Lab 5: Discrete Fourier Transform

Due: Check Moodle

Submission Format: Report PDF, source code, accompanying files

DFT is a tool that is widely used in many areas. Yet, it is often taken for granted without proper understanding. In this assignment, we are going to develop a deeper understanding of DFT.

DTFT

The spectrum of the discrete signal is evaluated with the help of a discrete-time Fourier transform (DTFT). The equation for DTFT has the form

$$X(j\omega) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}, \qquad \forall \omega \in \mathcal{R}$$

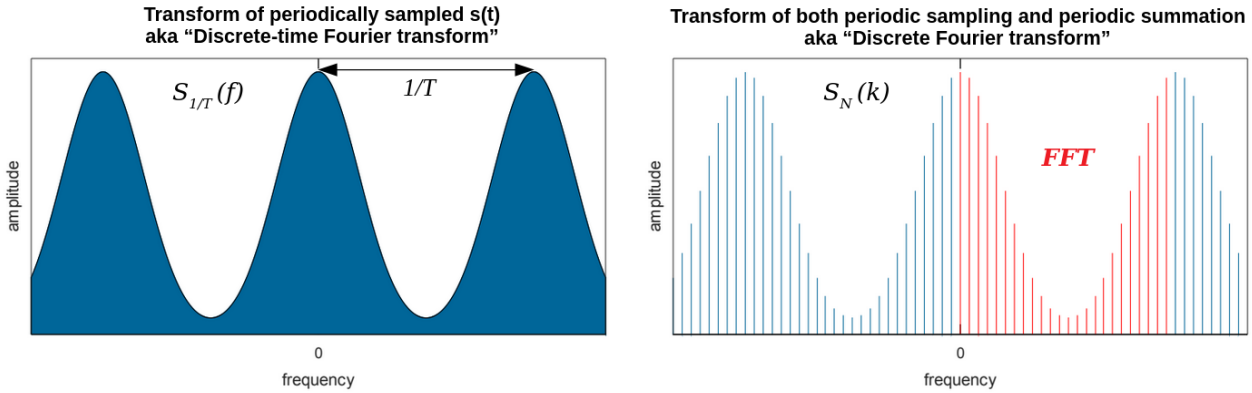
Unfortunately, DTFT is not a practical transformation. First, the signal is fully specified only when DTFT is computed for all frequencies in the range $\omega \in [-\pi, \pi]$, which includes an infinite number of points.

An engineer can pick only desired frequencies from the range and evaluate only them. This will give an evaluation of the correlation between the current signal and the desired frequency components. However, this can only be used for evaluation purposes. One cannot modify the spectrum and compute the inverse (IDTFT) because it requires integrating over the range of frequencies $\omega \in (-\infty, \infty)$ (see the definition of IDTFT).

DFT

Discrete Fourier Transform allows overcoming some practical problems that arise from using DTFT. The core idea of DFT is to take a signal of finite length N and introduce a periodic extension of this signal (refer to lecture slides for more details).

DTFT of a time-limited signal results in the signal with an unbounded spectrum. Periodic extension of the signal that takes place in DFT allows reducing the number of non-zero components in the spectrum. This notion is summarized in the following figure.



Source: Wikipedia

Because the spectrum of a periodic signal is discrete, we need to evaluate only a finite number of spectrum components to specify the signal spectrum completely. Specifically, the number of components is equal to the signal length N. When $N \to \infty$ DTFT and DFT become identical.

The most important practical implications of DFT are

- DFT can be computed for finite sequences in a finite number of steps
- DFT spectrum is a projection of the signal onto an orthogonal basis (bases $e^{-j2\pi k\frac{n}{N}}$ are orthogonal for $k = 0..N-1$)
- A signal can be reconstructed by computing IDFT
- DFT computes spectrum of a periodic extension of the original signal

Computing DFT

DFT Definition

The Discrete Fourier Transform (DFT) of a signal $x(n)$ may be defined by

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi k\frac{n}{N}}, \qquad n = 0, 1, 2, \ldots, N-1$$

where N is the signal length.

