

Laboratory Notes for

ECTE401/ECTE901

Fast Signal Processing Algorithms

Attending the Lab is essential, and it is also essential to be well prepared for the Lab. All Lab experiments use basic Matlab functions which are part of any Matlab installation. Thus, students will be able to prepare for the Lab wherever they have Matlab access.

You must record your laboratory findings, and you must hand in lab reports. Progress will be recorded on the Instructor Verification Sheet. After completion of a part of an experiment (e.g. Experiment 1.1), demonstrate your achievements to the lab instructor and let him sign your Instructor Verification Sheet. It is the students' responsibility to ensure that their Instructor Verification Sheet is signed off by their laboratory supervisor. After completion of all parts of an experiment (e.g. Experiments 1.1 - 1.4), hand in the report on the entire experiment. This report will be marked.

As a minimum, students are expected to complete all laboratories marked with an asterisk *. Completion of the marked experiments yields a mark of 50 out of 100, provided the results are correct. Completion of all experiments is required to get a mark of 100 out of 100. Marks are allocated to the experiments as follows:

1.1: 7	2.1: 6	3.1: 7	4.1: 12
1.2: 6	2.2: 7	3.2: 5	4.2: 4
1.3: 6	2.3: 12	3.3: 13	4.3: 9
1.4: 6			

These marks do not necessarily reflect the amount of work required to complete an experiment. They are distributed such that a mark of 50 can be achieved by completing only the marked experiments.

Note that some experiments require additional reading in [1] or material from the lecture notes.

References

[1] Burrus et. al, Computer-Based Exercises for Signal Processing. Prentice Hall, 1997.

1 The discrete Fourier transform (DFT)

1.1 DFT of some simple signals*

1. Consider the signal $x = [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$ and compute its DFT. In Matlab the DFT is computed with the FFT command, i.e. $X = \text{fft}(x)$;
2. Plot (using the `stem` command)
 - the signal itself: `subplot(221); stem(x)`
 - the real part of the DFT: `subplot(222); stem(real(fft(x)))`
 - the imaginary part of the DFT: `subplot(223); stem(imag(fft(x)))`
 - the magnitude of the DFT: `subplot(224); stem(abs(fft(x)))`
3. Comment on the results.
4. Repeat points 1. - 3. for the following signals:
 - (a) $x = [0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$
 - (b) $x = [1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0]$
 - (c) $x = [1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1]$
5. Comment on the results.

1.2 Zero Padding*

1. Consider the signal $x = [2 \ 1 \ 1 \ 1]$. Compute and note the DFT of the signal.
2. Zero-pad the signal to length 8 and compute the DFT. Note your result.
3. Zero-pad the signal to length 16 and compute the DFT. Note your result.
4. Plot the magnitude of the length-16 DFT. In your plot, indicate the points computed with the original length-4 DFT.
5. Taking into account the above results, comment on the effect of zero padding.

1.3 IFFT via FFT

Write a Matlab function (m-file) that uses the `fft` command to compute the IFFT. See Ref. [1] for hints on how to do that. Call your function `myifft.m`. The first line of your function has to be as follows:

```
function y=myifft(x);
```

Show that your function operates properly by showing that

$$\text{myifft}(\text{x})-\text{ifft}(\text{x})$$

is zero.

1.4 Fast Convolution

1. Implement the overlap save method of fast convolution. The first line of your function should read

```
function y=myfastfilter(x,h,N);
```

N is the FFT length, and h is a filter impulse response. You can choose your own filter and FFT length.

2. Demonstrate that your function works by comparing its output to the output of the `conv` function. Note: Your function may produce some tailing zeros. Don't worry about them.
3. Demonstrate the efficiency of your implementation by counting the number of operations required to filter a length-10000 signal. The comparison can be done by counting the number of flops (floating point operations) and comparing it to the number of flops required by the `conv` routine. The Matlab command is `flops`. With `flops(0)` you can reset the counter.

You will find that the fast convolution is better than `conv` when the filter h is long. For very short filters it is better to use `conv`. For which filter length are both methods equally good?

Note: Matlab has a function called `fftfilt.m` which contains the overlap add method. You may have a look at the source code to find out how your function might be structured.

