

#### **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 (*t*) , we only need the coefficients of the low-pass and high-pass filters and

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:

**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.

Explain

* 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 downsamples 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

1. 文本, 信件

   描述已自动生成

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

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

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

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

steps = 1; c2 = [2, 4]; d2 = [8, 4]

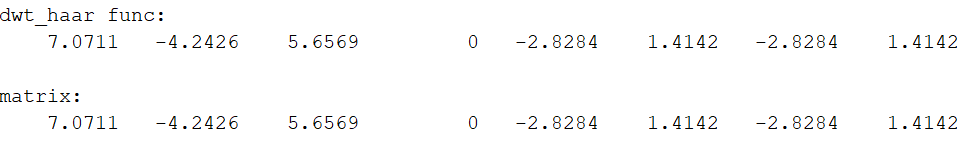
c2[1] = 0.5\* \*(c1[1]+c1[2]) = 2; c2[2] = 0.5\* \*( c1[3]+ c1[4]) = 4

d2[1] = 0.5\* \*(c1[1]-c1[2]) = 8 ; d2[2] = 0.5\* \*(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).

1. 

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];

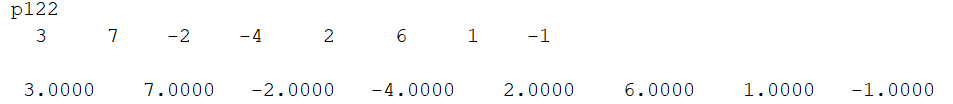
dwts = dwt\_haar(s, 2);

idwts = idwt\_haar(dwts, 2);

disp(s);

disp(idwts);

Results:



The original sequence is obtained after the inverse conversion.

Explain:

* 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:

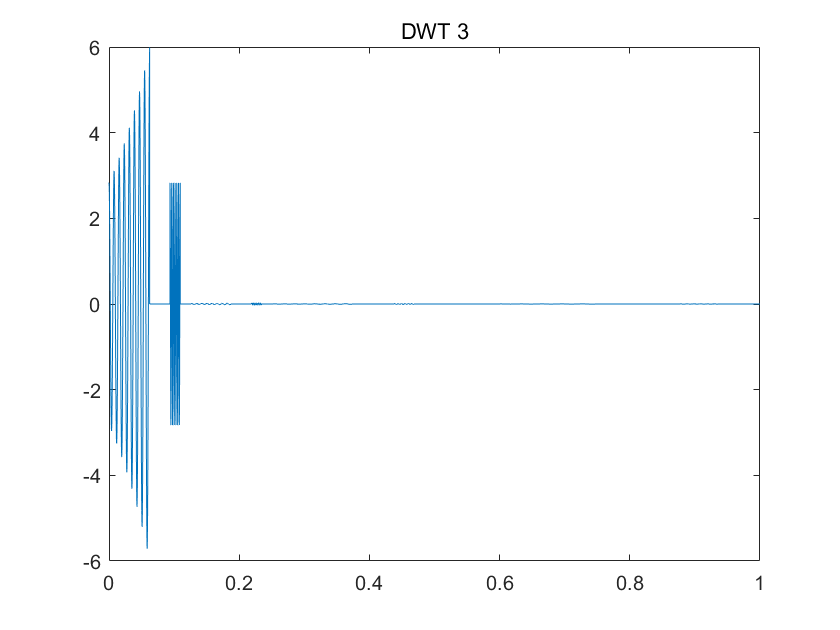
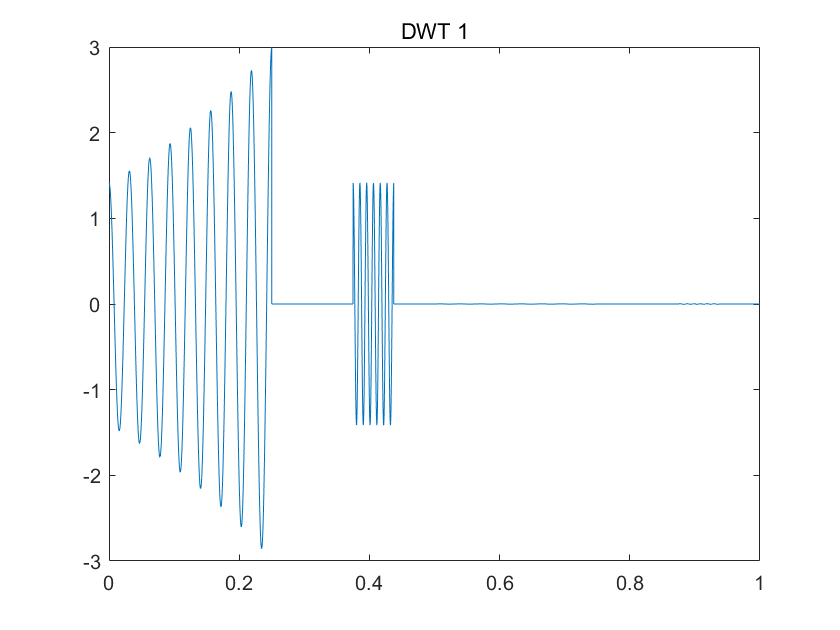
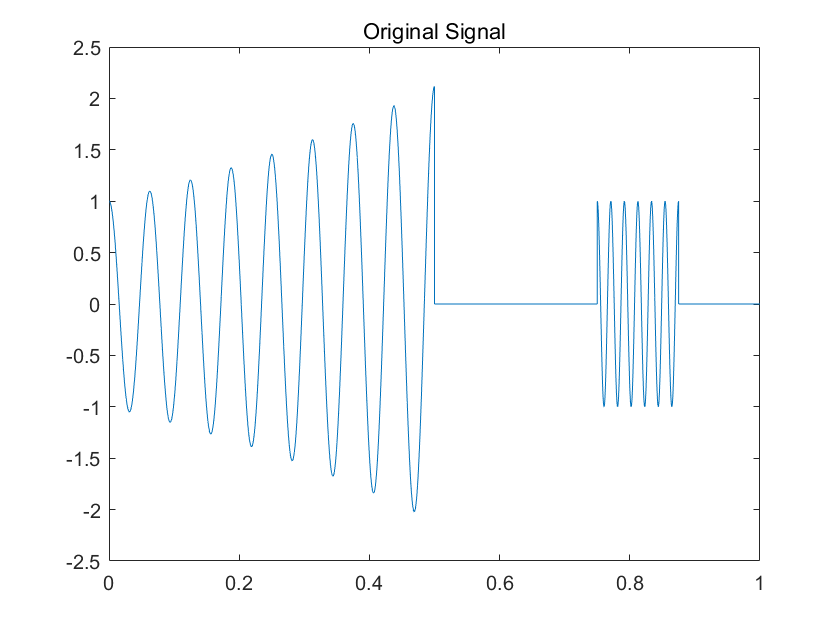
**Modify the file testsig.m so that the first part is now**

