Table of Contents


```
% Connor Dupuis
% Section: 28944
% TA: Noaki Sawahashi
```

QUESTION 1 COMMENTING

```
% DO NOT REMOVE THE LINE BELOW
% MAKE SURE 'eel3135_lab03_comment.m' IS IN SAME DIRECTORY AS THIS
 FILE
clear; close all;
type('eel3135_lab04_comment.m')
```

```
% Convolution can be seen as the mathematical expression of how one
 signal
% can modify the other, you will need to comment the answers to the
% questions on the following lines of code to demonstrate the basics
 of
% convolution. Also answer the question at the end in order to
 understand
% the effects that convolution may have on the original signal.
```

```
%% ONE-DIMENSIONAL CONVOLUTION

% INPUT SIGNAL
xx = [1 1 -1 -1 1 1 -1 -1 1 1 1 1 -1 -1 -1 -1];
% <-- Answer: What is the length of xx?
% xx has a length of 16
```

```
% FILTER COEFFICIENTS / IMPULSE RESPONSE
bk = [1/4 1/4 1/4 1/4];
% <-- Answer: What is the length of bk?
% bk has a length of 4

% FILTER OUTPUT
yy = conv(bk, xx);
% <-- Answer: What is the length of yy?
% yy has a length of 19

% PERFORM FILTERING IN ALTERNATIVE WAY
zz = 1/4*shift(xx, 0) + 1/4*shift(xx, 1) + 1/4*shift(xx, 2) +
 1/4*shift(xx, 3);
% <-- Answer: What is the length of zz?
% zz has a length of 16, but as a column vector

% --> ANSWER BELOW: Explain why yy and zz are different lengths. <--
% yy uses convolution with xx and bk, while zz uses shifts which
 doesn't
% casue additional inputs
%


% PLOT
figure(1)
subplot(411)
stem(xx); axis([0 20 -1 1])
ylabel('Amplitude')
subplot(412)
stem(bk); axis([0 20 -1 1])
ylabel('Amplitude')
subplot(413)
stem(yy); axis([0 20 -1 1])
ylabel('Amplitude')
subplot(414)
stem(zz); axis([0 20 -1 1])
ylabel('Amplitude')
xlabel('Samples')


%% TWO-DIMENSIONAL CONVOLUTION

% INPUT SIGNAL
x2 = [ones(15) -1*ones(15)]*255;

% FILTER COEFFICIENTS
b2 = (1/8)*[0 1 1  1  1  0; 0 1 1  1  1  0];
b3 =       [1 1 1 -1 -1 -1; 1 1 1 -1 -1 -1];

% OUTPUT
y2 = conv2(x2,b2);
y3 = conv2(x2,b3);
```

```
% --> ANSWER BELOW: Why does filter b2 affect the image as it does?
%     What are possible applications of filter b2? <--
% Filter b2 blurs the imnage slightly causing the borders to blend.
 This
% happens due to the fractional positive value of the kernel. This
 filter
% could be used as a blur in practice.
%

% --> ANSWER BELOW: Why does filter b3 affect the image as it does?
%     What are possible applications of filter b3? <--
% Filter b3 affects the image due to the negative values causing
 inversion
% in some of the areas. This filter could be used as an edge detection
 in
% practice.
%


% PLOT THE FIRST FILTER RESULTS
figure(2)
subplot(231)
image(x2)
xlabel('x'); ylabel('y'); zlabel('z');
title('x2')
axis equal; axis tight; colormap('gray');
subplot(232)
image(b2)
xlabel('x'); ylabel('y'); zlabel('z');
title('b2')
axis equal; axis tight; colormap('gray');
subplot(233)
image(y2)
xlabel('x'); ylabel('y'); zlabel('z');
title('y2')
axis equal; axis tight; colormap('gray');

% PLOT THE SECOND FILTER RESULTS
subplot(234)
image(x2)
xlabel('x'); ylabel('y'); zlabel('z');
title('x2')
axis equal; axis tight; colormap('gray');
subplot(235)
image(b3)
xlabel('x'); ylabel('y'); zlabel('z');
title('b3')
axis equal; axis tight; colormap('gray');
subplot(236)
image(y3)
xlabel('x'); ylabel('y'); zlabel('z');
title('y3')
axis equal; axis tight; colormap('gray');
```

```
function xs = shift(x, s)
%SHIFT   ===> Shifts each elements in the input by s <===

    % ====> Initializes xs <====
    xs = zeros(length(x), 1);

    for n = 1:length(x)
        % ====> Sets boundry conditions <====
        if n-s > 0 && n-s < length(x)
            % ====> Assigns values to xs(n) <====
            xs(n) = x(n-s);
        end
    end

end
```

QUESTION 2 AUDIO (DO NOT CHANGE)

```
%MAKE SURE 'SaveYourTears.wav' is in the same directory!
[x, fs] = audioread('SaveYourTears.wav');
soundsc(x, fs);
```

2(b) APPLY ECHO

```
xr = echo(x,10000,0.9);
soundsc(xr, fs);
```

2(c) ANSWER QUESTION

The delay is roughly 0.2268 seconds (10000/44100).

