ELECENG 3TP3 Lab 1

Raeed Hassan
hassam41
McMaster University

December 12, 2020

Question 1

To plot the discrete time signals listed, it was first necessary to create functions for the unit step function, $u[n]$, and the unit impulse function, $\delta[n]$. The MATLAB code for the unit step function is shown in Listing 1, and the Matlab code for the unit impulse function is shown in Listing 2.

Listing 1: unit step function

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

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

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

Listing 2: unit impulse function

```
function y = unitimpulse(x)
% The unit impulse function, delta(x)

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

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

Once the unit step and unit impulse functions have been defined in their own .m files, we can use these to define the specified discrete time signals. We can simply add the individual terms of each signal to calculate the resulting discrete time signal. The Matlab code to calculate each discrete time signal is shown in Listing 3.

Listing 3: Question 1 Signals

```
n = -10:10;

question1a = unitstep(n) - 2*unitstep(n-1) + unitstep(n-4);
question1b = (n+2).*unitstep(n+2) - 2*unitstep(n) - n.*unitstep(n-4);
question1c = unitimpulse(n+1) - unitimpulse(n) + unitstep(n+1) - unitstep(n-2);
question1d = (exp(0.8*n)).*unitstep(n+1) + unitstep(n);
```

The defined discrete time signals can be plotted in MATLAB using the stem function. The MATLAB code used to plot these graphs and create an image of the figure generated is shown in Listing 4, and the image of the plots is shown in 1.

Listing 4: Question 1 Plotting the Signals

```
fig = figure('Name','Question 1');
set(gcf,'WindowState','maximized'); set(0,'defaultAxesFontSize'
    ,20);
sgtitle('Raeed Hassan','FontSize',20);
xlim([-10 10]);

subplot(2,2,1)
stem(n,question1a); title('Question 1a');
xlabel('n'); ylabel('x[n]'); ylim([-1.5 1.5]);
subplot(2,2,2);
stem(n,question1b); title('Question 1b');
xlabel('n'); ylabel('x[n]'); ylim([-0.5 3.5]);
subplot(2,2,3);
stem(n,question1c); title('Question 1c');
xlabel('n'); ylabel('x[n]'); ylim([-0.5 2.5]);
subplot(2,2,4);
stem(n,question1d); title('Question 1d');
xlabel('n'); ylabel('x[n]');

exportgraphics(fig,'..\Figures\question1.png');
```

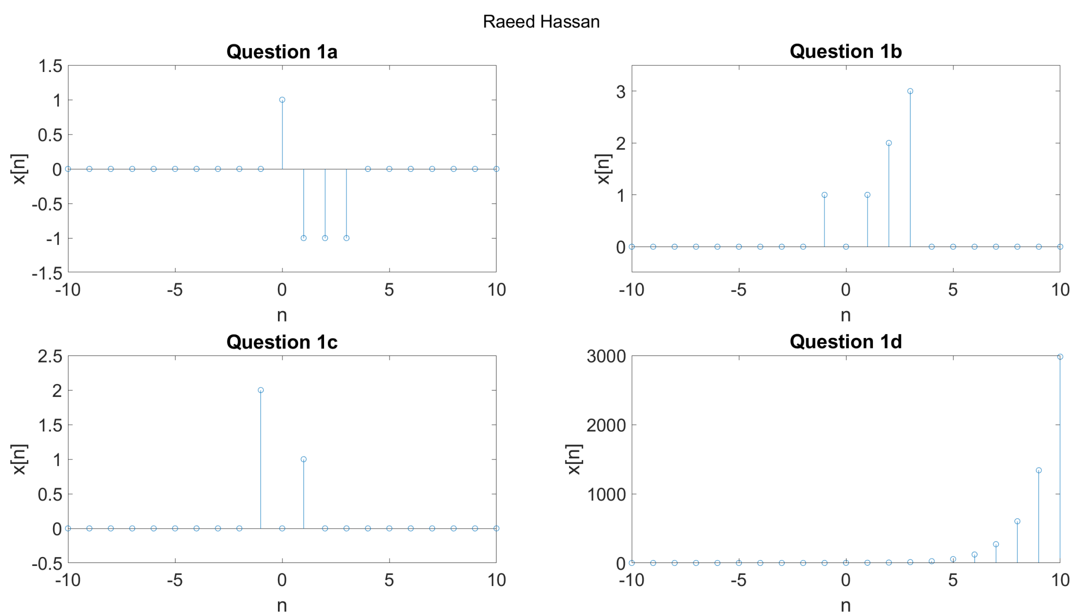

Figure 1: Question 1 Plots

Question 2

2a

The MATLAB function, named question2a, accepts three inputs: the set of student grade records as a matrix, the maximum grade vector, and a vector of column indices. The function performs element-wise division on every grade in the student grade records matrix, grades, by their maximum value in the maximum grade vector, maximum. The resulting matrix contains every grade relative to its maximum grade (a grade of 1). The sum of each row is calculated using Matlab's sum function, then divided by the number of columns to determine the average grade for each student. Every element in the vector is multiplied by 100 to convert the grades to a percentage. The Matlab code for the function is shown in Listing 5.

Listing 5: Question 2a