2 The DCT

2.1 DCT as a Matrix*

1. Write an m-file that creates the $N \times N$ DCT-II matrix with entries

$$[\mathbf{C}]_{k,n} = \sqrt{\frac{2}{N}} \gamma_k \cos \frac{k(n + \frac{1}{2})\pi}{N}, \quad k, n = 0, 1, \dots, N - 1$$

with

$$\gamma_k = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } k = 0, \\ 1 & \text{otherwise.} \end{cases}$$

A vector $\mathbf{y} = \mathbf{C}\mathbf{x}$ then contains the DCT of \mathbf{x} . The first line of your program should be

```
function C=cosmat(N);
```

2. Show (using Matlab) that $\mathbf{C}^{-1} = \mathbf{C}^T$.
3. Choose $N = 16$ and compute the DCT of $\mathbf{x} = [x(0), \dots, x(15)]^T$ with $x(n) = \cos(\omega_0 n)$ for six different frequencies ω_0 between 0 and π . Plot \mathbf{x} and \mathbf{y} for the six frequencies using the `stem` command. What do you observe?

2.2 DCT via FFT*

Write a program that implements the DCT-II via an FFT. For further information, see [1], Chapter 2, Related Transforms, Exercise 1.2.

2.3 DCT in Image Compression

Write a program that carries out a block-wise 2-D DCT of an image. You can load an image name .jpg into Matlab by using the command

```
X=imread('name.jpg');
```

This image will have three colour components. Extract the first colour component by typing `x=X(:, :, 1)` and display it as a b/w (black-and-white) image using

```
imagesc(x);colormap(gray(256));
```

Divide \mathbf{x} into 8×8 blocks and compute the 2-D DCT for each block. The 2-D DCT can be computed as

$$\mathbf{y}_{8 \times 8} = \mathbf{C} \mathbf{x}_{8 \times 8} \mathbf{C}^T.$$

Then rearrange the DCT coefficients of your transformed blocks in such a way that coefficients with the same meaning form blocks. For example, if your image has 200×200 pixels, you will have to form blocks of size 25×25 . Let \mathbf{y} be the result of your operations. Display \mathbf{y} using

```
imagesc(y);colormap(gray(256));
```

Write a second program that reconstructs \mathbf{x} from \mathbf{y} without error.

Reduce the amount of information in the transform domain by setting those parts of \mathbf{y} to zero which seem to be of little importance anyway. Reconstruct an image from the modified \mathbf{y} and look at the result. What do you observe?

3 Upsampling and Downsampling

3.1 Upsampling and Interpolation*

In this part of the experiment, we look at upsampling by a factor L , followed by an interpolation filter $h(n)$.

1. Design a Matlab function that carries out upsampling by L , followed by filtering with $h(n)$.

```
function y=upsamp(x,L,h);
```

Use an input signal of the form $1, -1, 1, -1, \dots$ and upsample it by $L = 3$. Choose the dummy filter $h(n) = \delta(n)$ to test your m-file (in Matlab: $h=1$;). Plot both the input and the upsampled signal, using the `stem` command.

2. Now use

(a) a zero-order hold operation. That is,

$$h(n) = \sum_{m=0}^{L-1} \delta(n - m) = \delta(n) + \delta(n - 1) + \dots + \delta(n - L + 1)$$

In Matlab this is simply `h = [1 1 1]` for `L=3`.

(b) a linear interpolator

$$h(n) = \sum_{m=-(L-1)}^{L-1} (1 - |m|/L) \delta(n - m - n_0)$$

with n_0 just large enough to make the filter causal.

For both filters, plot the output signal using `stem`.

3.2 Filtering and Downsampling*

1. Design an FIR lowpass filter with 50 taps and cutoff frequency $\pi/4$. Use the `remez` function to design the filter. Plot the impulse and frequency responses of your filter.
2. Create a random input signal of length 100000, using `randn`. Filter the signal with the previously designed filter and downsample the result by $M = 2$. Plot the spectra of the input, the filtered, and the filtered and downsampled signals. Use the `spectrum` command for this.

