Signals and Systems 3TP3

MATLAB #1

Instructor: Dr. Jun Chen

Guy-Jacques Isombe – L08 – isombeg - 400137394
Olayiwola Bakare – L08 – bakaro2- 400130008

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario. Submitted by **[Guy–Jacques Isombe, isombeg, 400137394]**

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario. Submitted by **[Olayiwola Bakare, bakaro2, 400130008]**

Question 1

The purpose of this exercise was to write a script which graphed the following discrete functions:

- $x[n] = u[n] - 2u[n-1] + u[n-4]$ (illustrated in Figure 1.1)
- $x[n] = (n+2)u[n+2] - 2u[n] - nu[n-4]$ (illustrated in Figure 1.2)
- $x[n] = \delta[n+1] - \delta[n] + u[n+1] - u[n-2]$ (illustrated in Figure 1.3)
- $x[n] = \delta[n+1] - \delta[n] + u[n+1] - u[n-2]$ (illustrated in Figure 1.4)

The script for question 1 was the following:

Code Sample 1.1: Script for Question 1

```
% Define discrete values to be used
t = [-10:10];

% Initialize matrix to store DTS values
x = zeros(4,21);

% Computing values and storing within matrix
x(1,:) = unitstep(t) - 2*unitstep(t) - 2*unitstep(t-1) + unitstep(t-4);
x(2,:) = (t+2).*unitstep(t) - 2*unitstep(t) - t.*(t-4);
x(3,:) = unitpulse(t+1) - unitpulse(t) + unitstep(t+1) - unitstep(t-2);
x(4,:) = exp(0.8*t) .* unitstep(t+1) + unitstep(t);

%Displaying DTSs
figure;
stem(x(1,:));
title('Guy-Jacques Isombe 400137394, Olayiwola Bakare 400130008');
xlabel('n');
ylabel('x[n]');

figure;
plot(x(2,:));
title('Guy-Jacques Isombe 400137394, Olayiwola Bakare 400130008');
xlabel('n');
ylabel('x[n]');

figure;
stem(x(3,:));
title('Guy-Jacques Isombe 400137394, Olayiwola Bakare 400130008');
xlabel('n');
ylabel('x[n]');

figure;
stem(x(4,:));
title('Guy-Jacques Isombe 400137394, Olayiwola Bakare 400130008');
xlabel('n');
ylabel('x[n]');
```

This script made use of the pre-defined unit-step function as well as unit-pulse function which we defined.

Code Sample 1.2: Unit-Step Function

```
function y = unitstep(x)
  %The unit step function, u(x)

  if (nargin != 1)
    disp('unit step requires 1 argument');
    return;
  endif

  y = cast(x >= 0, class(x));
endfunction
```

Code Sample 1.3: Unit-Pulse Function

```
function y = unitpulse(x)
  % Value will equal 1 when element in x is 0
  y = x == 0;
endfunction
```

The script produced the following plots:

Figure 1.1: $x[n] = u[n] - 2u[n-1] + u[n-4]$

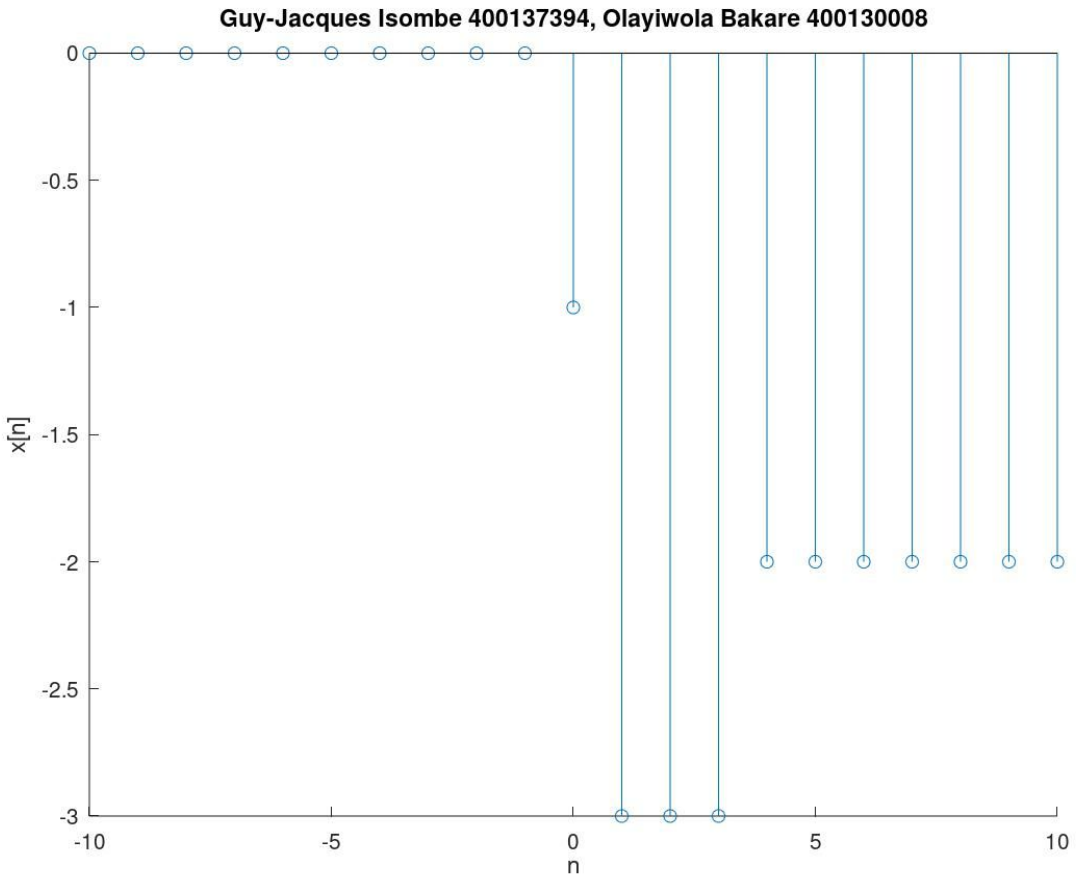


Figure 1.2: x[n] = (n + 2)u[n + 2] − 2u[n] − nu[n − 4]

