EBU6018 Advanced Transform Methods
EBU5303 Multimedia Fundamentals

Lab 2 – 16 November 2022

Introduction

This lab is common to EBU6018 and EBU5303.

In Part 1 of the lab you will:

- Investigate the Wavelet transform and its filter bank implementation for discrete-time signals
- Apply the Haar Discrete Wavelet Transform to compress signals.

Marks will go towards the assessment of EBU6018.

In Part 2 of the lab you will implement a simple JPEG encoder. Marks will go towards the assessment of EBU5303.

The MATLAB programming outcomes from the lab (i.e. MATLAB .m function files and plot figures) are to be handed in as a "folder" of results, showing that you have completed the steps of the lab successfully. A box in the right-hand margin indicates where such an outcome is expected – like this:

☐

Tick the box to indicate that the outcome has been created and placed in the result folder.

You must also answer some questions directly within this document, which must be saved and submitted with other outcomes in your folder of results.

Part 1 (EBU6018)

1 The Wavelet Transform: Introduction

In EBU6018 lectures, we introduced the wavelet transform, defined in terms of the wavelet's scaling function wavelet function. In those lectures we used Haar Functions as the wavelet function to perform a Discrete Wavelet Transform (DWT).

We also saw that for a discrete-time signal, the Discrete Wavelet Transform (DWT) can be implemented as a sequence of *filtering* followed by *downsampling*. Using this filterbank we do not need to calculate the mother wavelet $\psi(t)$, we only need the coefficients of the low-pass and high-pass filters $H_0(\omega)$ and $H_1(\omega)$.

Various files are provided for this lab.

2 DWT of an input vector s[n].

2.1 Discrete Wavelet Transform Filter Bank

Download the file **`dwt_haar.m`**. This implements a Haar Discrete Wavelet Transform using filtering (convolution) and subsampling.

Explain how this Matlab function works *in the text box provided*, and check that it implements a normalised Haar DWT using the recursion formulas given in the lecture notes and used in tutorial examples. ✓

Test this function by taking the Haar DWT of the following signal, to 2 steps:

```
s = [3, 7, -2, -4, 2, 6, 1, -1]
```

Check the following (*and record the values in the text box and save the plots obtained*):
(a) The "length" (Matlab: **`norm`**) of the DWT of **`s`**, and check it is the same as for **`s`** itself.
(b) The first half of the DWT of s contains (scaled) sums of pairs of elements of s, while the second half contains scaled differences of pairs of elements of s.
(c) The one-step DWT of **`s`** is the same as multiplying **`s`** (actually the column vector **`s'`**) by the following matrix:

$$W_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

✓

Hints: Make sure you always give row vectors to this simple **`dwt_haar()`** function.
We are using dyadic scaling so the number of elements in the input vector is a power of 2.

Your answer.

Explanation:

- Line 1, 38:

  These two blocks define the entry and exit of the function.

- Line 6 - 11:

```
N = length(c)-1;  % Max index for filter: 0 .. N
% If no steps to do, or the sequence is a single sample, the DWT is itself
if (0==N | steps == 0)
   dwtc = c;
   return
end
```

  This block determines the termination condition of the recursion. If the value of steps is 0, it means that the recursion of dwt has ended; specifically, if the length of the input signal is 1, then it has also ended. This judgment is needed in order to terminate at the completion of dwt.

- Line 13 - 17:

```
% Check that N+1 is divisible by 2
if (mod(N+1,2)~=0)
   disp(['Not divisible 2: ' num2str(N+1)]);
   return
end
```

  This block checks if the length of the input sequence matches the length that can be handled by this function.

- Line 19 - 25:

```
% Set the Haar analysis filter
h0 = [1/2  1/2];     % Haar Low-pass filter
h1 = [-1/2 1/2];     % Haar High-pass filter

% Filter the signal
lowpass_c = conv(h0, c);
hipass_c  = conv(h1, c);
```

  This block defines the coefficients of the two filters and applies (convolution) the filter to the original signal c to obtain the low-pass and high-pass signals.

- Line 27 – 29:

```
% Subsample by factor of 2 and scale
c1 = sqrt(2)*lowpass_c(2:2:end);
d1 = sqrt(2)*hipass_c(2:2:end);
```

This block down samples the signal to half of its original size and applies the coefficient root number 2.

- Line 31 – 35:

```
% Recursively call dwt_haar on the low-pass part, with 1 fewer steps
dwtc1 = dwt_haar(c1, steps-1);

% Construct the DWT from c1 and d1
dwtc = [dwtc1 d1];
```

This recursive block calls the next recursive transformation on c1 and subtracts 1 from the remaining steps value.

The final return value is the merge of the recursive function's return value dwtc1 with d1.

It does implements a normalised Haar DWT using the recursion formulas given in the lecture notes and used in tutorial examples.

