Digital Signal Processing - Lecture Notes - 3

E Yazıcı, SRH Uni Heidelberg

June 6, 2023

Introduction to the Discrete Fourier Transform

The Discrete Fourier Transform (DFT) is a way of figuring out which frequencies are present in a signal. By 'frequencies,' we mean how often a certain pattern repeats itself over time. You can think of the DFT like a detective, looking for clues about what's going on in a signal.

Consider a signal like a complex mystery. It's like a series of events or observations recorded over time. The DFT is like our detective's toolbox, helping us to make sense of that mystery. It allows us to see which patterns (frequencies) are present in our mystery (signal), and how strongly they're represented.

Imagine our signal is a stretch of ocean surface, with all kinds of waves of different sizes moving through it. Each wave is a different 'frequency': bigger waves change slowly, smaller waves change quickly.

When we perform a DFT, we're essentially measuring the heights of all the waves passing through a certain point. We're breaking down the complex, jumbled wave pattern into individual, recognizable waves. That's exactly what DFT does with signals: it breaks them down into individual frequencies.

Here's how it works in practice:

- We start with a sequence of numbers. This is our signal.

- We then multiply our signal by a series of sine and cosine waves of different frequencies. We can imagine this as a set of 'measuring sticks' that we're holding up to our signal to see which ones match.

- For each frequency, we add up all the numbers we get from the multiplication. This gives us a new number.

- We collect all these new numbers into a list. This is our DFT: a collection of measurements telling us how much each frequency is present in our signal.

The beauty of the DFT is that it allows us to move from the time-domain (looking at how the signal changes over time) to the frequency-domain (looking at which frequencies are present in the signal). This can be really helpful for understanding the signal better, and for processing the signal in fields like audio engineering, image processing, and telecommunications.

Fourier Analysis

The idea of decomposing a signal into sinusoidal elements, also known as sinusoidal components, is a cornerstone of signal processing and, more broadly, of Fourier analysis. At its core, this concept relies on the assumption that **any signal, no matter how complex, can be expressed as the sum of simple sinusoids (i.e., sine and cosine waves) of various frequencies, amplitudes, and phases**.

This decomposition is generally achieved through the Fourier Transform, a mathematical operation that transforms a signal from its original time or space domain into the frequency domain. The resulting frequency spectrum shows the constituent sinusoids and their corresponding amplitudes and phases.

To provide a mathematical description, let's start with a continuous time signal $x(t)$. Its Fourier Transform $X(f)$ is given by:

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} dt \tag{1}$$

In this expression, $e^{-j2\pi ft}$ is a complex exponential function, which we can understand as a rotating vector in the complex plane. As it rotates, it "probes" the signal at different frequency rates f. This integral, which is a continuous sum, essentially correlates the signal $x(t)$ with the complex exponential at each frequency f.

Now, the inverse Fourier Transform can be used to recover $x(t)$ from its Fourier Transform $X(f)$. The formula is:

$$x(t) = \int_{-\infty}^{\infty} X(f) e^{j2\pi ft} df \tag{2}$$

What this equation tells us is that we can recreate our signal $x(t)$ by adding up (i.e., integrating over) all the sinusoidal components represented by $X(f)$, where each sinusoid $e^{j2\pi ft}$ is weighted by its corresponding Fourier Transform coefficient. This is the essence of the decomposition of a signal into sinusoidal components.

Think about how you can create a rainbow by passing sunlight through a prism. The prism splits the white light into all the colors that make it up. This is because white light is actually a mix of different colors, and the prism allows us to see those individual colors.

Now, imagine if you could do the opposite. If you had lights of different colors and you mixed them all together in the right way, you could make white light.

This is similar to what the inverse Fourier transform equation is saying.

In the context of the equation, a signal (like a sound wave or radio wave) is like the white light. And the "colors" are like different frequencies (or pitches, in the case of sound) that make up the signal.

