Chapter 3, Representation of Discrete-Time Sequences (DFS, DFT): Problem Solutions

Problem 1

We have:

$$\begin{aligned} x[n] &= \frac{e^{j\phi}}{2}e^{j(2\pi/N)Ln} + \frac{e^{-j\phi}}{2}e^{-j(2\pi/N)Ln} \\ &= \frac{e^{j\phi}}{2}e^{j(2\pi/N)Ln} + \frac{e^{-j\phi}}{2}e^{-j(2\pi/N)Ln}e^{j(2\pi/N)Nn} \\ &= \frac{e^{j\phi}}{2}e^{j(2\pi/N)Ln} + \frac{e^{-j\phi}}{2}e^{j(2\pi/N)(N-L)n}. \end{aligned}$$

Therefore we can write in vector notation:

$$\mathbf{x} = \frac{e^{j\phi}}{2}\mathbf{w}^{(L)} + \frac{e^{-j\phi}}{2}\mathbf{w}^{(N-L)},$$

and the result follows from the linearity of the expansion formula

$$\begin{aligned} X[k] &= \langle \mathbf{w}^{(k)},\, \mathbf{x}\rangle \\ &= \left\langle \mathbf{w}^{(k)},\, \frac{e^{j\phi}}{2}\mathbf{w}^{(L)} + \frac{e^{-j\phi}}{2}\mathbf{w}^{(N-L)} \right\rangle = \frac{e^{j\phi}}{2}\langle \mathbf{w}^{(k)},\, \mathbf{w}^{(L)}\rangle + \frac{e^{-j\phi}}{2}\langle \mathbf{w}^{(k)},\, \mathbf{w}^{(N-L)}\rangle \\ &= \begin{cases} \frac{N}{2}e^{j\phi} & \text{if } k = L \\ \frac{N}{2}e^{-j\phi} & \text{if } k = N-L \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Problem 2

By simple visual inspection we can see that $x[n] = a[n] + b[n] + c[n]$ with

$$\begin{aligned} a[n] &= 2 \\ b[n] &= 3\cos(3(2\pi/64)n) \\ c[n] &= \sin(7(2\pi/64)n) = -\cos(7(2\pi/64)n + \pi/2). \end{aligned}$$

The DFT coefficients are $X[k] = A[k] + B[k] + C[k]$, with

$$\begin{aligned} A[k] &= 2N\delta[k] \\ B[k] &= (3N/2)\delta[k-3] + (3N/2)\delta[k-61] \\ C[k] &= -(jN/2)\delta[k-7] + (jN/2)\delta[k-57] \end{aligned}$$

and $N = 64$, so that in the end we have

$$\begin{aligned} X[0] &= 128 \\ X[3] &= 96 \\ X[7] &= -32j \\ X[57] &= 32j \\ X[61] &= 96 \end{aligned}$$

and $X[k] = 0$ for all the other values of k.

Problem 3

Let $f[n] = \text{DFT}\{x[n]\}$. We have:

$$\begin{aligned} y[n] &= \sum_{k=0}^{N-1} f[k] e^{-j\frac{2\pi}{N}nk} \\ &= \sum_{k=0}^{N-1} \left\{ \sum_{i=0}^{N-1} x[i] e^{-j\frac{2\pi}{N}ik} \right\} e^{-j\frac{2\pi}{N}nk} \\ &= \sum_{i=0}^{N-1} x[i] \sum_{k=0}^{N-1} e^{-j\frac{2\pi}{N}(i+n)k}. \end{aligned}$$

Now,

$$\sum_{k=0}^{N-1} e^{-j\frac{2\pi}{N}(i+n)k} = \begin{cases} N & \text{for } (i+n) = 0, N, 2N, 3N, \ldots \\ 0 & \text{otherwise} \end{cases} = N\delta[(i+n) \mod N]$$

so that

$$\begin{aligned} y[n] &= \sum_{i=0}^{N-1} x[i] N \delta[(i+n) \mod N] \\ &= \begin{cases} Nx[0] & \text{for } n = 0 \\ Nx[N-n] & \text{otherwise.} \end{cases} \end{aligned}$$

In other words, if $\mathbf{x} = [1\ 2\ 3\ 4\ 5]^T$ then $\text{DFT}\{\text{DFT}\{\mathbf{x}\}\} = 5[1\ 5\ 4\ 3\ 2]^T = [5\ 25\ 20\ 15\ 10]^T$.