Test & Check:

```
>> p121
Length of s:
   10.9545

Length of dwts:
   10.9545
```

a) `Length of s and dwts are equal`

The length of DWT of s is 10.9545, which is equal to itself.

b) s = [3, 7, -2, -4, 2, 6, 1, -1]

steps = 2; c1 = [7.0711, -4.2426, 5.6569, 0]; d1 = [-2.8284, 1.4142, -2.8284, 1.4142]

c1[1] = 0.5* $\sqrt{2}$*(s[1]+s[2]) = 7.0711; c1[2] = 0.5* $\sqrt{2}$*(s[3]+s[4]) = -4.2426 etc.

d1[1] = 0.5* $\sqrt{2}$*(s[1]-s[2]) = -2.828 ; d1[2] = 0.5* $\sqrt{2}$*(s[3]-s[4]) = 1.414 etc.

c1 = [7.0711, -4.2426, 5.6569, 0]

c2[1] = 0.5* $\sqrt{2}$*(c1[1]+c1[2]) = 2; c2[2] = 0.5* $\sqrt{2}$*(c1[3]+ c1[4]) = 4

d2[1] = 0.5* $\sqrt{2}$*(c1[1]-c1[2]) = 8 ; d2[2] = 0.5* $\sqrt{2}$*(c1[3]-c1[4]) = 4

dwtc1 = [c2 d2] = [2, 4, 8, 4]

dwt = [dwtc1 d1] = [2, 4, 8, 4, -2.8284, 1.4142, -2.8284, 1.4142]

The above calculation proves the description of Question (b).

c)

```
dwt_haar func:
    7.0711   -4.2426    5.6569         0   -2.8284    1.4142   -2.8284    1.4

matrix:
    7.0711   -4.2426    5.6569         0   -2.8284    1.4142   -2.8284    1.4
```

It can be considered that multiplying this matrix is equivalent to one step DWT.

2.2 Inverse DWT Filter Bank

Download the file **`idwt_haar.m`** , which implements the inverse DWT.

Apply this to the sequencies you transformed in the previous section. *Explain what happens in the text box provided.*
Explain how this Matlab function works, and check that it implements a recursion formula similar to the synthesis equation given in the lecture slide on the signal recovery filterbank. *Put your explanations in the text box provided and save your plots to confirm that the original sequence is obtained by applying the function idwt_haar.m.*

✓

Your answer.

Test code: p122.m

```
s = [3, 7, -2, -4, 2, 6, 1, -1];
dwts = dwt_haar(s, 2);
idwts = idwt_haar(dwts, 2);
disp(s);
disp(idwts);
```

Results:

```
    p122
      3     7    -2    -4     2     6     1    -1

    3.0000    7.0000   -2.0000   -4.0000    2.0000    6.0000    1.0000   -1.00
```

The original sequence is obtained after the inverse conversion.

Explanation:

- Line 1, 57:

  These two blocks define the entry and exit of the function.

- Line 6 – 11:

```
N = length(c)-1;  % Max index for filter: 0 .. N
% If no steps to do, or the sequence is a single sample, the DWT is itself
if (0==N | steps == 0)
   idwtc = c;
   return
end
```

This block determines the termination condition of the recursion. If the value of steps is 0, it means that the recursion of idwt has ended; specifically, if the length of the input signal is 1, then it has also ended. This judgment is needed in order to terminate at the completion of idwt.

- Line 13 - 17:

```
% Check that N+1 is divisible by 2
if (mod(N+1,2)~=0)
   disp(['Not divisible 2: ' num2str(N+1)]);
   return
end
```

This block checks if the length of the input sequence matches the length that can be handled by this function.

- Line 19 - 30:

```
% Set the Haar analysis filters
h0 = [1/2  1/2];     % Haar Low-pass filter
h1 = [-1/2 1/2];     % Haar High-pass filter

% Resynthesis filters are flip (time-reverse) of analysis filters
g0 = h0(end:-1:1);   % This will be [1/2  1/2] for Haar
g1 = h1(end:-1:1);   % This will be [1/2 -1/2] for Haar

% Split signal into the two halves
N2 = (N+1)/2;
c1 = c(1:N2);        % First half of signal
d1 = c(N2+1:end);    % Second half of signal
```

Redesign the filters in the high-pass and low-pass sections and flip them; split the signal into two parts.

- Line 32 – 33:

```
% Recursively call idwt_haar on the first half, with 1 fewer steps
idwtc1 = idwt_haar(c1, steps-1);
```

The idwt_haar function is called recursively to perform the inverse DWT transform on the first half of the signal.

- Line 35 – 44:

```
% Upsample the signals: first c ...
for i=1:N2
   up_c(2*i-1) = idwtc1(i);   % Transfer odd samples
   up_c(2*i)   = 0;           % Even samples set to zero
end
% ... then d
for i=1:N2
   up_d(2*i-1) = d1(i);
   up_d(2*i)   = 0;
end
```

Double up-sampling of idwtc1 and d1, interpolating zero between them.

- Line 46 – 48

```
% Filter the signals
filt_c = conv(g0, up_c);
filt_d = conv(g1, up_d);

% Add and scale
idwtc = sqrt(2)*(filt_c + filt_d);

% Answer is defined for n=0..N (in Matlab, 1 to N)
idwtc = idwtc(1:N+1);
```

The up-sampled signal is applied to the filter, combined and multiplied by the root number 2.

The result is then re-ranged and is the return value of the function, i.e., the sequence after this inverse DWT transform.

3 Signal Compression using the DWT

This exercise is to generate a test signal, plot its DWT, then compress the signal by discarding small values. By reducing the number of non-zero values the result is that fewer bits are needed to represent it. However, the result is some distortion of the original signal.

