An introduction to the Discrete Fourier Transform and its applications in signal processing

Laurent Nony, Jean-Marc Themlin

An introduction to the Discrete Fourier Transform and its applications in signal processing

Laurent NONY & Jean-Marc THEMLIN, Aix-Marseille Université

April 20, 2023

This Jupyter notebook is meant to introduce the concepts of Discrete Fourier Transform (DFT) as a fundamental tool of signal processing. The theoretical foundations of the Fourier transform are introduced, however with a minimal mathematical formalism. The reader must have been introduced to the Fourier transform concept and must have some mathematical background before going through this document.

Here, the focus is rather brought on the technical implementation of the DFT, as well as on illustrations of the pratical use of the DFT by means of illustrative codes implemented in Octave[1], a free programming langage that is compatible with Matlab.

The outline of this notebook is as follows:

---

[1] https://octave.org/

I- Foundations of the Fourier transform for continuous time signals

I-1- Definition and conventions

A Fourier transform (FT) is a mathematical transform that decomposes functions depending on time, or space, into functions depending on temporal, or spatial, frequency. The Fourier transform of a **function** (here a **signal**) is a complex-valued function representing the complex sinusoids out of which the original function might be reconstructed. For each frequency, the magnitude of the complex value (modulus) represents the amplitude of a constituent complex sinusoid with that frequency, and the argument of the complex value represents that complex sinusoid's phase offset.

Here, the focus is brought to one-dimensional, time-dependent signals only, but the framework can be extended to 2D and space-dependent signals.

Let's assume a **continuous time signal** (i.e. analog signal) $z(t)$, integrable in time (i.e. presenting no divergence), and its Fourier transform counterpart $Z(f) = FT\{z(t)\}$, forming a **Fourier transform pair**:

$$z(t) \rightleftharpoons Z(f) \tag{1}$$

By **definition**, $Z(f)$ and $z(t)$ are respectively derivable from each other according to :

$$Z(f) := FT\{z(t)\} = \int_{-\infty}^{+\infty} z(t)e^{-j2\pi ft}dt \tag{2}$$

and, upon inverse FT operation:

$$z(t) := FT^{-1}\{Z(f)\} = \int_{-\infty}^{+\infty} Z(f)e^{+j2\pi ft}df \tag{3}$$

Note: *Another definition of the Fourier transform using the pulsation $\omega = 2\pi f$ instead of the frequency can be found in the text books:*

$$Z(\omega) := \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} z(t)e^{-j\omega t}dt \tag{4}$$

But unless specified, we won't use this definition and we will stick to $Z(f)$.

I-2- Fourier series & Fourier transform

The FT formalism applies to any kind of signal, including periodic AND non-periodic signals. For **periodic signals** however, **Fourier series** are a valuable tool, whose formalism is easier than the FT one. Besides, since one can always consider a single period T of any T-periodic signal (frequency $f := 1/T$), the Fourier series formalism obviously also applies to finite-duration signals of duration T.

Fourier series can represent T-periodic signals as the sum of sinusoids whose frequencies are integer multiples of the fundamental frequency $f = 1/T$ of the periodic signal. The modulus and

phase of the Fourier series coefficients represent the amplitude and phase of each harmonically-related sinusoid. Hence the Fourier series coefficients spectrum of a T-periodic signal is discrete and its constituting frequencies are harmonics (i.e. integer multiples) of the fundamental frequency $f = 1/T$, whereas the Fourier transform spectrum of the same signal is continuous.

The table below summarizes the mathematical formalism for the Fourier series coefficients and for the Fourier transform.

Fourier Series formalism	**Fourier Transform formalism**
T-periodic continuous time signal z(t) in L^2	Any continuous time signal z(t) in L^2
$z(t) = \sum_{n=-\infty}^{+\infty} Z_n e^{j2\pi \frac{n}{T} t} = \sum_{n=-\infty}^{+\infty} Z_n e^{j2\pi n f t}$	$z(t) = \int_{-\infty}^{+\infty} Z(f) e^{+j2\pi f t} df$
Fourier Series coefficients	Fourier Transform
$Z_n = \frac{1}{T} \int_0^T z(t) e^{-j2\pi n f t} dt \in \mathbb{C}$	$Z(f) = \int_{-\infty}^{+\infty} z(t) e^{-j2\pi f t} dt \in \mathbb{C}$
Scalar product-based definition:	Scalar product-based definition:
$Z_n = \frac{1}{T} \langle z(t), e_n(t) \rangle \in \mathbb{C}$	$Z(f) = \langle z(t), e^{j2\pi f t} \rangle \in \mathbb{C}$
Discrete spectrum	Continuous spectrum

II- The Discrete time Fourier Transform (DtFT)

In most applications, the FT is to be computed out of **discrete time sampled signals**. Therefore, it's natural to introduce the discrete time Fourier Transform (DtFT).

II-1- Time sampling

Let T_s be the **sampling period** of the continuous time signal $z(t)$, hence its corresponding **sampling frequency**:

$$f_s := \frac{1}{T_s} \tag{5}$$

The discrete time sampled signal is denoted $z_s(t)$. At that stage, we make no assumption on the duration of its time support, that may be considered as infinite, hence an infinite number of samples. $z_s(t)$ is derived out of the **Dirac comb operator** of temporal period T_s, which is the sampling operator:

$$\delta_{T_s}(t) = \sum_{n=-\infty}^{+\infty} \delta(t - nT_s), \tag{6}$$

Therefore the continuous time t gets sampled according to:

$$t \to t_s[n] = t\delta_{T_s}(t)|_n = nT_s, \forall n \in \mathbb{Z} \tag{7}$$

and then the **discrete time sampled** version of the continuous time signal is:

$$z_s(t_s) = z(t)\delta_{T_s}(t) \tag{8}$$

Hence, the value of the n^{th} sample, with $n \in \mathbb{Z}$, of $z_s(t_s)$: $z_s(t_s[n]) \to z_s[n]$.

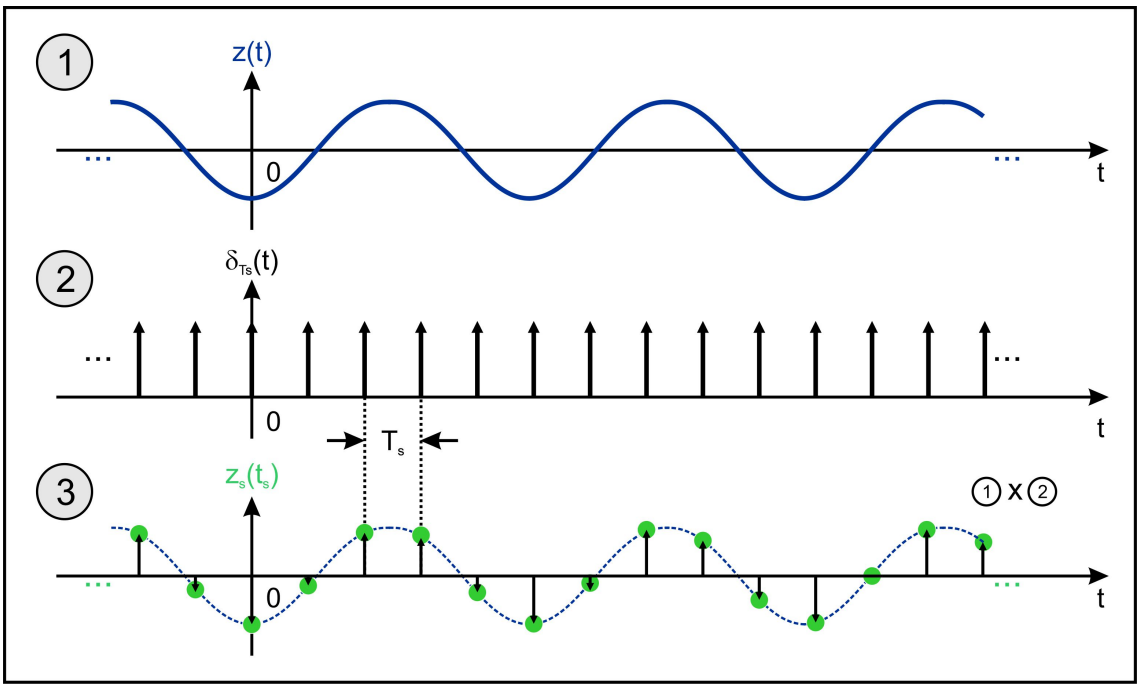
The figure below summarizes the sampling process of a continuous time signal $z(t)$ into its sampled version $z_s(t_s)$.

II-2- Formal expression of the DtFT

Let $Z_s(f)$ be the DtFT of the sampled signal $z_s(t_s)$, forming the FT pair:

$$z_s(t_s) \rightleftharpoons Z_s(f) \tag{9}$$

The formal expression of $Z_s(f)$ can be derived as follows. As we are about to demonstrate it, a straightforward consequence of the temporal sampling process is that the DtFT is f_s-periodic in the frequency space. Besides, it can be shown that the FT of a T_s-periodic Dirac comb operator in time ($T_s = \frac{1}{f_s}$), is a f_s-periodic Dirac comb operator in the frequency space, weighted by f_s. The corresponding FT pair is:



$$\left[\delta_{T_s}(t) = \delta_{1/f_s}(t)\right] \rightleftharpoons \left[f_s\delta_{f_s}(f) = \frac{1}{T_s}\delta_{1/T_s}(f)\right] \tag{10}$$

Thus, using the **convolution theorem**, the FT of the sampled signal $z_s(t_s)$ is:

$$Z_s(f) := FT\{z_s(t_s)\} = FT\{z(t)\delta_{T_s}(t)\} = FT\{z(t)\} * FT\{\delta_{T_s}(t)\} = FT\{z(t)\} * f_s\delta_{f_s}(f) = Z(f) * f_s\delta_{f_s}(f) \tag{11}$$

The sampling process turns the FT into a DtFT, now defined by the formal expression:

$$Z(f) \rightarrow Z_s(f) := Z(f) * f_s\delta_{f_s}(f) \tag{12}$$

Because the δ_{f_s} Dirac comb replicates $Z(f)$ every f_s, $Z_s(f)$ is f_s-periodic.

II-3- Explicit expression of the DtFT

To derive a more explicit expression of $Z_s(f)$, we consider the discrete time expression of the FT of a continuous time signal (equ.2):

$$Z(f) := \int_{-\infty}^{+\infty} z(t)e^{-j2\pi ft}dt \rightarrow Z(f) = \sum_{n=-\infty}^{+\infty} z_s[n]e^{-j2\pi fnT_s}T_s \tag{13}$$

That is, according to equ.12:

$$
\begin{aligned}
Z_s(f) := Z(f) * f_s\delta_{f_s}(f) &= \sum_{n=-\infty}^{+\infty} z_s[n] e^{-j2\pi f n T_s} T_s * f_s\delta_{f_s}(f) \\
&= \sum_{n=-\infty}^{+\infty} z_s[n] e^{-j2\pi f n T_s} * \delta_{f_s}(f) \\
&= \sum_{n=-\infty}^{+\infty} z_s[n] e^{-j2\pi n \frac{f}{f_s}} * \delta_{f_s}(f)
\end{aligned}
\tag{14}
$$

III- The Discrete Fourier Transform (DFT)

The discrete Fourier Transform (DFT) of a generic discrete time signal $z_s(t_s)$ is an approximation of its DtFT (equ.14) obtained by :

- truncating the summation over n
- discretizing the f parameter
- truncating the f_s-periodization

III-1- Truncating the summation: temporal windowing

When dealing with realistic discrete time sampled signals, it is impossible to handle an infinite number of samples. **The discrete time sampled signal to be analyzed, $z_s(t_s)$, always consists of a finite number of samples.** Let us assume that the sampling operation yields N samples of the discrete time signal. Hence the truncation of the summation over n in equ.14.

The N samples are obtained by **windowing operation** of $z_s(t_s)$, i.e. the multiplication of the signal by a windowing function $w(t)$ to form the **windowed sampled signal**:

$$z_{sw}(t_s) = z_s(t_s)w(t) \tag{15}$$

$w(t)$ may have a typical rectangular shape (but note that this definition is not exclusive), whose width sets the **windowing duration** T_w, i.e. the duration of the time support of $z_{sw}(t_s)$. $w(t)$ and its FT $W(f)$ form a FT pair defined as:

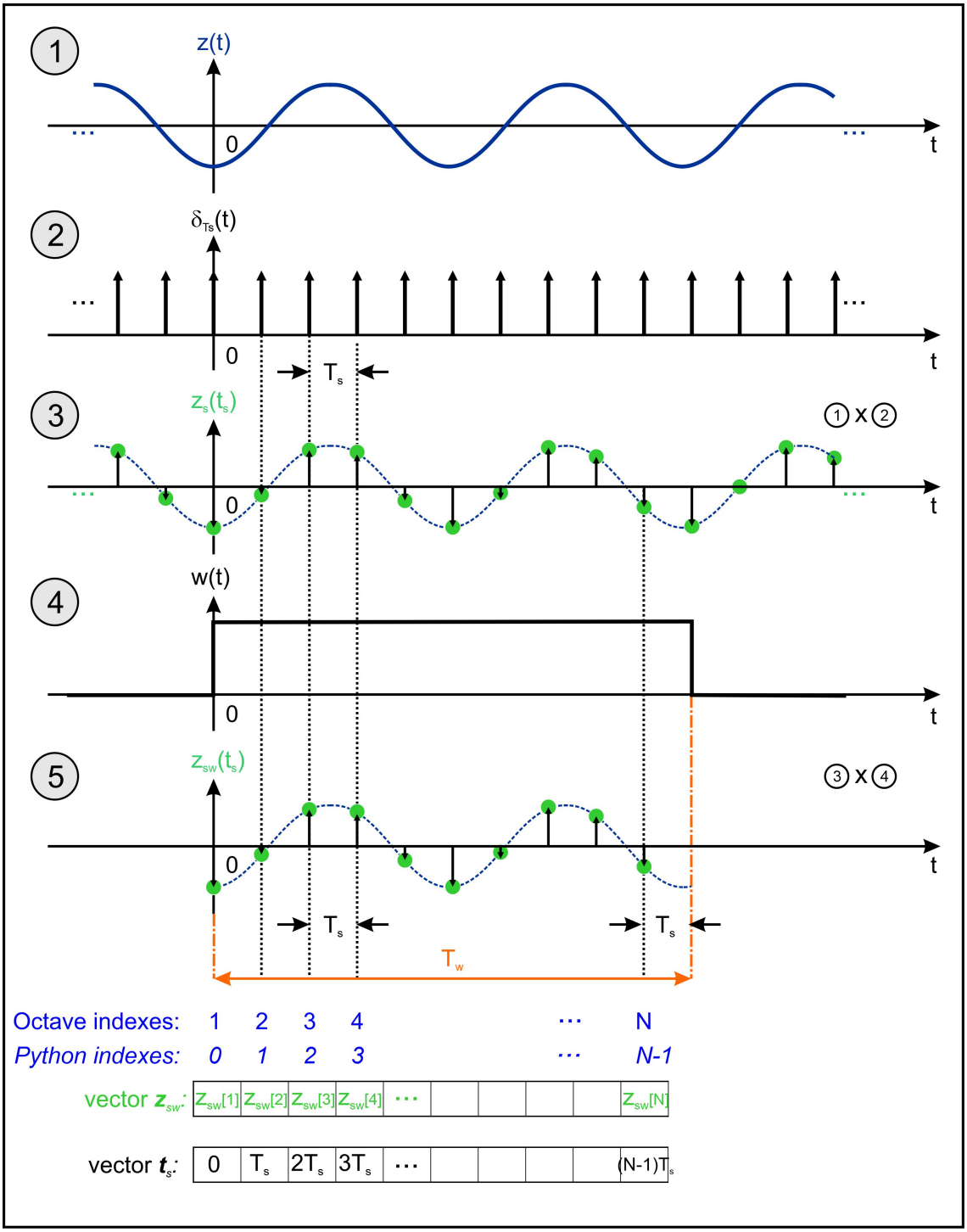
$$w(t) = \begin{cases} 1, \forall t \in [0; T_w] \\ 0, \text{ anywhere else} \end{cases} \rightleftharpoons W(f) = T_w \frac{\sin(\pi T_w f)}{\pi T_w f} := T_w \text{sinc}(T_w f) \tag{16}$$

By definition:

$$T_w := NT_s \tag{17}$$

Now, the n^{th} sample, with $n \in \mathbb{N}, n \in [0; N-1]$, of the windowed discrete time support t_s is $t_s[n] := nT_s$, and the corresponding sample of $z_{sw}(t_s)$ is $z_{sw}(t_s[n]) \rightarrow z_{sw}[n]$.

The figure below completes the former figure about the sampling process of a continuous time signal $z(t)$ into its sampled and windowed version $z_{sw}(t_s)$.



Important note: Although the last sample of the windowed discrete time support is written as $(N-1)T_s$, the total duration of the signal is $T_w = NT_s$. In other words, each sample covers a duration of T_s in time (cf. figure below, item 5).

III-2- Resuming the expression of the DtFT

The effect of the windowing operation is not trivial to the DtFT, which forces us to reconsider its expression (equ.14).

To that purpose, we resume the analysis of sections II-2 and II-3 out of the **FT of the windowed continuous time signal** $Z_w(t) := FT\{z(t)w(t)\}$, thus: $z(t)w(t) \rightleftharpoons Z_w(f)$. It is also reminded that: $w(t) \rightleftharpoons W(f)$.

We then have:

$$Z_w(f) := FT\{z(t)w(t)\} = Z(f) * W(f) \rightarrow Z_w(f) = \sum_{n=0}^{N-1} z_{sw}[n] e^{-j2\pi f n T_s} T_s * W(f) \tag{18}$$

That is, by analogy with equ.12:

$$\begin{aligned} Z_{sw}(f) := Z_w(f) * f_s\delta_{f_s}(f) &= \textstyle\sum_{n=0}^{N-1} z_{sw}[n] e^{-j2\pi f n T_s} T_s * W(f) * f_s\delta_{f_s}(f) \\ &= \textstyle\sum_{n=0}^{N-1} z_{sw}[n] e^{-j2\pi f n T_s} * W(f) * \delta_{f_s}(f) \\ &= \textstyle\sum_{n=0}^{N-1} z_{sw}[n] e^{-j2\pi n \frac{f}{f_s}} * W(f) * \delta_{f_s}(f) \end{aligned} \tag{19}$$

The previous equation constitutes the most complete expression of the DtFT of a continuous time signal $z(t)$, whose windowed and discrete time version is $z_{sw}(t_s)$.

The figure below illustrates the influence of the temporal windowing, both in the time space and in the frequency space, by FT pairing. It is important to notice that we have taken a symetric interval of definition for the continuous time signal $z(t)$.

III-3- Frequency sampling

III-3-a Sampled frequency

Both the T_s-sampling and the windowing in time force us to consider the sampling of the frequency, too. As seen before, the T_s-sampling in time imposes a f_s-periodization of the DtFT. In order to keep the same number of samples in the frequency domain, N samples will be used to sample the irreducible part (one period) of the DtFT. Therefore, the N samples of $z_s(t_s)$ resulting from the windowing operation feature a single f_s period of its DtFT.

Hence a frequency sampling rate: $\delta f = f_s/N$.

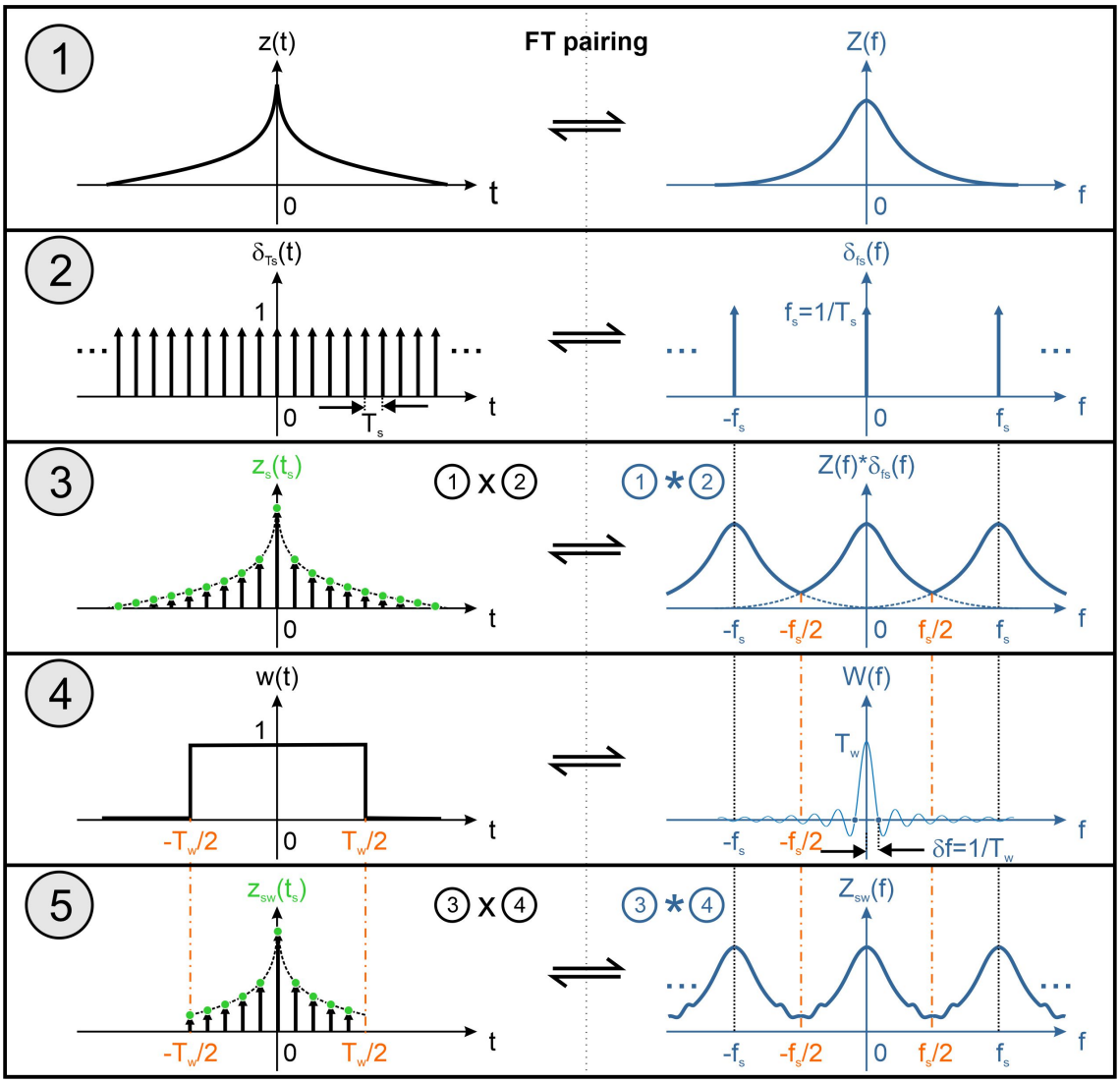
Therefore the continuous frequency gets sampled according to:

$$f \rightarrow f_{\delta f}[k] := f\delta_{\delta f}(f)|_k = k\delta f = k\frac{f_s}{N}, \forall k \in \mathbb{Z} \tag{20}$$

To simplify the notations, we will write in the following: $f_{\delta f}[k] \rightarrow f_k$.

Then the **discrete frequency sampled** version of the continuous frequency DtFT signal is:

$$Z_{sw}(f_{\delta f}[k]) = Z_{sw}(f_k) = Z_{sw}(f)\delta_{\delta f}(f)|_k \tag{21}$$



Hence, out of equ.19:

$$Z_{sw}(f_k) = \sum_{n=0}^{N-1} z_{sw}[n] e^{-j2\pi n \frac{f_k}{f_s}} * W(f_k) * \delta_{f_s}(f_k) \times \delta_{\delta f}(f)|_k \tag{22}$$

where the symbol $\times$ represents a product.

III-3-b Spectral resolution

The frequency sampling rate δf states the formal expression of the **spectral resolution** of the DFT (or frequency resolution), i.e. the shortest separation in frequency between two subsequent samples in the spectrum:

$$\delta f := f_{k+1} - f_k = \frac{f_s}{N} = \frac{1}{NT_s} = \frac{1}{T_w} \tag{23}$$

III-3-c Important consequence on the DtFT

Because $\delta f = \frac{1}{T_w}$, the FT pair $\delta_{\delta f}(f) \rightleftharpoons \frac{1}{\delta f}\delta_{1/\delta f}(t)$ may be written as:

$$\delta_{\delta f}(f) \rightleftharpoons T_w \delta_{T_w}(t) \tag{24}$$

Thus, we can establish the **FT pair between the discrete frequency DtFT and the windowed discrete time signal** as:

$$[Z_{sw}(f_k) = Z_{sw}(f)\delta_{\delta f}(f)] \rightleftharpoons z_{sw}(t_s) * T_w \delta_{T_w}(t) \tag{25}$$

The windowed discrete time signal is replicated every T_w duration, which means that it is actually T_w-periodic! Nevertheless, this operation affects the amplitude of $z_{sw}(t_s)$ as the signal is now scaling with T_w. To avoid this, $Z_{sw}(f_k)$ is to be normalized by T_w.

$$Z_{sw}(f_k) \rightarrow \frac{Z_{sw}(f_k)}{T_w} \tag{26}$$

Hence, out of equ.22:

$$Z_{sw}(f_k) = \frac{1}{T_w}\sum_{n=0}^{N-1} z_{sw}[n]e^{-j2\pi n\frac{f_k}{f_s}} * W(f_k) * \delta_{f_s}(f_k) \times \delta_{\delta f}(f)|_k \tag{27}$$

Introducing the explicit expression of $W(f_k)$ for a rectangular windowing (equ.16), we have now:

$$Z_{sw}(f_k) = \sum_{n=0}^{N-1} z_{sw}[n]e^{-j2\pi n\frac{f_k}{f_s}} * \mathrm{sinc}(T_w f_k) * \delta_{f_s}(f_k) \times \delta_{\delta f}(f)|_k \tag{28}$$

The consequence of the frequency sampling has properly been taken into accout, we get rid from the sampling operator $\delta_{\delta f}(f)|_k$ to lighten the previous equation, while keeping the frequency sampling in mind:

$$Z_{sw}(f_k) = \sum_{n=0}^{N-1} z_{sw}[n]e^{-j2\pi n\frac{f_k}{f_s}} * \mathrm{sinc}(T_w f_k) * \delta_{f_s}(f_k) \tag{29}$$

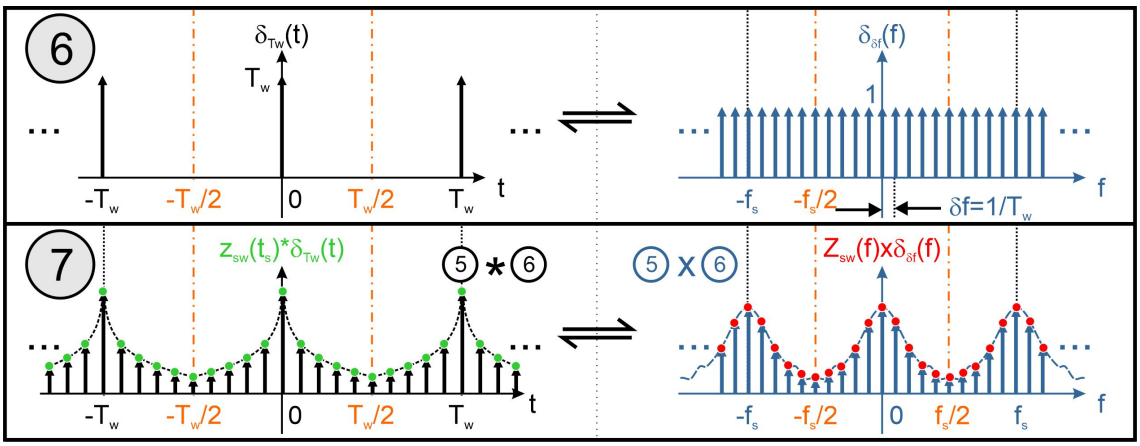
The figure below completes the previous one and illustrates the above elements.