Problem 4

[Implementing DFT in MATLAB]
Some comments:

- In the lectures all vectors are column vectors by default. In MATLAB, however, a command like `[1:N]` will create a row vector. Our `myDFT` function will therefore work on row vectors.
- If the length of the input `x` is less than the given length `N`, we need to do *z*ero-padding. This means that we concatenate the sequence `x` and `N-length(x)` zeros.
- Since all indices in MATLAB start at 1, some care needs to be taken when creating the matrix `W`.

An example of a complete function:

```
function X=myDFT(x,N)
if length(x)>N    % declare an error
    fprintf(' error: Length of x cannot exceed N\n');
else
    x=[x,zeros(1,N-length(x))];    % do zero-padding
    X=zeros(1,N);
    W=zeros(N,N);
    for n=1:N
```

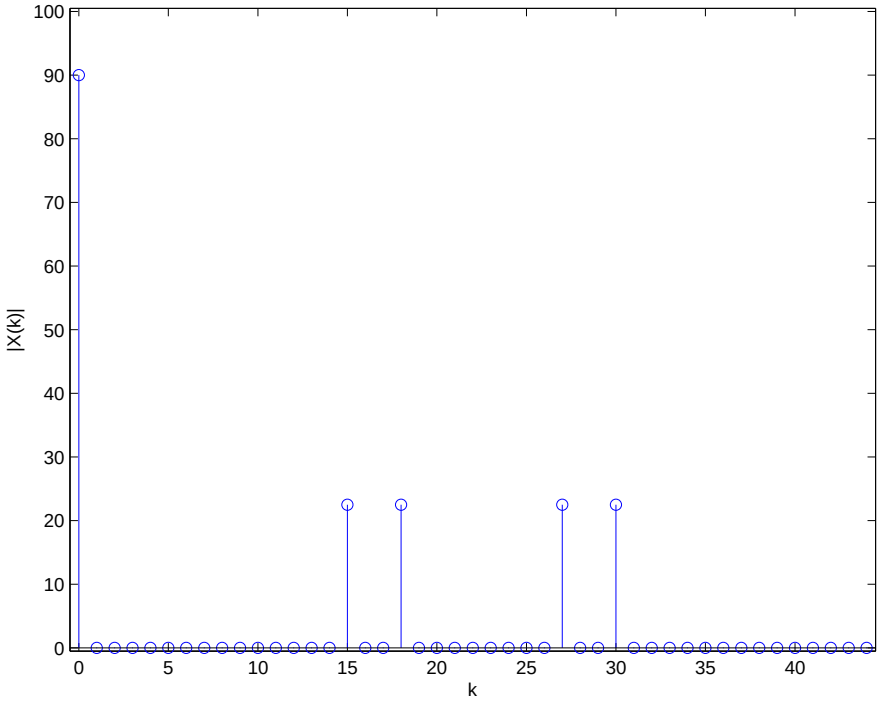

Figure 1: Problem 4.

```
        for m=1:N
            W(n,m)=exp(-i*2*pi*(n-1)*(m-1)/N);
        end
    end
    X=(W*x.').';

    % create the plot, on the x-axis put the values of k, not the index
    % in the matrix
    stem([0:N-1],abs(X));
    xlabel('k'); % add labels
    ylabel('|X(k)|');
    xlim([-.5 N-.5]); % make it look better by creating
    ylim(ylim+[-.5 .5]); % space around the data points.
end
```

