



Faculty of Engineering, Architecture and Science

Department of Electrical and Computer Engineering


Course Number	ELE532
Course Title	Signals and Systems I
Semester/Year	Fall 2020

Instructor	Javad Alirezaie
------------	------------------------

ASSIGNMENT No.	1
-----------------------	----------

Assignment Title	Working with MATLAB, Visualization of Signals
------------------	--

Submission Date	September 27, 2020
Due Date	September 27, 2020

Student Name	Nathan Agnew
Student ID	500819685
Signature*	
Student Name	Ahmad Fahmy
Student ID	500913092
Signature*	AF

**By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: www.ryerson.ca/senate/current/pol60.pdf.*

Table of Contents

A. Inline functions and plotting continuous functions	3
B. Time shifting and time scaling	4
C. Visualizing operations on the independent variable and algorithm Vectorization	8
D. Array indexing	9

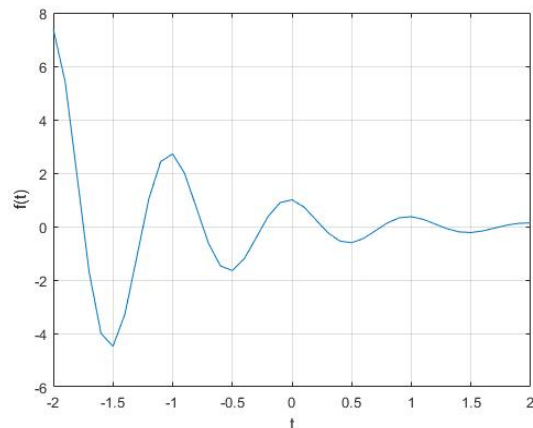
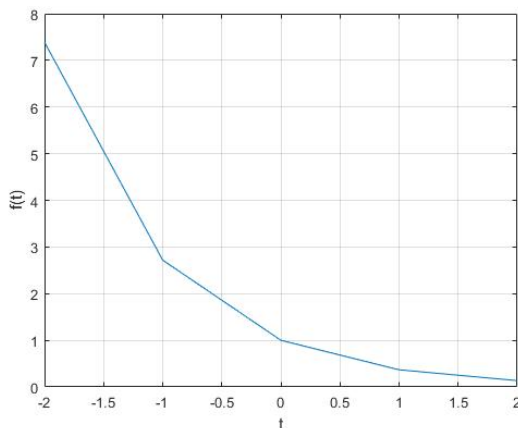
A. Inline functions and plotting continuous functions

Problem A.1 [0.5 Marks] Section 1.11-1 Anonymous Functions, *Lathi, 3rd Ed., page 126*. Generate and plot the graphs as shown in Figures 1.46 and 1.47 on page 127.

```
% Problem A.1a
% Figure 1.46
f = @(t) exp(-t).*cos(2*pi*t);
t=(-2:2);
figure('Name','Figure Ala','NumberTitle','off');
plot(t,f(t));
xlabel('t');ylabel('f(t)');grid;

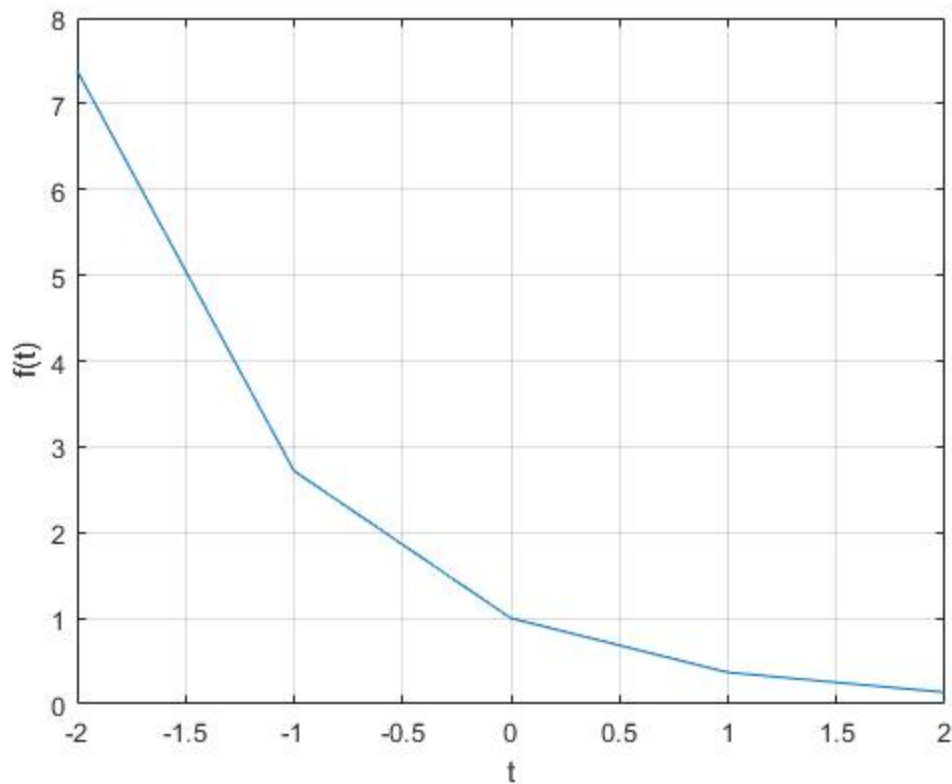
% Problem A.1b
% Figure 1.47
f = @(t) exp(-t).*cos(2*pi*t);
t=(-2:0.1:2);
figure('Name','Figure Alb','NumberTitle','off');
plot(t,f(t));
xlabel('t');ylabel('f(t)');grid;
```