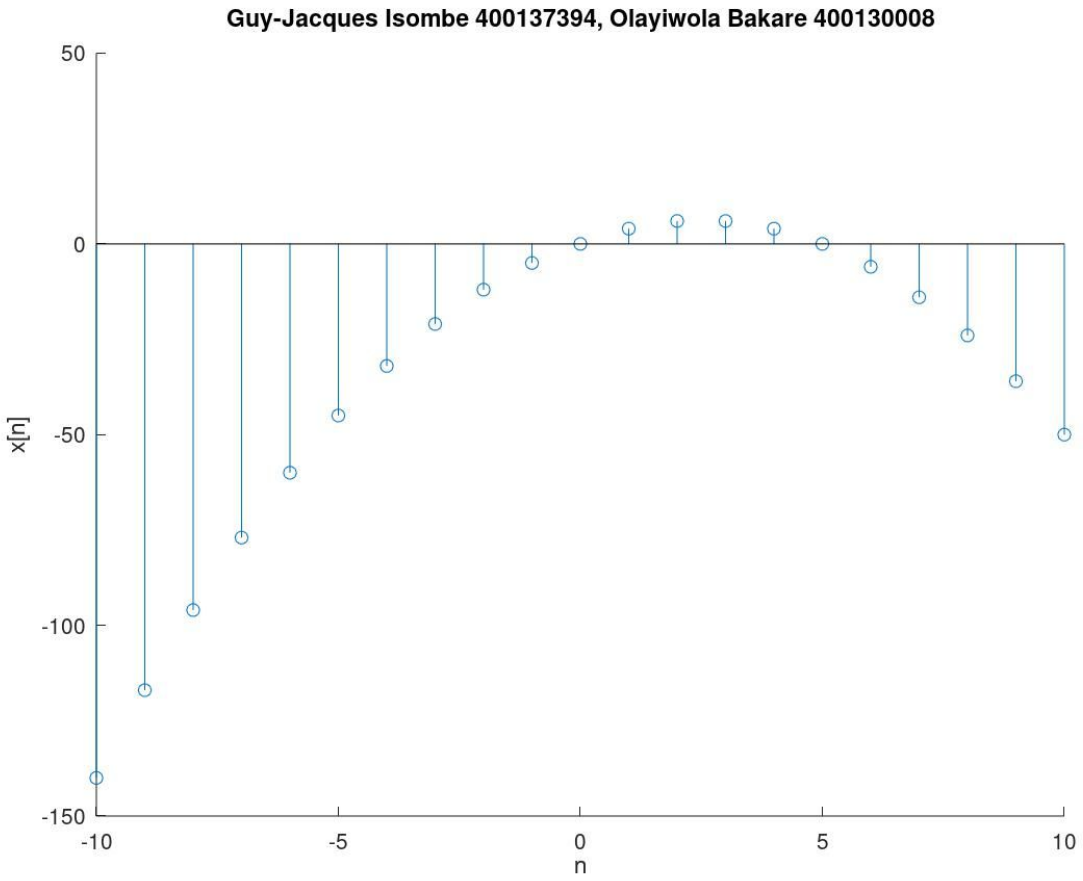


Figure 1.3: $x[n] = \delta[n + 1] - \delta[n] + u[n + 1] - u[n - 2]$

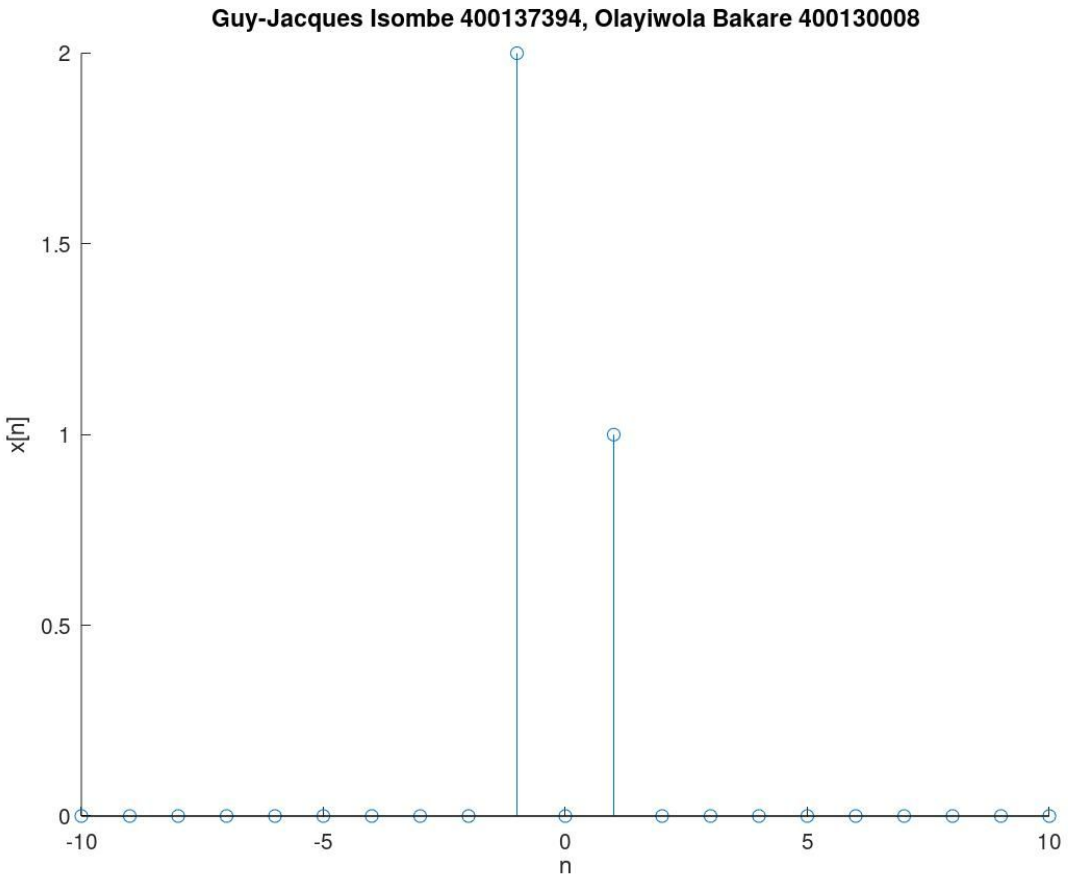
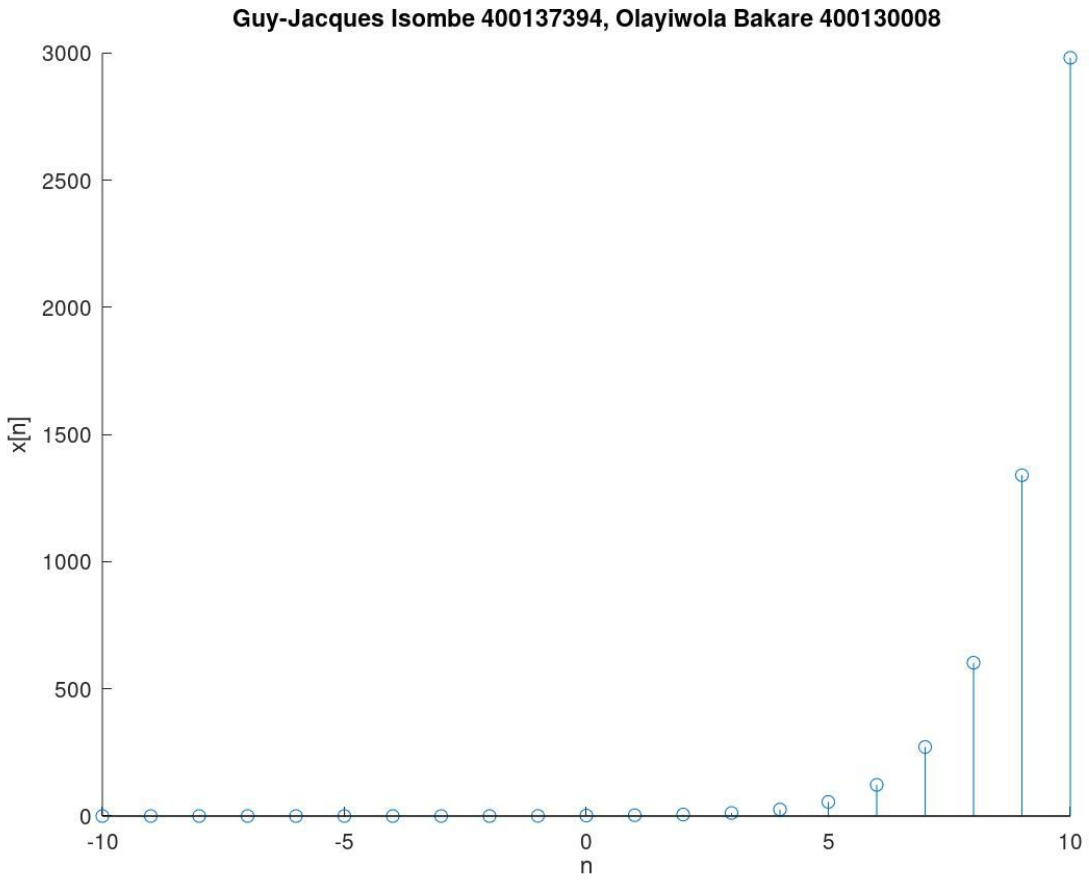