DFT Pseudocode:

```
for k = 0..N-1 do
    X[k]= 0
    for n = 0..N-1 do
        X[k] += x[n] * exp(-2 * i * pi * n * k / N)
    end do
    X[k]
end do
```

Source: pdf

Fast Fourier Transform

Fast Fourier Transform (FFT) rapidly computes such transformations by factorizing the DFT matrix into a product of sparse (mostly zero) factors. As a result, it manages to reduce the complexity of computing the DFT from $O\left(N^2\right)$ (follows from the definition of DFT) to $O(N \log N)$. The most common FFT algorithm is the Cooley–Tukey algorithm:

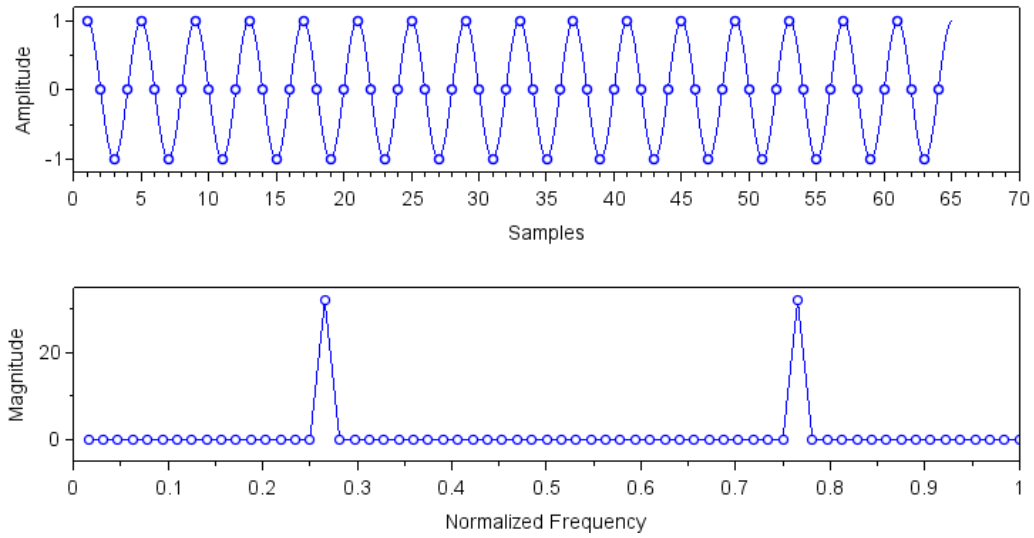
```
X0,...,N-1 ← ditfft2(x, N, s):       //DFT of (x0, xs, x2s, ..., x(N-1)s):
if N = 1 then
    X0 ← x0                    //trivial size-1 DFT base case
else
    X0,...,N/2-1 ← ditfft2(x, N/2, 2s)   //DFT of (x0, x2s, x4s, ...)
    XN/2,...,N-1 ← ditfft2(x+s, N/2, 2s) //DFT of (xs, xs+2s, xs+4s, ...)
    //combine DFTs of two halves into full DFT:
    for k = 0 to N/2-1 do
        t ← Xk
        Xk ← t + exp(-2πi k/N) Xk+N/2
        Xk+N/2 ← t - exp(-2πi k/N) Xk+N/2
    end for
end if
```

Source: Wikipedia

Exploring DFT

Spectral leakage

If we build a simple single-frequency cosine wave, as shown in the figure below, the transformation will result in non-zero magnitude values at the correct frequencies. We get such a good result because the sampling frequency matches peaks and valleys.