The Fourier transform is like the prism – it allows us to see what "colors" or frequencies are in the signal. Each frequency in the signal has a coefficient that tells us how "strong" or "bright" that part of the signal is.

So the inverse Fourier transform is telling us that we can recreate the original signal by adding up all these frequencies, where each frequency is weighted by its coefficient, just like we could recreate white light by adding up the right mix of different colors.

In essence, the equation is a mathematical way of saying that any signal can be broken down into (or built up from) a collection of simple sine waves of different frequencies.

This concept is at the heart of digital signal processing and is fundamental to technologies like MP3s and JPEGs, which compress audio and image data by selectively removing or reducing certain frequencies that are less important to our perception.

Discrete Signals and Fourier Transform:
This principle can be extended to discrete signals, as well. In the case of discrete signals, we use the Discrete Fourier Transform (DFT) instead of the Fourier Transform, and the integral is replaced by a sum. If $x[n]$ is our discrete signal, its DFT $X[k]$ is given by:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \tag{3}$$

And the signal $x[n]$ can be recovered from $X[k]$ using the inverse DFT:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j2\pi kn/N} \tag{4}$$

Here, N is the length of the signal, and n and k are the sample and frequency indices, respectively.

In the context of the Discrete Fourier Transform (DFT), the indices n and k are used in the following way:

- n is the index that is used to traverse the time-domain signal (the input signal to the DFT that you're analyzing). In other words, n is the discrete time index of your original signal.
- k is the index used to traverse the frequency-domain signal (the output of the DFT). In other words, k is the discrete frequency index of the Fourier transform of your signal.

So, n is used to go through your time-domain signal sample by sample, and k is used to go through the result of your DFT frequency by frequency.

Furthermore, N represents the total number of samples in your discrete-time signal (and also the number of samples in the DFT). Thus, n and k are simply indices used to traverse through your time-domain and frequency-domain signals, respectively, while N is the total number of samples.

To provide a simple example, let's assume we have a 4-point signal $x[n] = \{1, 2, 3, 4\}$. Here, $N = 4$ as we have four samples. The index n runs from 0 to 3, as it represents the time index for each sample in the signal. When we perform the DFT, the result is a sequence of complex numbers $X[k] = \{X[0], X[1], X[2], X[3]\}$ that gives us the frequency-domain representation of the signal. Here, the index k also runs from 0 to 3, representing the frequency index for each component in the transformed signal.

This fundamental concept of decomposing signals into sinusoidal components allows us to analyze and process signals in the frequency domain, which can provide insights and tools not immediately available in the time or space domain. For example, it is central to many applications of digital signal processing, such as filtering, compression, and feature extraction.

Time Domain and Frequency Domain

The concept of *time domain* and *frequency domain* are fundamental in understanding signal processing and systems.

The *time domain* is a view of a signal or a system, in which we observe how the signal or system changes over time. For instance, if we were to plot a signal that represents the pitch of a note changing over time, the resulting graph would be a time domain representation of that note. In the time domain, the independent variable is time, and the dependent variable is the quantity or quantities that are changing over time.