Download the files: testsig.m, compress.m, uncompress.m, comp_ratio.n.

The file testsig.m generates the following signal:

$$s(t) = \begin{Bmatrix} e^t \cos(32\pi t) & 0 \le t < 1/2 \\ 0 & 1/2 \le t < 3/4 \\ \cos(96\pi t) & 3/4 \le t < 7/8 \\ 0 & 7/8 \le t < 1 \end{Bmatrix}$$

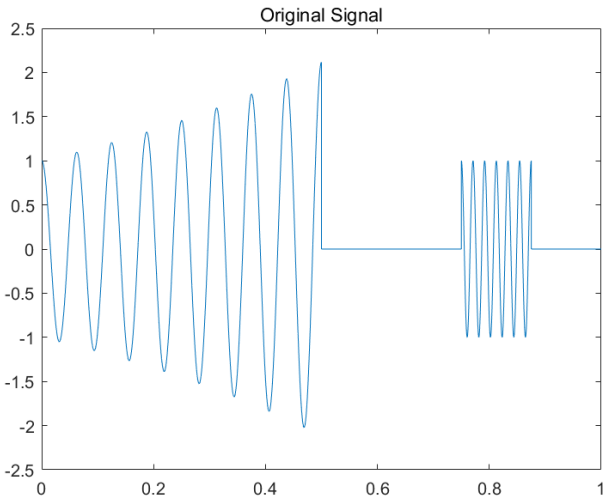
Modify the file testsig.m so that the first part is now $e^{1.5t}\cos(32\pi t)$

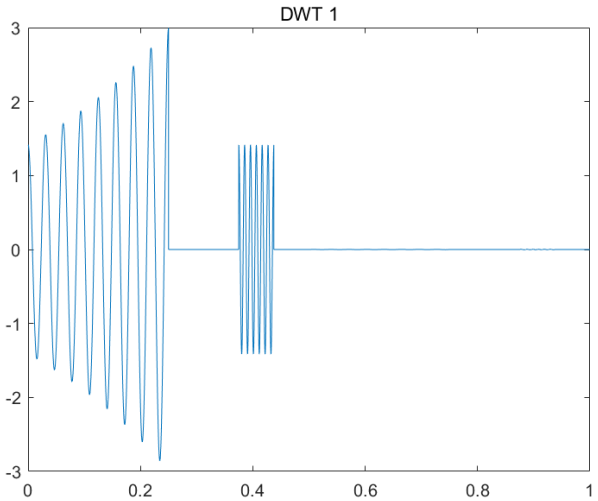
3.1 Plot this signal and its DWT for 4 levels.

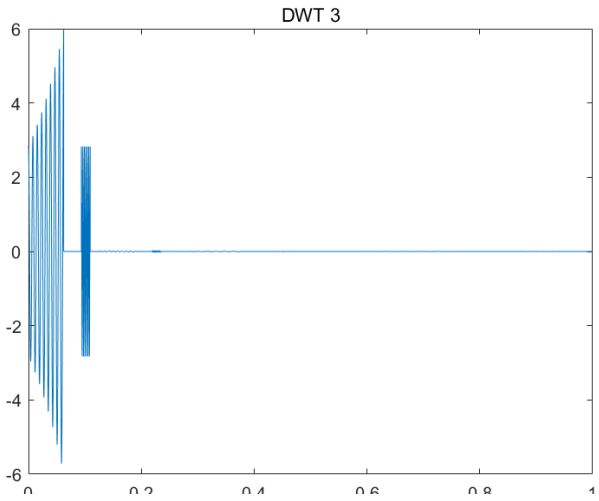
Plot the signal from the modified testsig.m and then its DWT after 1, 2, 3 and then 4 steps of decomposition

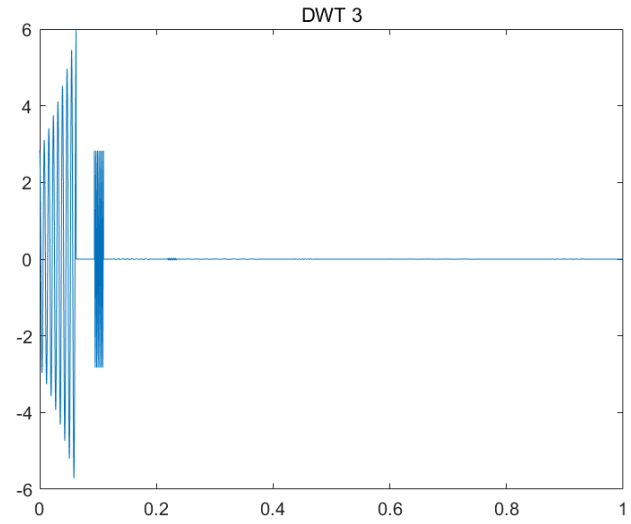
Your answer.