The plot in Figure 1 of the absolute value of the DFT of our sequence $x[n]$ can be produced by

```
>> n = [0:44];
>> x = 2 + sin(2*pi/3*n) + cos(4*pi/5*n);
>> X = myDFT(x,length(x));
```

Problem 5

[DFT with Different Lengths]

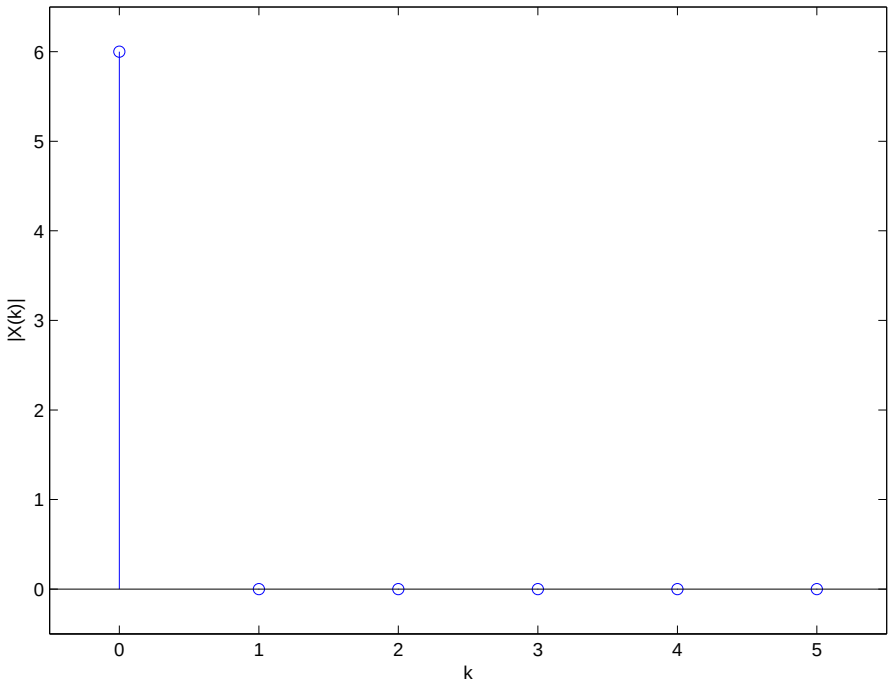

Figure 2: Problem 5(a).

(a) `x = ones(1,6), myDFT(x,6);` See Figure 2.

(b) `x = ones(1,6), myDFT(x,12);` See Figure 3.

(c) The function can for example be completed as follows:

```
function y=rcshift(x,k)
% this function makes a circular shift of length k to the right in vector x
n=length(x);
y=zeros(1,n);
y(1:k)=x(n-k+1:n);
y(1+k:n)=x(1:n-k);
```

For the circular shift of the sequence `t` we can do

```
>> t = sin([1:10]); % create sequence t
>> subplot(2,1,1), stem([0:9],t); % plot t
>> % add labels to the axes and add some space around the data points.
>> xlabel('k'), ylabel('t[k]'), xlim([-0.5 9.5]), ylim(ylim+[-.5 .5]);
>> subplot(2,1,2), stem([0:9],rcshift(t,3)); % plot circular shifted sequence
>> % and add labels, etc
>> xlabel('k'), ylabel('t[(k-3) mod 10]'), xlim([-0.5 9.5]), ylim(ylim+[-.5 .5]);
```

This gives us Figure 4.

