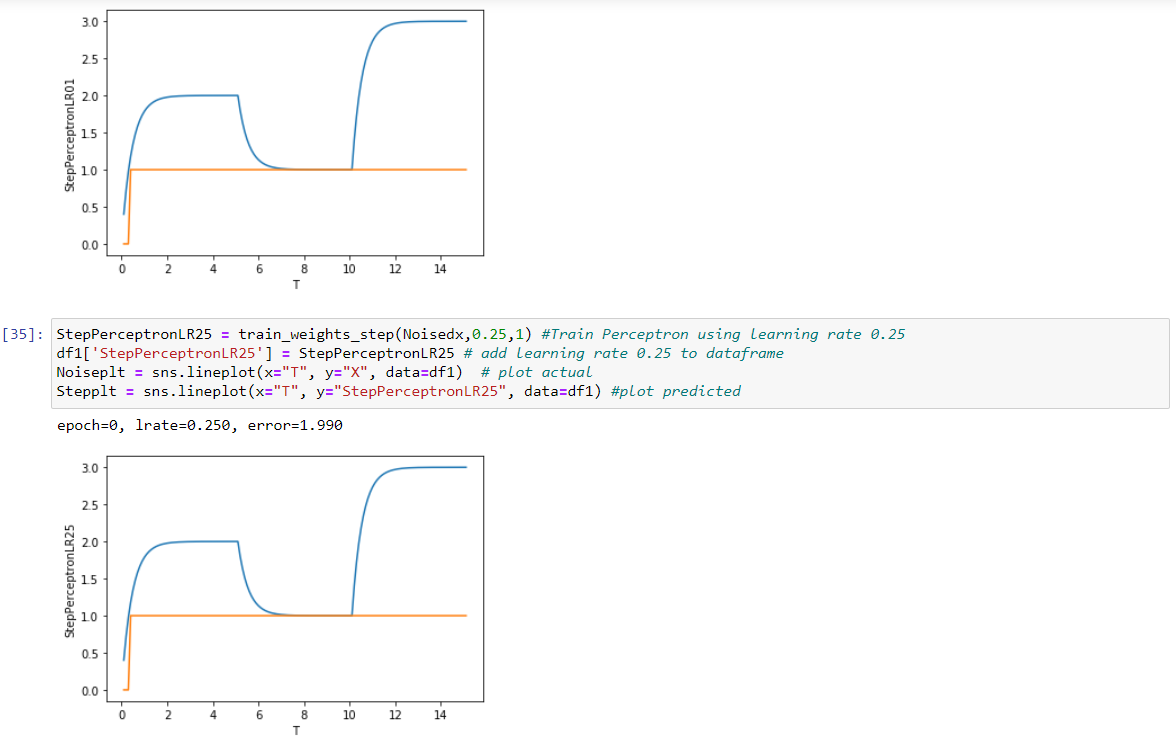


### Part 3: Step Function Perceptron Results



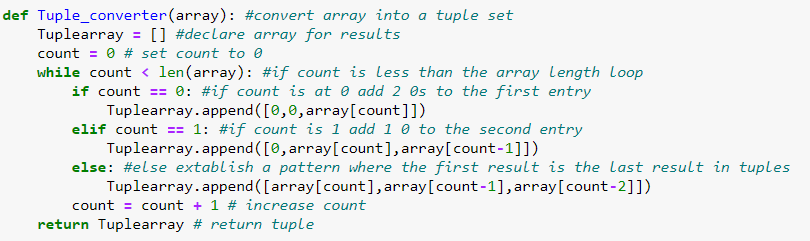
### Part 3: Step Function Change in Learning Rate



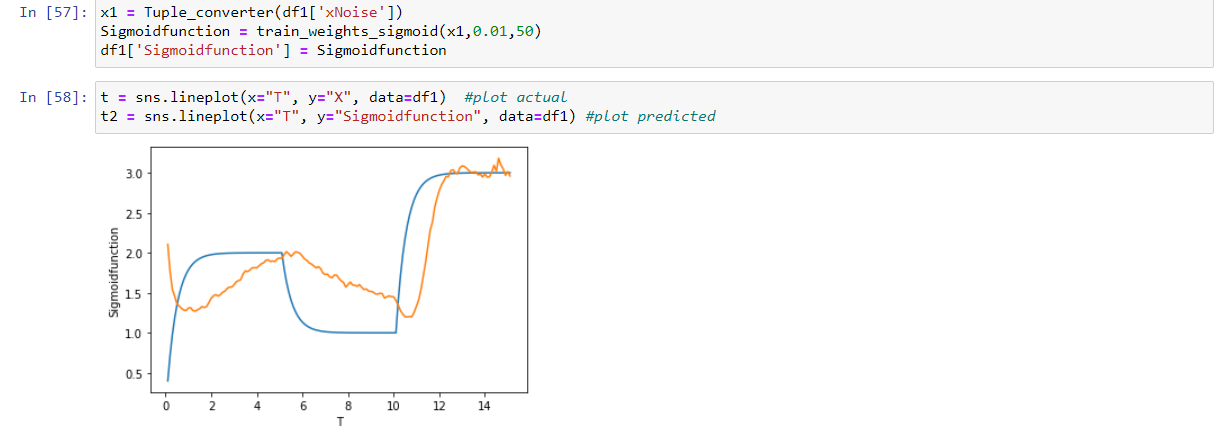


### Part 3: Sigmoid Function Perceptron Method





### Part 3: Sigmoid Perceptron Result



### Part 3: Effect of Learning Rates on Sigmoid Perceptron