Graphs for Figure 1.46 and 1.47 respectively



Problem A.2 [1 Mark] Plot the function e^{-t} for t taking on integer values contained in $-2 \leq t \leq 2$; you can generate these values using the MATLAB command `tt=[-2:2]`.

```
% Problem A.2
f = @(t) exp(-t);
tt=(-2:2);
figure('Name','Figure A2','NumberTitle','off');
plot(tt,f(tt));
xlabel('t');ylabel('f(t)');grid;
```



Problem A.3 [0.5 Marks] Compare the results of Problem A.2 with Figure 1.46 in Problem A.1.

The Results from the two graphs appear to be identical. This is because figure 1.46 uses values of t when \cos is 1, revealing the exponential function graphed on integers seen in problem A2.

B. Time shifting and time scaling

Problem B.1 [1 Mark] Section 1.11-2 *Lathi, 3rd Ed.*, page 128. Generate and plot $p(t)$ as shown in Figure 1.50 on page 129.

```
% Problem B1
t = (-1:0.01:2);
p = @(t) 1.0.*((t>=0)&(t<1));

figure('Name','Figure B1','NumberTitle','off');
plot(t,p(t));
xlabel('t');
ylabel('p(t) = u(t)-u(t-1)');
axis([-1 2 -.1 1.1]);
```

Problem B.2 [2 Marks] Use $p(t)$ defined in Problem B.1 to generate and plot functions $r(t) = tp(t)$ and $n(t) = r(t) + r(-t + 2)$.

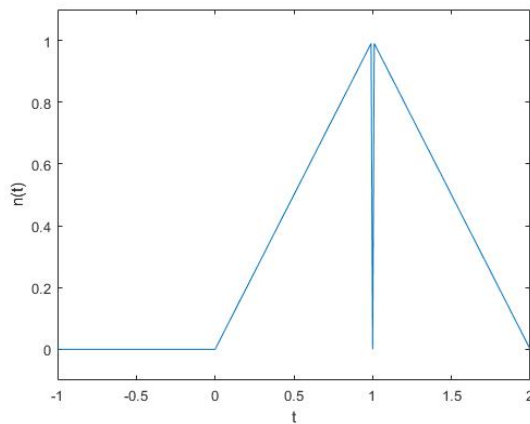
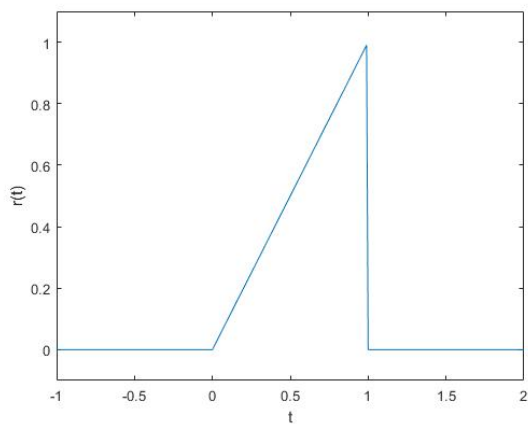
```
% Problem B2
r = @(t) t.*p(t);

figure('Name','Figure B2a','NumberTitle','off');
plot(t,r(t));
xlabel('t');
ylabel('r(t)');
axis([-1 2 -.1 1.1]);

n = @(t) r(t)+r(-t+2);

figure('Name','Figure B2b','NumberTitle','off');
plot(t,n(t));
xlabel('t');
ylabel('n(t)');
axis([-1 2 -.1 1.1]);
```