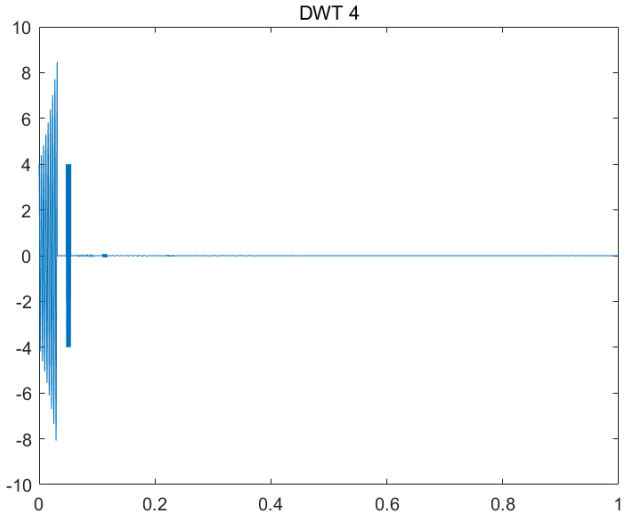
N = 2^16 = 65536








DWT 3
6
4
2
0
-2
-4
-6
0
0.2
0.4
0.6
0.8
1


DWT 4
10
8
6
4
2
0
-2
-4
-6
-8
-10
0
0.2
0.4
0.6
0.8
1

3.2 Compress the signal

Applying the file compress.m will compress this signal by transforming it and setting all values below a threshold to zero.

The file uncompress.m inverts the transform carried out by compress.m.

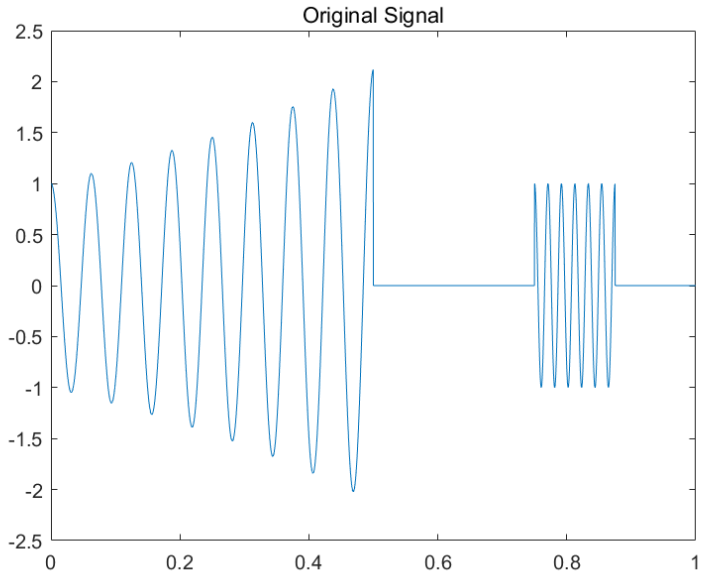
✓

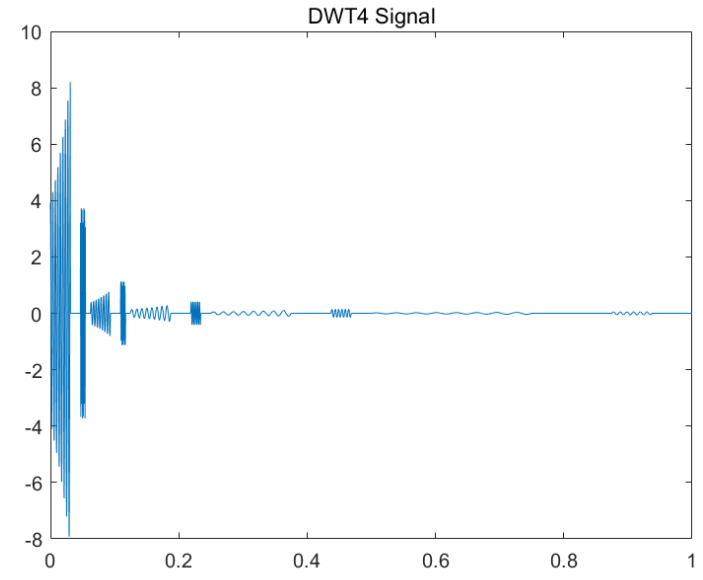
Modify the files compress.m and uncompress.m to include your transform of the signal where indicated.

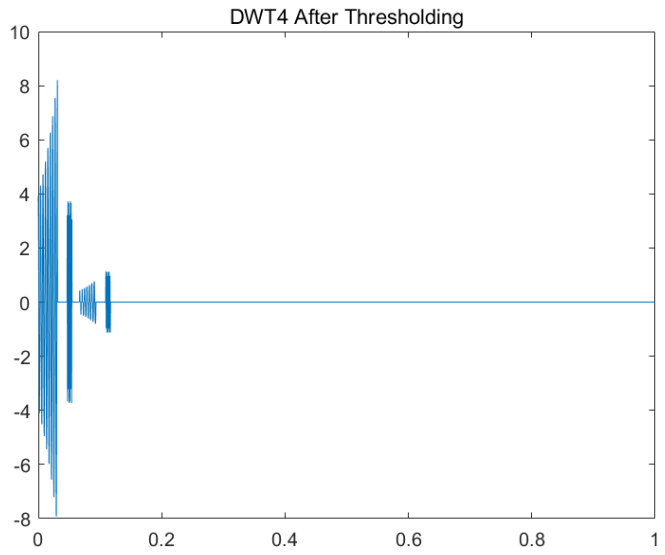
Plot the original modified signal, the 4-step DWT, the 4-step DWT after thresholding , the reconstructed signal from the inverse DWT and also the error introduced by the DWT thresholding (the difference between the original signal and the reconstructed one). Save the plots obtained.