Figure 1.4: $x[n] = \delta[n + 1] - \delta[n] + u[n + 1] - u[n - 2]$


Guy-Jacques Isombe 400137394, Olayiwola Bakare 400130008
3000
2500
2000
1500
1000
500
0
x[n]
-10
-5
0
5
10
n

Question 2

Posted below are our function and test files for question 2. The test files contained the solutions to the question. The function is just a generic function that returns the averages. We explained our code and thought process in the report below

Code Sample 2.1: Average Computing Function

```
function ans = Lab_1_2a(fileName, maxVec, col)
    %this function calculates the averages, given a
certain input column
    %and stores the re in a variable.
    ans = ((sum((fileName(2:size(fileName,
1),col)./maxVec(col)),2))*100)/length(col);
end
```

Code Sample 2.2: Exercise 2 Script

```
nameOfFile = csvread('course_grades_2019.csv');
maxRow = csvread('course_grades_2019.csv', 0,0,[0,0,0,11]);

%Average Lab Mark and Overall Avg Lab Mark-- From Columns 2 to 5
labVector = [2:5];
avgLabMarks = Lab_1_2a(nameOfFile, maxRow, labVector);
overallAvgLabGrade = sum(avgLabMarks)./ length(avgLabMarks);


%Average Exam Mark and Overall Avg Exam Mark-- From Columns 7 to 12
examVector = [7:12];
avgExamMarks = Lab_1_2a(nameOfFile, maxRow, examVector);
overallAvgExamGrade = sum(avgExamMarks)./ length(avgExamMarks);


%Final Grade -- From Columns 2 to 12
midtermVector = [6];
avgMidtermMarks = Lab_1_2a(nameOfFile, maxRow, midtermVector);
finalGrade = avgLabMarks.*(0.3) + avgExamMarks.*(0.4) + avgMidtermMarks.*(0.3)
```