Figure B2a and B2b respectively



Problem B.3 [2 Marks] Plot the following two signals: $n_1(t) = n(\frac{1}{2}t)$, $n_2(t) = n_1(t + \frac{1}{2})$.

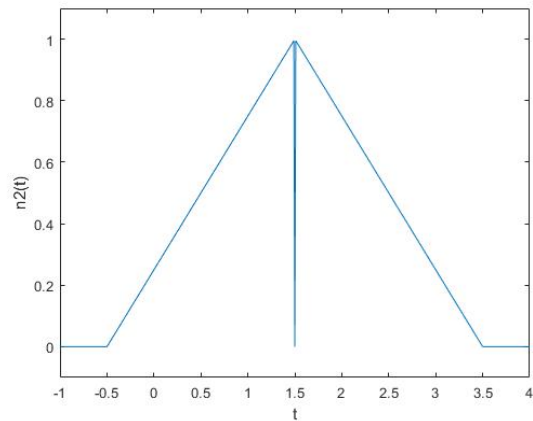
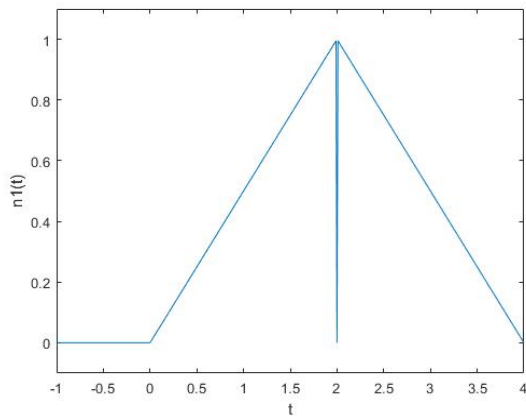
```
% Problem B3
t = (-4:0.01:4);
n1 = @(t) n(.5*t);

figure('Name','Figure B3a','NumberTitle','off');
plot(t,n1(t));
xlabel('t');
ylabel('n1(t)');
axis([-1 4 -.1 1.1]);

n2 = @(t) n1(t+.5);

figure('Name','Figure B3b','NumberTitle','off');
plot(t,n2(t));
xlabel('t');
ylabel('n2(t)');
axis([-1 4 -.1 1.1]);
```