3.3 Polyphase Filtering

1. Design a second Matlab function that carries out upsampling by L , followed by filtering with $h(n)$. This function should use a polyphase implementation and therefore should require less operations than the one above. The input and output variables are the same as above:

```
function y=upsamp2(x,L,h);
```

To solve this problem, your function will need to carry out the following operations: 1. decompose the filter impulse response into polyphase components; 2. filter the signal with the polyphase filters; 3. interleave your filtered sequences.

2. Create a Matlab function that carries out filtering with a filter $h(n)$, followed by downsampling by factor M .

```
function y=downsamp(x,M,h);
```

3. Create another Matlab function for filtering and downsampling that does uses a polyphase implementation and does not carry out obsolete operations.

```
function y=downsamp2(x,M,h);
```

Show that both functions operate properly and do the same job.

4 Filter Banks

4.1 Subband Coding Filter Bank*

Consider a two-channel filter bank based on the following filters:

$$h_0(n) = \frac{\sqrt{2}}{8} [1 + \sqrt{3}, 3 + \sqrt{3}, 3 - \sqrt{3}, 1 - \sqrt{3}]$$

$$h_1(n) = \frac{\sqrt{2}}{8} [-1 + \sqrt{3}, 3 - \sqrt{3}, -3 - \sqrt{3}, 1 + \sqrt{3}]$$

$$g_0(n) = \frac{\sqrt{2}}{8} [1 - \sqrt{3}, 3 - \sqrt{3}, 3 + \sqrt{3}, 1 + \sqrt{3}]$$

$$g_1(n) = \frac{\sqrt{2}}{8} [1 + \sqrt{3}, -3 - \sqrt{3}, 3 - \sqrt{3}, -1 + \sqrt{3}]$$

1. Create a Matlab function that carries out analysis filtering, followed by downsampling:

```
function [y0,y1]=analyze(x,h0,h1);
```

2. Create a second Matlab function that carries out upsampling and synthesis filtering:

```
function z=synthesize(y0,y1,g0,g1);
```

3. Create a non-trivial input signal $x(n)$ of your choice, analyze it, feed the subband signals $y_0(n)$ and $y_1(n)$ into your synthesis routine, and obtain the output signal $z(n)$. Plot the input, subband, and output signals.
4. Compute the energies of the input, subband, and output signals and comment on their relationships. Is the filter bank paraunitary? The energy of a vector \mathbf{x} can be computed as $E = \text{sum}(\mathbf{x} . * \mathbf{x})$.

4.2 Transmultiplexer

Create two signals $y_0(n)$ and $y_1(n)$ and feed them into your synthesis bank. Feed the output of the filter bank into a channel which performs a delay by m_0 taps.

- (a) Choose m_0 to be even. Analyze the channel output signal using the analysis bank. Let the output signals be $z_0(n)$ and $z_1(n)$. Plot $y_0(n)$, $y_1(n)$, $z_0(n)$, and $z_1(n)$. Comment on the result.
- (b) Do the same as before for an odd m_0 . Comment on the result.

4.3 Octave Filter Bank

1. Using the filters from Exercise 4.1, carry out a three-level octave decomposition by decomposing your lowpass subband signal into low and highpass components again and again. Plot all subband signals.
2. Carry out the synthesis operation, step by step. To get the correct output, you will have to adjust the delay between your subband components.
3. Using the Haar wavelet filters, carry out a seven-band decomposition of an image of your choice. Display the transformed image using the `imagesc` command.
4. Set all bands except the lowpass one to zero and reconstruct the image. Display the reconstructed image.

Haar filters:

$$H_0(z) = \frac{1}{\sqrt{2}} [1 + z^{-1}]$$

$$H_1(z) = \frac{1}{\sqrt{2}} [-1 + z^{-1}]$$

$$G_0(z) = \frac{1}{\sqrt{2}} [1 + z^{-1}]$$

$$G_1(z) = \frac{1}{\sqrt{2}} [1 - z^{-1}]$$

ECTE 401/901 Instructor Verification Sheet

Student Name: _____

Student No.: _____

Exercise	Date	Instructor's Signature
1.1		
1.2		
1.3		
1.4		
2.1		
2.2		
2.3		
3.1		
3.2		
3.3		
4.1		
4.2		
4.3		