Master 1 STLC

Practice # 4 : Discrete Fourier Transform (DFT)

1. Objectives:

- Specification of signal parameters.
- Frequency components of a noisy signal finding using the DFT.
- Comparisons between the DFT and the FFT (Fast Fourier Transform) algorithm.
- Calculation of the inverse TFD (IDFT) and inverse FFT.

2. Theoretical review

Mathematically the Fourier transform is defined on continuous functions from $-\infty$ to $+\infty$. The vast majority of signals are digital, such as the discrete Fourier transform (TFD) over a finite time interval corresponding to *N* samples. Obviously, when *N* becomes very large we can think that we are approaching the continuous case but we must keep in mind that the DFT actually assumes that the signal is periodic period *N*. DFT is then an essential tool in the field of digital signal processing. The basic basis is the Fast Fourier Transform (FFT), a method of calculating the DFT with a reduced execution time. Several routines containing the Z frequency response, spectral and spectral analysis, the design of some filters (e.g.: RIF) as well as the implementation of certain functions require the calculation of the FFT. For an input sequence $\tilde{x}(n)$ and its IDFT $\tilde{X}(k)$ which has an adequate interpretation as the samples of the unit circle spaced equally at the angle of a T.Z of a period of $\tilde{x}(n)$ the following two functions implement the TFD and the TFDI respectively.

$$\tilde{X}(k) = \begin{cases} \sum_{n=0}^{N-1} x(n) W_N^{kn} & 0 \le k \le N-1 \\ 0 & ailleurs \end{cases} \quad (1)$$

$$\tilde{x}(n) = \begin{cases} \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kn} & 0 \le n \le N-1 \\ 0 & ailleurs \end{cases} \quad (2)$$

Où $W_N = e^{-j\frac{2\pi}{N}}$

The FFT and IFFT algorithms aim to have faster calculations compared to equations (1) and (2) respectively. As an example, we present below the algorithm of Cooley-Tukey published in 1965 given by (see the course of DSP).

1- $\tilde{X}_p(Qs+r) = \sum_{p=0}^{P-1} \tilde{y}_r(p) W_P^{-sp}$

2- $\tilde{y}_r(p) = W_N^{-rp} \tilde{X}_p(r),\ 0 \le p \le P-1$

3- $\tilde{X}_p(r) = \sum_{q=0}^{Q-1} \tilde{x}_p(q) W_Q^{-rq}$

4- $\tilde{x}_p(q) = \tilde{x}(Pq+p),\quad 0 \le q \le Q-1,\ 0 \le p \le P-1$

3. Manipulations :

The Matlab environment provides the “FFT” and “IFFT” functions to determine the TFD and its TFDI respectively. In this case, the series indices in equations (1) and (2) must be started from "1" to "*N*" instead of "0" to "*N*-1".

Manip1 : Temporal representation of noisy signal :

- Specify the signal parameters with a sample rate f_s=1KHz of 1.5 second duration.

```
Fs=1000;            % Sampling frequency
T=1/Fs;             % Sampling period
L=1500;             % Length of signal
t=(0:L-1)*T;        % Time vector
```

- As an example, represent a signal containing a sum of two sine waves with frequencies and amplitudes (50Hz, 0.7) and (120Hz, 1).

```
S=0.7*sin(2*pi*50*t)+sin(2*pi*120*t);
```

- Contaminate (deform) this signal with an additive white Gaussian noise with a variance, $\sigma^2 = 4$.

```
X=S+2*randn(size(t));
```

- Plot the noisy signal in the time domain. As expected, it is difficult to identify the frequency components via direct observation on the resulting *x*(*t*) signal.

```
plot(1000*t(1:50),X(1:50))
title('Signal Corrupted with Zero-Mean Random Noise')
xlabel('t (milliseconds)')
ylabel('X(t)')
```

Manip2 : Frequency representation of a noisy signal using the FFT:

- Calculate the FFT of the signal $x(n)$:

```
Y=fft(X);
```

- Calculate the two bilateral spectra P_2. Calculate the unilateral spectrum P_1 from P_2 and the L even values of the signal.

```
P2=abs(Y/L);
P1=P2(1:L/2+1);
P1(2:end-1)=2*P1(2:end-1);
```