Figure B3a and B3b Respectively



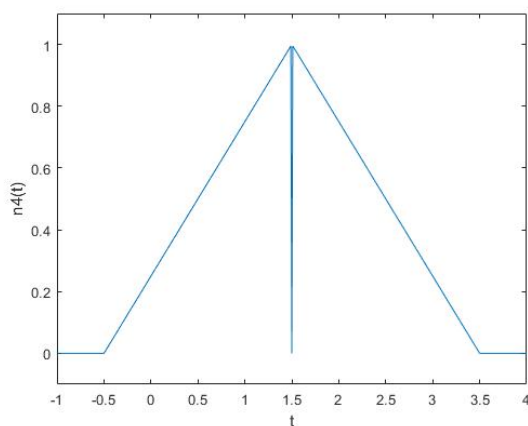
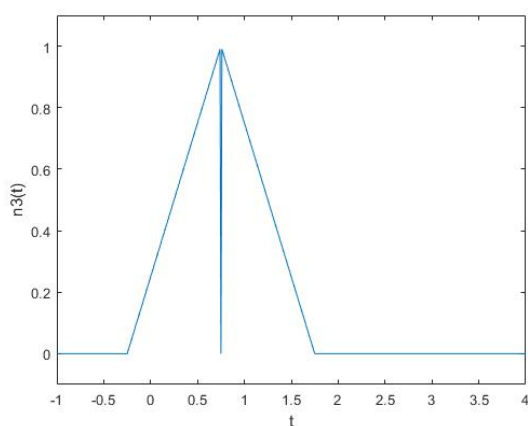
Problem B.4 [2 Marks] Plot the following two signals: $n_3(t) = n(t + \frac{1}{4})$, $n_4(t) = n_3(\frac{1}{2}t)$.

```
% Problem B4
n3 = @(t) n(t+.25);

figure('Name','Figure B4a','NumberTitle','off');
plot(t,n3(t));
xlabel('t');
ylabel('n3(t)');
axis([-1 4 -.1 1.1]);

n4 = @(t) n3(t*.5);

figure('Name','Figure B4b','NumberTitle','off');
plot(t,n4(t));
xlabel('t');
ylabel('n4(t)');
axis([-1 4 -.1 1.1]);
```



Problem B.5 [2 Marks] Compare $n_4(t)$ and $n_2(t)$; explain any observed differences and/or similarities.

The functions n_1 and n_2 produce the same graph, despite the different order and magnitude of operations performed.

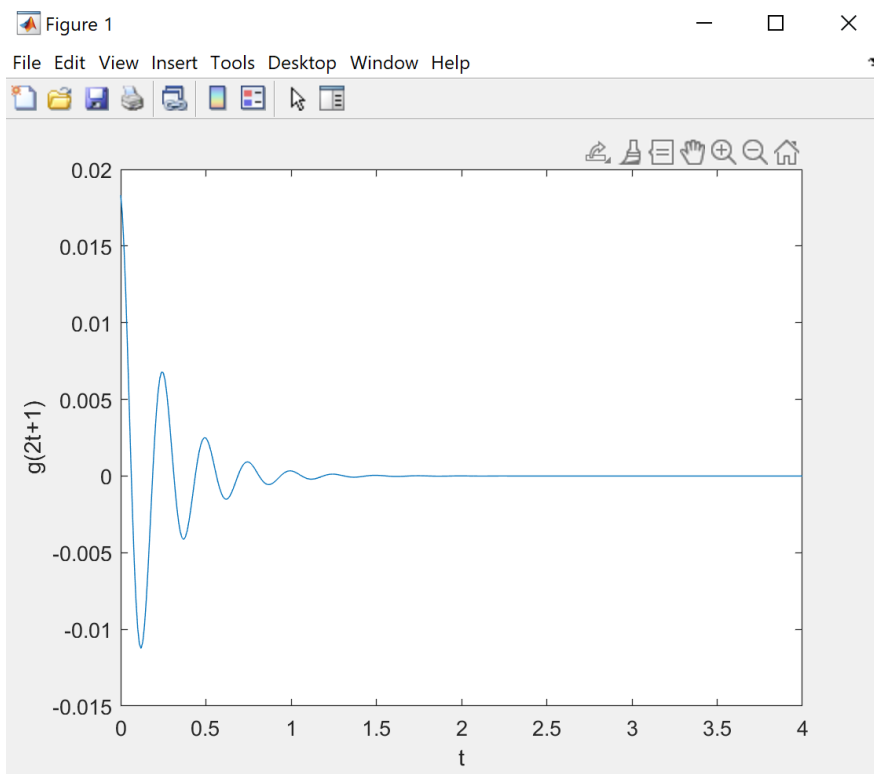
C. Visualizing operations on the independent variable and algorithm Vectorization

Problem C.1 [0.5 Marks] Section 1.11-3, *Lathi, 3rd Ed.*, page 130. Follow the steps, but instead, generate $g(t) = f(t)u(t)$ where $f(t) = e^{-2t} \cos 4\pi t$.

```
% Problem C.1
f = @(t) exp(-2*t).*cos(4*pi*t);
u = @(t) +(t>=0);
g = @(t) f(t).*u(t);
```