```
function averages = question2a(grades,maximum,columns)

averages = (100/length(columns))*sum(grades(:,columns-1)./
   maximum(:,columns-1),2);
```

2b

The input for the function created in 2a requires a matrix of the student grace records and a vector of the maximum grades, which were created by using MATLAB's csvread function on the appropriate rows and columns of the course_grades_2020.csv file and stored in `student_grades` and `max_grades`. The columns for lab grades and exam grades were defined in `lab_columns` and `exam_columns`. These were used as inputs for the function created in 2a to calculate average lab and exam marks for each student, stored in the vectors `lab_averages` and `exam_averages`. The overall course averages for labs and exams were calculated by using Matlab's mean function on the vectors. The Matlab code for question 2b is shown in Listing 6.

Listing 6: Question 2b

```
%% Create matrices for student grades and maximum grades from
   csv file
student_grades = csvread('../course_grades_2020.csv'
   ,1,1,[1,1,20,11]);
max_grades = csvread('../course_grades_2020.csv'
   ,0,1,[0,1,0,11]);


%% Define columns for labs and exam
lab_columns = 2:5;
exam_columns = 7:12;

```

```
%% Calculate averages for each student
lab_averages = question2a(student_grades,max_grades,lab_columns);
exam_averages = question2a(student_grades,max_grades,exam_columns);

%% Calculate overall course averages
average_lab_grade_course = mean(lab_averages);
average_exam_grade_course = mean(exam_averages);
```

2c

The MATLAB script for question 2c is similar to the script used in 2b. In addition to the code used to calculate the student averages for the labs and exams, it uses the column for the midterm grades, `midterm_column`, to calculate the student averages for the midterm, stored in the vector `midterm_averages`. The weighted average of the lab, midterm, and exam average vectors is calculated and stored in the vector `final_grades`. To create a plot of the final grades in decreasing order, the final grades are sorted using Matlab's sort function, then plotted using Matlab's bar function. The Matlab code for question 2c is shown in Listing 7. The plot of the final grades in decreasing order of final grade is shown in Figure 2.

Listing 7: Question 2c

```
%% Create matrices for student grades and maximum grades from csv file
student_grades = csvread('../course_grades_2020.csv',1,1,[1,1,20,11]);
max_grades = csvread('../course_grades_2020.csv',0,1,[0,1,0,11]);

%% Define columns for labs and exam
lab_columns = 2:5;
exam_columns = 7:12;
midterm_column = 6;

%% Calculate category averages for each student
lab_averages = question2a(student_grades,max_grades,lab_columns);
exam_averages = question2a(student_grades,max_grades,exam_columns);
midterm_averages = question2a(student_grades,max_grades,midterm_column);

%% Calculate overall average for each student
final_grades = 0.3*lab_averages + 0.3*midterm_averages + 0.4*exam_averages;
```