- Define the frequency domain f and plot the unilateral amplitude spectrum P_1. The amplitudes found are not equal to 0.7 and 1, this is due to the additive noise. Regularly, longer signals with a high L values produce a better frequency approximation.

```
f=Fs*(0:(L/2))/L;
plot(f,P1)
title('Single-Sided Amplitude Spectrum of X(t)')
xlabel('f (Hz)')
ylabel('|P1(f)|')
```

- Now, take the Fourier transform of the original signal (non-deformated signal) and then recover the exact amplitudes of 0.7 and 1.

```
Y=fft(S);
P2=abs(Y/L);
P1=P2(1:L/2+1);
P1(2:end-1)=2*P1(2:end-1);

plot(f,P1)
title('Single-Sided Amplitude Spectrum of S(t)')
xlabel('f (Hz)')
ylabel('|P1(f)|')
```

- Give your remarks.

Manip3 : Frequency representation of a noisy signal using the DFT:

- Repeat Manip 2 using equation (1) with the following Matlab code:

```
for k=0:L-1;
    som=0;
    for n=0:L-1;
        som=som+X(n+1)*exp(-j*2*pi*k*n/L);
    end;
    Y(k+1)=som;
end;
```

- Compare the results of the DFT et de la FFT

Manip4 : Calcul de la DFT, de l'amplitude et de la phase d'un signal:

- Calculate the amplitude and phase of the signal that contains a sum of two sinusoids.

```
t=0:1/100:10-1/100;                     % Time vector
x=sin(2*pi*15*t) + sin(2*pi*40*t);      % Signal
y=fft(x);                               % Compute DFT of x
m=abs(y);                               % Magnitude
p=unwrap(angle(y));                     % Phase
```

- To trace amplitude and phase, write the following commands and see the results.

f=(0:length(y)-1)*100/length(y); % Frequency vector

```
subplot(2,1,1)
plot(f,m)
title('Magnitude')
ax = gca;
ax.XTick = [15 40 60 85];

subplot(2,1,2)
plot(f,p*180/pi)
title('Phase')
ax = gca;
ax.XTick = [15 40 60 85];
```

Manip5 : The DFT with a specified number *n* :

- Write the following Matlab code and note the results obtained

```
N=512;
y=fft(x,N);
m=abs(y);
p=unwrap(angle(y));
f=(0:length(y)-1)*100/length(y);

subplot(2,1,1)
plot(f,m)
title('Magnitude')
ax=gca;
ax.XTick = [15 40 60 85];

subplot(2,1,2)
plot(f,p*180/pi)
title('Phase')
ax=gca;
ax.XTick = [15 40 60 85];
```

In this case, the FFT completes the input sequence with zeros if it is shorter at *N* and truncates the sequence if it is longer at N. If *N* is not given, the FFT defaults to the size of the input sequence. The FFT run time depends on the size N. The resulting FFT amplitude is A*N/2, where A is the amplitude of the original signal and N is the number of FFT points. This is just if and only if $N \geq$ "the size of the sequence". Otherwise, the amplitude of the FFT is less than the amplitude of the original signal.

Manip6 : Inverse FFT calculation:

The IFFT function of the Matlab code also accepts an input sequence (the desired number of points for the transformation).

- Find the IFFT from the following example.

```
t=0:1/255:1;
x=sin(2*pi*120*t);
y=real(ifft(fft(x)));

figure
plot(t,x-y)
```

The original sequence is x and the reconstructed sequence becomes identical with minimal error.

- Now use the inverse TFD given by equation (2) and compare the gaits found.

```
% DFT inverse
L=length(t);
for k=0:L-1;
    som=0;
    for n=0:L-1;
        som=som+x(n+1)*exp(-j*2*pi*k*n/L);
    end;
    y(k+1)=som;
end;

for n=0:L-1;
    som=0;
    for k=0:L-1;
        som=som+y(k+1)*exp(j*2*pi*k*n/L)/L;
    end;
    xr(n+1)=som;
end;

figure(2);
plot(t,x-xr);
```