Problem C.2 [0.5 Marks] Using $g(t)$ as described in Problem C.1, generate and plot $s(t) = g(t + 1)$ for $t = [0 : 0.01 : 4]$.

```
% Problem C.2
s = @(t) g(t+1);
figure('Name','Figure C2','NumberTitle','off');
t = (-4:0.01:4);
plot(t,s(t));
xlabel('t');
ylabel('s(t)');
grid;
```



Problem C.3 [0.5 Marks] Plot $s_\alpha(t) = e^{-2}e^{-\alpha t} \cos(4\pi t)u(t)$ for $\alpha \in \{1, 3, 5, 7\}$ in one figure for $t = [0 : 0.01 : 4]$. For this plot you can use the **for** command for a loop structure (to learn more about this command type **help for** at the MATLAB prompt). Also try to use matrix and vector operations to generate and plot the desired functions by following the steps of Section B.7-6, *Lathi, 3rd Ed.*, page 49.

```
% Problem C.3
sa = @(t) exp(-2*t).*exp(-1*a*t).*cos(4*pi*t).*u(t);

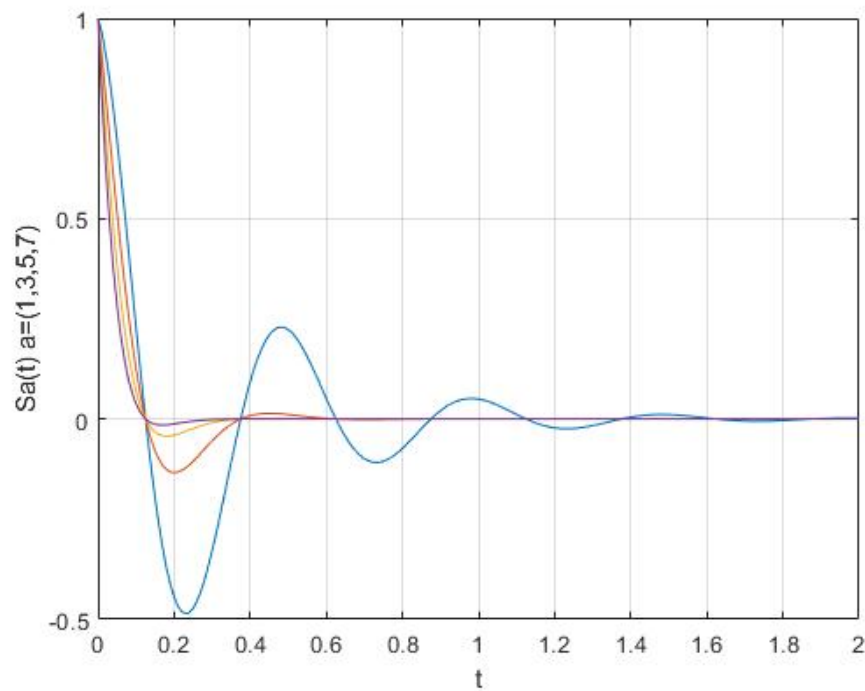
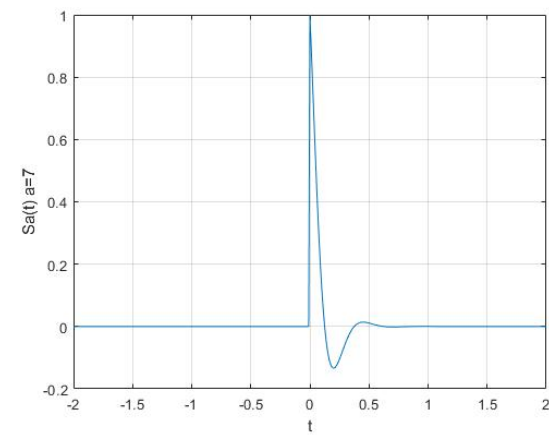
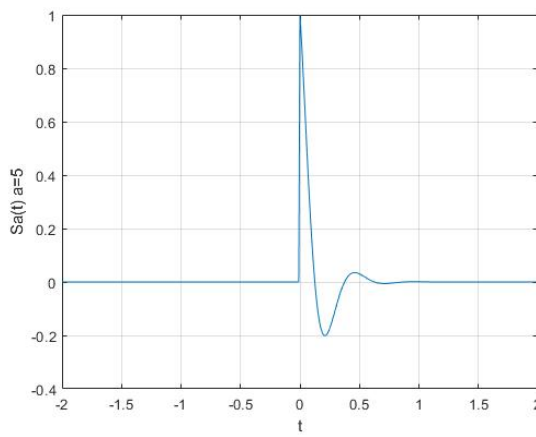
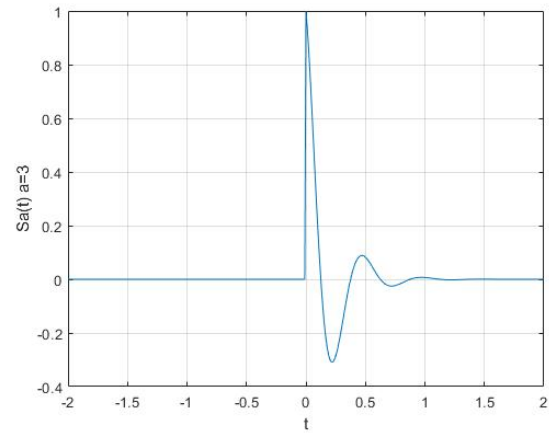
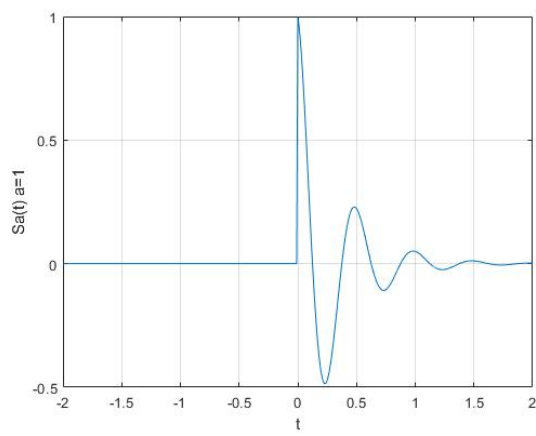
%for loop structure
for a = 1:2:7
    figure('Name',strcat('Figure C2 a= ',num2str(a)), 'NumberTitle','off');
    sa = @(t) exp(-2*t).*exp(-1*a*t).*cos(4*pi*t).*u(t);
    plot(t,sa(t));
    xlabel('t');
    ylabel(strcat('Sa(t) a= ',num2str(a)));
    grid;
end
```