```
sorted_final_grades = sort(final_grades,'descend');

fig = figure('Name','Question 2c');
sgtitle('Raeed Hassan','FontSize',20)
set(gcf,'WindowState','maximized');
b = bar(sorted_final_grades);
ylabel('Final grade','FontSize',20);
set(gca,'xticklabel',[])
set(gca,'xtick',[])
exportgraphics(fig,'..\Figures\question2c.png');
```

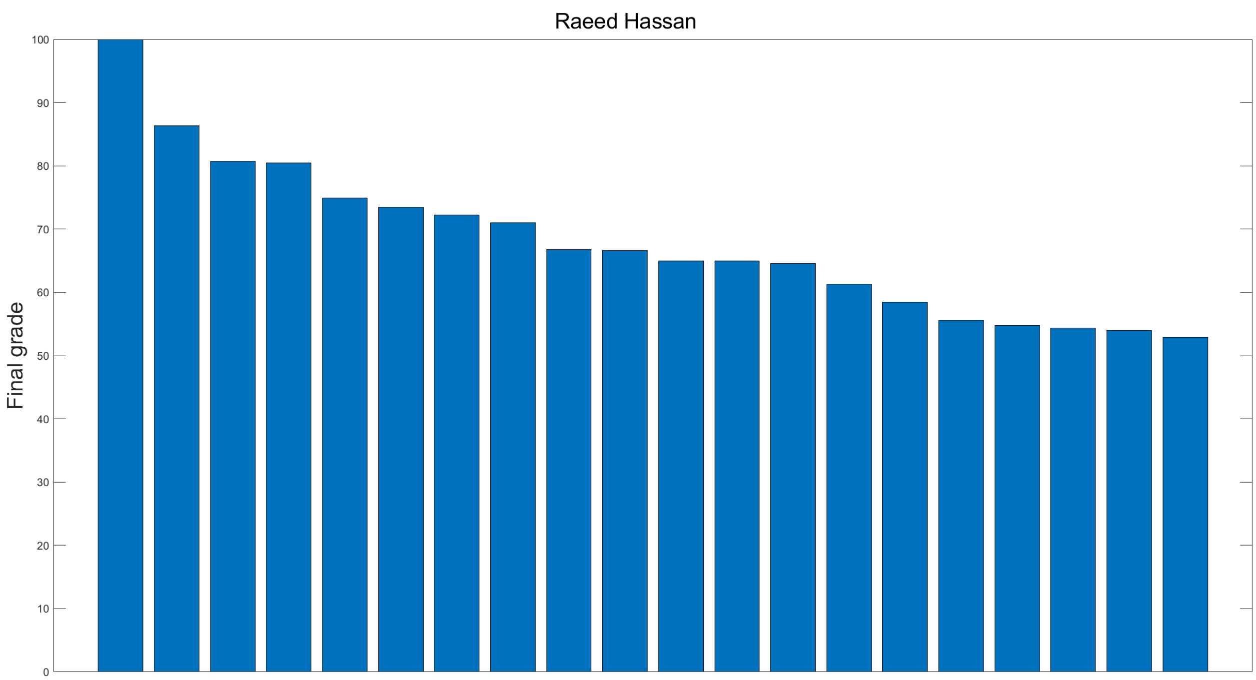

Figure 2: Question 2c

Question 3

3a

A visualization of the greyscale integer values is done in 3a, showing the range of the greyscale integer values of the ee3top3picture2020.png image. From this data, we can see that the values are very condensed and take up only a small portion of the entire greyscale range (0 to 255). Upon closer inspection, we can find that the minimum and maximum values of the image are 159 and 187, occupying only 29 of the possible 256 values.

Listing 8: Question 3a

```
num_bins = 20;

