The Discrete Fourier Transform

DSP Lab 09

Table of contents

1 Objectives

- Understand the basic principles of the Discrete Fourier Transform (DFT)
- Use the fft() and ifft() functions in MATLAB to perform DFTs and inverse DFTs on signals
- Understand the relationship between the time domain and frequency domain representations of signals

2 The Discrete Fourier Transform (DFT)

The Discrete Fourier Transform (DFT) is a powerful tool for analyzing and manipulating signals in a wide range of applications, from audio and image processing to telecommunications

and financial analysis. In this lab, we will learn how to use the `fft()` and `ifft()` functions in MATLAB to perform DFTs and inverse DFTs on signals.

The Discrete Fourier Transform (DFT) is a mathematical operation that decomposes a signal into its constituent frequencies.

Given a finite sequence of numbers $x[n] = \{x[0], x[1], x[2], ..., x[N-1]\}$, the DFT is defined as:

$$X_k = \sum_{n=0}^{N-1} x[n] \cdot e^{-j \cdot 2\pi \frac{k}{N} n}$$

where $k = 0, 1, 2, ..., N-1$.

The DFT has many useful properties, including the ability to recover the original signal from its frequency domain representation and the ability to efficiently perform various operations on signals in the frequency domain.

The original signal can be re-composed as a **sum of complex exponentials**:

$$x[n] = \sum_{k=0}^{N-1} X_k \cdot e^{j \cdot 2\pi \frac{k}{N} n}$$

Compared to the Fourier series of continuous signals:

- the fundamental frequency is $f_0 = 1/N$
- there are only N terms, with frequencies $k \cdot f_0$:
  - $0, f_0, 2f_0, ...(N-1)f_0$
- there are only N distinct coefficients X_k
- the N coefficients c_k can be chosen like $-\frac{N}{2} < k \leq \frac{N}{2}$ => the frequencies span the range $-1/2...1/2$

$$-\frac{1}{2} < f_k \leq \frac{1}{2}$$
$$-\pi < \omega_k \leq \pi$$

2.1 Basic properties of the DFT coefficients

1. Signal is **discrete** –> coefficients are **periodic** with period N

$$X_{k+N} = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-j2\pi(k+N)n/N} = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}$$

2. If signal is real $x[n] \in \mathbb{R}$, the coefficients are **even**:

- $X_k^* = X_{-k}$
- $|X_k| = |X_{-k}|$
- $\angle X_k = \angle D_{-k}$

- Together with periodicity:
  - $|X_k| = |X_{-k}| = |X_{N-k}|$
  - $\angle X_k = -\angle X_{-k} = -\angle X_{N-k}$

2.2 Expressing as sum of sinusoids

The DFT indicates that a discrete, periodical signals can always be written as a sum of sinusoidal signals.

- If N is odd:

$$x[n] = \frac{1}{N} X_0 + \frac{1}{N} \sum_{k=0}^{(N-1)/2} 2|X_k| \cos(2\pi k/N n + \angle X_k)$$

- If N is even:

$$x[n] = \frac{1}{N} X_0 + \frac{1}{N} \sum_{k=0}^{(N-2)/2} 2|X_k| \cos(2\pi k/N n + \angle X_k) + \frac{1}{N} X_{N/2} \cos(n\pi)$$

- Signal = DC value + a finite sum of sinusoids with frequencies kf_0
  - $|X_k|$ give the amplitudes (x 2)
  - $\angle X_k$ give the phases

3 Matlab functions

In Matlab, the DFT coefficients are computed with the function `fft()`. The inverse DFT is computed with `ifft()`.

Example:

```
x = [0 1 2 3 4 5 6];
S = fft(x);      % Compute the DFT
x2 = ifft(x)     % Compute back the signal
```

The DFT coefficients, like any Fourier transform, are **complex numbers** in general. Their modulus and phase can be obtained with `abs()` and `angle()`.

```
% Plot the modulus and phase of the Fourier coefficients
S_mod = abs(S)
S_phase = angle(S)
plot(S_mod)
figure
plot(S_phase)
```

The coefficients are returned as a vector $[X_0, X_1, ... X_{N-1}]$. They can be rearranged in the order $[X_{-N/2+1}, ... X_0, ... X_{N/2-1}]$ with the function `fftshift()`:

```
figure
plot(fftshift(S_mod)
figure
plot(fftshift(S_phase))
```

3.1 Matlab subplots

A figure in Matlab can be split into multiple parts with `subplot(a, b, c)`. The function takes 3 arguments:

- a = number of rows of the split
- b = number of columns of the split
- c = the current part we are in.

