Continuous and Discrete Signals

Jack Xin (Lecture) and J. Ernie Esser (Lab) [*]

Abstract

Class notes on signals and Fourier transform.

1 Continuous Time Signals and Transform

A continuous signal is a continuous function of time defined on the real line R denoted by $s(t)$, t is time. The signal can be complex valued. A *continuous signal* is called an *analog signal.* A stable (integrable) signal is one that satisfies:

$$\int_R |s(t)|\, dt < +\infty,$$

denoted by $s \in L^1(R)$.

1.1 Examples

Example 1: a stable signal, is the indicator function of the unit interval:

$$1_{[0,1]}(t) = \begin{cases} 1 & t \in [0,1] \\ 0 & \text{otherwise} \end{cases}$$

Analog sound signals are real oscillatory functions of time.
Example 2: sine wave (pure tone),

$$s(t) = A \sin(2\pi\, t/T + \phi), \tag{1.1}$$

where A is amplitude, T is period in seconds, ϕ is phase in radians. The reciprocal of the period T is frequency in Hertz (Hz) or cycles per second:

$$f = 1/T.$$

[*]Department of Mathematics, UCI, Irvine, CA 92617.

Angular frequency is:

$$\omega = 2\pi f.$$

The sine wave can be written as:

$$s(t) = A\,\sin(2\pi f t + \phi) = A\,\sin(\omega\, t + \phi).$$

Sound of a pure tone is a classical topic in hearing science [5]. The human audible frequency range is from 20 Hz to 20,000 Hz. Pure tones with frequencies lower than 200 Hz sound "dull", while higher frequency (above 2000 Hz) pure tones sound "bright". Ear is most sensitive in the range of 3000 to 5000 Hz. We'll hear pure tones played on Matlab later.

Example 3: a speech signal, see Fig. 1, is oscillatory with multiple frequencies. To analyze its energy distribution in frequencies, a decomposition into a linear combination of pure tones is necessary, which brings us to Fourier transform.

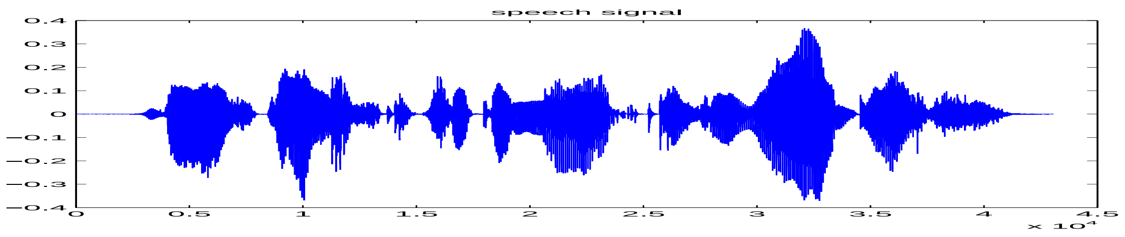

Figure 1: Illustration of a speech signal, oscillatory with multiple frequencies.

1.2 Fourier Transform

Fourier transform of a stable signal is:

$$\hat{s}(\nu) = \int_R s(t)\exp\{-2\pi i\nu t\}\,dt, \tag{1.2}$$

denoted by $F[s(t)] = \hat{s}(\nu)$.

A few elementary properties of Fourier transform are:

- delay:
$$F[s(t-t_0)] = \exp\{-2\pi i\nu t_0\}\,\hat{s}(\nu);$$
- modulation:
$$F[\exp\{2\pi i\nu_0 t\}\,s(t)] = \hat{s}(\nu-\nu_0),$$
- scaling:
$$F[s(c\,t)] = \frac{1}{|c|}\hat{s}(\nu/c),$$
- linearity:
$$F[c_1 s_1(t) + c_2 s_2(t)] = c_1\hat{s}_1(\nu) + c_2\hat{s}_2(\nu),$$
- symmetry ($*$ = complex conjugate):
$$F[s^*(t)] = \hat{s}(-\nu)^*.$$

Example 1: Let $s(t) = 1_{[-1/2,1/2]}(t)$, the indicator function of the interval $[-1/2, 1/2]$, also known as the rectangular pulse. Show in class that:
$$F[s(t)] = \mathrm{sinc}(\nu) \equiv \frac{\sin(\pi\,\nu)}{\pi\nu}.$$
By scaling property, for any positive number T:
$$F[1_{[-T/2,T/2]}(t)] = T\,\mathrm{sinc}(\nu\,T).$$

Note that the shorter (wider) the rectangular pulse, the wider (shorter) the spread of the transformed sinc function. This is known as uncertainty principle.