(d) For example:

```
function z=cir_conv(x,y,N)
```

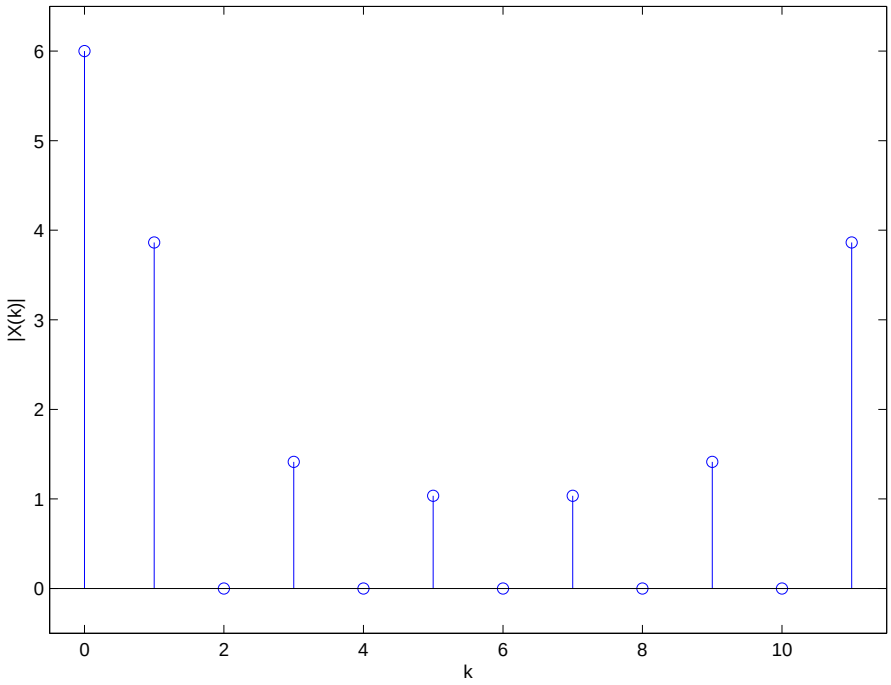

Figure 3: Problem 5(b).

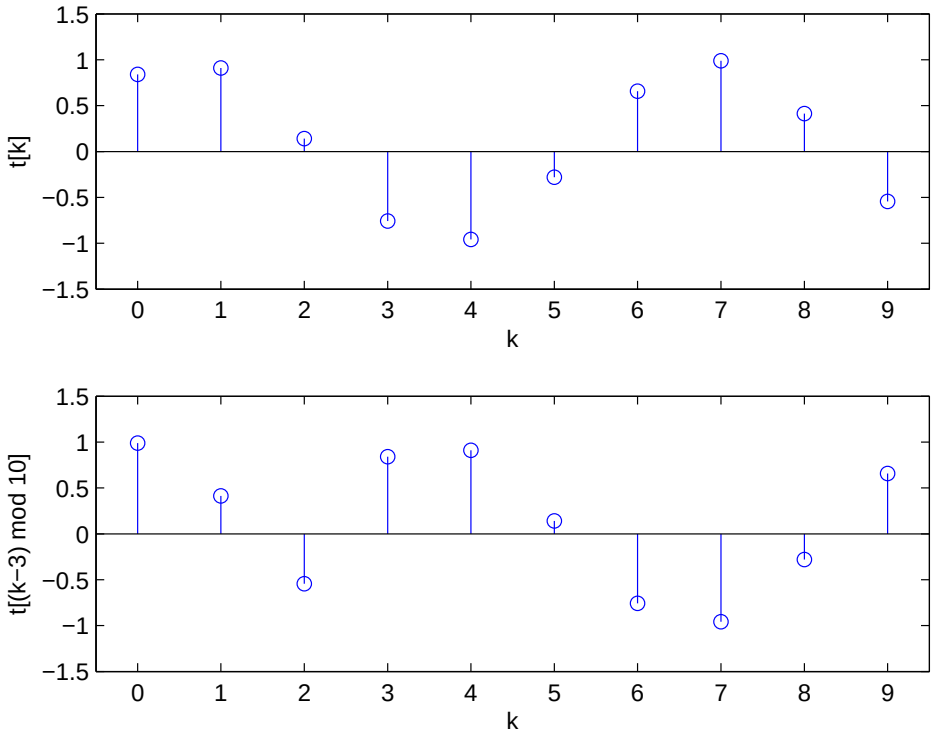

Figure 4: Problem 5(c).