Example:

```
figure                % Make new figure window
subplot(2, 1, 1)      % Split in 2 rows, 1 column. We are now in part 1 of the split
plot(S_mod)                % Plot is displayed in the first part
subplot(2, 1, 2)      % Use same split in 2 rows, 1 column. But we move now to part 2 of t
plot(S_mod)                % Plot is displayed in the second part
```

4 Exercises

1. In this first exercise, we will use the `fft()` function to compute the Discrete Fourier Transform (DFT) of a signal.

   a. Generate a 100 samples long signal `x` defined as $x[n] = 0.7\cos(2\pi f_1 n) + 1.2\sin(2\pi f_2 n)$, with $f_1 = 0.05$ and $f_2 = 0.1$.

   b. Plot the signal in the top third of a figure (use `subplot()`).

c. Compute the DFT coefficients with `fft()` and plot their magnitude in the middle third, and their phase in the lower third.

d. Explain the relationship between the time domain and frequency domain representations of the signal.

2. Repeat Exercise 1 for a signal length of 93 samples. Why do additional frequency components appear in the spectrum?

3. Repeat the plot but do the FFT in N=1000 points (use `fft(x, N)`). What changes?

4. Repeat exercise 1 for the ECG signal loaded from the file `ECG_Signal.mat`. What is the dominant frequency here? Why?

   Find the average heartbeat rate (in beats per minute), as follows:

   - locate the highest frequency component in the spectrum (ignore the DC component)
   - identify the frequency f associated with its
   - knowing that the ECG signal was sampled with $F_S = 300$Hz, $f = \frac{F}{F_S}$, compute the analog frequency F in Hz
   - translate F in beats-per-minute

5. Repeat Exercise 1 for:

   a) a signal x containing the first 10 samples of a square wave with period 8, i.e. $x =$ [1, 1, 1, 1, -1, -1, -1, -1].

   b) a constant signal $x = 7$, 100 samples long

   c) an impulse of height 5: $x = [5, 0, 0, 0, 0, ...]$, 100 samples long in total

   d) A random signal of length 1000

   e) A piece from the middle of the song `Kalimba.mp3`, 1024 samples long

6. In this exercise, we will use the `ifft()` function to compute the inverse DFT of a signal.

   a. Create a vector X containing the first 3 coefficients of the DFT of a square wave with period 4.
   b. Compute the inverse DFT of X using the ifft() function and store the result in a variable x.
   c. Plot the time domain signal x using the stem() function.
   d. Explain the relationship between the time domain signal and its frequency domain representation.

7. In this exercise, we will use the fft() and ifft() functions to manipulate a signal in the frequency domain.

a. Create a vector `x` containing the first 10 samples of a square wave with period 4, i.e. x = [1, 1, 1, 1, -1, -1, -1, -1, 1, 1].
b. Compute the DFT of `x` using the fft() function and store the result in a variable `X`.
c. Set the first 5 coefficients of `X` to 0.
d. Compute the inverse DFT of X using the ifft() function and store the result in a variable `y`.
e. Plot the time domain signals `x` and `y` using the `stem()` function.
f. Explain how the manipulation of the frequency domain representation of the signal affected the time domain signal.

8. Generate a 39 samples long **triangular** signal `x` defined as:

   - first 10 samples are zeros
   - next, `x` increases linearly from `x(10) = 0` up to `x(19) = 4`, then decreases linearly to `x(29) = 0`.
   - last 10 samples are 0

   a. Plot the signal in the top third of a figure, the magnitude of the DFT coefficients in the middle third, and their phase in the lower third.

   b. What is the amplitude of the **third harmonic component** in the signal's spectrum?

   c. Concatenate 50 zeros at the end of the signal and redo the exercise. What do you observe?

9. Generate two 10-long random signals `x` and `y`.

   a. Perform **linear convolution** with `conv()`.

   b. Perform **circular convolution** via the frequency domain, using `fft()` and `ifft()`.

   c. Perform **linear convolution** via the frequency domain using the `fft` in N points, with N larger then 19.

   d. Which method of linear convolution is is faster, `conv()` or via `fft()`? Use long signals (e.g. length 40000).

5 Final questions

1. How do you expect the amplitudes of the Fourier coefficients to be for:

   - a slow varying signal
   - a rapid varying signal