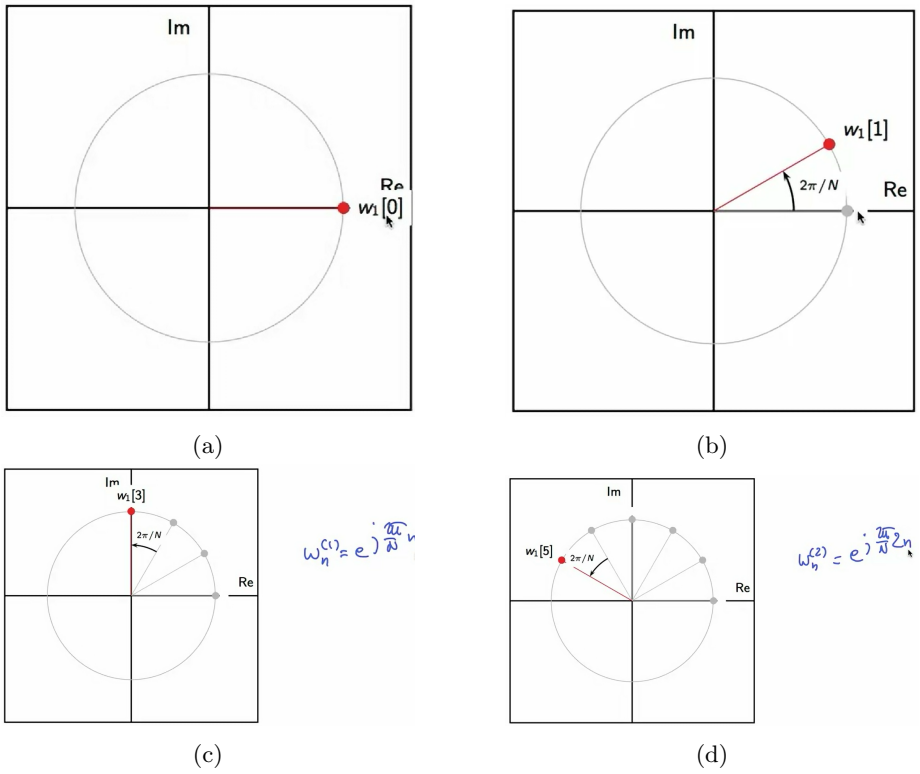

Figure 1: DFT Vector Basis Visualization on a complex plane

$$s(t) = A\sin(\omega t + \phi) \tag{5}$$

In this equation, $s(t)$ represents a sinusoidal signal, where A is the amplitude, ω is the angular frequency, and ϕ is the phase shift.

The *frequency domain* is another view of a signal or a system, in which we observe how much of the signal lies within each given frequency band over a range of frequencies. If we were to analyse the same note as before, but this time in the frequency domain, we would see a graph showing how much of the note's sound lies within each frequency. The frequency domain representation is a decomposition of the signal into a sum of sinusoidal components, each with a different frequency.

This can be mathematically represented using the Fourier Transform:

$$X(f) = \int_{-\infty}^{\infty} s(t)e^{-j2\pi ft}dt \tag{6}$$

In this equation, $X(f)$ represents the Fourier Transform of the signal $s(t)$, which gives us a frequency domain representation of the signal.

DFT Vector Basis

In the context of Fourier analysis, the phrase "Fourier analysis is simply a change of basis" refers to the interpretation of a function or a signal in terms of a different set of basis functions, specifically the set of sine and cosine functions.

Remember that a *basis* of a vector space is a set of vectors that are linearly independent and span the space. When we talk about a signal or function in the time domain, we are usually thinking of it as a point in an infinite-dimensional vector space where the basis vectors can be thought of as the delta functions centered at each point in time.

On the other hand, when we take the Fourier transform of a signal, we are representing that same signal in terms of a different set of basis functions, specifically the complex exponential functions. Each frequency in the Fourier transform corresponds to a specific complex exponential basis function.

So, the action of the Fourier transform is essentially just a *change of basis* from the delta functions in the time domain to the complex exponentials in the frequency domain.

This is why Fourier analysis is often described as a change of basis.

The Discrete Fourier Transform of a sequence of N complex numbers x_n is defined as follows:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{2\pi i}{N} nk}, \quad k = 0, \ldots, N-1 \tag{7}$$

The functions $\phi_k(t) = e^{\frac{2\pi i}{N} kt}$, $k = 0, \ldots, N-1$ form a basis for the space of complex-valued functions on the interval [0, 1]. The components of a function f in this basis are given by the DFT of the sequence $f(0), f(1/N), \ldots, f((N-1)/N)$. These basis functions are orthogonal with respect to the standard inner product on this function space:

$$\langle \phi_k, \phi_l \rangle = \int_0^1 \phi_k(t) \overline{\phi_l(t)} dt = \delta_{kl} \tag{8}$$

where δ_{kl} is the Kronecker delta, which is 1 if $k = l$ and 0 otherwise.