**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



图表, 直方图

描述已自动生成

图表

描述已自动生成

**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.

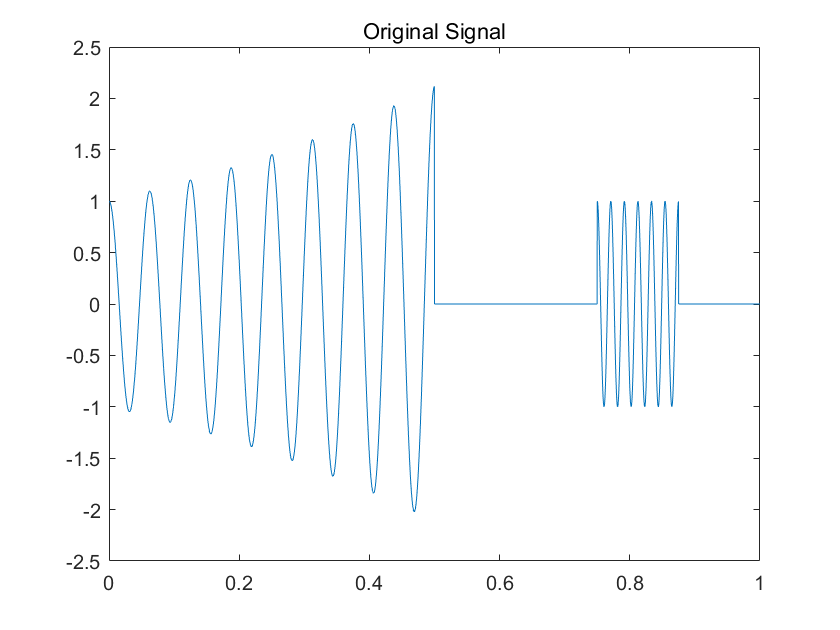
The file uncompress.m inverts the transform carried out by compress.m.

**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

图表

描述已自动生成

图表

描述已自动生成图表, 直方图

描述已自动生成图表, 条形图, 直方图

描述已自动生成

**Part 2 (EBU5303)**

**4. Image preparation**

1. **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.

1. 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 filling edge: ceil(250/8)\*ceil(360/8) = 32\*45 = 1440 blocks

1. 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.
2. **Print** the first two 8x8 matrices of pixel values as examples (use the disp matlab function).

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

Second 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

First Matrix

Copy and paste the matrices

1. In myJPEG.m, **subtract** 128 from each pixel values so they range from -128 to 127. Comment your code.
2. **Print** (disp) again the first two 8x8 matrices of pixel values and check they have been “levelled off”.

-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

Second 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

First Matrix

Copy and paste the “levelled off” matrices

**5. Applying 2D DCT to each block**

1. 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.
2. **Print** (disp) the first two 8x8 matrices of DCT coefficients values.

