Homework 5 GSI: Pulkit Grover
Due: Thursday, October 4, 2007, at 5pm

Reading OWN Section 7.1-7.3.

Practice Problems *(Suggestions.)* OWN 7.6, 7.7. **Problem 1** *(DTFT.)*

(a) OWN Problem 5.21, Part (e)

(b) OWN Problem 5.22, Part (a)

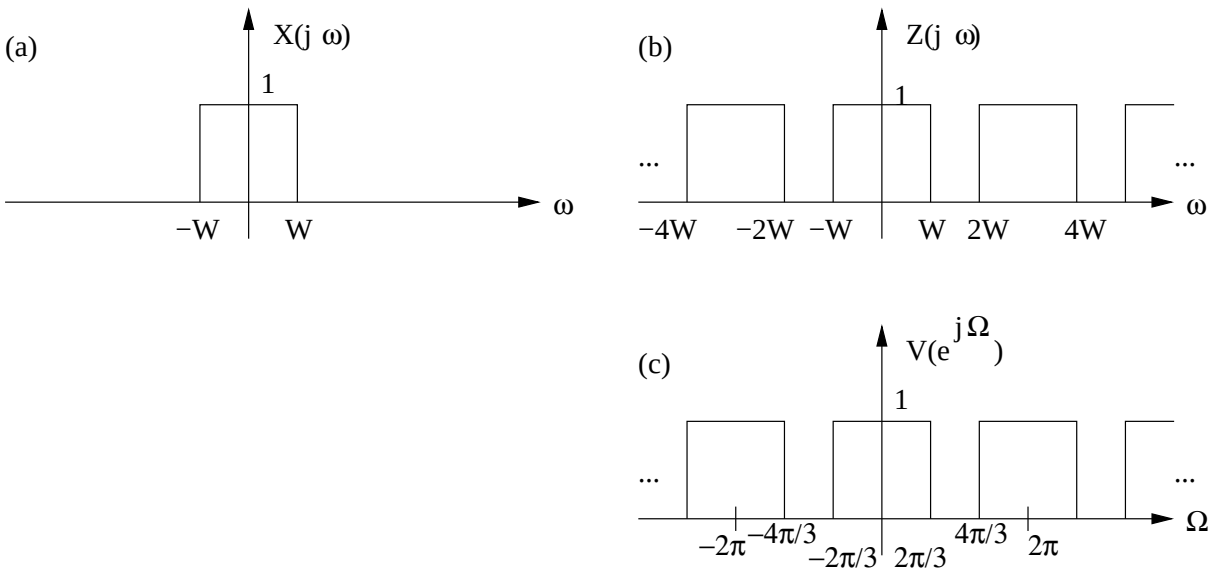
Problem 2 *(More DTFT.)*

(a) OWN Problem 5.26, Part (a)

(b) OWN Problem 5.50, Part (a)

Problem 3 *(Fourier Representations and their interconnections.)*

For each of the spectra in plots (a), (b), and (c), find and sketch the corresponding time-domain signal. *Remark.* Since we have both continuous-time and discrete-time signals in this problem, we distinguish the continuous-time and the discrete-time frequency variables by using ω in continuous time and Ω in discrete time. (See OWN p.535, last paragraph.)



Problem 4 *(Fourier via Matlab.)*

The Discrete Time Fourier Series (DTFS), often called the *Discrete Fourier Transform (DFT)*, is an

extremely important tool in signal processing and communications. It is essentially the only one of the four Fourier transforms that can be computed *exactly* for any signal, and moreover, there exists a very efficient algorithm, called the *fast Fourier transform (FFT)*, to calculate the DFT. If you want to learn more about the DFT and its implementation, you should take EE 123 (Digital Signal Processing).

(a) The definition of the DFT (or DTFS) in our textbook is

$$X[k] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}kn} \quad \text{and} \quad x[n] = \sum_{k=0}^{N-1} x[n]e^{j\frac{2\pi}{N}kn}. \tag{1}$$

In MATLAB, the definition is

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}kn} \quad \text{and} \quad x[n] = \frac{1}{N}\sum_{k=0}^{N-1} x[n]e^{j\frac{2\pi}{N}kn}. \tag{2}$$

Apparently, people cannot agree as to where the factor $\frac{1}{N}$ should go. Another definition that is sometimes used is the so-called *unitary form of the DFT*, defined as

$$X[k] = \frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi k}{N}n} \quad \text{and} \quad x[n] = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} x[n]e^{j\frac{2\pi k}{N}n}. \tag{3}$$

This form of the DFT is particularly good to gain geometric insight, as we have seen in class.

