L.EEC025 - FUNDAMENTALS OF SIGNAL PROCESSING

Academic year 2024-2025, week 10
TP (Recitation) problems

Topics: Exercises on the DFT

Exercise 1

Find the DFT, of length N, of the discrete-time signal $x[n]=\frac{1}{2}\left(1-\cos\frac{2\pi}{N}n\right)$. Validate your analytical result in Matlab by making, for example, N=8.

Exercise 2

Consider the discrete-time signal $x[n]=2^{-n}u[n]$.

a) Find its Fourier transform $X\left(e^{j\omega}\right)$.

b) Show that the discrete-time signal $y[n]$ of length N and whose DFT is obtained as $Y[k]=X\left(e^{jk\frac{2\pi}{N}}\right),\quad k=0,1,\ldots,N-1$, is given by $y[n]=\frac{2^{-n}}{1-2^{-N}},\quad n=0,1,\ldots,N-1$.

c) Verify numerically in Matlab the previous result using N=64.

Exercise 3

Consider the following set of Matlab commands:

```
x=[3 2 1 0 0 0];
X=fft(x);
Y=real(X);
y=x-ifft(Y);
Z=X.*X;
z=ifft(Z)
```

a) Say what the purpose of this Matlab code is, and identify the main DFT properties that are implied.

b) Without computing the DFT or the IDFT, find the contents of vector `y`. Explain.

c) Replace the two lines of the above code involving `Y` and `y` by different Matlab commands delivering the same result. Explain.

d) Without computing the DFT or the IDFT, find the contents of vector `z`. Explain.

e) The result in vector `z` is equivalent to the result of a linear convolution. Would this hold true if `x=[3 2 1 1 0 0]` ?

Exercise 4

Consider the following discrete-time signal $x_0[n] = [0 \quad 1 \quad 1 \quad 0]$.

a) Find its DFT.

b) Express the DFT of the signal $x[n] = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0]$ as a function of the DFT of $x_0[n]$.

c) Confirm your results using Matlab.

Exercise 5

Use a microphone correctly plugged to your computer and the following Matlab code (or an alternative audio recorder) in order to record on a WAV file the sound of your voice during 10 seconds. During this recording time you should utter different vowels, for example, a-e-i-o-u, as well as your name. This Matlab code sets the sampling frequency to 22050 Hertz and sets the sample resolution to 16 bits.

```
FS=22050; duration=10; NBITS=16;
r = audiorecorder(FS, NBITS, 1);
fprintf('Start speaking...\n');
record(r);     % speak into the microphone...
pause(duration); stop(r);
% p = play(r);   % listen to the complete recording
x = getaudiodata(r, 'single'); % get data normalized to +-1.0
x=x(FS:end); % avoid first second, may contain noise
fprintf('Stop speaking. Now playing...\n');
sound(x,FS);
fprintf('Stop playing.\n');
audiowrite(soundfile.wav', x, FS, 'BitsPerSample', NBITS);
```

After you create the WAV file (you only need to do this once), use the following Matlab code to read and play the recorded sound:

```
[x,FS,NBITS]=wavread('soundfile.wav'); % or
     [x,FS]=audioread('soundfile.wav');
sound(x,FS,NBITS); %NOTE: x values are in the range [-1, 1]
N=length(x);
samples=[0:N-1];
figure(1)
plot(samples/FS, x);
xlabel('Time (s)');
ylabel('Amplitude');
title('soundfile.wav');
```

Our objective now is to filter the recorded sound using different filters and to listen to the result. We will use the (incomplete) Matlab code `aeiou_name.m` as a baseline. In order to complete this Matlab command file, you should replace ‘`COMPLETEHERE`’ by the appropriate Matlab commands according to the following indications:

a) design an FIR “equiripple” filter (comand `firpm,` or `remez`), or order 126 (i.e., length 127) having passband between 300 Hz and 3200 Hz, and stopbands between 0 Hz e 200 Hz, and between 4000 Hz and the Nyquist frequency,

b) design a low-pass FIR filter using the window method (comand `fir1`) whose cut-off frequency is 2000 Hz,

c) using the filter designed in **b)** modify its impulse response vector in order to obtain a high-pass filter (i.e., without using again command `fir1`).

For each of the designed filters (one at a time!) first check, using the instructions already available in the supplied `.m` file, whether the resulting frequency response corresponds to the desired frequency response. Listen to each filtered signal and compare it to the original sound. Discuss what changes you notice as a result of each filtering.

Optional: You may use the Matlab filer design environment `fdatool` (or `filterDesigner` in the most recent Matlab version) in order to repeat each one of the above designs and check their characteristics.