Example 2: Gaussian pulse is invariant:
$$F[\exp\{-\pi\,t^2\}] = \exp\{-\pi\nu^2\}.$$

Inversion of Fourier transform:

Theorem 1.1. *Let $s \in L^1$ and $\hat{s} \in L^1$. Then:*
$$s(t) = \int_R \hat{s}(\nu)\,\exp\{2\pi i\nu t\}\,dt.$$
Moreover:
$$\int_R |s(t)|^2\,dt = \int_R |\hat{s}|^2\,d\nu,$$
Parseval identity.

Another useful property of the Fourier transform is that it can turn convolution into multiplication. The Fourier transform of the convolution between two functions is the product of their Fourier transforms. Recall that the convolution $f(x) * g(x)$ is defined by

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(y)g(x-y)dy \tag{1.3}$$

Theorem 1.2 (Convolution-Multiplication Rule)**.**

$$F[f(x) * g(x)] = \hat{f}(\nu)\hat{g}(\nu)$$

2 Discrete Time Signals: Sampling and Transform

A discrete time signal is denoted $s(n)$ or s_n, where n is an integer and the value of s can be real or complex. It comes from a sampling or discretization of a continuous signal $s(t)$ with $t = n\Delta$, where $\Delta > 0$ is a discrete time step known as the *sampling interval.* A *discrete signal* is called *digital.* It is written as:

$$s(n) = s(n\Delta).$$

Some signals occur naturally at discrete times without analog to digital conversion, such as warehouse inventories, daily stock market prices.

A transform maps a discrete signal to another. A related concept is discrete-time system that maps an input signal to an output signal by a set of rules. We shall consider only linear system, denoted by $T[\cdot]$, satisfying *Linearity*:

$$T[as_1(n) + bs_2(n)] = aT[s_1(n)] + bT[s_2(n)], \tag{2.1}$$

for any two constants a and b.

2.1 Examples

We list 3 simple and useful discrete signals.
Example 1: unit sample, denoted by $\delta(n)$,

$$\delta(n) = \begin{cases} 1 & (n=0) \\ 0 & (\text{otherwise}) \end{cases}$$

The unit sample is used for decomposition of arbitrary signal into sums of weighted and delayed unit samples:

$$s(n) = \sum_{k=-\infty}^{+\infty} s(k)\,\delta(n-k). \tag{2.2}$$

Example 2: unit step, denoted by $u(n)$,

$$u(n) = \begin{cases} 1 & (n \geq 0) \\ 0 & (\text{otherwise}) \end{cases}$$

related to unit sample by:

$$u(n) = \sum_{k=-\infty}^{n} \delta(k).$$

Example 3: complex exponential, given by:

$$s(n) = \exp\{i\, n\, \omega_0\} = \cos(n\omega_0) + i\, \sin(n\omega_0),$$

where ω_0 is a real number.

Combining (2.1)-(2.2), we see that the output of a linear discrete time system $y(n) = T[s(n)]$ is represented as:

$$y(n) = \sum_{k=-\infty}^{+\infty} s(k)\, T[\delta(n-k)] = \sum_{k=-\infty}^{+\infty} s(k)\, h_k(n), \tag{2.3}$$

where $h_k(n) = T[\delta(n-k)]$ is the system response to the delayed unit sample $\delta(n-k)$. One can think of $\delta(n-k)$ as "basis vectors", a linear transform is completely determined when its action on basis vectors is known as in linear algebra.

The system is *shift invariant* if the output $y(n)$ goes to $y(n-n_0)$ when the input signal $s(n)$ becomes $s(n-n_0)$ for any time shift n_0. For a linear shift invariant (LSI) system, $h_k(n) = h(n-k)$ and formula (2.3) becomes:

$$y(n) = \sum_{k=-\infty}^{+\infty} s(k)\, h(n-k) \equiv s(n) * h(n), \tag{2.4}$$

the convolution sum, discrete version of (1.3).