This code is a loop structure and plots each graph individually.

```
%matrix strucutre
a = (1:2:7);
t = (0:0.01:2)';
T = t*ones(1,4);
sa2 = exp(-2*T*diag(a)).*cos(4*pi*T).*u(T).*exp(-1*T*diag(a));
figure('Name','Figure C3 Matrix','NumberTitle','off');
plot(t,sa2);
xlabel('t');
ylabel('Sa(t) a=(1,3,5,7)');
grid;
```

This code creates a matrix with multiple plots, then all the plots are put on one graph.

See Graphs on next page



Problem C.4 [0.5 Marks] Determine the size of the matrix $s(t)$ generated in Problem C.3.

```
disp('Matrix Size, rows, columns, repectively');
disp(size(sa2));
```

```
Matrix Size, rows, columns, repectively
    201         4
```

We used the size function to determine how big the matrix was.

D. Array indexing

Problem D.1 [0.5 Marks] Let A be a 5×4 matrix array with real-valued elements:

$$\mathbf{A} = \begin{bmatrix} 0.5377 & -1.3077 & -1.3499 & -0.2050 \\ 1.8339 & -0.4336 & 3.0349 & -0.1241 \\ -2.2588 & 0.3426 & 0.7254 & 1.4897 \\ 0.8622 & 3.5784 & -0.0631 & 1.4090 \\ 0.3188 & 2.7694 & 0.7147 & 1.4172 \end{bmatrix} \quad (1)$$

For the matrix A in Equation (??) implement the following operations:

- (a) $A(:)$
- (b) $A([2 \ 4 \ 7])$
- (c) $[A \geq 0.2]$
- (d) $A([A \geq 0.2])$
- (e) $A([A \geq 0.2]) = 0$