III-4- Truncating the f_s-periodization

It was mentioned above that the N samples of $z_s(t_s)$ resulting from the windowing operation feature a single f_s period of its DtFT. Thus, the f_s-periodization is truncated to the frequency range $f_k \in \left[-\frac{f_s}{2}; \frac{f_s}{2}\right[$.

The **discrete frequency support** of the DFT is then defined as:

$$f_k := -\frac{f_s}{2} + k\frac{f_s}{N}, \forall k \in \mathbb{N}, k \in [0; N-1] \tag{30}$$



or equivalently, using the spectral resolution δf:

$$f_k := -\frac{f_s}{2} + k\delta f, \forall k \in \mathbb{N}, k \in [0; N-1] \tag{31}$$

Because the frequency range is restricted to $f_k \in \left[-\frac{f_s}{2}; \frac{f_s}{2}\right[$, the convolution with the Dirac comb operator $\delta_{f_s}(f)$ can be removed from the expression of the DtFT (cf. equ.29):

$$Z_{sw}(f_k) = \sum_{n=0}^{N-1} z_{sw}[n] e^{-j2\pi n \frac{f_k}{f_s}} * \text{sinc}(T_w f_k), \text{ with } f_k := -\frac{f_s}{2} + k\frac{f_s}{N}, \text{ and } k \in \mathbb{N}, k \in [0; N-1] \tag{32}$$

An equivalent expression is obtained when introducing the **dimensionless frequency**:

$$\hat{k} := \frac{f_k}{f_s} \tag{33}$$

Hence, $\hat{k} \in \left[-\frac{1}{2}; +\frac{1}{2} - \frac{1}{N}\right]$. The expression of $Z_{sw}(f_k)$ now becomes:

$$Z_{sw}(f_k) = \sum_{n=0}^{N-1} z_{sw}[n] e^{-j2\pi n \hat{k}} * \text{sinc}(T_w f_k), \text{ with } \hat{k} := -\frac{1}{2} + \frac{k}{N}, \text{ and } k \in \mathbb{N}, k \in [0; N-1] \tag{34}$$

The previous equation constitues a possible definition of the DFT, however the regular algorithm implemented in most of softwares to compute the DFT does not perform the straight implementation of equ.34. To earn computation time, a further step is considered.

III-5- Pre-implementation step

In equ.34, the summation is performed over negative and positive values of the dimensionless frequency $\hat{k} \in \left[-\frac{1}{2}; +\frac{1}{2} - \frac{1}{N}\right]$. To avoid the handling of signed and unsigned numbers, the implementation actually performs a shift of $\hat{k}$ by $1/2$, such that $\hat{k} \in \left[0; 1 - \frac{1}{N}\right]$, and therefore:

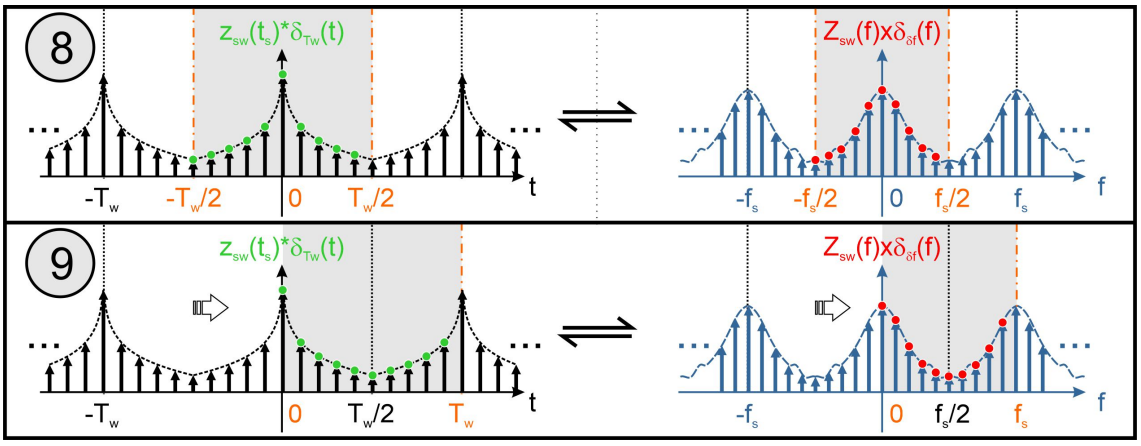
$$Z_{sw}(f_k) = \sum_{n=0}^{N-1} z_{sw}[n] e^{-j2\pi n\hat{k}} * \mathrm{sinc}(T_w f_k), \text{ with } \hat{k} := \frac{k}{N}, \text{ and } k \in \mathbb{N}, k \in [0; N-1] \tag{35}$$

The previous equation now constitues the DFT, as implemented in the regular algorithm *"fft"* implemented in most of softwares.

Important notes:

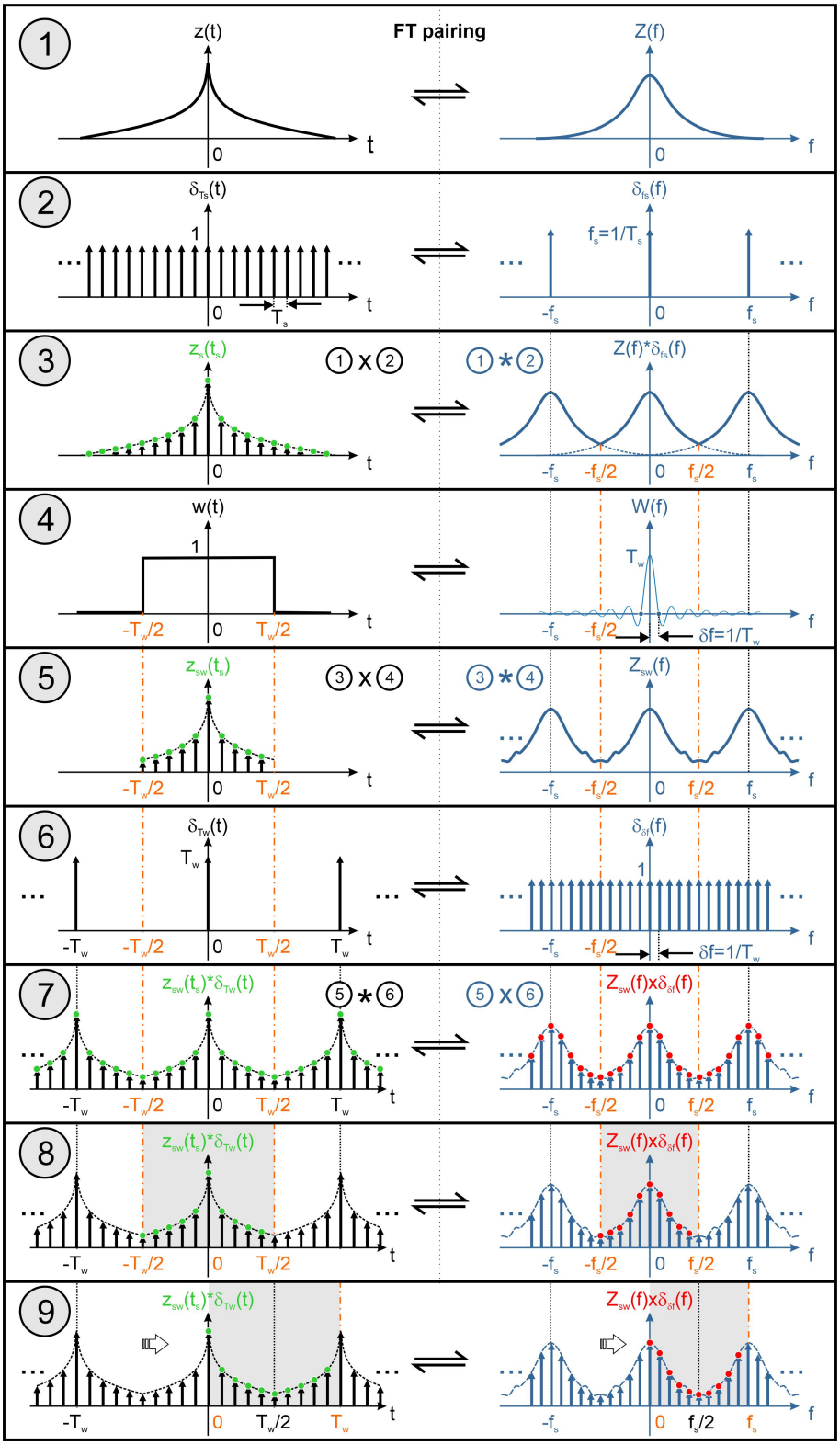
- In equ.35, **the DFT does not primarily depend on** f_s. The numerical implementation does not require the value of that parameter to compute the DFT of a sampled signal. However, **the quantitative interpretation of the DFT does mandatorily require to know** f_s**!** The f_s-dependence of the DFT will actually be concealed in the abscissa axis of the spectrum, as well as in its **normalization procedure** (cf. below).
- In equ.35, due to the shift of $\hat{k}$ that now varies in the range $\left[0; 1-\frac{1}{N}\right]$, **the first** $\frac{N}{2}$ **samples of the DFT and of the frequency support stand for** $\hat{k} \in \left[0; \frac{1}{2}-\frac{1}{N}\right]$, hence $f_k = \hat{k} f_s \in \left[0; \frac{f_s}{2} - \delta f\right[$, that is **the positive frequencies range**. Because this range is restricted to $\left[-\frac{f_s}{2}; +\frac{f_s}{2}\right[$, **the next** $\frac{N}{2}$ **samples of the DFT and of the frequency support** $\left(\hat{k} \in \left[\frac{1}{2}; 1-\frac{1}{N}\right]\right)$, **will stand for the negative frequencies range**: $\left[-\frac{f_s}{2}; -\delta f\right[$. **The concept applies with the samples of** $z_{sw}(t_s)$ **and those of the time support too.**
- It is most important to notice that, strictly speaking, **the physical unit of** $Z_{sw}(f_k)$**, as computed by equ.35, is similar to the physical unit of** $z_s(t_s)$**. Yet, the regular physical unit of the Fourier transform is the one of signal multiplied by a time.** This issue has to do with the **normalization of the DFT**, which will be discussed hereafter.
- The convolution with $ sinc(T_w f_k)$ is not effectively realized in a typical computation of the DFT, which generally only uses the sum $\sum_{n=0}^{N-1} z_{sw}[n] e^{-j2\pi n\hat{k}}$. It merely reflects the fact that the (time) windowing of the original signal has two consequences on the computed spectra : it increases the width of the spectral features in the DtFT and the DFT, and it introduces some oscillations in the spectra, corresponding to the sidelobes of the window function. Apart from the natural rectangular window, other window shapes can be used, that can reduce either the widening of the spectral features, either the residual oscillations.

The figure below illustrates the last sequence to calculate the DFT.



IV- DtFT & DFT: synopsis

The figure below summarizes the discretization process of a continuous time signal $z(t)$ yielding its DtFT (step 5) and lastly, its DFT (step 9).



V- Computation of the DFT

In most of softwares, the computation of the DFT, as given by equ.35, is performed by the instruction *"fft"*. E.g. if $z_{sw}[n]$ is a windowed sampled signal (a "vector") consisting of N samples in Octave, its DFT will be given by the instruction $fft(z_{sw})$.

We hereafter give couple of examples of such an implementation, compare it to the result returned by the *"fft"* instruction, and highlight some traps to avoid.

V-1- Raw computation

We examplify the calculation of the DFT with a simple cosine waveform signal of period T_1 (frequency $f_1 = 1/T_1$) over a windowing duration: $T_w = 1$ s:

$$z(t) = \cos(2\pi f_1 t), \forall t \in [0;1] \tag{36}$$

whose FT is well-known:

$$Z(f) = \frac{1}{2}\left[\delta(f - f_1) + \delta(f + f_1)\right] \tag{37}$$

On the other hand, the signal being T_1-periodic, it can be expanded in Fourier series, whose set of coefficients are real:

$$Z_1 = Z_{-1} = \frac{1}{2} \tag{38}$$

Below is given an Octave code that implements equ.35 and compares the result to the one returned by the *"fft"* instruction. We discuss the DFT spectra in the monolateral range $\left[0; \frac{f_s}{2}\right[$ (i.e. $\left[0; \frac{f_s}{2} - \delta f\right]$) for now.