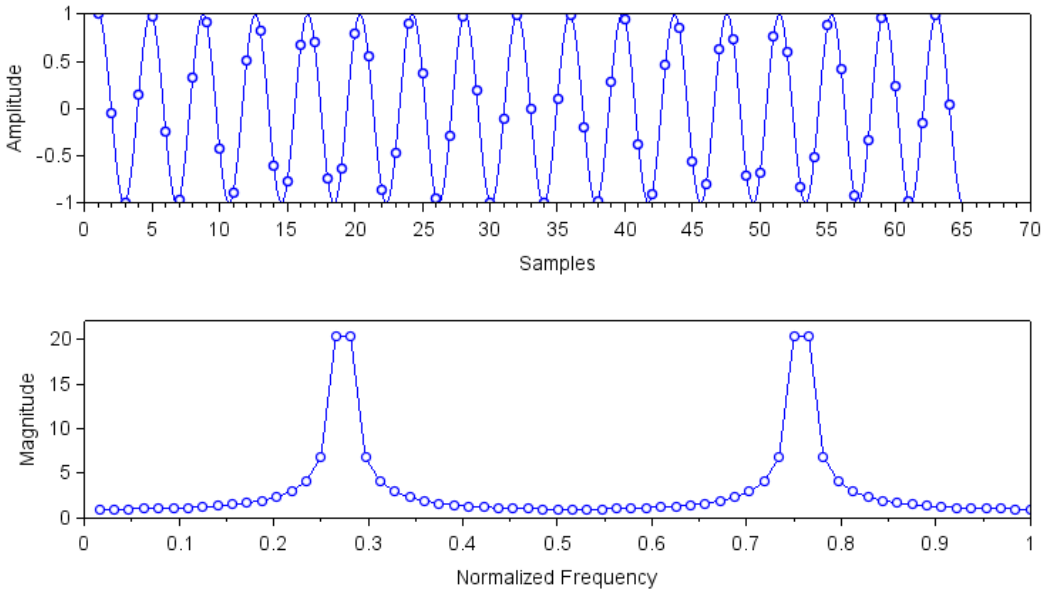


The signal has a single frequency spectrum under the condition that the ratio of the number of samples in the sequence N by the number of samples in a single period of the signal N_T is an integer:

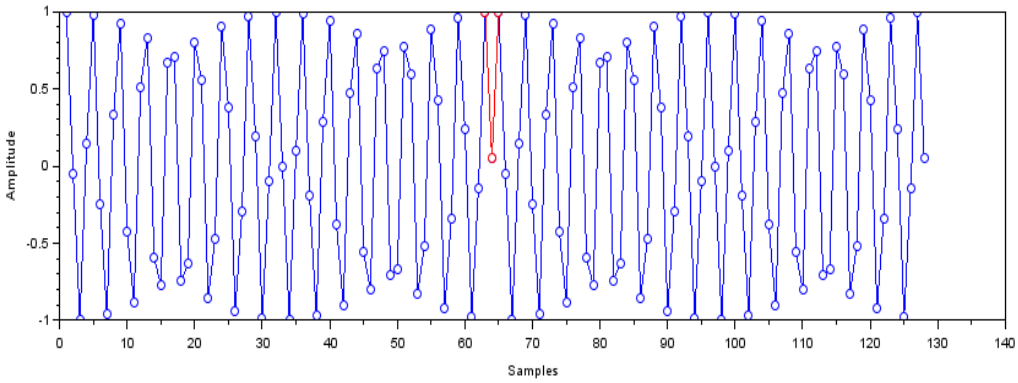
$$\frac{N}{N_T} = k, \;\; k = 0..N-1$$

This comes from the fact that when these conditions are met, the periodic extension of the signal results in an uninterrupted sinusoidal function.

If we change the frequency of the cosine wave such that this number is rational, spectral leakage is observed. The result of the transformation will show the presence of other frequencies.



To get an idea of where this is coming from, look at the periodic extension of the time waveform. Note the 'glitch' in the middle where the signal begins its forced repetition.



DFT is based on a periodic signal $e^{-j2\pi nk/N}$ with an integer number of periods N. It is assumed that the signal is repetitive outside the interval on which DFT is performed. In the example above, the signal was not described on the duration up to an integer number of periods. Thus it was fractured. Fractional periods cause signal discontinuity, which in turn causes spectral leakage.