The main trick is now to write both the (time domain) signal and its (frequency domain) spectrum as vectors of length N :

$$\mathbf{x} = \begin{pmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ \vdots \\ x[N-1] \end{pmatrix} \quad \text{and} \quad \mathbf{X} = \begin{pmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \\ \vdots \\ X[N-1] \end{pmatrix} \tag{4}$$

Write a matlab program that generates the Fourier matrix F of dimensions $N \times N$ such that $\mathbf{X} = F\mathbf{x}$. Do this for all of the three definitions above, so that you obtain three Fourier matrices, F_{OWN} (for the textbook version), F_M (for the Matlab version), and F_u (for the unitary form).

(b) Verify (paper-pencil) that the columns of the matrix F_u are orthonormal to each other (that is, their dot products are 0, and their lengths are one). Then, verify that the columns of F_M and F_{OWN}, respectively, are also orthogonal, but that their lengths are not one. Now what is the DFT really doing to your signal? This notion can be extended to the other Fourier transforms, but it is easiest to see in this case.

(c) Now, create a new m-file. Matlab's function for the DFT is called `'fft'` (Fast Fourier Transform). It computes the same transform, just faster.

Let's look at some DFTs of length 16 signals.

```
x1 = [1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0];
x2 = [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
x3 = sin(3*pi*(0:15)/8); x4 = exp(j*14*pi*(0:15)/16);
```

Compute the FFT of these signals using `'fft'` as well as the matrix multiplication you derived in part a. Be careful because Matlab starts indexing at 1 rather than 0. Verify that they give you the same answers and plot the absolute value of the FFTs next to the signals. Use the command `'subplot'` to put all the plots in one figure.

Do the FFTs of the signals (particularly signals 2, 3 and 4) make sense to you?

Problem 5 *(Fourier Transform via Matlab.)*

In Homework 1, Problem 8, you had to compute a continuous-time convolution (or at least an approximation thereof) via matlab. Here, we want to explore the same for the Fourier Transform.

We want to use the Matlab command `fft` to give an instructive plot of the Fourier transform of

$$x(t) \quad = \quad \cos(30t)sinc^2(t). \tag{5}$$

We will use the following matlab file to get started:

```
T = 10;
dt = 0.01;
t = [ -T : dt : T-dt];
x1 = cos(30*t).*sinc(t).^2;
figure(1); plot(t, x1);
```

To our eyes, this function looks nice and even-symmetric, and hence, should have a purely real-valued Fourier representation. So, let's go for it:

```
Xnotquite = fft(x1);
sum(imag(Xnotquite))
```

Clearly, the spectrum `Xnotquite` is *not* real-valued! The explanation is very simple: Matlab does not know where to put the time origin! More precisely, it simply puts it all the way in the left corner (i.e., the first element of the vector). From that perspective, our function is clearly no longer even-symmetric. To fix this problem, the (nearly trivial) shifting function fftshift repositions the signal such that the old middle point now comes to lie in the first element of the vector. Everything else is shifted in a cyclic manner, reflecting the fact that really, the FFT is thinking about a *periodic* discrete-time signal with one period given by the vector `x`. Hence, we continue by doing

```
x = fftshift(x1);
figure(2); plot(t, x);
X = fft(x);
plot(real(X));
sum(imag(X))
```

Now, we are happy: the spectrum is (up to numerical precision) real-valued. Now, explain (with justifications!) how to modify `X` to get the right values on the amplitude axis with respect to the original problem, i.e., finding the Fourier transform of $x(t)$. Then, annotate the frequency axis with actual frequencies. Again, justify your answer.

Problem 6 *(Sampling theorem.)*

OWN Problem 7.21, Parts (a), (b), (g)

Problem 7 *(Sampling.)*

OWN Problem 7.23, all parts.