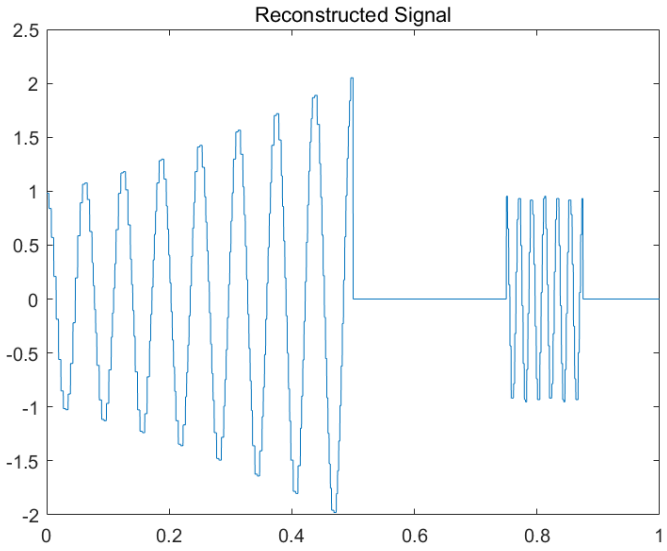
Your answer.

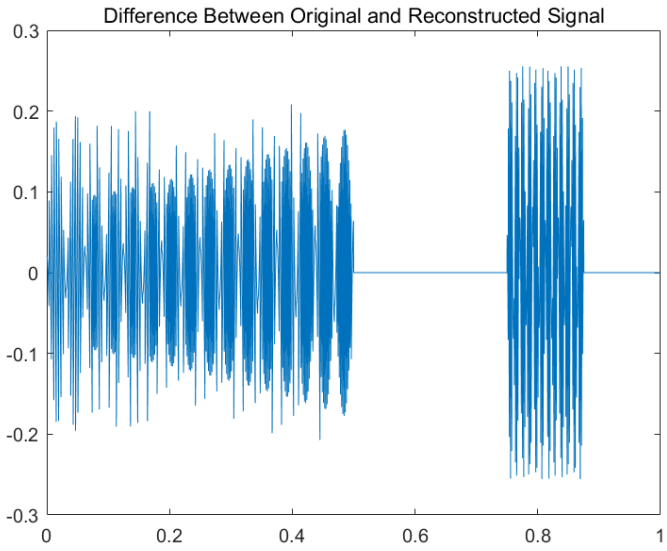
N = 2^12 = 4096; threshold = 0.42






DWT4 After Thresholding
10
8
6
4
2
0
-2
-4
-6
-8
0
0.2
0.4
0.6
0.8
1


Reconstructed Signal
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
0.2
0.4
0.6
0.8
1


Difference Between Original and Reconstructed Signal
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
0
0.2
0.4
0.6
0.8
1

Part 2 (EBU5303)

4. Image preparation

a. **Create a Matlab function** in the file "**myJPEG.m**". **Read** the grayscale image file mona-lisa.png using the `imread` matlab function and display it with `imshow`.

✓

b. What is the dimension of the image (width x hight)? What is its size in bits? Calculate the number of 8x8 blocks of pixels contained in the image.

Your answer.

- The size is 250 x 360.
- 250*360*8 = 720000 bits
- int(250/8)*int(360/8) = 31*45 = 1395 blocks

  (If the image is filled in order to preserve the full image: ceil(250/8)*ceil(360/8) = 32*45 = 1440 blocks)

c. In myJPEG.m, for one of the channels only and working from left to right, top to bottom, **split** the image into 8x8 blocks of pixels. Comment your code.

✓

d. **Print** the first two 8x8 matrices of pixel values as examples (use the `disp` matlab function).

Copy and paste the matrices

First Matrix

104	103	105	99	92	92	94	100
109	103	105	100	96	92	93	94
109	105	99	98	98	102	105	93
106	110	104	99	103	112	112	97
113	113	112	103	105	109	107	91
121	121	119	107	103	99	95	90
114	117	112	105	105	95	92	93
101	107	103	103	105	96	92	96

Second Matrix

100	94	85	84	89	103	103	95
88	86	88	84	85	96	99	95
88	86	86	89	93	100	102	98
91	92	90	97	97	105	105	101
90	90	94	97	95	102	105	98
90	91	93	94	92	93	98	100
94	94	89	89	94	98	96	98
97	97	89	89	95	98	94	93

e. In myJPEG.m, **subtract** 128 from each pixel values so they range from -128 to 127. Comment your code. ✓

f. **Print** (`disp`) again the first two 8x8 matrices of pixel values and check they have been "levelled off".