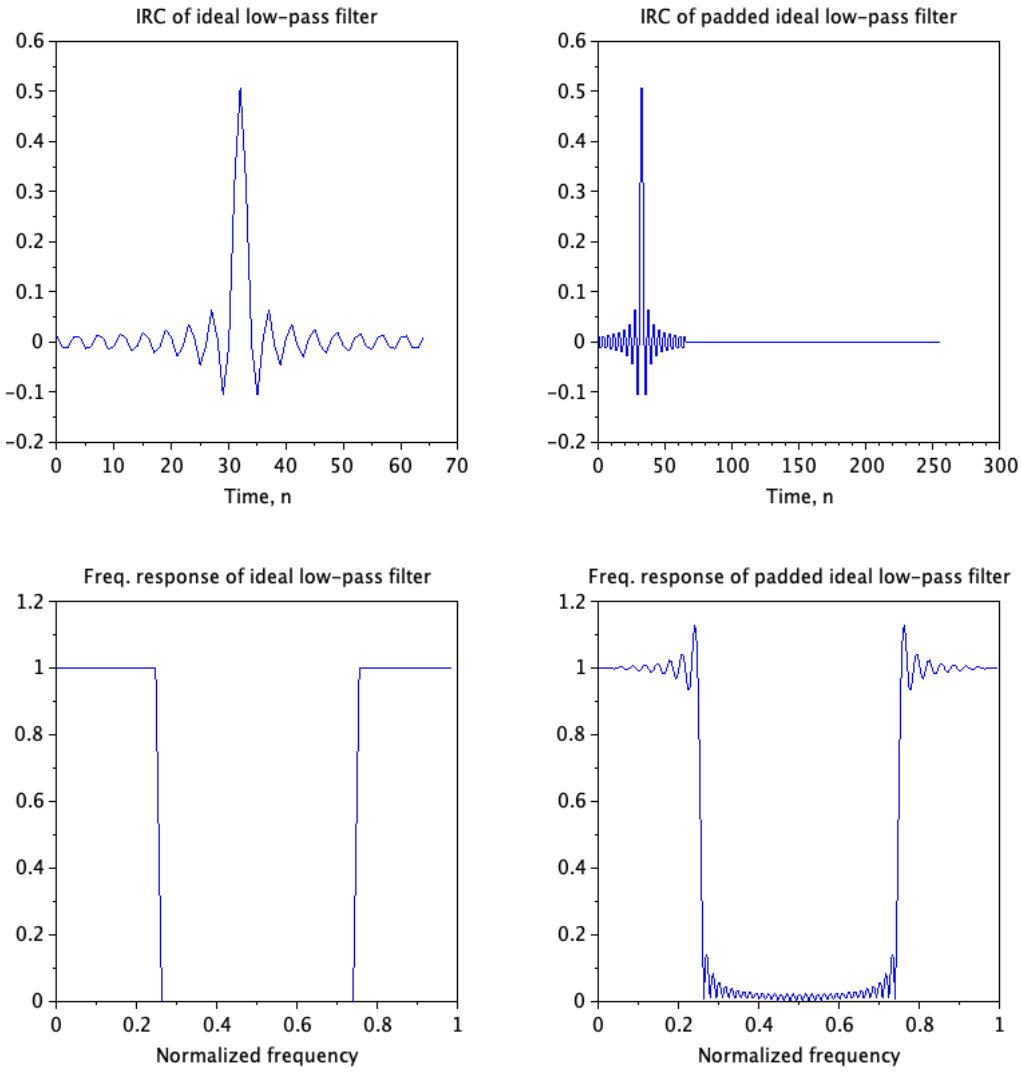
Zero Padding and Spectral Interpolation

In the example above, you have observed that the DFT spectrum is not always what you think. Now let us overview the implication of this observation towards a familiar problem: FIR filter design.