2(e) APPLY TREMOLO

```
xt = tremolo(x,20/fs,0.5);
soundsc(xt, fs);
```

2(f) ANSWER QUESTION

The tremolo effect varies the volume of the song through a cos wave. As the value approaches and leaves -1, the song becomes quiter and then louder again.

2(g) APPLY TREMOLO AND ANSWER QUESTION

```
xtt = tremolo(x,1/length(x),1);
soundsc(xtt, fs);
% The sound gest quiet because at halfway through n*m evaluates to 1/2
 which
```

```
% causes the cos to evaluate to -1. cos(2*pi*(1/440998)*220499) = -1.
 As
% the value approaches and leaves -1, the song becomes quiter and then
% louder again.
```

2(h) ASSESS FOR TIME-VARYING SYSTEM

Shifitng outputs from parts b and g

```
b = shift(xr,220499);
g = shift(xtt,220499);

soundsc(b, fs);

soundsc(g, fs);

%Shifting original input and then applying the effects
xs = shift(x,220499);
b = echo(xs,10000,0.9);
g = tremolo(xs,1/length(x),1);

soundsc(b, fs);

soundsc(g, fs);
```

2(i) ANSWER QUESTION

The system with the tremolo effect is time varying. In part g, the song starts off loud and gets quiet, while in part i, the song starts quiet and gets louder.

QUESTION 3 Image Set-up (DO NOT CHANGE!)

```
%MAKE SURE 'flower.pgm' is in the same directory!
img = imread('flower.pgm');
img = double(img);          % Convert image from integers to doubles
```

3(a) APPLY FILTER 1

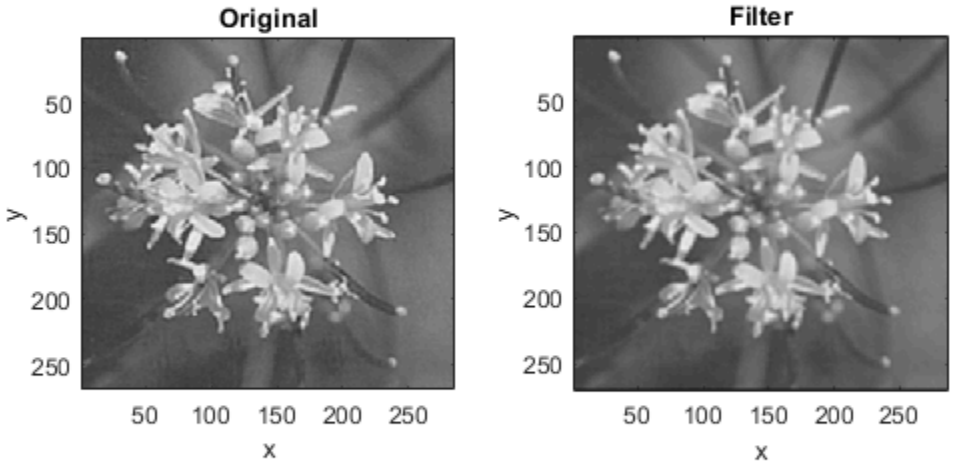
```
% Filter Coefficients
b1 = (1/9)*[1 1 1; 1 1 1; 1 1 1];

% Output
y1 = conv2(img,b1);

figure(1)

subplot(121)
image(img)
xlabel('x'); ylabel('y'); zlabel('z');
title('Original')
axis equal; axis tight; colormap('gray');
subplot(122)
image(y1)
```

```
xlabel('x'); ylabel('y'); zlabel('z');
title('Filter')
axis equal; axis tight; colormap('gray');
```



3(b) ANSWER QUESTION

This filter blurs the image. This can be seen by the coeficients being a fractional value.

