# **ECOR 1010 – Introduction to Engineering**

**Dorian Wang**

**Assignment 8**

**Assignment Title: Programming with Matlab**

***TO Marking TA:***

Name: Neil Cory

Email: […@....ca](mailto:...@....ca)

**Lab Section: L17**

**Room: 4301CB – Carleton University**

**FROM:**

Email: [dorianwang@cmail.carleton.ca](mailto:dorianwang@cmail.carleton.ca)

Student Number: 101009020

Number of Figures and Tables (Including handwritten ones): 7

Last Date and Time of Revision: DD/MM/YYYY @ HH:MM (AM/PM)

# Introduction

Matlab may be used as a simple programming language. It is faster to code than many other languages, and allows simple code for manipulating matrices and vectors.

# Materials and Methods

Matlab

Coding knowledge

# Results

The results are all as expected. The first function outputs x, y and x\*y^2.

The second function creates a plot from the piecewise function given.

The third function may have bugs, but I haven’t been able to isolate it.

The last function runs quite fast, and produces the series required.

# Conclusions

Matlab is somewhat useful for programming real life tasks. The support for matrix input allows for quick coding of matrix and vector manipulating scripts. However, other languages are faster (c/c++) and simpler to understand (python, lua). Matlab should not be used for general purpose coding.

Matlab functions are more useful for running against wide arrays of input, with expected input and expected output, while scripts do not. This makes functions more useful for larger tasks, where pieces of code need to be reused.

# APPENDIces- Figures and Tables

|  |
| --- |
| function Q2 = s(x, y)  xDim = size(x);  yDim = size(y);    if (xDim(1, 2) ~= 1) %checks to see if the x input is correct  if (xDim(1, 1) == 1)  x = transpose(x); %Makes it correct if it isn’t  else  'The input is not correct.'  return  end  end    if (yDim(1, 2) ~= 1) %ditto above, just with y  if (yDim(1, 1) == 1)  y = transpose(y);  else  'The input is not correct.'  return  end  end    if (size(x) ~= size(y)) %If the sizes aren’t the same, return error  'Input is incompatable.'  return  end    xy2 = [];  disp(size(x))  disp(size(y))  for n = 1:size(x) %Makes xy^2 column  xy2 = [xy2; x(n, 1)\*(y(n, 1)^2)];  end    disp(' x y xy^2')  answ = [x, y, xy2];  disp(answ)    end |

**Figure 1.**

|  |
| --- |
| for i = -40:0.1:20%Counts up from -40 to 20 in increments of 0.1  if (i >= 9.0)  plot(i, 1.5\*sqrt(4\*i)+10, '\*'); hold on;  elseif((i < 9.0) && (i >= 0.0))  plot(i, 38/(11 - i), '\*'); hold on;  else  plot(i, i\*sin(i) + 38/11, '\*'); hold on;  end  end      xlabel('x')  ylabel('y')  title('Dorian Wang Matlab plot of piecewise function') |

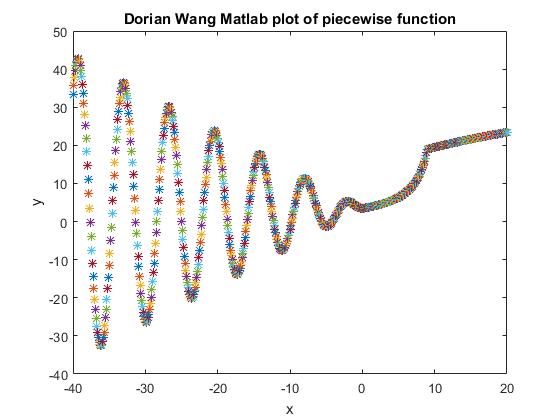
**Figure 2.**

|  |
| --- |
| function Q4 = y(distance, fuel)    sizeD = size(distance);    KMpLToMpG = 2.35214583;    if ((size(distance) ~= size(fuel)) | (sizeD(1, 2) ~= 1))  disp('Input is not the correct size.')  return %error  end    KmPer100L = zeros(sizeD(1,1), sizeD(1,2));    for n = 1:sizeD(1,1)  KmPer100L(n, 1) = distance(n, 1)/fuel(n, 1) \* 100.0;  end    MlPerG = zeros(sizeD(1,1), sizeD(1,2));    for n = 1:sizeD(1,1)  MlPerG(n, 1) = KmPer100L(n, 1) \* KMpLToMpG / 100.0;  end    months = zeros(sizeD(1, 1), 1);  for n = 1:sizeD(1, 1)  months(n, 1) = n;  end    month = months;  MilesPerGallon = MlPerG;  fuelUsed = fuel;  distanceTraveled = distance;    T = table(month, distanceTraveled, fuelUsed, MilesPerGallon, KmPer100L);    disp(T)  fprintf('The average fuel efficiency is %f MPG.\n', sum(MilesPerGallon(:))/ sizeD(1, 1));  fprintf('The average fuel consumption is %f L/100 km.\n', sum(KmPer100L(:))/ sizeD(1, 1));    end |

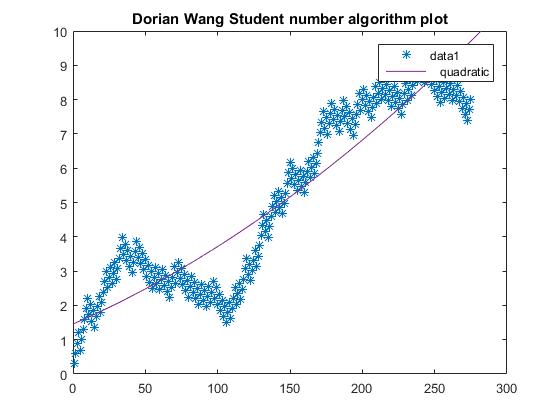
**Figure 3.**

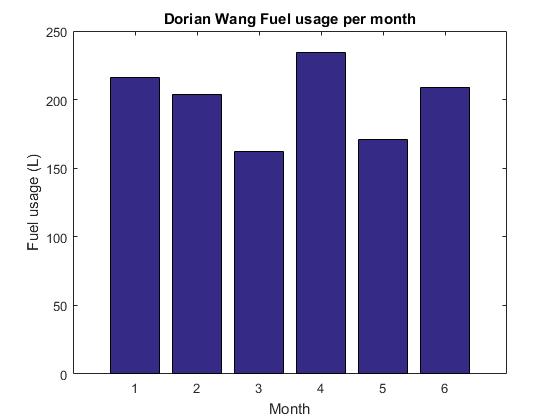
|  |
| --- |
| function Q5 = y(input)    if size(input) ~= size(1)  return % Error  end    inputLog = zeros(275, 1);  count = 0;    while input ~= 1 %While the number is not zero, run algorithm  count = count + 1;  inputLog(count, 1) = input; %Record numbers  if mod(input, 2) == 0  input = input / 2;  else  input = input \* 3 + 1;  end  end    revSeries = flipud(inputLog); %flips the vector from top to bottom  revSeries = log10(revSeries); %applies the function log10 to each element    plot((1:275), revSeries, '\*'); %Makes a pretty plot. I liked the rainbows, so I kept it.    end |

**Figure 4.**



**Figure 5.**

**Figure 6.**

****

**Figure 7.**