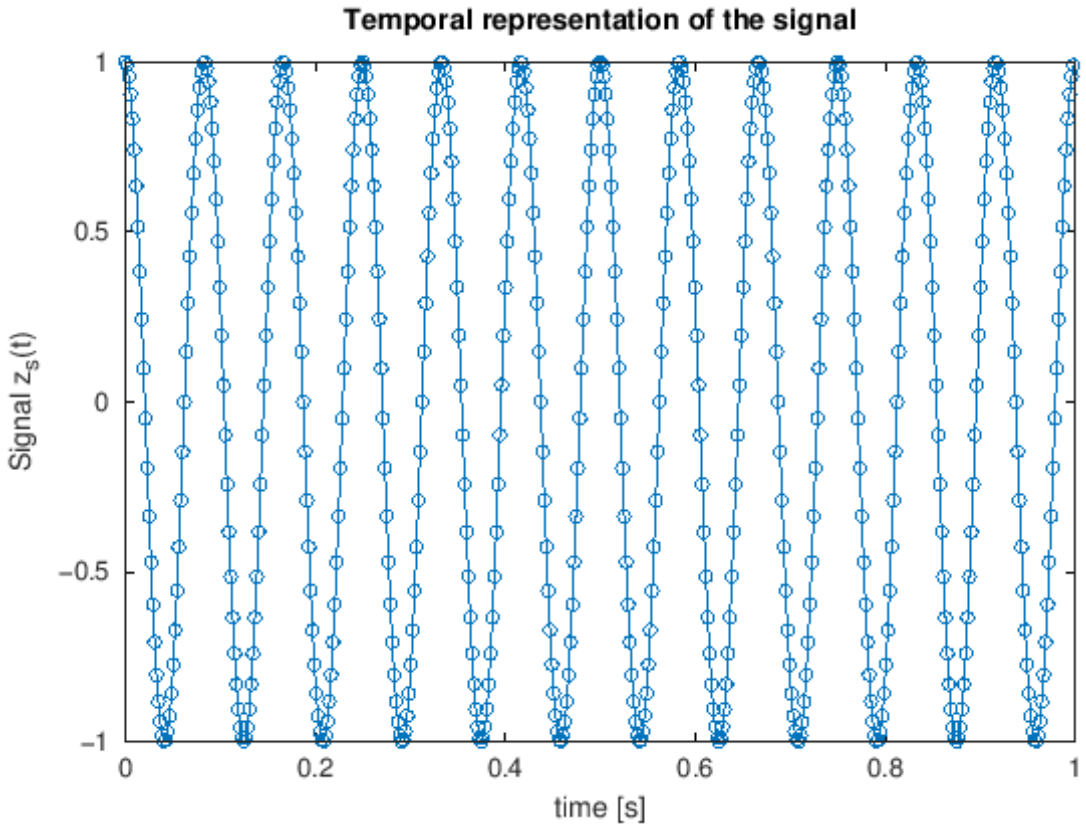
Illustrative code: The code below computes half the bilateral spectrum of a sinusoidal signal, windowed over an integer number of periods, without normalization.

[1]:
```
##########################################################
# Illustration of the fft instruction from hard coding #
##########################################################

##### Initializations
# Regular initializations lines in Octave, quite useless in CoCalc
clc
clear all
close all

%matplotlib qt

# Parameters
N = 512;  # number of samples
```

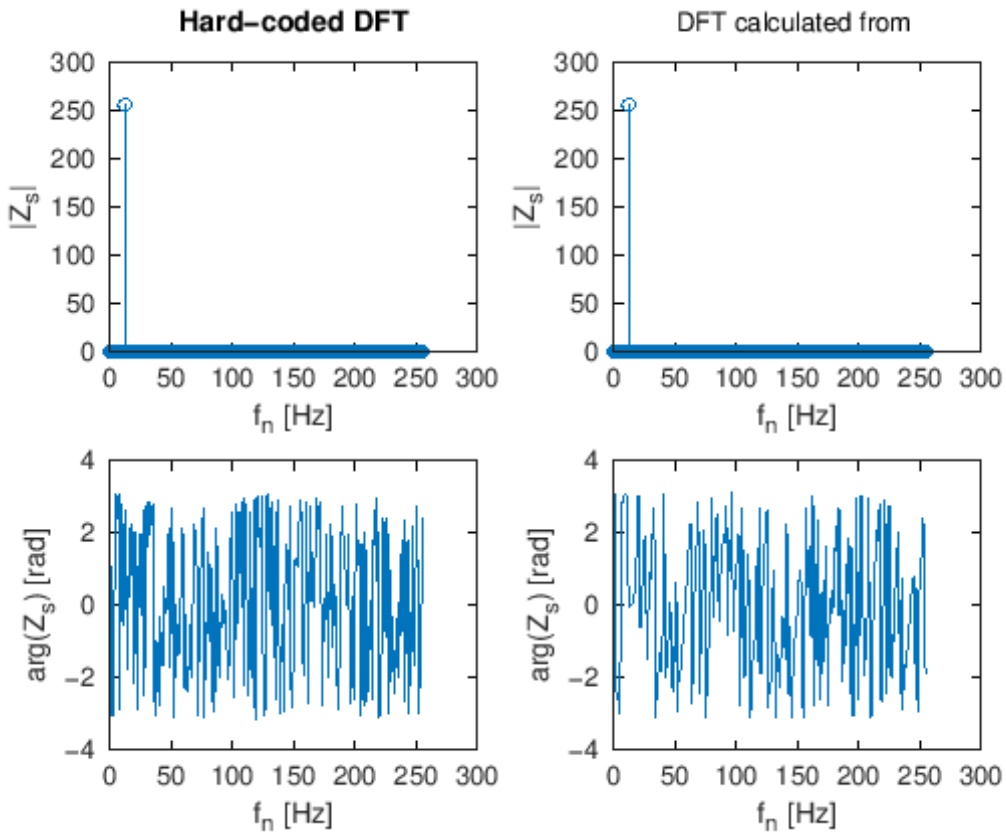
```
Tw= 1;    # windowing duration in natural units, i.e. seconds

f1= 12;   # frequency of the cosine waveform signal in Hz

#In-built variables
Ts = Tw/N; # sampling period
fs = 1/Ts; # sampling frequency
df = fs/N; # spectral resolution

# support vectors
n  = 0:(N-1);       # n ranging from 0 to N-1, as defined in eq.18 e.g.
t_s= n*Ts;          # time support, of duration Tw
f_n= n(1:(N/2))*df; # frequency support, here restricted to [0;fs/2[␣
 ↪(monolateral)


##### Building the test discrete time sampled signal
z_s = cos(2*pi*f1*t_s); # a simple cosine waveform signal, of frequency f1


##### DFT calculation
# 1-hard coding
for k = 1:N           # k index, as defined in eq.18. It starts from 1 instead␣
 ↪of 0 as we'll use it for indexation, which starts at 1 in Octave
    k_hat = (k-1)/N;  # k_hat, as defined in eq.18. Note the (k-1), to make the␣
 ↪definition of the variable consistent with eq.18.
    Z_s1(k) = sum(z_s.*exp(-j*2*pi*n*k_hat)); # DFT, as defined in eq.18
endfor
Z_s1 = Z_s1(1:(N/2)); # truncation of the DFT to its first N/2 elements to␣
 ↪restrict the frequency spectrum to [0;fs/2[

# 2-fft instruction
Z_s2 = fft(z_s);      # fft-based DFT
Z_s2 = Z_s2(1:(N/2)); # truncation of the DFT to its first N/2 elements to␣
 ↪restrict the frequency spectrum to [0;fs/2[


##### Displays
figure
subplot(2,2,1), stem(f_n, abs(Z_s1)),   xlabel('f_n [Hz]'), ylabel('|Z_s|'),␣
 ↪title('Hard-coded DFT')
subplot(2,2,3), plot(f_n, angle(Z_s1)), xlabel('f_n [Hz]'), ylabel('arg(Z_s)␣
 ↪[rad]')
subplot(2,2,2), stem(f_n, abs(Z_s2)),   xlabel('f_n [Hz]'), ylabel('|Z_s|'),␣
 ↪title('DFT calculated from "fft" instruction')
```

```
subplot(2,2,4), plot(f_n, angle(Z_s2)), xlabel('f_n [Hz]'), ylabel('arg(Z_s)␣
 ↪[rad]')

figure
plot(t_s, z_s, 'o-'), xlabel('time [s]'), ylabel('Signal z_s(t)'),␣
 ↪title('Temporal representation of the signal')
```

[1]:



[1]:



The modulus and phase of the hard-coded DFT from equ.35 and those returned by the *"fft"* instruction are identical, indeed.

The modulii feature a unique peak at the frequency of the cosine waveform, f_1 (here 12 Hz) in the range $\left[0; \frac{f_s}{2}\right[$ (here $[0; 256[$ Hz). However, unexpectedly, the magnitude of the peak does not match the magnitude of the Fourier series coefficient $|Z_1| = Z_1 = \frac{1}{2}$, but rather $\frac{N}{2}$ (here 256, since we have set $N = 512$), which is discussed in the section below.

V-2- Normalization: Fourier series & Fourier transform

The discrepancy between the magnitude of the modulus of the computed peaks and the theoretical, expected, one is due to the lack of **normalization** of the DFT, as computed by equ.35, or by the *"fft"* instruction.

The **normalization of the DFT is usually performed upon division of the result of the *"fft"* instruction by the number of samples $z_s(t_s)$ consists of (e.g. N).**

Doing so, the normalized DFT stands for the Fourier series coefficients of the T_w-periodic, or finite, duration sampled signal $z_s(t_s)$, whose physical units are those of $z_s(t_s)$.

However, to use the FT to evaluate the (continuous) Fourier Transform of an infinite-duration signal, whose physical units are those of the signal times time (*e.g.* $V.s$), it is necessary to use the normalizing factor $T_s = \frac{N}{T_w}$.

To normalize properly the DFT, we will remind:

- If *"fft"* is meant to compute the **Fourier series coefficients** of a T-periodic discrete time signal $z_s(t_s)$**, then the normalization is performed upon division of the result of the *"fft"* instruction by the number of samples** $z_s(t_s)$ **consists of. The physical units of Fourier series coefficients are consistently those of** $z_s(t_s)$.
- If *"fft"* is meant to compute the **DFT** of a discrete time signal $z_s(t_s)$ of finite duration T**, then the normalization is again performed upon multiplication of the result of the *"fft"* instruction by the number of samples** $z_s(t_s)$ **consists of. The physical units of Fourier series coefficients are consistently those of** $z_s(t_s)$.
- If *"fft"* is meant to compute the **Fourier Transform** of an infinite duration time signal, then **the normalization is performed upon division of the result of the *"fft"* instruction by the sampling period** $T_s = \frac{N}{T_w}$ **used to sample** $z_s(t_s)$**. The physical units of the DFT are now consistent with those of the FT of the continuous time signal** $z(t)$.
- Note that if $T_w = 1$ s, the two normalization processes are equivalent, since $\frac{1}{T_s} = \frac{N}{T_w} = N$.

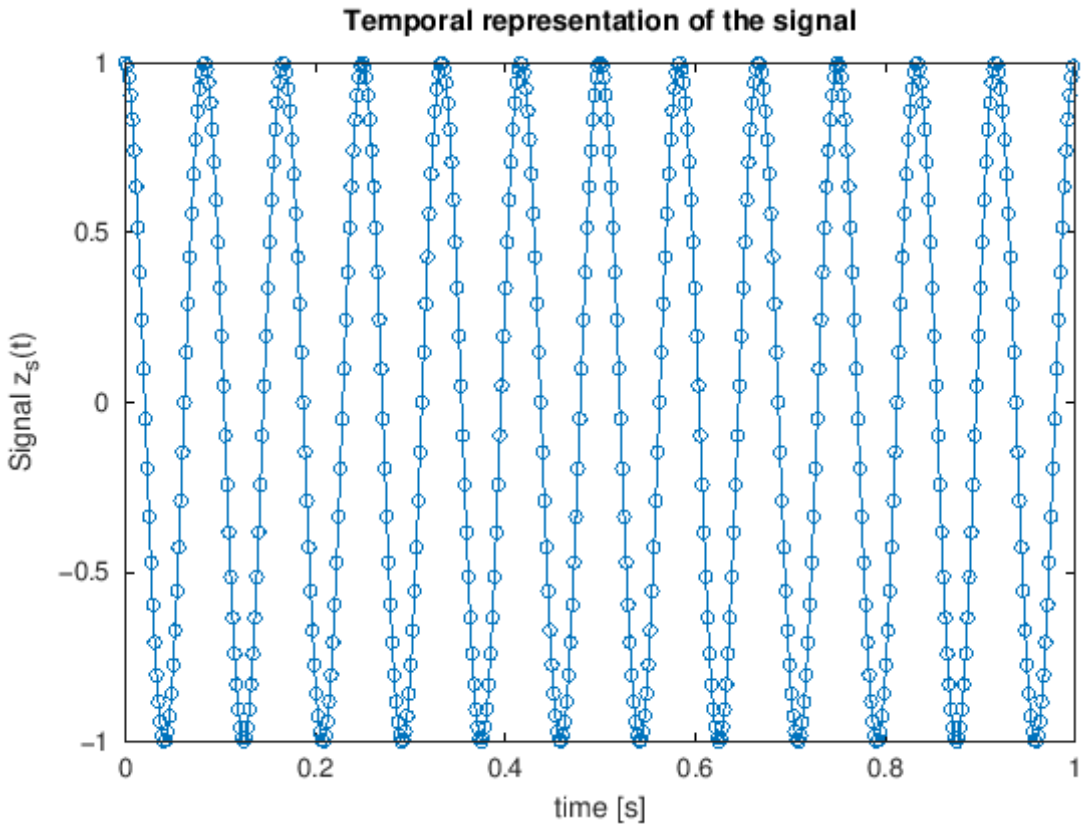
The corresponding modified Octave code showing both the properly normalized Fourier series coefficients AND DFT of the cosine waveform signal, calculated out the *"fft"* instruction only, is given below.

Illustrative code: The code below computes half the bilateral spectrum of a sinusoidal signal, windowed over an integer number of periods, and compares the two ways of normalizing the *DFT*.