Describe the outcome of each operation stated in parts (a)–(e).

```
%Problem D.1
```

```
%a)
```

```
A(:);
```

```
%this operation specifies all the elements in the specified array
```

```
%b
```

```
B = A([2 4 7]);
```

```
%this operation loaded cells 2,4, and 7 for placement in another Matrix
```

Matrix B:

	1	2	3
1	1.8339	0.8622	-0.4336

```
%c)
```

```
C=[A >= 0.2];
```

```
%this operation checks to see if the current cell value is greater
%than or equal to 0.2 and gives 0 or 1 for false and true,
%respectively
```

Matrix C:

	1	2	3	4
1	1	0	0	0
2	1	0	1	0
3	0	1	1	1
4	1	1	0	1
5	1	1	1	1

```
%d)
```

```
D=A([A >= 0.2]);
```

```
%this operation checks to see if the current cell value is greater
%than or equal to 0.2 and gives 0 or 1 for false and true, then
%removes only the cells that gave boolean 1 for placement into
%another matrix
```

	1
1	0.5377
2	1.8339
3	0.8622
4	0.3188
5	0.3426
6	3.5784
7	2.7694
8	3.0349
9	0.7254
10	0.7147
11	1.4897
12	1.4090
13	1.4172

```
%e)
```

```
A([A >= 0.2]) = 0;
```

```
%this operation checks to see if the current cell value is greater
%than or equal to 0.2 and gives 0 or 1 for false and true, then
%replaces all the cells that gave boolean 1 with 0's
```

	1	2	3	4
1	0	-1.3077	-1.3499	-0.2050
2	0	-0.4336	0	-0.1241
3	-2.2588	0	0	0
4	0	0	-0.0631	0
5	0	0	0	0

Problem D.2 [1 Mark] Let \mathbf{B} be a 1024×100 data matrix representing 100 blocks of non-overlapping 1024-element input samples from a particular data source.

- (a) Write a simple MATLAB program using two nested `for` loops that will set all elements of the data matrix \mathbf{B} with magnitude values below 0.01 to zero:

$$\mathbf{B}(i, j) = 0, \quad \text{if } |\mathbf{B}(i, j)| < 0.01, \quad (2)$$

where $\mathbf{B}(i, j)$ is element of the data matrix \mathbf{B} in i -th row and j -th column.

- (b) Repeat part (a) using MATLAB's indexing features as described in Problem D.1.
 (c) Use the MATLAB commands `tic` and `toc` to compare the execution time of the code you wrote in parts (a) and (b).

The written code generated an array of size 1024x100 and populated it with values between 0 and 1, the matrix was then scanned for value that were less than 0.01 and changed them to 0.

```
%Problem D.2
%rng('default')
%B = rand(1024,102);
```

```
%a)
tic
for i = 1:1024
    for j = 1: 100
        if (B(i, j) < 0.01)
            B(i, j) = 0;
        end
    end
end
toc
```