Example 4: causal system

$$y(n) = s(n) + s(n-1), \tag{2.5}$$

the response at present time $n = n_1$ depends on the input only at present and past times $n \leq n_1$.

Example 5: non-causal system

$$y(n) = s(n) + s(n+1) + s(n-1).$$

A system is causal if and only if $h(n) = 0$ for $n < 0$.

The LSI is a stable system if the output is bounded in n when input is bounded in n. LSI system is stable if

$$\sum_{k=-\infty}^{+\infty} |h(n)| < +\infty,$$

for example:

$$h(n) = a^n\, u(n), \quad |a| < 1,$$

gives a stable and causal system.

2.2 Sampling

Sampling is the process of discretizing the domain of a continuous signal to produce a discrete signal which can then be processed on a computer. Usually, information is lost in the sampling process. Information may also be lost via quantization, which discretizes the range of the signal, rounding or truncating $s(n)$ to the nearest value in some finite set of allowed values. The samples might also be corrupted by random noise. For now, we will ignore quantization and noise and focus on the sampling process.

The *sampling rate* is defined to be $\frac{1}{\Delta}$, where Δ the sampling interval. An immediate question is what the sampling rate should be to represent a given signal. It's not surprising that if the sampling rate is too low, information is lost and the continuous signal is not uniquely determined by the samples. This kind of error is called aliasing. More surprising is the fact that for certain kinds of signals, it is possible to choose the sampling rate high enough so that no information is lost in the sampling process. This is the subject of the Shannon Sampling Theorem. To see what can happen when the sampling rate is too low, consider the periodic function $\sin(2\pi\nu t)$. Its period is $\frac{1}{\nu}$ and its frequency is ν. Now suppose it is sampled at $t = n\Delta$. From these samples alone, it is impossible to distinguish between functions of the form $\sin(2\pi\tilde{\nu}t)$ with $\tilde{\nu} = \nu + \frac{m}{\Delta}$ where m is any integer. This is because

$$\sin(2\pi(\nu + \frac{m}{\Delta})n\Delta) = \sin(2\pi\nu n\Delta).$$

In particular, when $\sin(2\pi\nu t)$ is sampled at rate $\frac{1}{\Delta}$, any frequency ν outside the range $\frac{-1}{2\Delta} < \nu \leq \frac{1}{2\Delta}$ is indistinguishable from a frequency in that range. This phenomenon is called aliasing and it can be said that higher frequency waveforms have lower frequency aliases depending on the sampling rate. When trying to reconstruct continuous signals from their discrete samples, aliasing error occurs when these lower frequency aliases are recovered instead of the original higher frequency components.

Even at a sampling rate of 2ν, $\sin(2\pi\nu t)$ ends up being sampled at $\sin(2\pi\nu n\frac{1}{2\nu}) = \sin(n\pi) = 0$ and is indistinguishable from the zero function. However, any higher sampling rate suffices to represent $\sin(2\pi\nu t)$ unambiguously. In general, the types of continuous signals that can be completely recovered from their sampled versions are band limited signals, namely those whose frequency content is bounded. More precisely, $s(t)$ is band limited if there is some $\nu_{\max}$ such that the Fourier transform $\hat{s}(\nu)$ is zero for $|\nu| > \nu_{\max}$.

Theorem 2.1 (Shannon Sampling Theorem)**.** *A continuous band limited function $s(t)$ with frequency content bounded by $\nu_{\max}$ ($|\nu| \leq \nu_{\max}$) can be completely recovered from samples taken at any sampling rate strictly greater than $2\nu_{\max}$. Moreover, a formula for the continuous signal in terms of its discrete samples can be given by:*

$$s(t) = \sum_{n=-\infty}^{\infty} s(n\Delta)\operatorname{sinc}(\frac{t-n\Delta}{\Delta}).$$