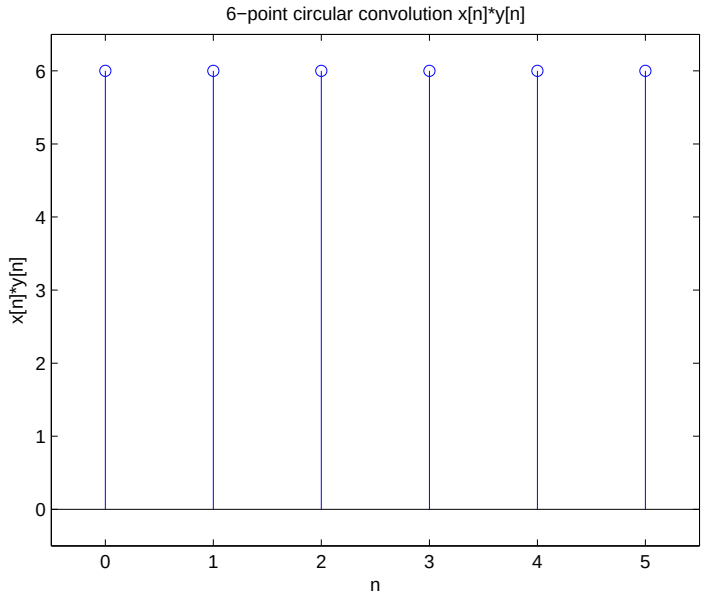

Figure 5: Problem 5(e).

```
% This function comutes the circular convolution of vectors x and y with
% the same lengths, and saves the result in the vector z of length N.
if length(x)~=length(y) | length(x)>N   % check the length compatibility
    fprintf(' error: error about the lengths\n');
else
    n=length(y);
    x=[x,zeros(1,N-n)];  % make x & y of length N
    y=[y,zeros(1,N-n)];
    y1=zeros(1,N);       % produce y1(n)=y((0-n) mod N)
    y1(1)=y(1);
    y1(2:N)=y(N:-1:2);
    for m=1:N  % each element of the result is produced in one iteration of the loop
        y2=rcshift(y1,m-1);
        z(m)=x*y2.';  % summation of the element-wise multiplication of 2 vectors
    end
end
```

(e) Figure 5 produced by

```
>> x = ones(1,6); y=x;
>> z = cir_conv(x,y,6);
>> stem([0:5],z);
>> xlabel('n');
>> xlabel('n'); ylabel('x[n]*y[n]'); title('6-point circular convolution x[n]*y[n]');
>> xlim([-.5 5.5]); ylim(ylim+[-.5 .5]);
```

(f) Figure 6 produced by

```
>> x = ones(1,6); y=x;
>> z = cir_conv(x,y,12);
>> stem([0:11],z);
```