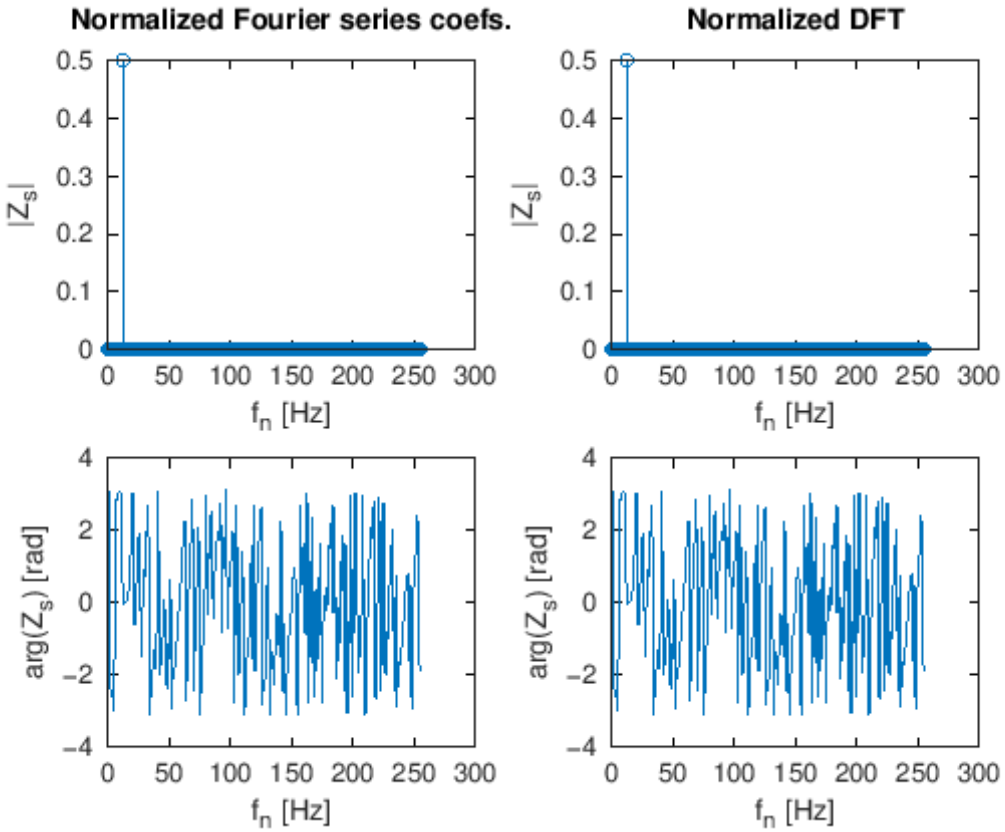
[3]:
```
# Parameters
N = 512;  # number of samples
Tw= 1;    # windowing duration in natural units, i.e. seconds

f1= 12;   # frequency of the cosine waveform signal in Hz

#In-built variables
Ts = Tw/N; # sampling period
fs = 1/Ts; # sampling frequency
df = fs/N; # spectral resolution

# support vectors
n  = 0:(N-1);       # n ranging from 0 to N-1, as defined in eq.18 e.g.
t_s= n*Ts;          # time support, of duration Tw
f_n= n(1:(N/2))*df; # frequency support, here restricted to [0;fs/2[␣
 ↪(monolateral)


##### Building the test discrete time sampled signal
z_s = cos(2*pi*f1*t_s); # a simple cosine waveform signal, of frequency f1


##### DFT calculation
# 2-fft instruction
```

```
Z_s_FSC = fft(z_s)/N;  # fft-based normalized Fourier series coefficients (FSC)
Z_s_DFT = fft(z_s)*Ts; # fft-based normalized DFT
Z_s_FSC = Z_s_FSC(1:(N/2));  # truncation of the FSC to its first N/2 elements␣
 ↪to restrict the frequency spectrum to [0;fs/2[
Z_s_DFT = Z_s_DFT(1:(N/2));  # truncation of the DFT to its first N/2 elements␣
 ↪to restrict the frequency spectrum to [0;fs/2[


##### Displays
figure
subplot(2,2,1), stem(f_n, abs(Z_s_FSC)),   xlabel('f_n [Hz]'), ylabel('|Z_s|'),␣
 ↪title('Normalized Fourier series coefs.')
subplot(2,2,3), plot(f_n, angle(Z_s_FSC)), xlabel('f_n [Hz]'), ylabel('arg(Z_s)␣
 ↪[rad]')
subplot(2,2,2), stem(f_n, abs(Z_s_DFT)),   xlabel('f_n [Hz]'), ylabel('|Z_s|'),␣
 ↪title('Normalized DFT')
subplot(2,2,4), plot(f_n, angle(Z_s_DFT)), xlabel('f_n [Hz]'), ylabel('arg(Z_s)␣
 ↪[rad]')

figure
plot(t_s, z_s, 'o-'), xlabel('time [s]'), ylabel('Signal z_s(t)'),␣
 ↪title('Temporal representation of the signal')
```

[3]:



[3]:



Conclusion: The magnitude of the modulus of the coefficient at f_1 now matches the expected magnitude $|Z_1| = Z_1 = \frac{1}{2}$. As to the DFT, its spectrum is now consistent with that of the Fourier series coefficients, too. However, this coincidence between the two normalizations is due to the fact that $T_w = T_{w_0} = 1$. If you change that value to kT_{w_0}, the amplitude of the fundamental frequency will increase by k. In fact, since we are clearly dealing with a periodic signal, there is no reason to use the normalisation by T_s in this case, and the normalization by N should therefore be used.

V-3- Bilateral spectra

Whether we deal with the Fourier series coefficients of a T-periodic or time-limited signal, or the DFT of an infinite duration signal, their spectrum is naturally bilateral and spreads, as stated above, over $\left[-\frac{f_s}{2}; \frac{f_s}{2}\right[$, or equivalently over $\left[-\frac{f_s}{2}; \frac{f_s}{2} - \delta f\right]$.

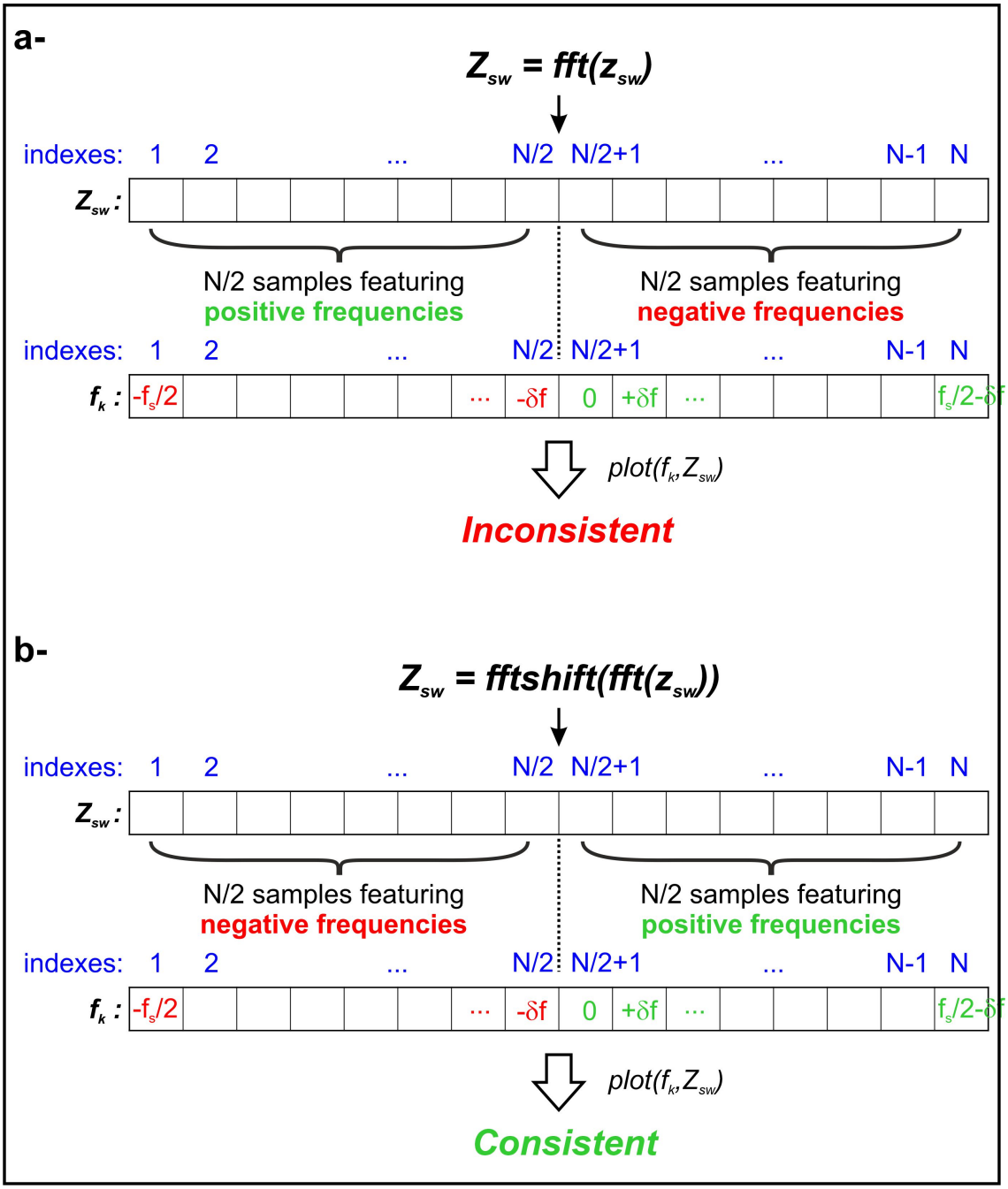
Several approaches lead to a consistent bilateral representation. Ours is a three-steps procedure:

1. The bilateral frequency range is first defined.
2. It has been stated in section III-4 that the first $N/2$ samples of the DFT stand for the positive range of frequencies whereas the second half of the samples stand for the negative range

of frequencies. In Octave, the instruction *"fftshift"* allows for swaping the second half of the elements of a vector to the first half part of it and vice versa. Therefore, applying the instruction *"fftshift"* to the DFT-calculated vector makes its representation consistent over the bilateral frequency range.

3. The bilateral spectrum can be plotted consistlently (modulus, phase, real part, imaginary part).

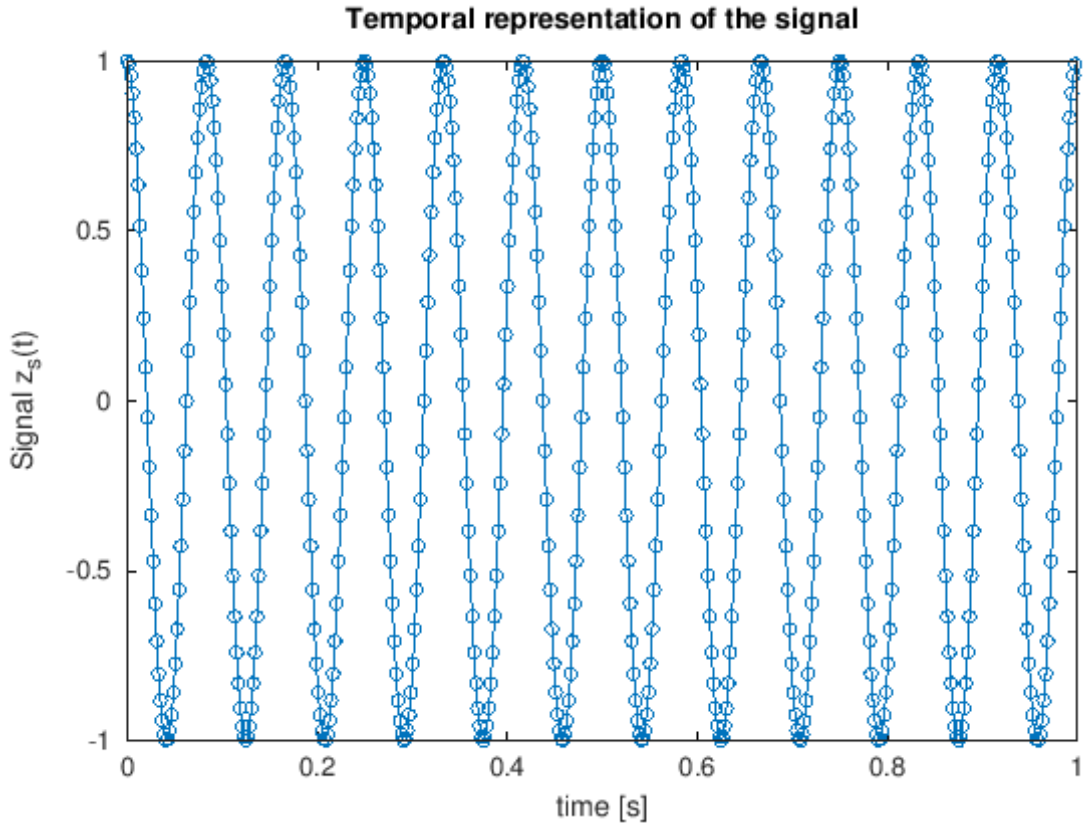
The figure and the Octave code below illustrate the bilateral representation of the Fourier series coefficients and DFT spectrum of the cosine waveform signal.