However, the basis functions ϕ_k are not orthonormal because

$\|\phi_k\| = \sqrt{\langle \phi_k, \phi_k \rangle} = \sqrt{1} = 1$ for all k.

To obtain an orthonormal basis, we can simply normalize each basis function by dividing it by its norm, which in this case is unnecessary as the norm equals to 1.

An interesting property of the DFT is that the inverse DFT can be expressed in a form very similar to the DFT itself. This is due to the orthogonality of the basis functions, and it makes the DFT a particularly easy transform to invert:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{\frac{2\pi i}{N} nk}, \quad n = 0, \ldots, N-1 \tag{9}$$

In other words, if we know the DFT X_k of a sequence x_n, we can easily reconstruct the original sequence.

* * *

In summary, any natural and man-made phenomena exhibit a periodic behavior: after a certain amount of time, called the period, these phenomena repeat exactly. The inverse of the period, that is, the number of repetitions per unit of time, is called the frequency. Sines and cosines, simple oscillatory functions, are natural candidates as the basic building blocks to represent more complicated oscillatory signals. This is the basic goal of Fourier analysis: to decompose a signal in terms of sines and cosines.

We have two kinds of Fourier tools, Fourier analysis and Fourier synthesis. Fourier analysis allows determining the weight of each periodic basic building block in a given signal; with the analysis, we move from the time domain to

the frequency domain. Fourier synthesis allows building signals with a "user-defined" frequency content: with this we move from the frequency domain to the time domain.

The time domain and the frequency domain are simply different ways of looking at the same signal. They contain the same information, just in different forms.

Example Problems on Discrete Fourier Transform

Sampling period: In digital signal processing, a continuous signal is usually converted into a discrete signal through a process known as sampling. The sampling period, usually denoted as T_s, refers to the time interval at which the continuous signal is sampled to obtain the discrete signal. In other words, it's the time difference between two consecutive samples in the discrete signal.

For instance, if you sample a continuous signal every second, your sampling period T_s is 1 second.

EXAMPLE 1:

We will find the 4-point DFT of a simple discrete-time sequence $x[n] = [1, 2, 3, 4]$.

First, let's recall the formula for the DFT:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j2\pi kn/N} \tag{10}$$

For a 4-point DFT, N = 4. The k values range from 0 to N-1 (or 0 to 3).

Step 1: Calculate $X[0]$

$$X[0] = x[0]\cdot e^{-j2\pi\cdot 0\cdot 0/4} + x[1]\cdot e^{-j2\pi\cdot 1\cdot 0/4} + x[2]\cdot e^{-j2\pi\cdot 2\cdot 0/4} + x[3]\cdot e^{-j2\pi\cdot 3\cdot 0/4} \tag{11}$$

Simplifying, we get $X[0] = x[0] + x[1] + x[2] + x[3] = 1 + 2 + 3 + 4 = 10$

Step 2: Calculate $X[1]$

$$X[1] = x[0]\cdot e^{-j2\pi\cdot 0\cdot 1/4} + x[1]\cdot e^{-j2\pi\cdot 1\cdot 1/4} + x[2]\cdot e^{-j2\pi\cdot 2\cdot 1/4} + x[3]\cdot e^{-j2\pi\cdot 3\cdot 1/4} \tag{12}$$

Step 3: Continue the same way for $X[2]$ and $X[3]$. The $e^{-j2\pi kn/N}$ part can be calculated using Euler's formula $e^{ix} = cos(x) + isin(x)$.

So, by calculating all these, we would obtain the DFT of the sequence.

Please note that for $X[1]$, $X[2]$, and $X[3]$ you would have complex numbers as results because of the exponential term. In this case, you would usually represent the DFT as magnitude (which is the absolute value of the complex number) and phase (which is the angle of the complex number). This gives you an idea of the 'strength' of the frequencies present in the signal (magnitude) and their phase offsets (phase).