Copy and paste the "levelled off" matrices

First Matrix

```
-24  -25  -23  -29  -36  -36  -34  -28
-19  -25  -23  -28  -32  -36  -35  -34
-19  -23  -29  -30  -30  -26  -23  -35
-22  -18  -24  -29  -25  -16  -16  -31
-15  -15  -16  -25  -23  -19  -21  -37
 -7   -7   -9  -21  -25  -29  -33  -38
-14  -11  -16  -23  -23  -33  -36  -35
-27  -21  -25  -25  -23  -32  -36  -32
```

Second Matrix

```
-28  -34  -43  -44  -39  -25  -25  -33
-40  -42  -40  -44  -43  -32  -29  -33
-40  -42  -42  -39  -35  -28  -26  -30
-37  -36  -38  -31  -31  -23  -23  -27
-38  -38  -34  -31  -33  -26  -23  -30
-38  -37  -35  -34  -36  -35  -30  -28
-34  -34  -39  -39  -34  -30  -32  -30
-31  -31  -39  -39  -33  -30  -34  -35
```

5. Applying 2D DCT to each block

a. In myJPEG.m, working from left to right, top to bottom, **apply** the 2D DCT Matlab function (`dct2`) to each block of pixel values. Save the results in 8x8 matrices of DCT coefficients values. Comment your code. ✓

b. **Print** (`disp`) the first two 8x8 matrices of DCT coefficients values.

Copy and paste the DCT matrices

First Matrix

```
-201.8750   40.7379    2.5126    3.5727   -9.1250   -0.1836   -1.4467    3.4005
 -13.3163  -12.5239    8.7008   -0.1176   -0.4433    7.4667    2.2557   -0.9583
 -19.6236    2.9356    0.1402  -14.4638   13.6649   -5.3012    1.9294   -0.8980
   8.3138   16.6129    1.0360   -6.8044   -0.6615    0.3763    4.2425    1.4944
  -0.1250  -15.7590   -2.1628    1.2498   -4.3750   -0.9280   -0.3218   -1.3851
   2.6773   -3.0854    3.0435    0.0228   -2.0441   -1.6778   -1.6912   -1.3525
  -0.6661    0.1560    1.4294    0.4091   -0.6541    1.1602   -0.3902   -1.3632
  -1.4836   -1.3118    0.0035    2.4986   -0.0045    1.4734   -1.0667    0.5061
```

Second Matrix

```
-271.5000  -26.0595   10.2119   12.3171   -7.7500    3.2627   -0.1709   -3.2665
  -5.0256   -8.1248    4.9878    4.9707   -4.9738    5.2796    1.1810   -0.2532
  -8.5391   10.4890    9.6051    7.8289   -1.6004   -4.8424    1.0821    0.4498
   2.0890    2.3203    2.9602   -2.8853   -0.6709    2.3292    0.1812    0.4780
  11.0000    2.5018   -1.1316    6.4881   -1.7500    0.6649   -1.4254   -2.2534
   6.1915   -1.0335    1.9780    4.6095    1.8175   -2.7308   -1.0854   -1.9704
   0.2898    0.8183    0.3321    0.2427    2.0159   -0.1773   -1.3551    0.5273
   0.5425   -1.4729   -0.5351    3.5111   -0.8709    0.5262    0.2349    0.7409
```

6. Quantisation

a. In myJPEG.m, **create** a standard quantisation matrix named Q_{std} similar to the matrix shown below. ✓

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

b. In myJPEG.m, **create** two more quantisation matrices, one for lower quantisation (Q_{low}) and one for higher quantisation (Q_{high}). Comment your code. ✓

Hint: use quantisation factors to multiply Q_{std}. The scaled matrices must then be rounded and clipped to have positive integer values ranging from 1 to 255.

c. **Print** (`disp`) Q_{low} and Q_{high} and explain how you created them (i.e., what quantisation factor you chose and why).

Copy and paste the quantisation matrices and explain your choice of quantisation factors.

Q_{low}

```
 7   5   4   7  10  17  21  26
 5   5   6   8  11  24  25  23
 6   5   7  10  17  24  29  24
 6   7   9  12  21  37  34  26
 8   9  16  24  29  46  43  32
10  15  23  27  34  44  47  39
21  27  33  37  43  51  50  42
30  39  40  41  47  42  43  42
```

Q_{high}

```
 32   22   20   32   48   80  102  122
 24   24   28   38   52  116  120  110
 28   26   32   48   80  114  138  112
 28   34   44   58  102  174  160  124
 36   44   74  112  136  218  206  154
 48   70  110  128  162  208  226  184
 98  128  156  174  206  242  240  202
144  184  190  196  224  200  206  198
```

I chose factor_low = 0.42, because I think the number 42 has a special meaning and is moderately sized as a low quantization factor.

factor_high = 2, because the coefficient is large enough and can keep the values in the quantization matrix within 255 (the maximum value of unsigned int8) to avoid more distortion.

d. In myJPEG.m, **apply** quantisation to all the 8x8 matrices of DCT coefficients, using Q_{std}, Q_{low} and Q_{high}. Comment your code.

✓

Hint: quantisation is achieved by dividing each DCT coefficient by the corresponding element in the quantisation matrix, and then rounding to the nearest integer value.

e. **Print** (`disp`) the first two 8x8 matrices of quantised DCT coefficients for the three different levels of quantisation (standard, low and high), i.e., 6 matrices in total.

Copy and paste the six quantised matrices here.