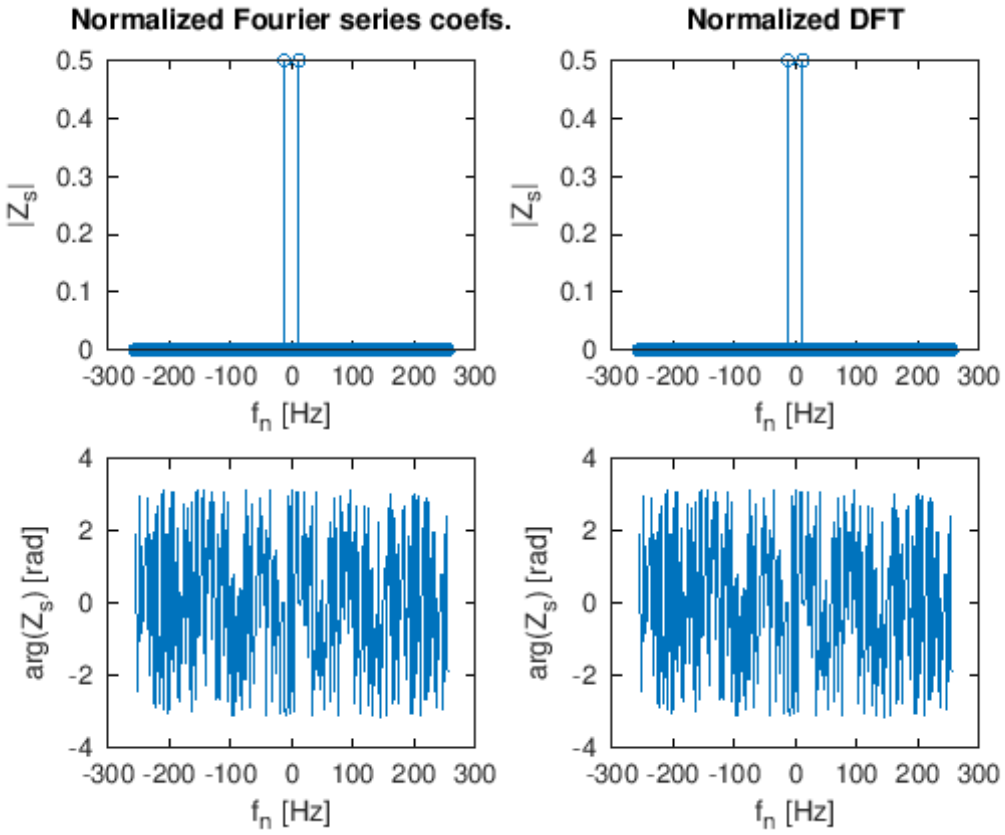
Illustrative code:

[6]:
```
# Parameters
N = 512;  # number of samples
Tw= 1;    # windowing duration in natural units, i.e. seconds

f1= 12;   # frequency of the cosine waveform signal in Hz

#In-built variables
Ts = Tw/N; # sampling period
fs = 1/Ts; # sampling frequency
df = fs/N; # spectral resolution

# support vectors
n  = 0:(N-1);          # n ranging from 0 to N-1, as defined in eq.18 e.g.
t_s= n*Ts;             # time support, of duration Tw
f_n= -fs/2:df:fs/2-df; # bilateral frequency support


##### Building the test discrete time sampled signal
z_s = cos(2*pi*f1*t_s); # a simple cosine waveform signal, of frequency f1


##### DFT calculation
# 2-fft instruction
Z_s_FSC = fftshift(fft(z_s)/N);  # fft-based normalized Fourier series␣
 ↪coefficients (FSC) to be represented over a bilateral spectrum
Z_s_DFT = fftshift(fft(z_s)*Ts); # fft-based normalized DFT to be represented␣
 ↪over a bilateral spectrum


##### Displays
figure
subplot(2,2,1), stem(f_n, abs(Z_s_FSC)),   xlabel('f_n [Hz]'), ylabel('|Z_s|'),␣
 ↪title('Normalized Fourier series coefs.')
subplot(2,2,3), plot(f_n, angle(Z_s_FSC)), xlabel('f_n [Hz]'), ylabel('arg(Z_s)␣
 ↪[rad]')
subplot(2,2,2), stem(f_n, abs(Z_s_DFT)),   xlabel('f_n [Hz]'), ylabel('|Z_s|'),␣
 ↪title('Normalized DFT')
subplot(2,2,4), plot(f_n, angle(Z_s_DFT)), xlabel('f_n [Hz]'), ylabel('arg(Z_s)␣
 ↪[rad]')

figure
plot(t_s, z_s, 'o-'), xlabel('time [s]'), ylabel('Signal z_s(t)'),␣
 ↪title('Temporal representation of the signal')
```

[6]:


Temporal representation of the signal
Signal $z_s(t)$
time [s]
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1

[6]:



The spectra have been calculated consistently and are bilateral in the range $\left[-\frac{f_s}{2};+\frac{f_s}{2}\right[$ (here $[-256;+256[$ Hz).

V-4- Phase issues

In the spectra calculated so far, the phase trace shows apparent random fluctuations. These stem from the numerical fluctuations of the imaginary and real parts of the DFT. Indeed, the phase is calculated out of the regular definition:

$$\arg(Z_{sw}) = \varphi(Z_{sw}) := \text{atan}\left(\frac{Im\{Z_{sw}\}}{Re\{Z_{sw}\}}\right) \tag{39}$$

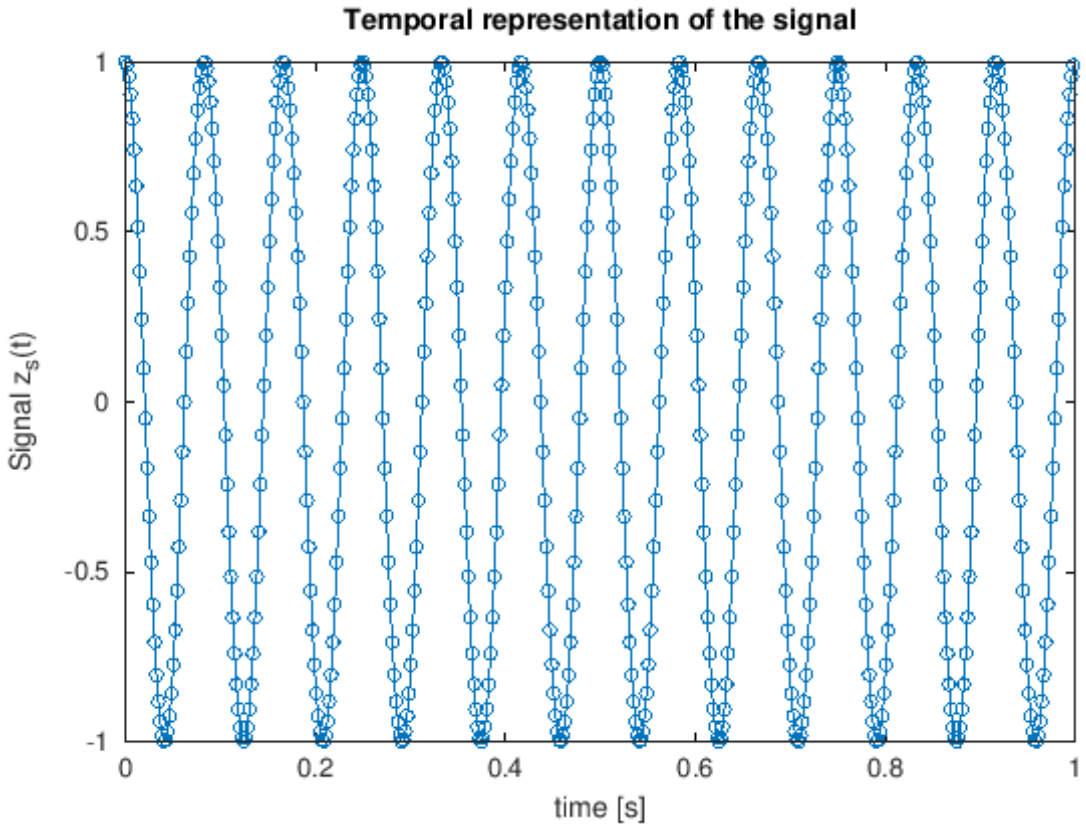
Therefore tiny fluctuations of the real part of the DFT, will introduce large fluctuations of the phase, but fluctuations of the imaginary part of the DFT can cause unwanted fluctuations of the phase too. To overcome that, we state a "zero-phase" criterion: if the modulus of a given sample of the DFT is larger than an arbitrary small value, then the sample is expected to be valuable and its phase must be computed consistently (equ.39), otherwise we force it to be zero. This can easily be implemented in Octave by using a boolean test condition when computing the phase of the DFT vector.

The Octave code below illustrates our "zero-phase" criterion on the Fourier series coefficients and DFT spectrum of the cosine waveform signal.

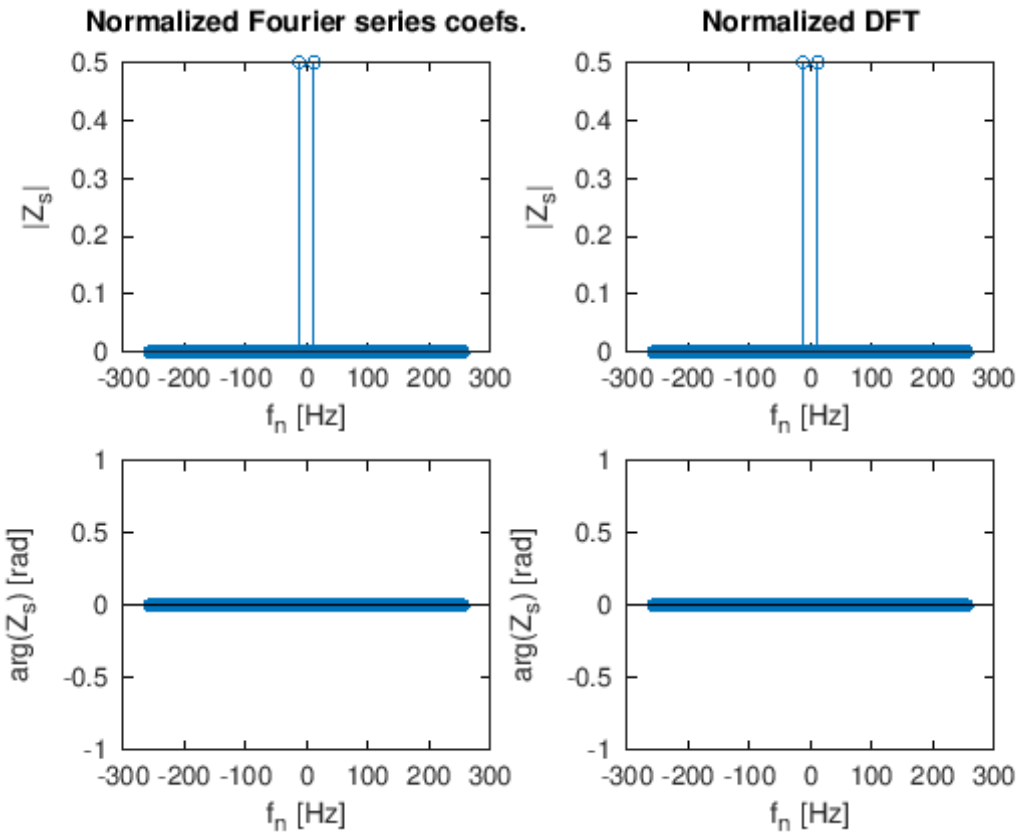
[4]:
```
# Parameters
N = 512;  # number of samples
Tw= 1;    # windowing duration in natural units, i.e. seconds

f1= 12;   # frequency of the cosine waveform signal in Hz

#In-built variables
Ts = Tw/N; # sampling period
fs = 1/Ts; # sampling frequency
df = fs/N; # spectral resolution

# support vectors
n  = 0:(N-1);          # n ranging from 0 to N-1, as defined in eq.18 e.g.
t_s= n*Ts;             # time support, of duration Tw
f_n= -fs/2:df:fs/2-df; # bilateral frequency support


##### Building the test discrete time sampled signal
z_s = cos(2*pi*f1*t_s); # a simple cosine waveform signal, of frequency f1


##### DFT calculation
# 2-fft instruction
Z_s_FSC = fftshift(fft(z_s)/N);  # fft-based normalized Fourier series␣
 ↪coefficients (FSC) to be represented over a bilateral spectrum
Z_s_DFT = fftshift(fft(z_s)*Ts); # fft-based normalized DFT to be represented␣
 ↪over a bilateral spectrum

zero_phase_criterion = 1e-8;     # arbitrary threshold w.r.t which the absolute␣
 ↪value of the imaginary part of the DFT will be compared

Phi_Z_s_FSC= angle(Z_s_FSC).*(abs(Z_s_FSC)>zero_phase_criterion); # the phase is␣
 ↪computed by the "angle" instruction, for each Fourier series coefficient␣
 ↪sample, but it is multiplied by our zero phase criterion that either outputs␣
 ↪0, or 1.
Phi_Z_s_FSC= Phi_Z_s_FSC.*(abs(Phi_Z_s_FSC) > zero_phase_criterion); # the␣
 ↪resulting phase is forced to be 0 if it is smaller than the␣
 ↪zero_phase_criterion
Phi_Z_s_DFT= angle(Z_s_DFT).*(abs(Z_s_DFT)>zero_phase_criterion); # Idem for DFT
Phi_Z_s_DFT= Phi_Z_s_DFT.*(abs(Phi_Z_s_DFT) > zero_phase_criterion);


##### Displays
figure
subplot(2,2,1), stem(f_n, abs(Z_s_FSC)), xlabel('f_n [Hz]'), ylabel('|Z_s|'),␣
 ↪title('Normalized Fourier series coefs.')
```

```
subplot(2,2,3), stem(f_n, Phi_Z_s_FSC),  xlabel('f_n [Hz]'), ylabel('arg(Z_s)␣
 ↪[rad]')
subplot(2,2,2), stem(f_n, abs(Z_s_DFT)), xlabel('f_n [Hz]'), ylabel('|Z_s|'),␣
 ↪title('Normalized DFT')
subplot(2,2,4), stem(f_n, Phi_Z_s_DFT),  xlabel('f_n [Hz]'), ylabel('arg(Z_s)␣
 ↪[rad]')

figure
plot(t_s, z_s, 'o-'), xlabel('time [s]'), ylabel('Signal z_s(t)'),␣
 ↪title('Temporal representation of the signal')
```

[4]:



[4]:



The bilateral phase spectrum now features only zeros, as expected for the cosine waveform signal.