Remark 2.1. *The lower bound $2\nu_{\max}$ on the necessary sampling rate for recovering a band limited signal is known as the Nyquist rate. It is twice the bandwidth of the band limited signal. This is different from the Nyquist frequency $\frac{1}{2\Delta}$, which is half the sampling rate. If the original signal contains frequencies greater in magnitude than the Nyquist frequency, then they are aliased with lower frequencies less than or equal to the Nyquist frequency in magnitude.*

If $s(t)$ is band limited and the sampling rate is high enough so that $\nu_{\max} < \frac{1}{2\Delta}$, then

$$\hat{s}(\nu) = 1_{(\frac{-1}{2\Delta},\frac{1}{2\Delta})}(\nu)\hat{s}(\nu) = 1_{(\frac{-1}{2\Delta},\frac{1}{2\Delta})}(\nu)\sum_{n=-\infty}^{\infty}\hat{s}(\nu - \frac{n}{\Delta}) \tag{2.6}$$

where $\sum_{n=-\infty}^{\infty}\hat{s}(\nu - \frac{n}{\Delta})$ is a periodic extension of $\hat{s}$ with period $\frac{1}{\Delta}$. Crucially, since $\nu_{\max} < \frac{1}{2\Delta}$, this periodic extension is actually a string of non-overlapping copies of $\hat{s}$. This identity leads to the sinc interpolation formula given in the Shannon sampling theorem. To see how, we can use the fact that a periodic function $f(x)$ with period P that is square integrable over one period can be represented as a Fourier series, namely as an infinite series of the form

$$\sum_{n=-\infty}^{\infty} c_n e^{\frac{2\pi i n x}{P}} \qquad \text{where} \qquad c_n = \frac{1}{P}\int_{\frac{-P}{2}}^{\frac{P}{2}} f(x)e^{\frac{-2\pi i n x}{P}}\,dx.$$

Since $\sum_{n=-\infty}^{\infty}\hat{s}(\nu - \frac{n}{\Delta})$ is periodic with period $\frac{1}{\Delta}$ it can be represented as $\sum_{n=-\infty}^{\infty} c_n e^{2\pi i n\nu\Delta}$ with

$$c_n = \Delta\int_{\frac{-1}{2\Delta}}^{\frac{1}{2\Delta}}\sum_{k=-\infty}^{\infty}\hat{s}(\nu - \frac{k}{\Delta})e^{-2\pi i n\nu\Delta}\,d\nu = \Delta\int_{-\infty}^{\infty}\hat{s}(\nu)e^{-2\pi i n\nu\Delta}\,d\nu = \Delta s(-n\Delta).$$

Therefore

$$\sum_{n=-\infty}^{\infty} \hat{s}(\nu - \frac{n}{\Delta}) = \Delta \sum_{n=-\infty}^{\infty} s(n\Delta) e^{-2\pi n\nu\Delta},$$

which is known as the Poisson summation formula. Substituting in this expression, we can then take the inverse Fourier transform of (2.6) to recover $s(t)$.

$$\begin{aligned} s(t) &= \int_{-\infty}^{\infty} 1_{(\frac{-1}{2\Delta},\frac{1}{2\Delta})}(\nu)\Delta \sum_{n=-\infty}^{\infty} s(n\Delta)e^{-2\pi n\nu\Delta}e^{2\pi i\nu t}\,d\nu \\ &= \Delta \sum_{n=-\infty}^{\infty} s(n\Delta) \int_{-\infty}^{\infty} 1_{(\frac{-1}{2\Delta},\frac{1}{2\Delta})}(\nu)e^{2\pi i\nu(t-n\Delta)}\,d\nu \\ &= \sum_{n=-\infty}^{\infty} s(n\Delta)\,\mathrm{sinc}(\frac{t-n\Delta}{\Delta}) \end{aligned}$$

A complication that arises in practice is that we must work with finite duration signals. Such signals cannot be band limited even if they are obtained by restricting band limited functions to a finite interval. Therefore, aliasing is inevitable. However, band limited periodic functions are determined by a single period, so if the finite duration signal happens to correspond to a period of a band limited periodic function, then it's technically possible for it to be determined by discrete samples satisfying the sampling theorem.