Consider the following code example that generates a simple low-pass filter.

```
H_l = ideal_lowpass(64, 0.25, 0) // Ideal rectangular IRC
h_l = ifft(H_l) // Ideal impulse response
h_l_padded = resize_matrix(h_l, 1, 256) // Pad IRC with zeros
plot(h_l) // Plot ideal IRC
plot(h_l_padded) // Plot padded IRC
plot(abs(fft(h_l))) // Plot spectrum of ideal IRC
plot(abs(fft(h_l_padded))) // Plot spectrum of padded IRC
```

On the first line of this code snippet, a frequency response for an ideal filter is generated. On the second, a time-domain IRC is computed. Then, we create a padded copy of this IRC with $256 - 64 = 210$ zeros in the end. Then we plot these two figures and their corresponding frequency responses. The frequency response of the ideal IRC should be rectangular (the way it was generated), but the form of the frequency response for the padded IRC is unknown. Speculatively, we are not changing the response itself by introducing zeros in the end. Hence, it should be the same.



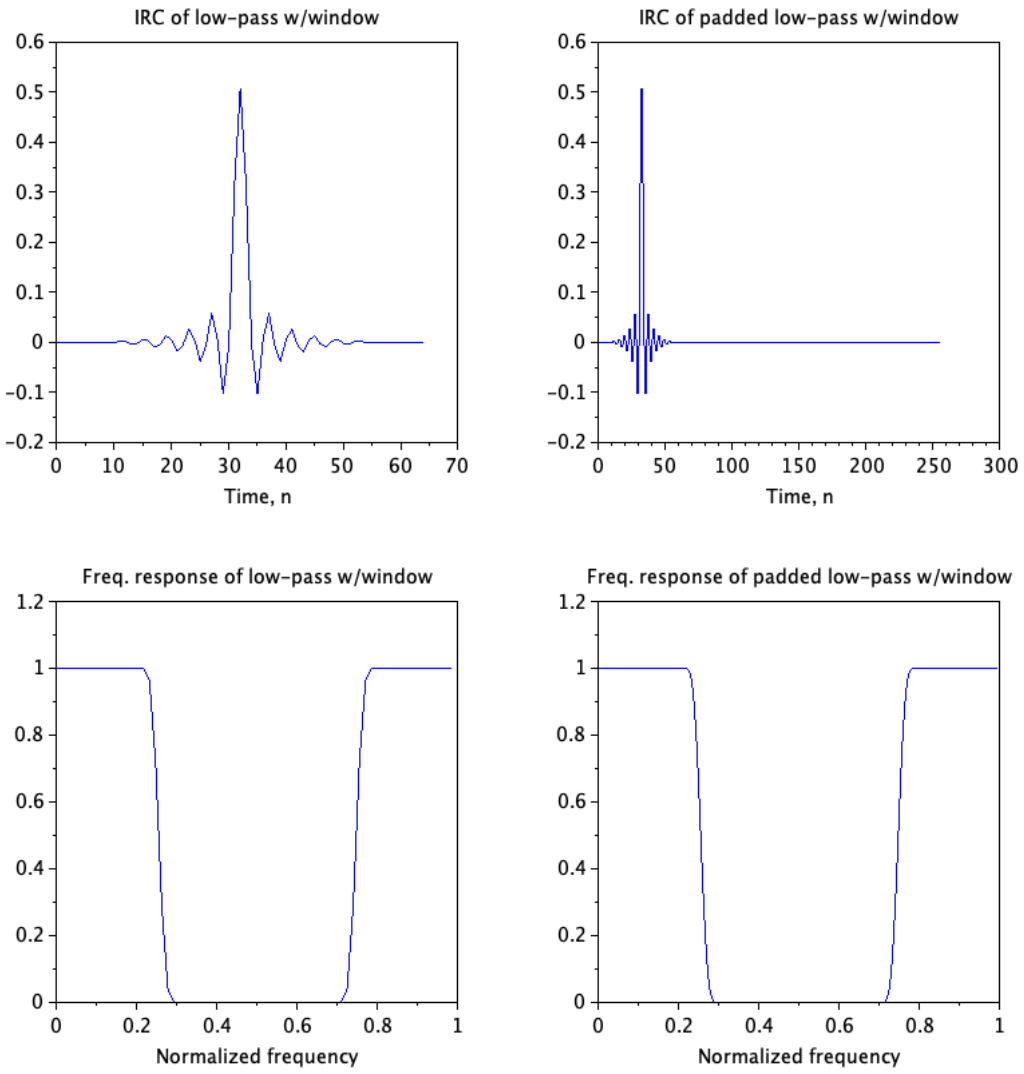
As it appears, adding zeros at the end of the signal changes its frequency response. This happens because the DFT spectrum computes circular convolution with the periodic extension of the original finite-length signal. After we introduce zeros, the period of the signal changes, and DFT spectrum changes as well.