We exemplify again the calculation of the phase, but now with a sine waveform signal of frequency f_1 over a similar windowing duration: $T_w = 1$ s:

$$z(t) = \sin(2\pi f_1 t), \forall t \in [0; 1[\tag{40}$$

whose FT is well-known, too:

$$Z(f) = \frac{1}{2}\left[\delta(f - f_1)e^{-j\frac{\pi}{2}} + \delta(f + f_1)e^{+j\frac{\pi}{2}}\right] \tag{41}$$

and whose set of Fourier series coefficients are imaginary:

$$Z_1 = Z_{-1}^* = \frac{1}{2}e^{-j\frac{\pi}{2}} \tag{42}$$

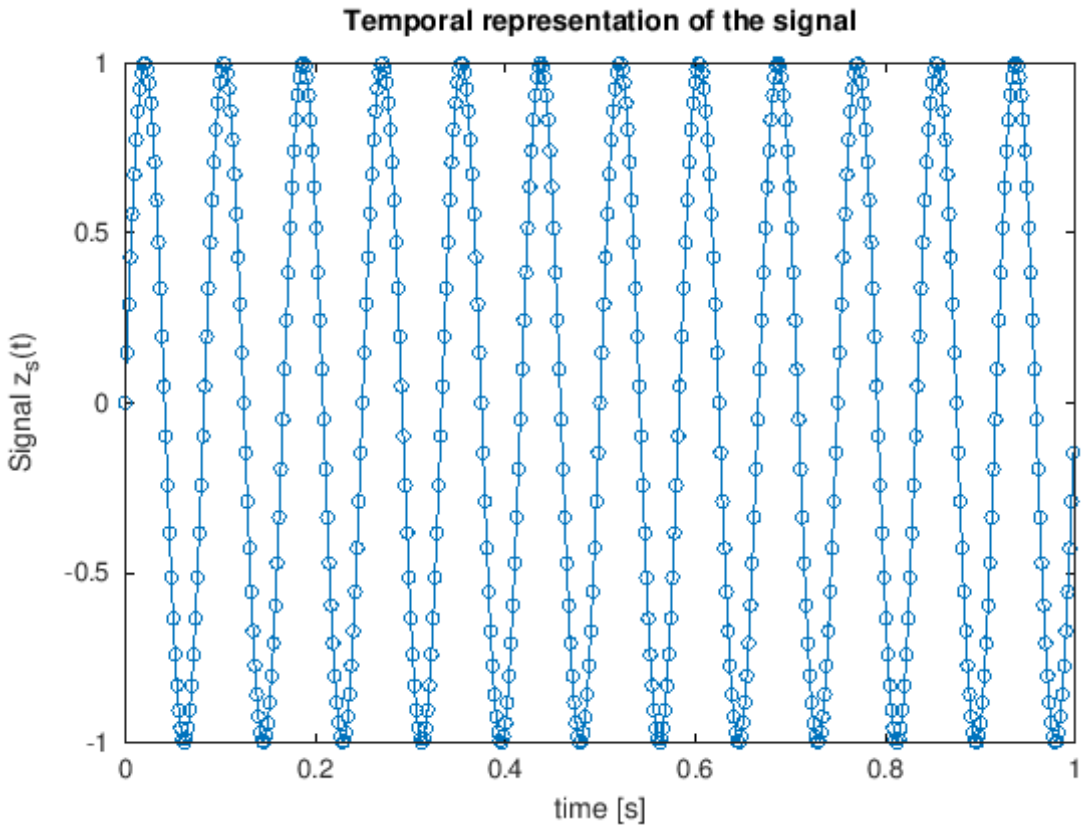
Illustrative code:

[7]:
```
# Parameters
N = 512;   # number of samples
Tw= 1;     # windowing duration in natural units, i.e. seconds
```

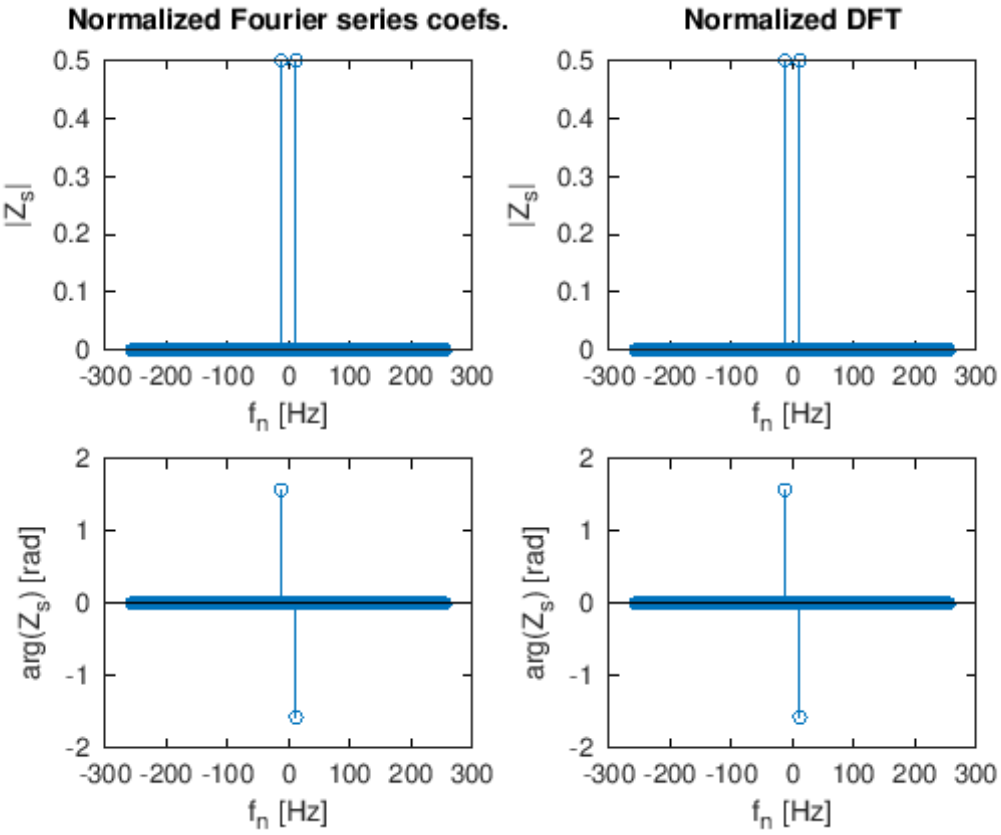
```
f1= 12;   # frequency of the cosine waveform signal in Hz

#In-built variables
Ts = Tw/N; # sampling period
fs = 1/Ts; # sampling frequency
df = fs/N; # spectral resolution

# support vectors
n  = 0:(N-1);          # n ranging from 0 to N-1, as defined in eq.18 e.g.
t_s= n*Ts;             # time support, of duration Tw
f_n= -fs/2:df:fs/2-df; # bilateral frequency support


##### Building the test discrete time sampled signal
z_s = sin(2*pi*f1*t_s); # a simple sine waveform signal, of frequency f1


##### DFT calculation
# 2-fft instruction
Z_s_FSC = fftshift(fft(z_s)/N);  # fft-based normalized Fourier series␣
 ↪coefficients (FSC) to be represented over a bilateral spectrum
Z_s_DFT = fftshift(fft(z_s)*Ts); # fft-based normalized DFT to be represented␣
 ↪over a bilateral spectrum

zero_phase_criterion = 1e-8;     # arbitrary threshold w.r.t which the absolute␣
 ↪value of the imaginary part of the DFT will be compared

Phi_Z_s_FSC= angle(Z_s_FSC).*(abs(Z_s_FSC)>zero_phase_criterion); # the phase is␣
 ↪computed by the "angle" instruction, for each Fourier series coefficient␣
 ↪sample, but it is multiplied by our zero phase criterion that either outputs␣
 ↪0, or 1.
Phi_Z_s_FSC= Phi_Z_s_FSC.*(abs(Phi_Z_s_FSC) > zero_phase_criterion); # the␣
 ↪resulting phase is forced to be 0 if it is smaller than the␣
 ↪zero_phase_criterion
Phi_Z_s_DFT= angle(Z_s_DFT).*(abs(Z_s_DFT)>zero_phase_criterion); # Idem for DFT
Phi_Z_s_DFT= Phi_Z_s_DFT.*(abs(Phi_Z_s_DFT) > zero_phase_criterion);


##### Displays
figure
subplot(2,2,1), stem(f_n, abs(Z_s_FSC)), xlabel('f_n [Hz]'), ylabel('|Z_s|'),␣
 ↪title('Normalized Fourier series coefs.')
subplot(2,2,3), stem(f_n, Phi_Z_s_FSC),  xlabel('f_n [Hz]'), ylabel('arg(Z_s)␣
 ↪[rad]')
```

```
subplot(2,2,2), stem(f_n, abs(Z_s_DFT)), xlabel('f_n [Hz]'), ylabel('|Z_s|'),␣
 ↪title('Normalized DFT')
subplot(2,2,4), stem(f_n, Phi_Z_s_DFT),  xlabel('f_n [Hz]'), ylabel('arg(Z_s)␣
 ↪[rad]')

figure
plot(t_s, z_s, 'o-'), xlabel('time [s]'), ylabel('Signal z_s(t)'),␣
 ↪title('Temporal representation of the signal')
```

[7]:



[7]:



The bilateral phase spectrum features the expected $\pm\pi/2$ values (± 1.57 rad) for the phase at the $\mp f_1$ frequencies, as expected from the theoretical calculation of the FT of that signal, which states the consistency of our phase criterion.

V-5 "Seeing" the time & frequency shift

To figure out the time shift as well as the frequency shift effect that is implictly introduced by the DFT computation, we consider a basic signal consisting in a time-shifted Dirac peak by a factor τ, hence:

$$z_\tau(t) = z(t-\tau) = \delta(t-\tau) \tag{43}$$

Considering the property of the FT regarding time-shifted signals, we have the FT pair:

$$[z_\tau(t) = z(t-\tau)] \rightleftharpoons [Z_\tau(f) = Z(f)e^{-j2\pi f\tau}] \tag{44}$$

Therefore, here, with our Dirac peak:

$$[z_\tau(t) = \delta(t-\tau)] \rightleftharpoons [Z_\tau(f) = e^{-j2\pi f\tau}] \tag{45}$$

The FT has a constant modulus of 1 and a phase scaling linearly with the frequency f, however with a slope scaling with $-\tau$. The phase slope is therefore expected to be negative with $\tau > 0$ and vice versa.

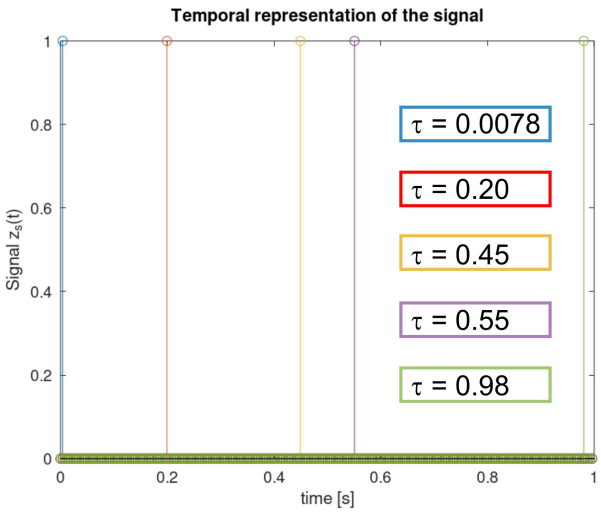
The figure and the Octave code below illustrate the time and frequency shift that the DFT computation intoduces on the above-defined time-shifted Dirac signal that we windowed and sampled over $T_w = 1$ s.

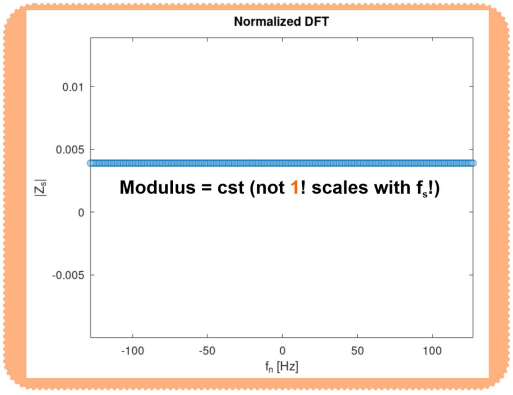
$$FT\{z_{sw}(t-\tau)\} = FT\{z_{sw}(t)\}\, e^{-j2\pi f\tau}$$
$$= Z_{sw}(f)\, e^{-j2\pi f\tau}$$
$$= |Z_{sw}(f)|\, e^{-j(\varphi(f)+2\pi f\tau)}$$

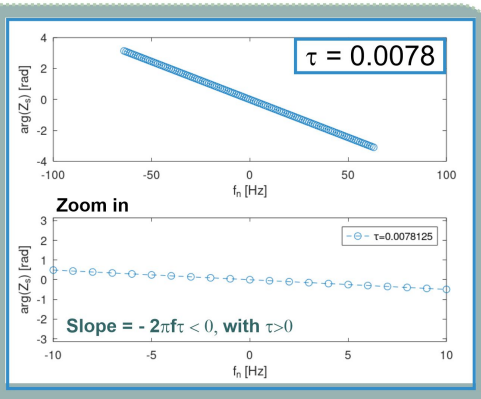
Modulus Phase

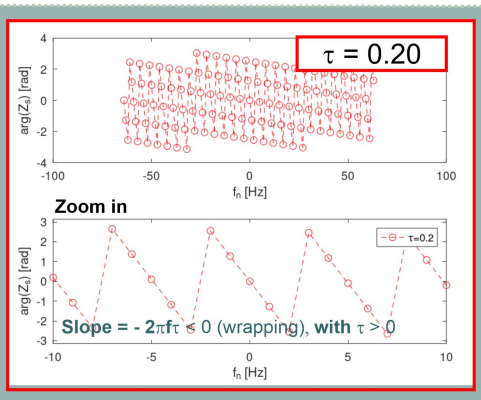
Here:

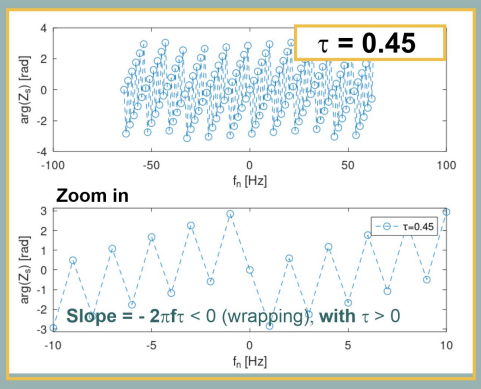
$$FT\{\delta(t-\tau)\} = 1.e^{-j2\pi f\tau}$$

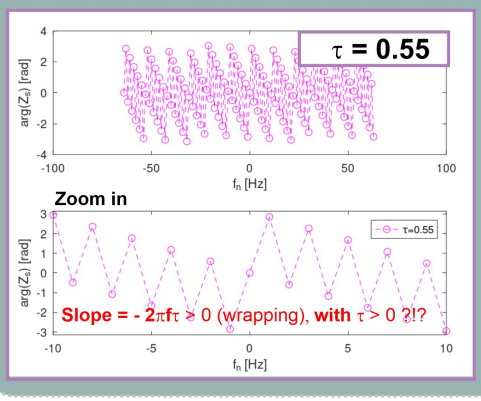


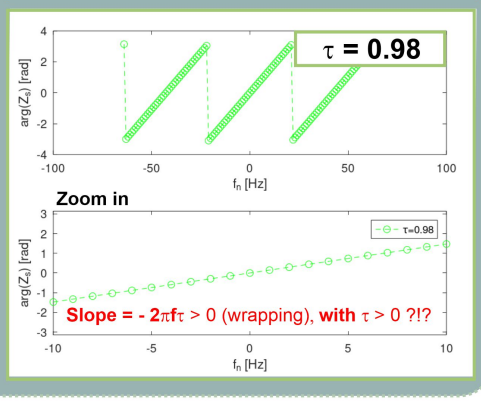












For values of τ smaller than half of the windowing duration $\frac{T_w}{2} = 0.5$, the slope of the DFT phase is negative (leaving aside the wrapping of the phase within the region $[-\pi; +\pi]$). And the slope gets more and more negative as τ increases. As τ crosses $\frac{T_w}{2}$, the slope of the DFT phase gets positive, whereas there should be no reason for that.