%% Histogram
image_hist = figure('Name','Question 3a');
[n_elements, centers] = hist(image_of_doubles(:), num_bins);
bar(centers, n_elements);
title('Raeed Hassan');
xlim([0 255]);
exportgraphics(image_hist,'..\Figures\question3a.png');
```

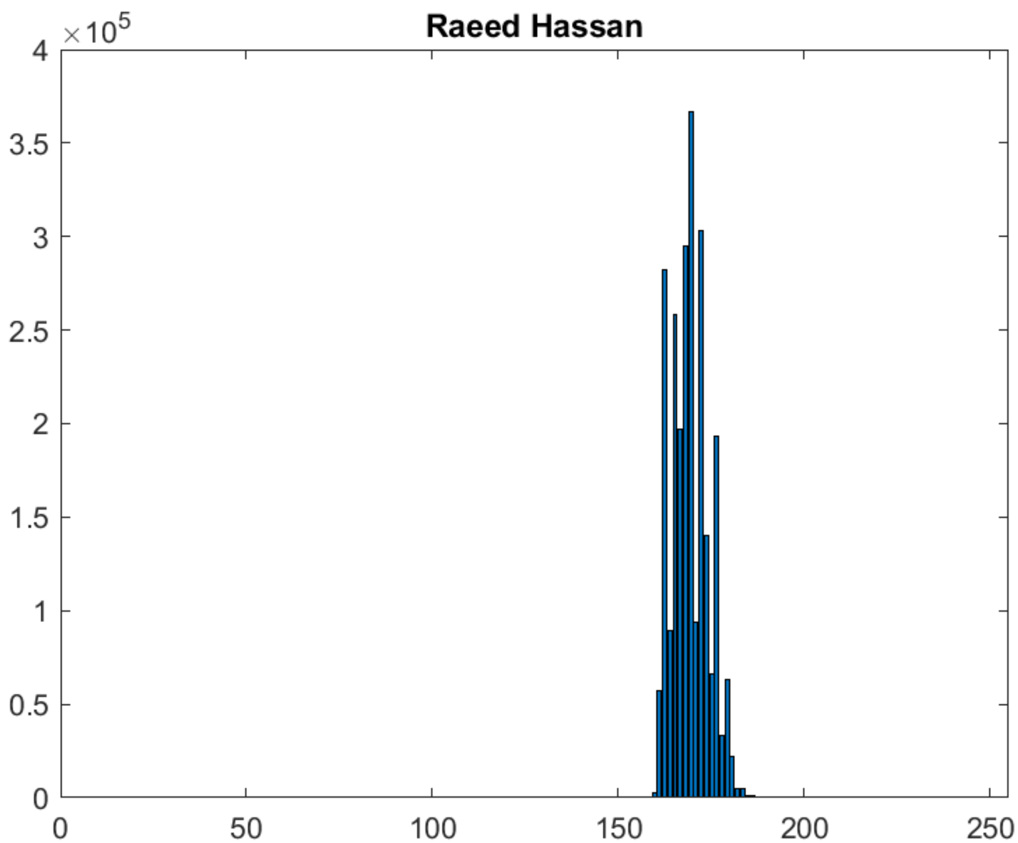

Figure 3: Question 3a

3b

The MATLAB code for displaying and printing the image is shown in Listing 9. The image exported by MATLAB is shown in Figure 4. The image has very poor contrast, likely due to the limited range of greyscale values that was determined in 3a. It is difficult to dintiguish parts of the image because their greyscale values are very close, resulting in the image have very little difference in colour or darkness.

Listing 9: Question 3b

```
%% Display original image
original_image = figure('Name','Question 3b');
imshow(uint8(image_of_doubles));
imwrite(uint8(image_of_doubles),'..\Figures\question3b.png');
```

Figure 4: Question 3b

3c

To improve the image quality, we simply increase the range of the greyscale values. The constants α and β can be chosen by determining the transformation that we want to apply to the original histogram. To improve the image quality, we need to increase the contrast of the image by increasing the range of greyscale values. To increase the range of the histogram, we need to multiply all the values by a constant, α, which can be determined by dividing our desired range (255) by the range of the original image (28). The adjusted values have to be translated so they start at 0, which is done by changing the value of β, which must be negative as the values are shifted down towards 0. β can be determined by multiplying the minimum greyscale value of the original image (159) and multiplying by α. Therefore $\alpha = {}^{255}/_{28}$ and $\beta = -159 \cdot ({}^{255}/_{28})$. The Matlab code to perform the transformation is shown in Listing 10.

Listing 10: Question 3c