3(c) APPLY FILTER 2

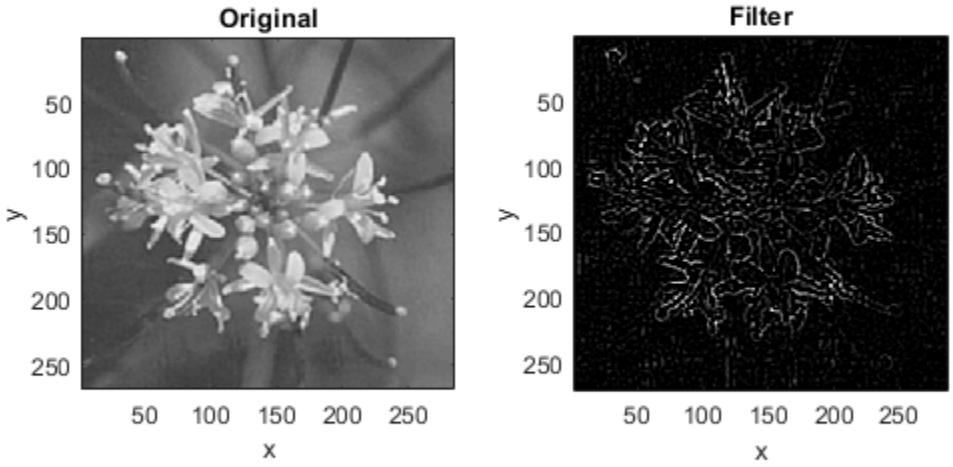
```
% Filter Coefficients
b2 = [1 1 1; 1 -8 1; 1 1 1];

% Output
y2 = conv2(img,b2);

figure(1)

subplot(121)
image(img)
xlabel('x'); ylabel('y'); zlabel('z');
title('Original')
axis equal; axis tight; colormap('gray');
subplot(122)
image(y2)
```

```
xlabel('x'); ylabel('y'); zlabel('z');
title('Filter')
axis equal; axis tight; colormap('gray');
```



3(d) ANSWER QUESTION

This filter extracts the edges of the image. This can be seen by the coeficients present. The w[0,0] element inverts and scales the signal while the others around lighten.

3(e) APPLY UNSHARP MASKING

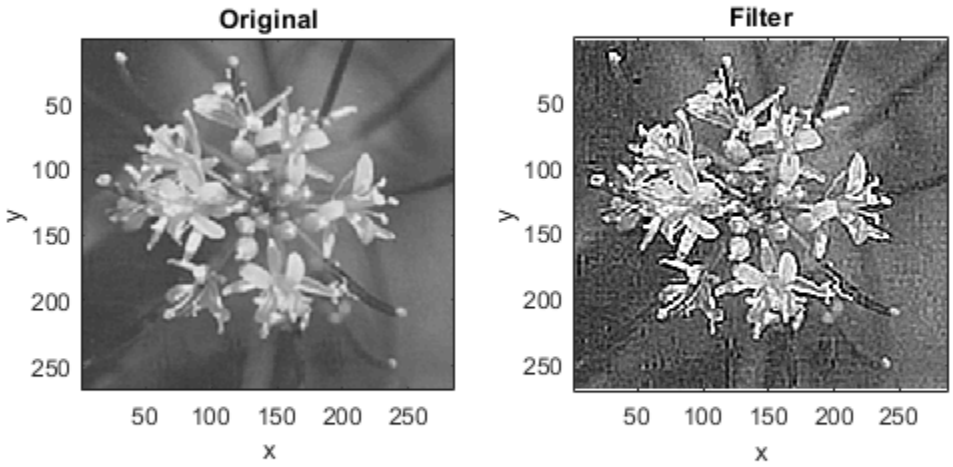
```
% Filter Coefficients
b3 = [0 0 0; 0 1 0; 0 0 0];

% Output
y3 = conv2(img,b3);

y3 = y3 - y2;
figure(1)

subplot(121)
image(img)
xlabel('x'); ylabel('y'); zlabel('z');
title('Original')
axis equal; axis tight; colormap('gray');
```

```
subplot(122)
image(y3)
xlabel('x'); ylabel('y'); zlabel('z');
title('Filter')
axis equal; axis tight; colormap('gray');
```



3(f) ANSWER QUESTION

This filter sharpens the image. This filter adds an almost grainy effect to the image but also "enhances" the details. This enhancement comes at the cost of noise.

ALL FUNCTIONS SUPPORTING THIS CODE % %

```
function y = echo(x, s, A)
%ECHO   ===> Repeats the samples starting at sample s in the future,
 casuing an echo effect.<===
    xs = shift(x, s);
    y = x + A*xs;
end

function y = tremolo(x, m, A)
%TREMOLO   ===> Adjusts the volume of a sample with a cosine wave.
 <===
```

```
    y = zeros(size(x));
    for n = 1:length(x)
        y(n) = x(n) + A*cos(2*pi*m*n)*x(n);
    end


end

function xs = shift(x, s)
%SHIFT   ===> Shifts each elements in the input by s <===

    % ====> Initializes xs <====
    xs = zeros(length(x), 1);

    for n = 1:length(x)
        % ====> Sets boundry conditions <====
        if n-s > 0 && n-s < length(x)
            % ====> Assigns values to xs(n) <====
            xs(n) = x(n-s);
        end
    end

end
```

Published with MATLAB® R2020a