The purpose of question 2 was to come up with a function that can easily calculate every students individual average grade, given a certain beginning and end column. The columns are used to represent students' grades. Being able to compute the average of all students' grades would make it easier to work

with the results. For example, if we had an average of all grades that we wanted then we would be able to easily calculate each students final mark.

Our function to calculate the average grades has three input parameters fileName, maxVec, and col. fileName is the name of the csv file. maxVec represents the column with total marks and col is the specific column that we are working with.

We used the size function to get the size of our fileName. This returned the total size of the row and column. We then indexed it at 1, so we could have the overall size of the row. We then put this result into fileName. We passed (2:size(fileName, 1) and the specific column to fileName, which resulted in us getting the row where the students and their specified column marks are. We then divided this result by maxVec(col), which resulted in a matrix of what the students got over the total mark. For example if maxVec was 25, 20, and 40 respectively and a student's mark were 20, 15, and 35. Then we would get 20/25, 15/20, 35/40 in a matrix, this was what we got for every single student. We then used the sum function and summed up each row. So continuing with the example stated above, our results would be (20/25 + 15/20 + 35/40). Then we multiplied by 100 and divided by the length of the column that was passed into our function. This would then give us each student's average.This was implemented using the code from Code Sample 2.1.

Since we already have a function that can calculate each students average given a certain input column, then the remainder of question 2 was relatively straightforward to compute. We know that the lab Marks are from column 2 to 5. So we passed this to our function and stored the result in a variable. The variable then represented the average lab marks. To get the overall average lab grade, we simply just divided the sum of the result by the length of our result. To get the answer for the exam marks we did the same thing that we did for the average and overall average lab mark, but the exam marks started from column 7 to 12 which is what we passed to the function.
Lastly, to get the Final Grade, we were told that the lab and midterm marks are both worth 30%, and the exam marks were worth 40%. Since we already have a variable that stores the average lab marks and the average exam marks, we made use of those variables, but to get the average midterm mark, we simply just passed column 6 to our function and stored the result in a variable. We then multiplied the average lab marks, average exam marks, and average midterm marks by 0.3, 0.4, and 0.3 respectively. Which then returned the final grade that we wanted.

Question 3

The purpose of this exercise was to write a script to improve the quality of Figure 3.1's left image through processing. It was identified that the issue with it was that the contrast was low. This is illustrated in Figure 3.2 left diagram, a histogram, which illustrates the frequency of pixel values within given ranges. It can be observed that most pixel values are clustered together. This causes the objects with the image to be difficult to differentiate. Therefore, we were instructed to put the pixel values through the following function: $v^0(x, y) = \alpha\, v(x, y) + \beta$ (implemented in Code Sample 3.2) where $v(x, y)$ is the pixel's value

at coordinates x and y, α and β are chosen constants and $v^0(x, y)$ is the new pixel value at coordinates x and y. It was observed that increasing α increases the contrast of the picture since the difference between pixel values in the image also increases by the same factor they're are being multiplied with. After a few attempts, it was decided that 6.0 was an appropriate value for α. Clustering was minimized without the differential between pixels being exaggerated. The issue with this was that pixel values which exceed 255 are represented as completely white. Therefore, we had to reduce the brightness. It was observed that β was the parameter that influenced brightness. Setting it to -890 managed to offset all pixel values sufficiently so that most values were not only below 250 but also of appropriate brightness for the given image.

The numerical results of this manipulation is represented in Figure 3.2's bottom image. It can be observed that the range of pixel values is much broader in the right diagram than the upper diagram. This means that the new image is much more contrasted than the original. Additionally, it can be observed that the median value is much more centered in the right diagram than the left. This means that the new image should be slightly less bright than the original. This can be confirmed by the comparison of images in Figure 3.1.