In this example, we didn't specify a sampling period because we were working directly with a digital signal defined by the sequence $x[n] = [1, 2, 3, 4]$. This sequence was already in a digital format, so we directly applied the DFT formula to it.

However, let's talk about what the sampling period means in a real-world situation. The sampling period (often denoted as T) is the time interval at which an analog signal is sampled to convert it into a digital signal. It is the reciprocal of the sampling frequency.

The sampling frequency (often denoted as fs) is the number of samples taken per second. So if fs = 100 Hz, this means we're taking 100 samples of the signal per second, and thus our sampling period T would be 1/fs = 1/100 = 0.01 seconds.

In the context of our simple DFT example, if we said our sampling period was T = 0.01 seconds, then we're saying that each element in our sequence represents a sample taken every 0.01 seconds. So $x[0]$ would be the sample at time 0, $x[1]$ would be the sample at time 0.01 seconds, $x[2]$ would be the sample at time 0.02 seconds, and so on.

EXAMPLE 2:
Analog Signal (Continuous Time Domain): Suppose we have a simple sine wave signal $x(t) = sin(2\pi ft)$ where the frequency $f = 1$ Hz. This signal is continuous in time.

Sampling: To convert this analog signal into a digital signal, we sample it at regular time intervals, which is determined by the sampling rate f_s. Let's say we choose a sampling rate of $f_s = 10$ Hz. The sampling period T (time interval between samples) is the reciprocal of the sampling rate, $T = 1/f_s = 0.1$ s. Hence, we take samples every 0.1 seconds. If we do this for 1 second, we end up with 10 samples because our sampling rate is 10 Hz.

Digital Signal (Discrete Time Domain): Let's denote our samples as $x[n]$. For $n = 0, 1, 2, ..., 9$, we have $x[n] = sin(2\pi nT) = sin(2\pi n/10)$. Therefore, we have $x[n] = [0, 0.5878, 0.9511, 0.9511, 0.5878, 0, -0.5878, -0.9511, -0.9511, -0.5878]$.

Apply DFT: The Discrete Fourier Transform (DFT) takes a sequence of complex numbers (our sampled signal) in the time domain and transforms it to the frequency domain. Here's how we calculate the DFT $X[k]$ of our signal $x[n]$:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j2\pi kn/N}$$

where N is the number of samples (10 in our case), n is the time index (ranging from 0 to $N-1$), k is the frequency index (also ranging from 0 to $N-1$), and j is the imaginary unit.

We won't compute the full DFT here as it can be quite involved, but you'd do this for each k to get your frequency domain representation of the signal $X[k]$.

Frequency Domain: The result, $X[k]$, represents the frequency content of our original signal. Each k corresponds to a specific frequency, and the magnitude of $X[k]$ tells us how much of that frequency is present in our original signal. This is a simplified example but it captures the core steps in the process from an analog signal to a DFT. In real applications, the signals are usually more complex and the mathematics can be more involved.

Problem Set - 3

Problem 1

Consider a discrete time signal $x[n] = \sin(2\pi f_0 n T_s)$, where f_0 is the frequency of the signal, and T_s is the sampling period. Compute its Discrete Fourier Transform (DFT).

Solution:

The DFT of a sequence is calculated by summing the products of the sequence values and complex exponential terms. When you have a sinusoidal sequence such as $x[n] = \sin(2\pi f_0 n T_s)$, the result of the DFT will essentially indicate the frequency components present in that sequence.

For this specific sequence, the DFT will show components at frequencies $+f_0$ and $-f_0$. These correspond to the positive and negative frequency components of the sinusoid, respectively. If the frequency f_0 corresponds to a discrete frequency k_0 in the DFT (meaning $f_0 = k_0/N$ where N is the number of samples), these components will be seen at the indices $k = k_0$ and $k = N - k_0$ in the DFT result.