```
%% Improved image
improved_image = image_of_doubles*(255/28) - 159*(255/28);
```

3d

The MATLAB code to plot and print the final histogram is shown in Listing 11. The final histogram is shown in Figure 5.

Listing 11: Question 3d

```
%% Improved histogram
image_hist = figure('Name','Question 3d');
[n_elements, centers] = hist(improved_image(:), num_bins);
bar(centers, n_elements);
title('Raeed Hassan');
xlim([0 255]);
exportgraphics(image_hist,'..\Figures\question3d.png');
```

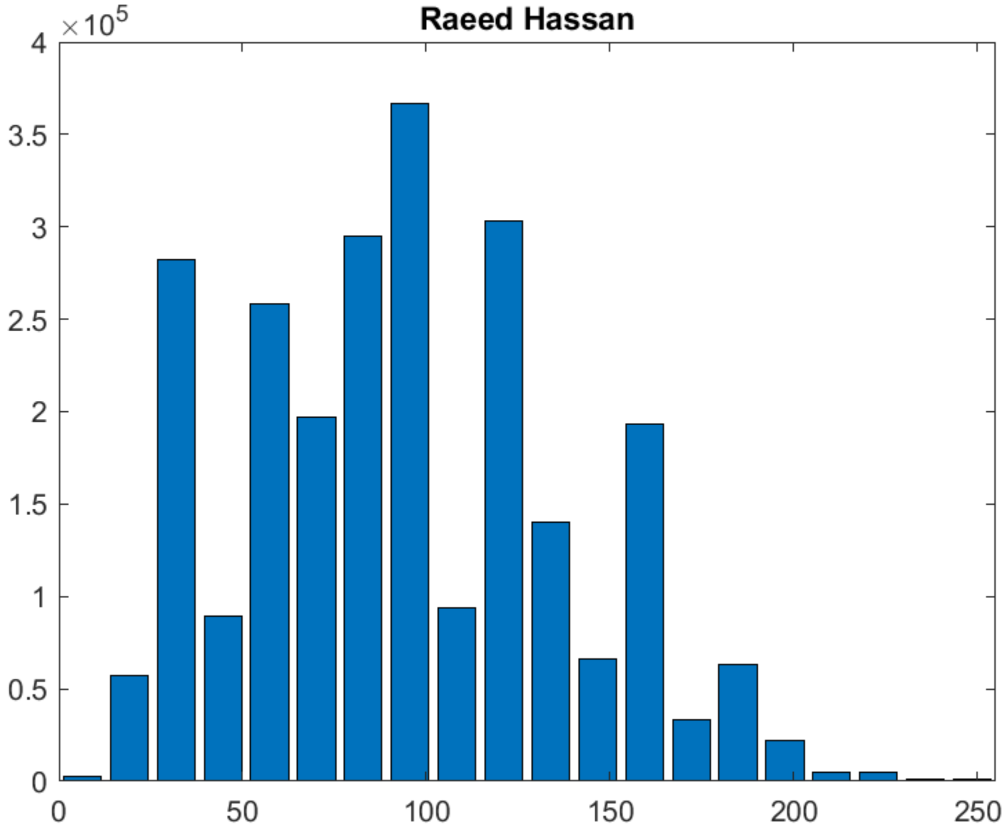

Figure 5: Question 3d

3e

The MATLAB code to display and print the final image is shown in Listing 12. The final image is shown in Figure 6. The improved image has greater contrast due to the larger range of greyscale values.

Listing 12: Question 3e

```
%% Display improved image
improved_image_fig = figure('Name','Question 3e');
imshow(uint8(improved_image));
imwrite(uint8(improved_image),'..\Figures\question3e.png');
```

Figure 6: Question 3e