Remember that DFT approaches DTFT when $N \to \infty$. This is equivalent to padding our IRC with an infinite number of zeros. This means that when no circular convolution is involved, the frequency response is much less than ideal.

Now, remember that when applying an FIR filter, one performs a regular and not cyclic convolution. This means that the effective frequency response of the filter is more similar to the frequency response of the padded IRC.

Now, let us look at the effect of the window.

```
H_l = ideal_lowpass(64, 0.25, 0) // Ideal rectangular IRC
h_l = ifft(H_l) .* window("kr", 64, 8)// windowed impulse response
h_l_padded = resize_matrix(h_l, 1, 256) // Pad windowed IRC
plot(h_l) // Plot windowed IRC
plot(h_l_padded) // Plot padded windowed IRC
plot(abs(fft(h_l))) // Plot spectrum of windowed IRC
plot(abs(fft(h_l_padded))) // Plot spectrum of padded windowed IRC
```



A window function is applied by multiplying each sample by window function coefficients. It suppresses the tails of IRC. Remember that multiplication in the time domain corresponds to convolution in the frequency domain. Given the Fourier Transforms of particular window functions, this procedure effectively smoothens the frequency response. As a result, we see that the frequency response of windowed IRC and its padded versions are nearly identical.

Practical implications:

- DFT performs circular convolution with the periodic extension of the finite-length sequence. This periodic extension is not the same as the original sequence
- DTFT describes how to compute the [correct] spectrum of the finite-length sequence, but it is not practical
- One can interpolate [correct] DTFT spectrum from DFT spectrum by padding the finite-

length sequence with zeros

- This is often referred to as Spectral Interpolation theorem:

> For time-limited signal x, taking the DFT of the entire non-zero portion of x extended by zeros yields exact interpolation [not an approximation] of the complex spectrum.

- DFT computes a subset of what DTFT computes
- This should be taken into consideration when using DFT for design
- Is this discrepancy between DFT and DTFT a problem? Probably not.

Aliasing

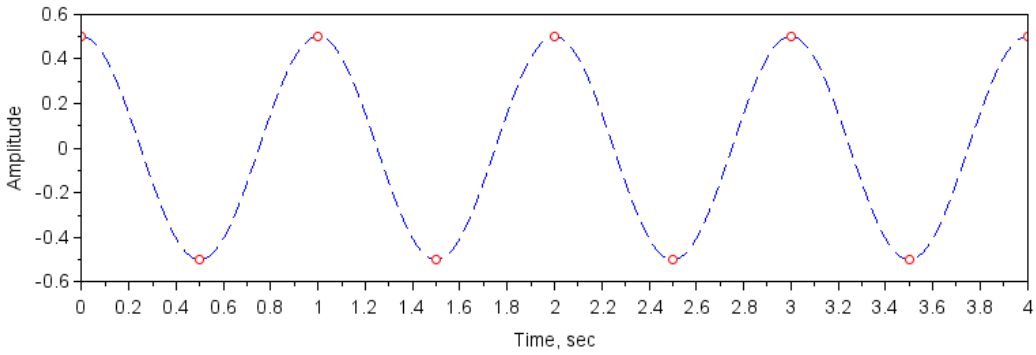
Nyquist sampling theorem

The Nyquist sampling theorem (*Kotelnikov theorem*) provides a prescription for the nominal sampling interval required to avoid aliasing. It may be stated simply as follows: *The sampling frequency should be at least twice the highest frequency contained in the signal.*