The DFT of a sequence $x[n]$ is given by:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N}$$

For a sinusoidal signal like $x[n] = \sin(2\pi f_0 n T_s)$, the DFT will result in frequency components at $\pm f_0$. More specifically, if the frequency f_0 is such that $f_0 = k_0/N$, then the non-zero components of the DFT will be at $k = k_0$ and $k = N - k_0$.

Problem 2

Compute the DFT of a sequence which is a sum of two sinusoids: $x[n] = \sin(2\pi f_1 n T_s) + \sin(2\pi f_2 n T_s)$, where f_1 and f_2 are the frequencies of the sinusoids, and T_s is the sampling period.

Solution:

Here, the sequence is a sum of two sinusoids with frequencies f_1 and f_2. Similar to the first problem, the DFT will reveal components at frequencies $+f_1$, $-f_1$, $+f_2$, and $-f_2$. If the frequencies f_1 and f_2 correspond to discrete frequencies k_1 and k_2 in the DFT (meaning $f_1 = k_1/N$ and $f_2 = k_2/N$), these components will be seen at the indices $k = k_1$, $k = N - k_1$, $k = k_2$, and $k = N - k_2$ in the DFT result.

The DFT of this signal will contain frequency components at $\pm f_1$ and $\pm f_2$. More specifically, if the frequencies f_1 and f_2 are such that $f_1 = k_1/N$ and $f_2 = k_2/N$, then the non-zero components of the DFT will be at $k = k_1, N - k_1$ and $k = k_2, N - k_2$.

Problem 3

Let's assume $f_0 = 2Hz$, $f_1 = 3Hz$, $f_2 = 4Hz$, and we have a total of $N = 100$ samples. Using these values:

Given a sinusoidal sequence $x[n] = \sin(2\pi \cdot 2 \cdot n \cdot T_s)$, calculate the DFT of $x[n]$.

Solution:
The DFT will show components at frequencies $+2Hz$ and $-2Hz$. These correspond to the positive and negative frequency components of the sinusoid, respectively. If the frequency $2Hz$ corresponds to a discrete frequency k_0 in the DFT (meaning $2Hz = \frac{k_0}{100}$), these components will be seen at the indices $k = k_0$ and $k = 100 - k_0$ in the DFT result.

Problem 4

Given a sequence $y[n] = \sin(2\pi \cdot 3 \cdot n \cdot T_s) + \sin(2\pi \cdot 4 \cdot n \cdot T_s)$, calculate the DFT of $y[n]$.

Solution:
Similar to the first problem, the DFT will reveal components at frequencies $+3Hz$, $-3Hz$, $+4Hz$, and $-4Hz$. If the frequencies $3Hz$ and $4Hz$ correspond to discrete frequencies k_1 and k_2 in the DFT (meaning $3Hz = \frac{k_1}{100}$ and $4Hz = \frac{k_2}{100}$), these components will be seen at the indices $k = k_1$, $k = 100 - k_1$, $k = k_2$, and $k = 100 - k_2$ in the DFT result.

Problem 5

Given a sequence $z[n] = \cos(2\pi \cdot 2 \cdot n \cdot T_s) + \cos(2\pi \cdot 3 \cdot n \cdot T_s)$, calculate the DFT of $z[n]$.

Solution:
The DFT of the sequence will reveal components at frequencies $+2Hz$, $-2Hz$, $+3Hz$, and $-3Hz$. If the frequencies $2Hz$ and $3Hz$ correspond to discrete frequencies k_1 and k_2 in the DFT (meaning $2Hz = \frac{k_1}{100}$ and $3Hz = \frac{k_2}{100}$), these components will be seen at the indices $k = k_1$, $k = 100 - k_1$, $k = k_2$, and $k = 100 - k_2$ in the DFT result.