In practice one works not only with signals of finite duration but with discrete signals of finite duration, ie: just a finite number of samples. It's still useful to analyze the frequency content of such signals. A discrete analogue of the Fourier transform called the discrete Fourier transform (DFT) can be used to do this. A discrete signal of finite duration consisting of N samples can be thought of as a vector in $\mathbb{C}^N$. Analogous to the Fourier transform, the DFT can be used to represent this vector as a linear combination of vectors $e_k \in \mathbb{C}^N$ of the form

$$e_k = (1, e^{\frac{2\pi ik}{N}}, e^{\frac{2\pi i2k}{N}}, ..., e^{\frac{2\pi i(N-1)k}{N}}) \qquad k = 0, 1, ..., N-1. \tag{2.7}$$

Following [3], the DFT can be motivated as a numerical approximation of the sampled Fourier transform of a finite duration signal. Suppose $s(t)$ is defined on $[0, T]$ and we take samples at $t = \frac{nT}{N}$, $n = 0, 1, ..., N-1$. Since $s(t)$ is restricted to $[0, T]$, the sampling theorem (applied to $\hat{s}$ instead of s) says it can be reconstructed completely from samples of $\hat{s}(\nu)$ sampled at $\nu = \frac{k}{T}$ for integer k. Using the available samples of s, the Fourier transform

$$\hat{s}(\frac{k}{T}) = \int_0^T e^{\frac{-2\pi ikt}{T}} s(t)\,dt$$

can be approximated by a Riemann sum, dividing T into intervals of length $\frac{T}{N}$. The resulting approximation is given by

$$\hat{s}(\frac{k}{T}) \approx \sum_{n=0}^{N-1} e^{\frac{-2\pi ikn}{N}} s(\frac{nT}{N})\frac{T}{N}.$$

It only makes sense to take $k = 0, 1, ..., N-1$ since $e^{\frac{-2\pi in(k+mN)}{N}} = e^{\frac{-2\pi ink}{N}}$ for integer m. Note that the approximation is better when k is much smaller than N.

Let $x \in \mathbb{C}^N$ be defined by $x_n = s(\frac{nT}{N})$, $n = 0, 1, ..., N-1$. Then $\text{DFT}(x) = X \in \mathbb{C}^N$ with

$$X_k = \sum_{n=0}^{N-1} e^{\frac{-2\pi ikn}{N}} x_n \qquad k = 0, 1, ..., N-1.$$

We will see in the next section that X_k are related to the coefficients for representing x in terms of the orthogonal basis consisting of the vectors e_k for $k = 0, 1, ..., N-1$.

2.3 Discrete Fourier Transform

The N-point discrete Fourier transform is a linear map from $\mathbb{C}^N$ to $\mathbb{C}^N$ defined by

$$\text{DFT}(x) = X \qquad \text{with} \qquad X_k = \sum_{n=0}^{N-1} e^{\frac{-2\pi ikn}{N}} x_n \qquad k = 0, 1, ..., N-1. \tag{2.8}$$

In terms of e_k defined by Equation 2.7, $X_k = \langle x, e_k \rangle$. The vectors $\{e_k\}_{k=0}^{N-1}$ form an orthogonal basis for $\mathbb{C}^N$, and the DFT can be understood as computing the coefficients for representing a vector in this basis. To see that the e_k are orthogonal, note that if $k \neq l$

$$\langle e_l, e_k \rangle = \sum_{n=0}^{N-1} e^{\frac{2\pi iln}{N}} \overline{e^{\frac{2\pi ikn}{N}}} = \sum_{n=0}^{N-1} e^{\frac{2\pi i(l-k)n}{N}} = \frac{1 - e^{2\pi i(l-k)}}{1 - e^{\frac{2\pi i(l-k)}{N}}} = 0$$

by summing the geometric series.