Standard Quantization 1

-13	4	0	0	0	0	0	0
-1	-1	1	0	0	0	0	0
-1	0	0	-1	0	0	0	0
1	1	0	0	0	0	0	0
0	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Standard Quantization 2

-17	-2	1	1	0	0	0	0
0	-1	0	0	0	0	0	0
-1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Low Quantization 1

-29	8	1	1	-1	0	0	0
-3	-3	1	0	0	0	0	0
-3	1	0	-1	1	0	0	0
1	2	0	-1	0	0	0	0
0	-2	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Low Quantization 2

-39	-5	3	2	-1	0	0	0
-1	-2	1	1	0	0	0	0
-1	2	1	1	0	0	0	0
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

High Quantization 1

-6	2	0	0	0	0	0	0
-1	-1	0	0	0	0	0	0
-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

High Quantization 2

-8	-1	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

f. **Compare** the 6 matrices and **comment**.

Your answer.

When a lower quantization factor is chosen, larger values within the matrix represent higher accuracy; more non-zero values of the matrix, i.e., more information is retained, and therefore more storage space may be required.

When a higher quantization factor is chosen, smaller values within the matrix represent lower precision; fewer non-zero values of the matrix, i.e., more information is discarded, and therefore less storage space may be required.

When a standard quantization factor is chosen (i.e. factor = 1), the matrix is somewhere in between, i.e. the quality and storage space are relatively balanced.

g. Comment on how the choice of quantisation matrix affects the compression rate.

Your answer.

The higher the value of the quantization matrix, the more effective information in the quantized matrix, the lower the compression rate, and the higher the image quality.

The lower the value of the quantization matrix, the less valid information in the quantized matrix, the higher the compression rate, and the lower the image quality.

7. Decompression

a. In myJPEG.m, **multiply** the quantised DCT values by the corresponding element of the quantisation matrix originally used (i.e., Q_{std}, Q_{low} or Q_{high}). Comment your code.

✓

b. **Print** (`disp`) the first two 8x8 matrices of DCT coefficients for the three different levels of quantisation (standard, low and high), i.e., 6 matrices in total. **Compare** with the two matrices of question 5b. and **comment**.

Copy and paste the matrices, and comment.

Standard Dequantization 1

-208	44	0	0	0	0	0	0
-12	-12	14	0	0	0	0	0
-14	0	0	-24	0	0	0	0
14	17	0	0	0	0	0	0
0	-22	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Standard Dequantization 2

-272	-22	10	16	0	0	0	0
0	-12	0	0	0	0	0	0
-14	13	16	0	0	0	0	0
0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Low Dequantization 1

-203	40	4	7	-10	0	0	0
-15	-15	6	0	0	0	0	0
-18	5	0	-10	17	0	0	0
6	14	0	-12	0	0	0	0
0	-18	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Low Dequantization 2

-273	-25	12	14	-10	0	0	0
-5	-10	6	8	0	0	0	0
-6	10	7	10	0	0	0	0
0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

High Dequantization 1

-192	44	0	0	0	0	0	0
-24	-24	0	0	0	0	0	0
-28	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

High Dequantization 2

-256	-22	20	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Compared with the matrices in 5b, terms with larger absolute values are retained with partial errors; values with smaller absolute values are discarded as 0.

Matrices with lower quantile coefficients have more accurate values and discard fewer values; matrices with higher quantile coefficients have higher errors in values and discard more values; matrices with standard quantile coefficients are in between.

c. In myJPEG.m, **apply** the inverse DCT (`idct2`) to all 8x8 matrices of DCT coefficients (standard, low and high quantisation). **Add** 128 to each matrix element and **round** the values. Comment your code. ✓

d. **Print** (`disp`) the first two 8x8 matrices of pixels for the three different levels of quantisation (standard, low and high), i.e., 6 matrices in total. **Compare** with the two matrices of question 4.d and **comment**.

Copy and paste the matrices, and comment.

Standard IDCT 1

103	106	106	100	93	90	96	103
108	107	105	99	93	91	93	95
109	104	100	97	98	98	96	94
109	102	97	99	105	107	103	98
117	110	105	105	108	108	101	94
124	120	115	110	106	100	92	86
113	115	115	110	102	94	89	88
95	102	108	107	99	94	95	98

Standard IDCT 2

98	92	87	87	91	97	100	100
91	87	84	85	90	95	97	97
89	87	86	90	95	99	99	97
92	91	92	97	103	106	105	102
93	92	93	98	103	106	104	101
92	89	88	90	95	97	96	94
96	91	87	86	89	92	93	92
103	97	90	88	90	94	95	95

Low IDCT 1

103	102	101	100	95	90	93	100
108	105	101	98	96	95	94	93
109	107	100	95	100	106	102	92
106	108	103	97	102	113	110	97
110	115	112	102	102	109	107	96
118	122	118	107	101	100	96	89
117	116	112	107	101	96	92	89
109	104	102	104	103	98	95	95

Low IDCT 2

99	92	84	83	93	102	102	96
91	87	82	81	89	98	98	94
90	89	87	87	93	100	101	98
90	93	94	93	97	103	105	103
87	91	92	92	94	99	101	100
89	92	92	91	93	97	98	96
95	95	93	91	94	98	98	95
97	96	92	89	92	97	96	92