1024x100 double

	1	2	3	4	5	6	7	8	9	10	11	12	
1	-1.7467	0	-1.6048	-0.1358	0	-1.2890	-0.6321	-0.2275	-0.1007	0	0	-0.7146	^
2	0	-0.3761	0	0	-0.1487	-0.8223	0	-0.2656	-0.0165	0	0	0	
3	-1.1262	-0.2719	0	0	-1.5093	-0.6804	-0.0962	0	-0.5904	-1.3184	0	-0.2458	
4	-0.4282	-1.5331	-0.6670	0	-0.1237	-0.7232	-0.3604	0	-0.6356	0	-1.0312	0	
5	0	-1.4043	-0.9066	-0.5863	0	-2.7286	-0.4429	-1.3997	0	-0.5095	-0.0825	0	
6	0	0	0	-1.1993	-0.1739	-0.5826	-0.0947	-0.9430	-0.5423	0	0	0	
7	-0.4766	-0.4049	-0.3453	0	-1.4501	-0.2731	0	-0.8395	-1.3256	0	0	-0.2908	
8	-0.0590	-0.4552	0	-0.0765	0	-0.7725	0	0	-0.0454	0	-0.4139	-0.1349	
9	-0.8150	-1.0034	0	-0.9107	-0.6292	0	0	-1.3904	0	-0.3749	0	-0.2047	
10	-1.6369	-0.9882	0	0	-0.6013	0	0	-0.1478	-0.1532	0	-1.6676	-1.0858	
11	-0.7564	-0.5980	0	-0.3317	-0.3003	-0.2802	-0.6340	0	0	0	0	-0.0192	
12	0	0	-1.4102	-0.7936	0	-1.0915	0	-1.4981	-1.2279	-0.0327	-0.3994	0	...
13	0	-1.5250	-0.3256	0	0	0	-0.7998	-1.6360	0	-0.2505	-0.9485	0	
14	0	-0.1969	-1.2854	-0.9057	0	-0.5370	-0.2627	0	0	0	0	0	
15	-0.7578	0	0	-0.5270	0	-0.3978	0	-1.4079	0	0	-0.0157	-0.6433	
16	0	0	0	-1.0635	0	0	-0.9151	-0.0741	-0.8850	-1.1656	-2.6054	-0.3349	
17	-0.4376	-0.2514	-0.9090	0	0	0	-1.4481	0	-0.7303	-0.2530	-1.1540	0	
18	0	0	-0.2893	-0.5545	0	0	0	0	-0.4121	0	-0.1772	-0.1383	
19	-0.8523	-0.4755	-0.7948	0	0	-0.7018	-0.5353	0	0	0	0	-1.2780	
20	-1.9693	0	-0.5649	-1.0277	0	-0.2187	-0.0150	-0.8654	-1.5992	-0.1534	-0.8492	-0.3462	
21	-0.2074	0	-0.5162	-0.2977	-0.4040	-1.9619	-0.5548	-1.1630	-0.5236	0	-1.0310	0	
22	-0.2825	-2.2428	0	0	0	0	-0.2827	0	-1.3140	0	-0.4243	-0.3157	
23	-1.0954	0	0	-1.1797	0	0	0	0	0	-1.4213	-0.5592	-1.9574	
24	0	0	-1.1436	-0.2385	0	-1.7881	-1.0485	-0.0320	0	0	-2.0909	0	
25	0	-0.1100	0	-1.2124	0	0	2.8606e-04	-1.0585	-0.3696	-0.2905	-0.3226	-0.4468	...

```
%b)
load ELE532_Lab1_Data
tic
    B([B > 0.01]) = 0;
toc
```

c) Following are the elapsed times for parts a) (the nested for loop) and b) (matlab function) respectively.

Elapsed time is 0.007949 seconds.

Elapsed time is 0.002836 seconds.

Problem D.3 [1 Mark] Let **x_audio** be a 20,000 sample-long row vector representing 2.5 sec of an audio signal sampled at 8 kHz. A simple data compression algorithm can be implemented by setting all elements of the data array **x_audio** with magnitude values below a threshold to zero. ***Note:** The actual compression algorithm will code the zero-valued samples more efficiently thus achieving a certain degree of compression. In this exercise, however, we only want to investigate the compressibility of the data array **x_audio** as measured by the number of samples with magnitude values below the threshold and the resulting sound quality as a function of the threshold.*

Write such a data compression algorithm and listen to the processed audio file. You may want to consider the following points:

- Do not work directly on the data array **x_audio**; otherwise after each process you will need to reload the data file. Instead, copy the data array **x_audio** to another working array and process that array.
- Devise a simple method of counting the number of elements of data array that are set to zero.
- You can listen to an audio array by using the MATLAB command **sound**. For example, if you want to listen to the original, unprocessed data array **x_audio**, issue the command **sound(x_audio, 8000)** at the MATLAB prompt.

The quote was “Members of the public service Alliance of Canada” and if you remove a threshold that was very large the audio becomes very unclear, whereas if you remove a reasonable threshold the audio will sound the same but will have less values and therefore be more compressible.

```
%Problem D.3
copy_x_audio = x_audio;
counter = 0;

%Self-given threshold of 0.01
for i = 1:20000
    if (copy_x_audio(i) < 0.01 && copy_x_audio(i) > -0.01)
        copy_x_audio(i) = 0;
        counter = counter + 1;
    end
end
counter 3399
```