Let c_k be the coefficients of x in the basis $\{e_k\}_{k=0}^{N-1}$ so that

$$x = c_0 e_0 + c_1 e_1 + ... + c_{N-1} e_{N-1}.$$

We can solve for the c_k by taking the inner product of the entire expression with e_k. This implies

$$\langle x, e_k \rangle = c_k \langle e_k, e_k \rangle.$$

Noting that $\langle e_k, e_k \rangle = N$ and $\langle x, e_k \rangle = X_k$ we have that $c_k = \frac{X_k}{N}$. Thus

$$x = \frac{1}{N} \sum_{k=0}^{N-1} X_k e_k.$$

This is exactly the inverse discrete Fourier transform (IDFT). The IDFT is also a linear map from $\mathbb{C}^N$ to $\mathbb{C}^N$ defined by

$$\mathrm{IDFT}(X) = x \qquad \text{with} \qquad x_n = \frac{1}{N} \sum_{k=0}^{N-1} e^{\frac{2\pi ikn}{N}} X_k. \tag{2.9}$$

Remark 2.2. *The DFT and IDFT both produce periodic functions in the sense that if $X = DFT(a)$, then $X_{k+mN} = X_k$ and if $x = IDFT(b)$ then $x_{n+mN} = x_n$ for integer m.*

The DFT can also be expressed as a $N \times N$ matrix F_N whose row k, column n entry is given by $e^{\frac{2\pi ikn}{N}}$. Thus application of the DFT and IDFT can be interpreted as matrix multiplication by F_N and F_N^{-1} respectively.

$$X = F_N x \qquad\qquad x = F_N^{-1} X$$

Since $\{e_k\}_{k=0}^{N-1}$ is an orthogonal basis and $\langle e_k, e_k \rangle = N$, $F_N^{-1} = \frac{1}{N} F_N^*$, where $*$ denotes conjugate transpose ($F_N^* = \overline{F_N}^T$). Since F_N is symmetric, this simplifies further to $F_N^{-1} = \frac{1}{N}\overline{F_N}$.

A drawback of computing the DFT and IDFT by direct matrix multiplication is that this requires $O(N^2)$ operations. When N is large, this can be computationally impractical. Fortunately, there is a faster algorithm for computing the DFT called the fast Fourier transform (FFT), which takes advantage of the special structure of F_N and only requires $O(N \log N)$ operations.

The DFT has many analogous properties as the continuous Fourier transform. For example, a discrete analogy of Parseval's equations holds. Since $x = \frac{1}{N}\bar{F}_N F_N x = \frac{1}{N}\bar{F}_N X$,

$$\|x\|^2 = x^T \bar{x} = \frac{1}{N^2} X^T \bar{F}_N^T F_N \bar{X} = \frac{1}{N} X^T \bar{X} = \frac{1}{N}\|X\|^2.$$

A discrete analogue of the delay and modulation properties of the Fourier transform also apply to the DFT. Let τ_s denote translation by s such that $(\tau_s x)_n = x_{n-s}$. Then

$$\mathrm{DFT}(\tau_s x) = X\bar{e}_s \tag{2.10}$$

$$\mathrm{DFT}(x e_s) = \tau_s X \tag{2.11}$$

To verify that Equation 2.10 holds, note that

$$\mathrm{DFT}(\tau_s x)_k = \sum_{n=0}^{N-1} e^{\frac{-2\pi ikn}{N}} x_{n-s} = \sum_{m=-s}^{N-1-s} e^{\frac{-2\pi ik(s+m)}{N}} x_m = \sum_{m=0}^{N-1} e^{\frac{-2\pi ikm}{N}} x_m e^{\frac{-2\pi iks}{N}} = X_k e^{\frac{-2\pi iks}{N}} = (X\bar{e_s})_k.$$

Similarly, Equation 2.11 follows by noting that

$$\mathrm{DFT}(xe_s)_k = \sum_{n=0}^{N-1} e^{\frac{-2\pi ikn}{N}} x_n e^{\frac{2\pi ins}{N}} = \sum_{n=0}^{N-1} e^{\frac{-2\pi in(k-s)}{N}} x_n = X_{k-s} = (\tau_s X)_k.$$

There is also a discrete analogy to the convolution theorem. The discrete convolution is defined by

$$(x * y)_n = \sum_{j=0}^{N-1} x_j y_{n-j}, \tag{2.12}$$

where x and y are understood to be periodic with period N. $x * y$ is then also periodic. Analogous to the continuous case, the DFT turns convolution into pointwise multiplication.