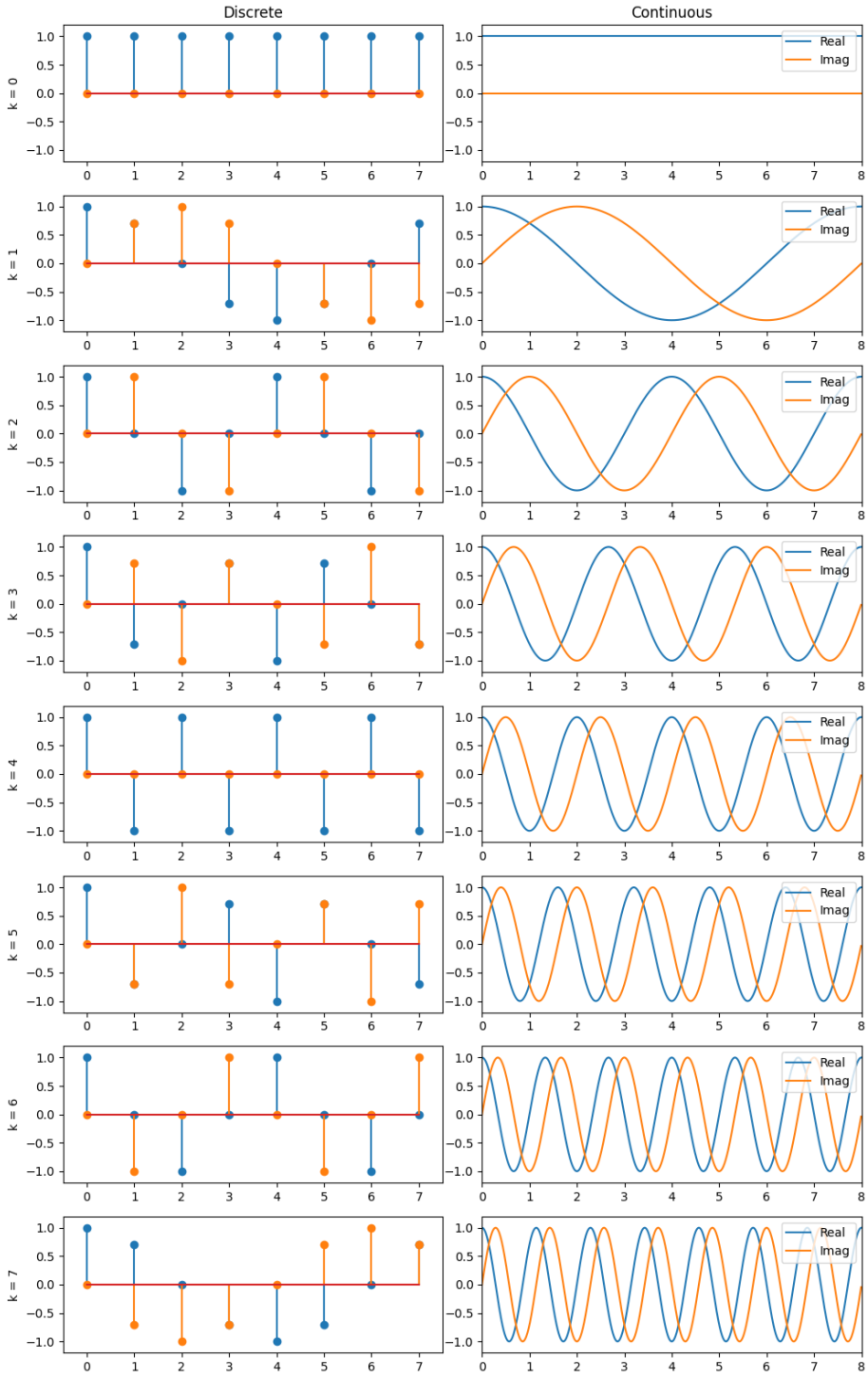

Figure 2: DFT Basis Vectors for N=8