High IDCT 1

97	97	96	96	95	94	94	93
101	101	100	99	98	97	96	96
108	107	106	104	103	101	100	99
114	113	111	109	106	104	102	101
118	117	114	111	108	104	102	101
119	117	114	110	106	102	99	97
118	116	113	108	103	99	95	93
117	115	111	106	101	96	92	90

High IDCT 2

95	94	92	92	93	97	101	103
95	94	92	92	93	97	101	103
95	94	92	92	93	97	101	103
95	94	92	92	93	97	101	103
95	94	92	92	93	97	101	103
95	94	92	92	93	97	101	103
95	94	92	92	93	97	101	103
95	94	92	92	93	97	101	103

Compared with the matrix of 4d, the values of the matrix are closer but with errors; among them, the recovery matrix with low quantization coefficients has smaller errors, the high quantization coefficients have larger errors, and the standard quantization coefficients are in between.

8. Image reconstruction

a. In myJPEG.m, **reconstruct** the images from the 8x8 blocks of pixel values you obtained in question 7 for the three different levels of quantisation (standard, low and high). Comment your code. ✓

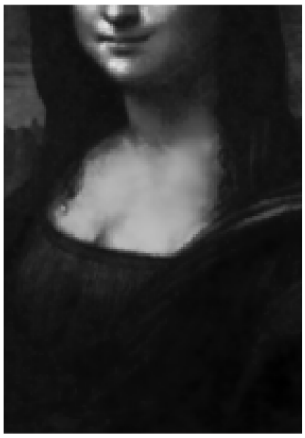
b. **Display** and **save** the four images: original, standard, low and high compression. ✓

c. **Compare the images and comment**. *You should clearly see blocky artefacts in the high compression image. If not, go back to Question 6 and modify the Q_{high} matrix.*

Hint: You may have to cast your pixel values to uint8 to display the images.

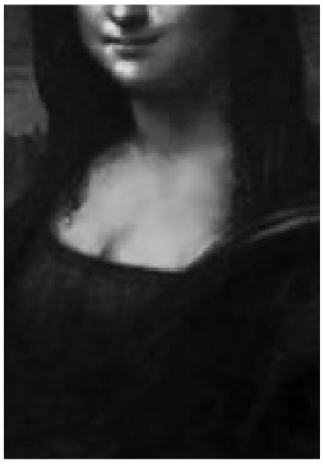
Your comments.

original image



standard quantization image

low quantization image



high quantization image

Comparing the three compressed images with the original image, we can see that the higher the quantization factor, the less detail is retained in the image; the image with high quantization factor has obvious blocking artifacts; the image with high quantization factor has obvious ringing artifacts at the edges; the image with low quantization factor is closest to the original image; the image with standard quantization factor is in between.

Handing In

Compile the answers to the exercises, including the answers to specific questions, program listings (including comments), and plots from experiments, into a "folder" of results showing that you have completed the lab.

Name the folder: EBU6018_5303_Lab2

Rename this document 'Lab2_xxxxxxxx' where xxxxxxxx is your QM student number and save it in your result folder together with your MATLAB files and plot images.

Submit the folder as a zip archive on QMplus in **BOTH the EBU6018 and EBU5303 course areas** before the deadline (i.e., submit twice the same zip archive).

IMPORTANT: Plagiarism (copying from other students or copying the work of others without proper referencing) is cheating, and **will not be tolerated**.

IF TWO "FOLDERS" ARE FOUND TO CONTAIN IDENTICAL MATERIAL, BOTH WILL BE GIVEN A MARK OF ZERO.

Marking scheme

Part 1 EBU6018 (max 45 marks):

Q2.1 : up to 16 marks

Explanations up to 5 marks

Plots up to 6 marks

Calculations up to 5 marks

Q2.2 : up to 9 marks

Plots up to 6 marks

Explanations up to 3 marks

Q3.1 : up to 10 marks for plots.

Q3.2 : up to 10 marks for plots

Part 2 EBU5303 (max 80 marks):

Q4. Image Preparation: up to 15 marks
Q4.b up to 3 marks
Q4.c up to 4 marks
Q4.d up to 4 marks (2 per matrix)
Q4.e up to 2 marks
Q4.f up to 2 marks (1 per matrix)

Q5. 2D DCT: up to 8 marks

Q5.a up to 4 marks
Q5.b up to 4 marks (2 per matrix)

Q6. Quantisation: up to 27 marks
Q6.a up to 1 mark
Q6.b up to 5 marks
Q6.c up to 5 marks (1 per matrix, and 3 for the explanations)
Q6.d up to 5 marks
Q6.e up to 3 marks (0.5 per matrix)
Q6.f up to 5 marks
Q6.g up to 3 marks

Q7. Decompression: up to 18 marks
Q7.a up to 3 marks
Q7.b up to 5 marks
Q7.c up to 6 marks
Q7.d up to 4 marks

Q8. Reconstruction: up to 12 marks
Q8.a up to 5 marks
Q8.b up to 3 marks (1 per image)
Q8.c up to 4 marks

Updated by MPD, MEPD
Modified ARW for EBU6018.
Modified MLB for EBU5303.