Theorem 2.2. *Discrete Convolution Theorem:*

$$\mathrm{DFT}(x * y) = \mathrm{DFT}(x)\mathrm{DFT}(y) \tag{2.13}$$

$$\mathrm{DFT}(xy) = \frac{1}{N}\mathrm{DFT}(x) * \mathrm{DFT}(y) \tag{2.14}$$

Proof. Equation 2.13 follows from the delay property and the definition of the DFT.

$$\begin{aligned}\mathrm{DFT}(x * y)_k &= \sum_{n=0}^{N-1} e^{\frac{-2\pi ikn}{N}} \sum_{j=0}^{N-1} x_j y_{n-j} \\ &= \sum_{j=0}^{N-1} x_j Y_k e^{\frac{-2\pi ikj}{N}} = X_k Y_k\end{aligned}$$

Similarly, equation 2.14 follows with the help of the modulation property.

$$\begin{aligned}\mathrm{DFT}(xy)_k &= \sum_{n=0}^{N-1} x_n y_n e^{\frac{-2\pi ikn}{N}} \\ &= \sum_{n=0}^{N-1} \frac{1}{N} \sum_{j=0}^{N-1} e^{\frac{2\pi ijn}{N}} X_j y_n e^{\frac{-2\pi ikn}{N}} \\ &= \frac{1}{N} \sum_{j=0}^{N-1} X_j \sum_{n=0}^{N-1} e^{\frac{-2\pi in(k-j)}{N}} y_n \\ &= \frac{1}{N} \sum_{j=0}^{N-1} X_j Y_{k-j} = \frac{1}{N}(X * Y)_k\end{aligned}$$

□

The DFT can also be extended to two dimensions as well and analogous properties still hold. This will be useful when extending our analysis of 1D signals to 2D images.

2.4 Hands-On Matlab Exercises

Here are hands-on Matlab Exercises to consolidate the math concepts discussed so far, such as LSI system, sampling, DFT, also with an application (sound compression). Exercises 2 and 3 are drawn from [2].

Exercise 1: Consider the LSI system given by:

$$y(n) = [s(n+1) + s(n) + s(n-1)]/3. \tag{2.15}$$

If the input signal is:

$$s(1:50) = 0, \;\; s(51:100) = 1.$$

1. Write a for-loop to compute $y(1:100)$, with $y(1) = 0$, $y(100) = 1$.
2. Do: plot(s(1:100); hold on; plot(y(1:100),'r'); what does the system do to $s(n)$?
3. Feed the output $y(n)$ back into the system as new input, repeat this process 20 times, how is the final output compared with the original input s (plot and comment) ?

Exercise 2:
Consider sampling the function $f(t) = \sin(2\pi(440)t)$ on the interval $0 \le t < 1$, at 8192 points (sampling interval $\Delta = \frac{1}{8192}$ to obtain samples $f_k = f(k\Delta) = \sin(2\pi(440)k/8192)$ for $0 \le k \le 8191$. The samples can be arranged in a vector f. You can do this in MATLAB with

```
f = sin(2*pi*440/8192*(0:8191));
```

1. What is the frequency of the sinewave $\sin(2\pi(440)t)$, in Hertz?
2. Plot the first 100 samples with `plot(f(1:100))`.
3. At the sampling rate 8192 Hertz, what is the Nyquist frequency? Is the frequency of $f(t)$ above or below the Nyquist frequency?

4. Type `sound(f)` to play the sound out of the computer speaker. By default, MATLAB plays all sound files at 8192 samples per second, and assumes the sampled audio signal is in the range -1 to 1. The MATLAB command **soundsc(f)** automatically scales and centers signal f, and plays it out. Compare **soundsc(2*f)** and **soundsc(4*f)**.

5. As an example of aliasing, consider a second signal $g(t) = \sin(2\pi(440 + 8192)t)$. Repeat parts 1 through 4 with sampled signal

   ```
   g = sin(2*pi*(440+8192)/8192*(0:8191));
   ```

   The analog signal $g(t)$ oscillates much faster than $f(t)$, and we could expect it to yield a higher pitch. However, when sampled at frequency 8192 Hertz, $f(t)$ and $g(t)$ are aliased and yield precisely the same sampled vectors f and g. They should sound the same too.