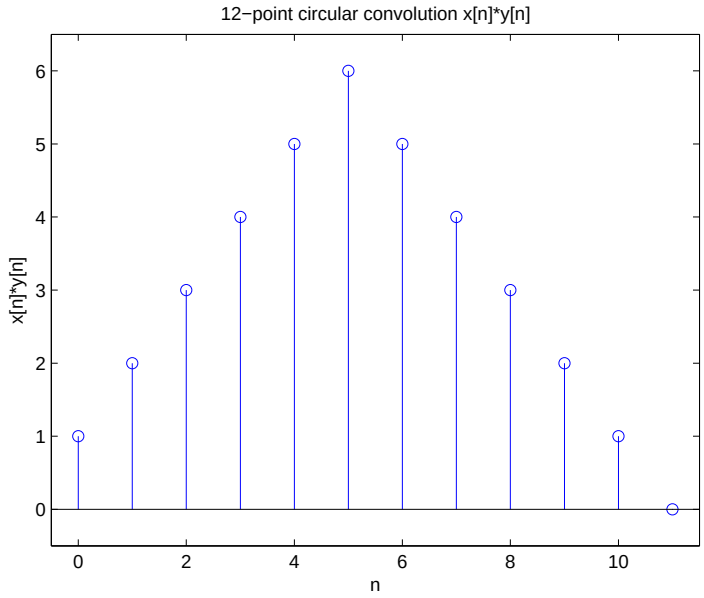

Figure 6: Problem 5(f).

```
>> xlabel('n');
>> xlabel('n'); ylabel('x[n]*y[n]'); title('12-point circular convolution x[n]*y[n]')
>> xlim([-.5 11.5]); ylim(ylim+[-.5 .5]);
```

(g) We see that the two are significantly different. What is important to see is that the 12-point convolution of part (f) is exactly the linear convolution of the sequences $x[n]$ and $y[n]$ if we extend them beyond N. For the 6-point circular convolution this is not the case.

If we choose the length of our circular convolution properly, we can compute a linear convolution by means of a circular convolution.

(h) They are the same. This can be checked by

```
>> Z = myDFT(z,12); ylabel('|Z(k)|');
>> figure
>> X = myDFT(x,12);
>> Y = myDFT(y,12);
>> Zmult = X.*Y;
>> stem([0:11],abs(Zmult))
```

We give the (first) plot in Figure 7. Similarly to the normal Fourier transform, a circular convolution in the time-domain is equivalent to a multiplication in the Fourier domain.

Extending the idea from part (g) we see that we can compute a linear convolution by performing appropriate (inverse) Discrete Fourier transforms. We will see later in class that there are very efficient algorithms to perform DFT. This means that we have found an efficient way to perform a linear convolution.

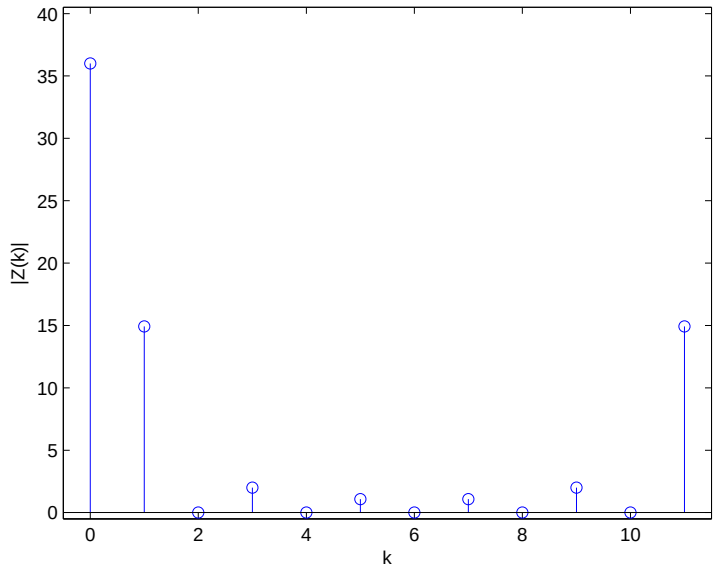

Figure 7: Problem 5(h).

Problem 6

We are looking for N such that for all n, $\tilde{x}[n+N] = \tilde{x}[n]$. This means that we need to find N for which

$$2 + \sin\left(\frac{2\pi}{3}(n+N)\right) + \cos\left(\frac{4\pi}{5}(n+N)\right) = 2 + \sin\left(\frac{2\pi}{3}n\right) + \cos\left(\frac{4\pi}{5}n\right). \tag{1}$$

We have that $\sin\left(\frac{2\pi}{3}(n+N_1)\right) = \sin\left(\frac{2\pi}{3}n\right)$ for $N_1 = 3$ and $\sin\left(\frac{4\pi}{5}(n+N_2)\right) = \sin\left(\frac{4\pi}{5}n\right)$ for $N_2 = 5$. If we take N equal to the least common multiple of N_1 and N_2 we satisfy (1). Hence $N = 15$.