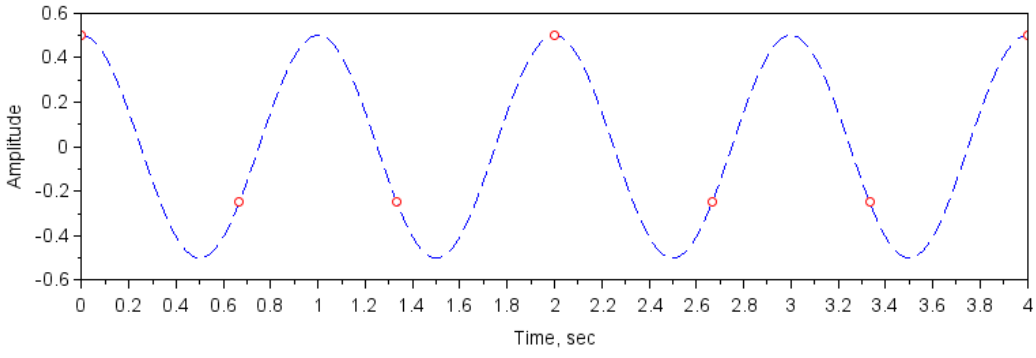
$$f_s \geq 2f_c$$

Aliasing Effect

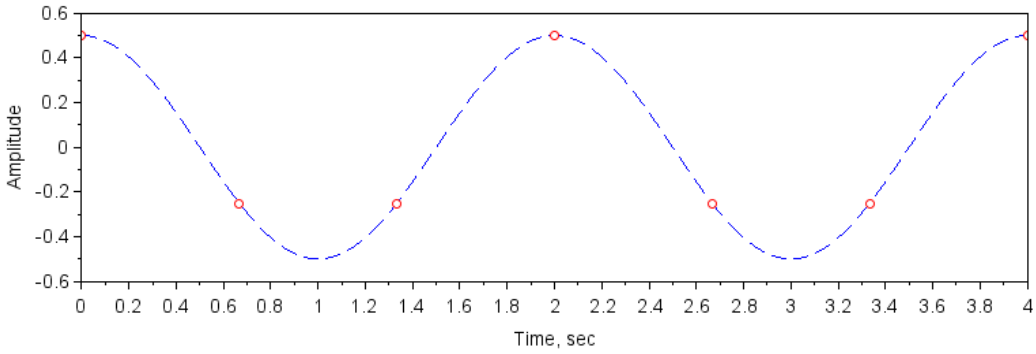
Consider, for example, a signal composed of a single sinusoidal wave at a frequency of 1 Hz. If we sample this waveform at 2 Hz (as dictated by the Nyquist theorem), that is sufficient to capture each peak and trough of the signal:



If however we sample at a frequency lower than 2 Hz, for example at 1.5 Hz, then there are now not enough samples to capture all the peaks and troughs in the signal:



Note here that we are not only losing information, but we are getting the wrong information about the signal. The person receiving these samples, without any previous knowledge of the original signal, may well be misled into thinking that the signal has quite a different form:



From this example, we can see the reason for the term *aliasing*. That is, the signal now takes on a different "persona" or a false presentation, due to being sampled at an insufficiently high frequency.

Source: *Bruno A. Olshausen: PSC 129 - Sensory Processes*

Sampling Sinusoidal Functions

When sampling a function at frequency fs (at time intervals $1/fs$ seconds), the following functions yield identical sets of samples:

$$sin(2\pi(f + Nfs)t + \varphi), N = 0, \pm1, \pm2, \pm3, \dots$$

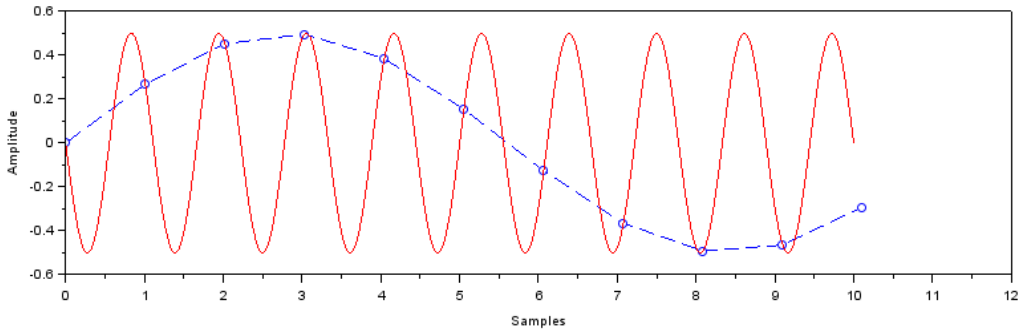
A frequency spectrum of the samples produces equally strong responses at all those frequencies. Without collateral information, the frequency of the original function is ambiguous. So the functions and their frequencies are said to be aliases of each other. Noting the trigonometric identity:

$$\sin(2\pi(f + Nf_s)t + \phi) = \begin{cases} +\sin(2\pi(f + Nf_s)t + \phi), & f + Nf_s \geq 0 \\ -\sin(2\pi|f + Nf_s|t - \phi), & f + Nf_s < 0 \end{cases}$$

we can write all the alias frequencies as positive values:

$$f_N(f) \triangleq |f + Nf_s|.$$

For example, here, a plot depicts a set of samples with parameter $fs = 1$ (represented with blue dots), and two different sinusoids that could have produced such samples. Nine cycles of the red sinusoid and one cycle of the blue sinusoid span an interval of 10 samples. The corresponding number of cycles per sample are $f_{red} = 0.9f_s$ and $f_{blue} = 0.1f_s$, where f_s the sampling frequency. So the $N = -1$ alias of f_{red} is f_{blue} (and vice versa).



Source: Wikipedia

Task

1. Implement your function for DFT in SciLab. Verify your functions. Be ready to explain how DFT algorithm works. (You will get 1 bonus point if you implement and are able to explain [one of] FFT algorithm[s])
2. Produce examples of spectral leakage in the DFT spectrum [**use SciLab's built-in function**]. Generate a signal with three sinusoidal components of equal amplitude, two of which produce spectral leakage, and the third does not. How do the amplitudes of these sinusoidal signals correspond in the frequency domain?
3. Compute spectral interpolation of the signals you created in task 2. Are the spectrums identical? If not, describe the differences.
4. In a time range of 0.5 seconds, plot cosine signal $x_1(n)$ with a frequency $f_1 = 190\ Hz$ and amplitude $A_1 = 0.5$, sampled at a rate of $f_s = 200\ S/s$, and cosine signal $x_2(n)$ with $f_2 = 10\ Hz$ and amplitude $A_2 = 2$ sampled at the same f_s.
    - Compute DFT/FFT. Describe your observations.
    - By adjusting the sampling rate, make the DFT reflect the correct frequency for both sinusoidal components.

Self-check questions

1. What is the difference between DFT and FFT algorithms?
2. What is the cause of spectral leakage?
3. How to make an ideal interpolation in the frequency domain?
4. How can ideal interpolation clarify the frequency value in the case of spectral leakage?
5. What may occur if the sampling rate is incorrectly selected?

Additional References

> If you still do not understand the Fourier transform and where the glitches come from, here are some video explanation.

- YouTube: An animated introduction to the Fourier Transform
- YouTube: One more explanation of Fourier Series and Fourier Transform with easy to

understand 3D animations

- YouTube: DFT
- YouTube: FFT
- YouTube: Aliasing
- YouTube: Uncertainty principle
- Spectal Interpolation: 1, 2, 3

Bonus Task

(1 point) Show that basis vectors for DFT are orthogonal, i.e.

$$\frac{1}{N}\sum_{n=0}^{N-1}(e^{-j2\pi k\frac{n}{N}})^* \cdot e^{-j2\pi \hat{k}\frac{n}{N}} \neq 0 \iff k = \hat{k}$$