Exercise 3:

1. Load in the "train" signal with the command `load('train');`. The sampling rate is 8192 Hertz, and the signal contains $12,880$ samples. If we consider this signal as sampled on an interval $[0, T]$, then $T = 12880/8192 \approx 1.5723$ seconds.

2. Compute the DFT of the signal with `Y=fft(y);`. Display the magnitude of the Fourier transform with `plot(abs(Y))`. The DFT should have length 12880 and be symmetric about the center.

   Since MATLAB indexes from 1, the DFT coeficient Y_k as defined by equation 2.8 is actually `Y(k+1)` in MATLAB. Also, Y_k corresponds to frequency $k/1.5723$, and so `Y(k)` corresponds to frequency $(k-1)/1.5723$ Hertz.

3. You can plot only the first half of the DFT with `plot(abs(Y(1:6441)))`. Use the data cursor button on the plot window to pick out the largest frequency and amplitude of the three largest frequencies in the train signal. Compute tha actual value of each frequency in Hertz.

4. Let f_1, f_2 and f_3 denote these largest frequencies, in Hertz, and let A_1, A_2, A_3 denote the corresponding amplitudes from the plot above. Define these variables in MATLAB. Synthesize a new signal using only these frequencies, sampled at 8192 Hertz on the interval $[0, 1.5]$, with

```
t = [0:1/8192:1.5];

ysynth = (A1*sin(2*pi*f1*t) + A2*sin(2*pi*f2*t) +
A3*sin(2*pi*f3*t))/(A1+A2+A3);
```

The division by $(A1 + A2 + A3)$ guarantees that the synthesized signal `ysynth` lies in the range $[-1, 1]$, which is the range MATLAB uses for audio signals.

5. Play the original train sound with `sound(y)`, and the synthesized version of only three frequencies with `sound(ysynth)`. Note that our computations do not take into account the phase information at these frequencies, merely the amplitude. Does the artificially generated signal capture the tone of the original?

6. Here is a simple approach to compressing an audio or other one-dimensional signal. The idea is to transform the audio signal to the frequency domain with the DFT. We then eliminate the insignificant frequencies by "thresholding," that is, zeroing out and Fourier coefficients below a given threshold. This becomes the compressed version of the signal. To recover an approximation to the signal, we use the IDFT to take the thresholded transform back to the time domain.

   For the train audio signal we can threshold as follows: First, we compute the maximum value of $|Y_k|$, with

   ```
   M = max(abs(Y))
   ```

   Then choose a threshold parameter `thresh` between 0 and 1. Let's start with

   ```
   thresh = 0.1
   ```

   Finally, we zero out all frequencies in Y that fall below a value `thresh*M` in magnitude. This can be done with

   ```
   Ythresh = (abs(Y)>thresh*M).*Y;
   ```

which installs the thresholded transform into `Ythresh`. Plot the thresholded transform with `plot(abs(Ythresh))`. You can also see what fraction of the Fourier coefficients survived the cut with

```
sum(abs(Ythresh)>0)/12880
```

We'll call this the compression ratio.

To recover an approximation to the original signal, inverse transform with

```
ythresh = real(ifft(Ythresh));
```

and play the compressed audio with `sound(ythresh)`. The `real` command truncates any vestigial imaginary round-off error in the `ifft` command.

You can compute the distortion of the compressed signal with

```
100*norm(y-ythresh)^2/norm(y)^2
```

Repeat the computations above for threshold values 0.001, 0.01, 0.1 and 0.5. In each case compute the compression ratio, the distortion, and of course play the audio signal and rate its quality.

References

[1] P. Brémaud, *Mathematical Principles of Signal Processing*, Springer, 2002.

[2] S. A. Broughton and K. Bryan, *Discrete Fourier Analysis and Wavelets*, Wiley, 2009.

[3] G. B. Folland, *Fourier Analysis and its Applications*, Wadsworth and Brooks/Cole, 1992.

[4] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Third Edition, Pearson Prentice Hall, 2008.

[5] W. Hartmann, *Signals, Sound, and Sensation*, Springer, AIP Series in Modern Acoustics and Signal Processing, 4th edition, 2000.

[6] B. Porat, *A course on digital signal processing*, John Wiley and Sons, 1997.