-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

Second 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

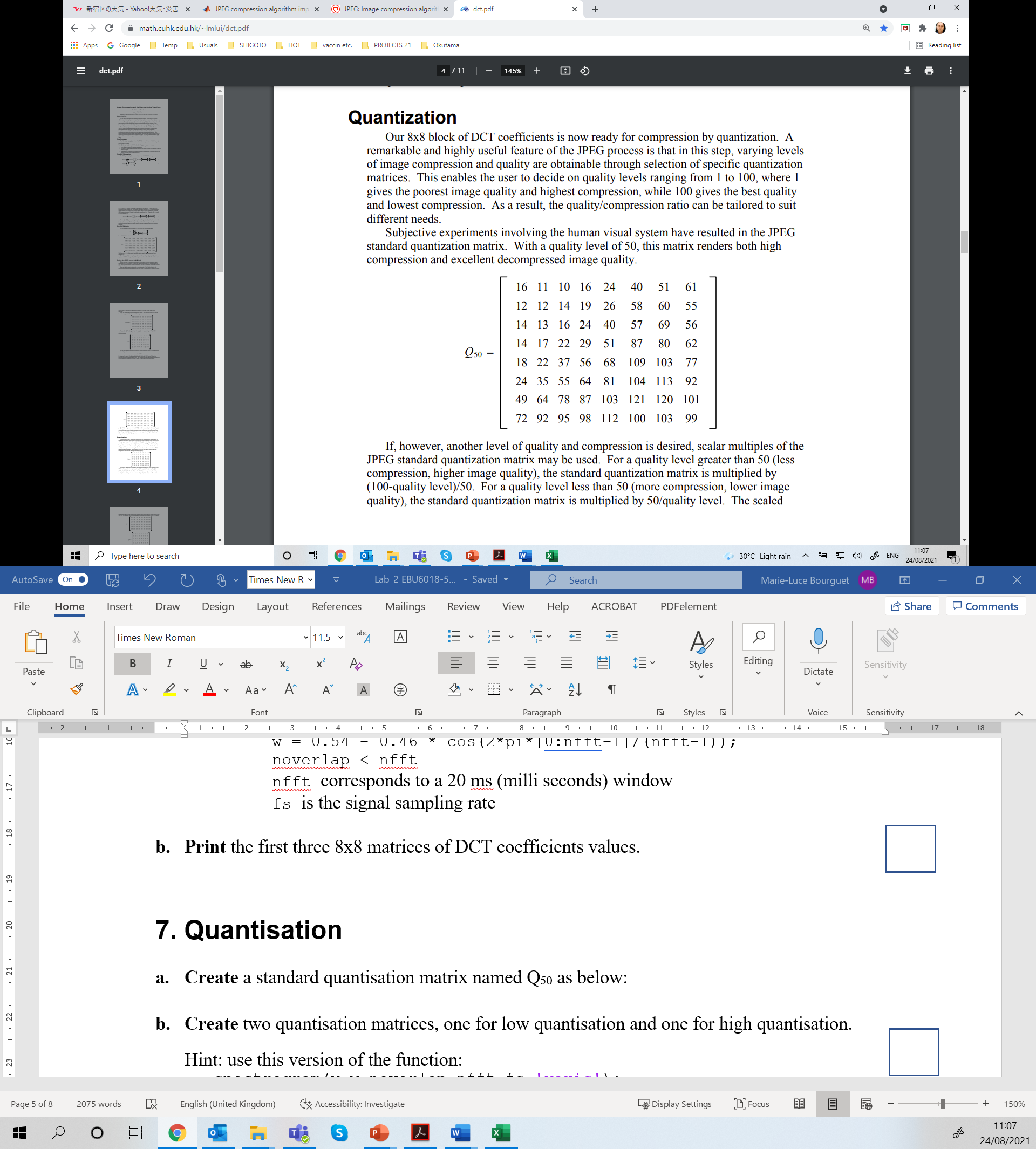
First Matrix

Copy and paste the DCT matrices

Paste the matrices

**6. Quantisation**

1. In myJPEG.m, **create** a standard quantisation matrix named Qstd similar to the matrix shown below.



1. In myJPEG.m, **create** two more quantisation matrices, one for lower quantisation (Qlow) and one for higher quantisation (Qhigh). Comment your code.

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

1. **Print** (disp) Qlow and Qhigh 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.

1. In myJPEG.m, **apply** quantisation to all the 8x8 matrices of DCT coefficients, using Qstd, Qlow and Qhigh. 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.*

1. **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.

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

Your answer.

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

Your answer.

**7. Decompression**

1. In myJPEG.m, **multiply** the quantised DCT values by the corresponding element of the quantisation matrix originally used (i.e., Qstd, Qlow or Qhigh). Comment your code.
2. **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.

1. 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.
2. **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.

**8. Image reconstruction**

1. 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.
2. **Display** and **save** the four images: original, standard, low and high compression.
3. **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 Qhigh matrix.*

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

Your comments.

**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.