Code Sample 3.1: Question 3 Script

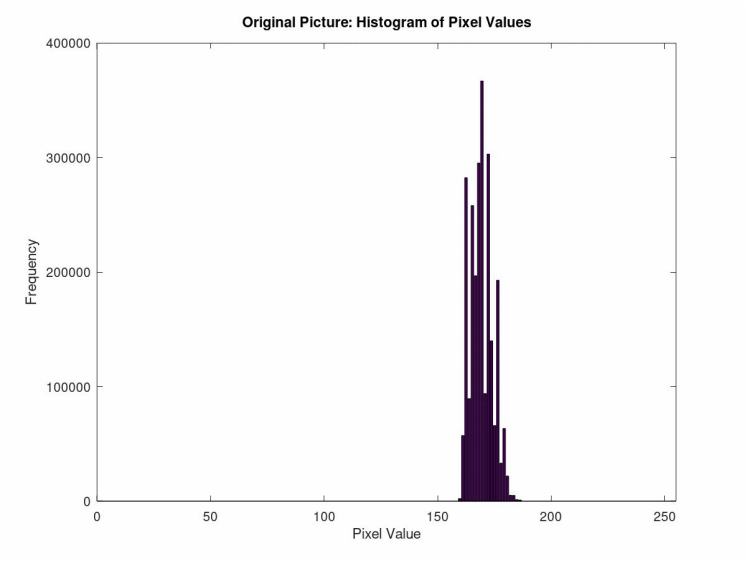
```
% Retrieve and format data
image = imread('ee3tp3picture2019.png');
image_of_doubles = double(image);

% Edit and fix picture
fixed_image = contrast_image(image_of_doubles, 6.0, -890);

% Plotting histogram of original image's pixel values
figure;
[n_elements, centers] = hist(image_of_doubles(:),20);
bar(centers, n_elements);
xlim([0 255]);
title('Original Picture: Histogram of Pixel Values');
xlabel('Pixel Value')
ylabel('Frequency');

% Plotting histogram of fixed image's pixel values
figure;
[n_elements, centers] = hist(fixed_image(:),20);
bar(centers, n_elements);
xlim([0 255]);
title('Fixed Picture: Histogram of Pixel Values');
xlabel('Pixel Value')
ylabel('Frequency');

% Display original image
figure;
imshow(uint8(image_of_doubles));

% Display fixed image
figure;
imshow(uint8(fixed_image));

% Save edited picture
imwrite(uint8(fixed_image), 'saved_image.png');
```

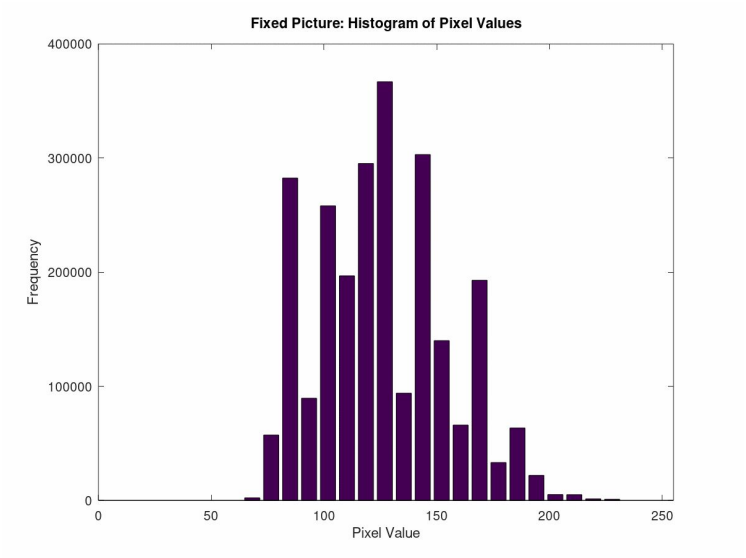
Code Sample 3.2: Function to fix the image

```
function v = contrast_image(img, alpha, beta)
  v = alpha*img + beta;
endfunction
```

Figure 3.1: Original Image (upper) and Fixed Image (lower)

Figure 3.2: Original Image's Histogram (left) and Fixed Image's Histogram (below)




Fixed Picture: Histogram of Pixel Values
400000
300000
200000
100000
0
Frequency
0
50
100
150
200
250
Pixel Value