The only interpretation is to assume that $\tau \to -\tau$. The DFT algorithm interprets the second half of the signal ($N/2$ samples) as standing for negative times. Consequently the second half of the frequency support stands for negative frequencies. Hence the necessity to built the frequency support consistently!

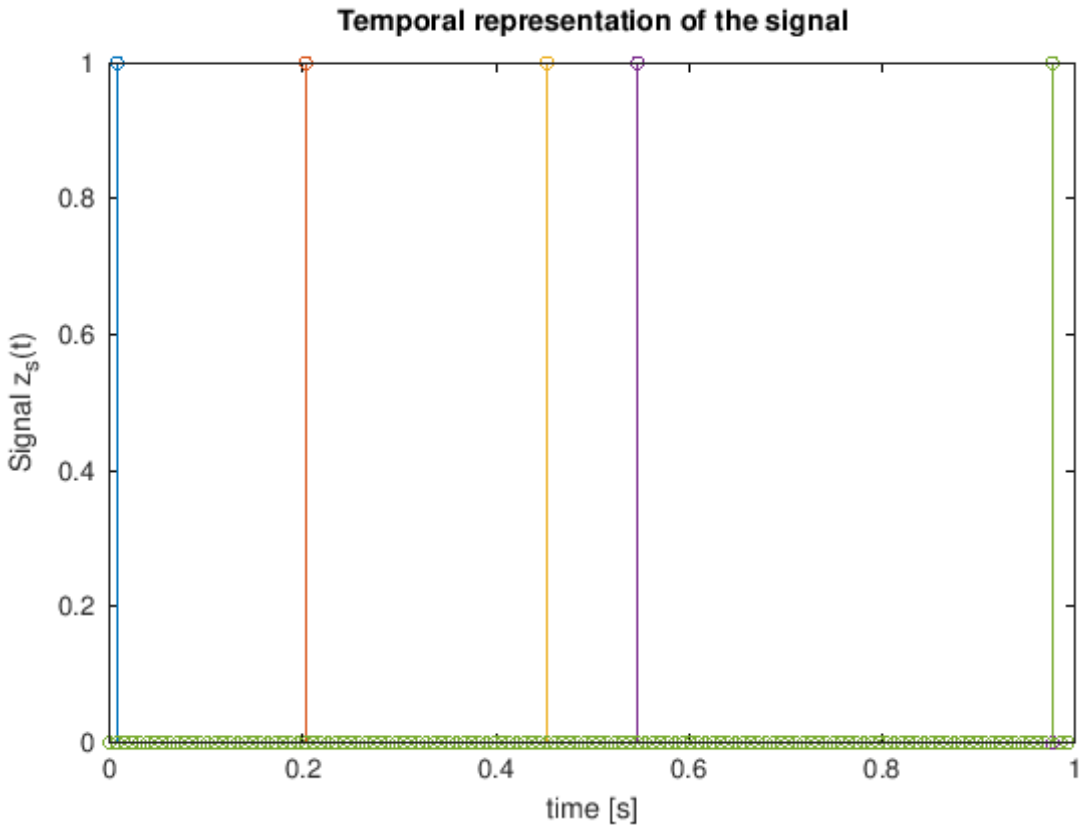
Illustrative code:

[3]:
```
# Parameters
N  = 128;  # number of samples
Tw = 1; # windowing duration now defined over one period only

width = Tw/N;
tau   = [width 0.2 0.45 0.55 0.98];
A     = 1;

#In-built variables
Ts = Tw/N; # sampling period
fs = 1/Ts; # sampling frequency
df = fs/N; # spectral resolution


# support vectors
t_s = 0:Ts:Tw-Ts;        # time support, of duration Tw
f_n = -fs/2:df:fs/2-df; # bilateral frequency support


##### Building the test discrete time sampled signal & DFT calculation
for i=1:length(tau)
  z_s(i,:) = A*(abs(t_s-tau(i))<width/2);

  ##### DFT calculation
  # 2-fft instruction
  Z_s_DFT(i,:) = fftshift(fft(z_s(i,:)))*Ts;        # fft-based normalized DFT of␣
 ↪the causal signal.

  zero_phase_criterion = 1e-8;

  Phi_Z_s_DFT(i,:) = angle(Z_s_DFT(i,:)).*(abs(Z_s_DFT(i,:)) >␣
 ↪zero_phase_criterion);
  Phi_Z_s_DFT(i,:) = Phi_Z_s_DFT(i,:).*(abs(Phi_Z_s_DFT(i,:))>␣
 ↪zero_phase_criterion);
endfor
```

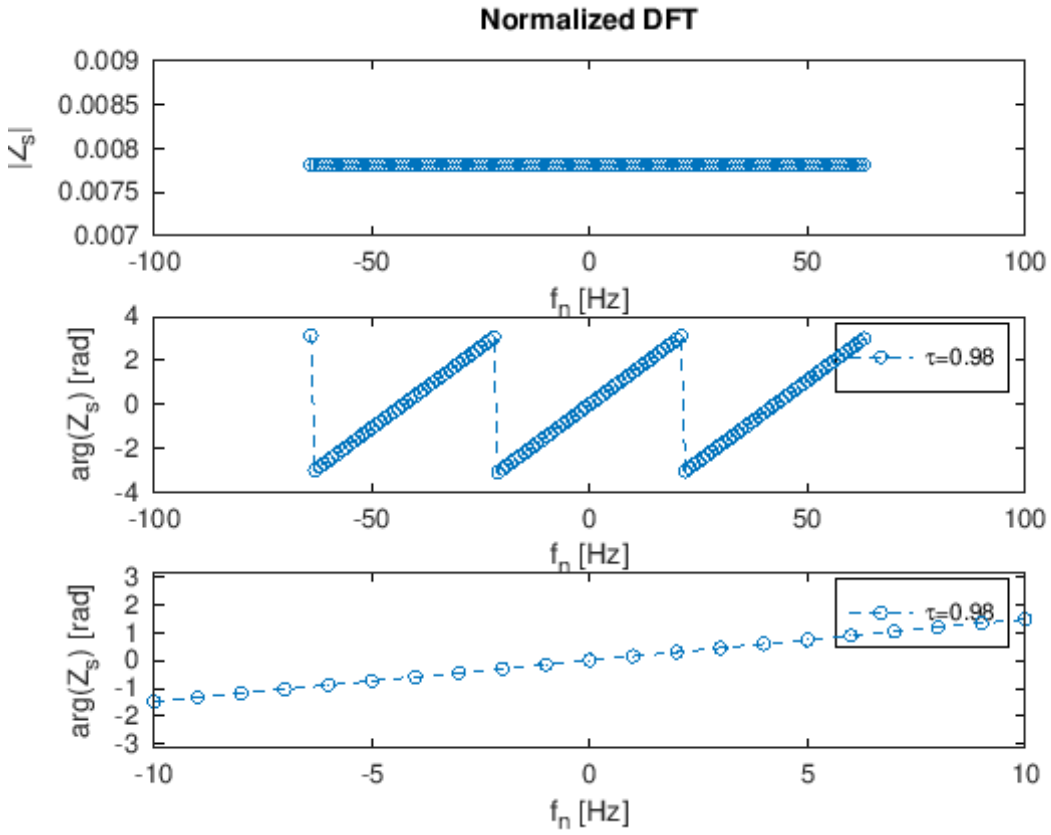
```
##### Displays
i=5; # Change that index from 1 to length(tau) to select the desired value of tau
figure
subplot(3,1,1), plot(f_n, abs(Z_s_DFT(i,:)),'o'), xlabel('f_n [Hz]'),␣
 ↪ylabel('|Z_s|'), title('Normalized DFT')
subplot(3,1,2), plot(f_n, Phi_Z_s_DFT(i,:),'o--'),  xlabel('f_n [Hz]'),␣
 ↪ylabel('arg(Z_s) [rad]')
legend(strcat('\tau=', num2str(tau(i))))
subplot(3,1,3), plot(f_n, Phi_Z_s_DFT(i,:),'o--'),  xlabel('f_n [Hz]'),␣
 ↪ylabel('arg(Z_s) [rad]')
axis([fmin fmax -pi pi])
legend(strcat('\tau=', num2str(tau(i))))


figure
for i=1:length(tau)
  stem(t_s, z_s(i,:)), hold on
endfor
xlabel('time [s]'), ylabel('Signal z_s(t)'), title('Temporal representation of␣
 ↪the signal')
```

```
ans = -63.248
ans = -48.925
```

[3]:


Temporal representation of the signal
Signal $z_s(t)$
time [s]
0
0.2
0.4
0.6
0.8
1

[3]:


Normalized DFT
|Z_s|
0.009
0.0085
0.008
0.0075
0.007
-100
-50
0
50
100
f_n [Hz]
arg(Z_s) [rad]
4
2
0
-2
-4
τ=0.98
f_n [Hz]
arg(Z_s) [rad]
3
2
1
0
-1
-2
-3
-10
-5
0
5
10
τ=0.98
f_n [Hz]

VI- Applications

Although the theoretical framework of the DFT has been established for the general case of **non-causal signals, that is signals that exist for positive as well as negative time** $t \in \left[-\frac{T_w}{2}; +\frac{T_w}{2}\right[$, so far we have treated examples of signals built as **causal signals, that is signals defined for positive time only**.

In the examples given hereafter, we focus on periodic, or non-periodic, non-causal signals.

VI-1 Fourier series coefficients of a periodic, non-causal signal

We start with the case of periodic, non-causal signals.

VI-1-a Problem positioning

Signal 1:

To that end, we consider the former T_1-periodic sine waveform signal defined over a windowing duration $T_w = 1$ s, but we arbitrary define it as:

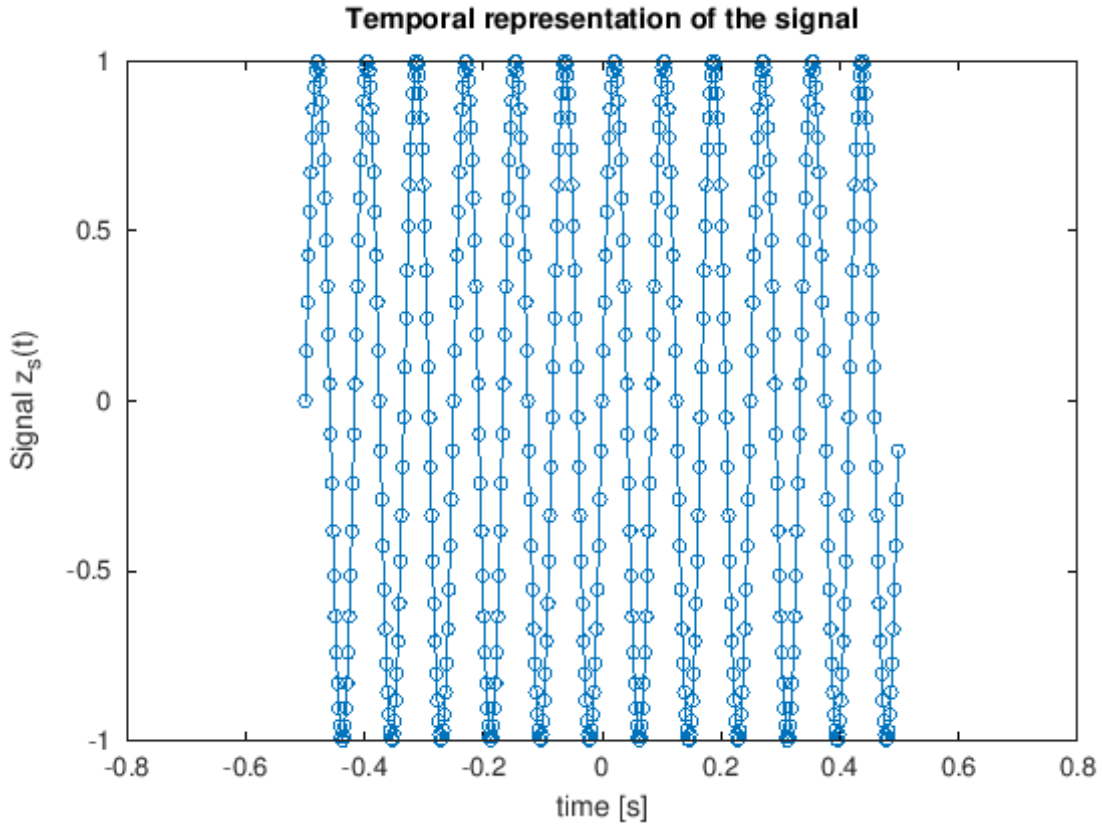
$$z(t) = \sin(2\pi f_1 t), \forall t \in \left[-\frac{1}{2}; \frac{1}{2}\right[\tag{46}$$

Hence, the non-causal nature of that signal. The code below reproduces the previous code for that signal, except that, our signal being periodic, we only focus of the Fourier series coefficients spectrum.

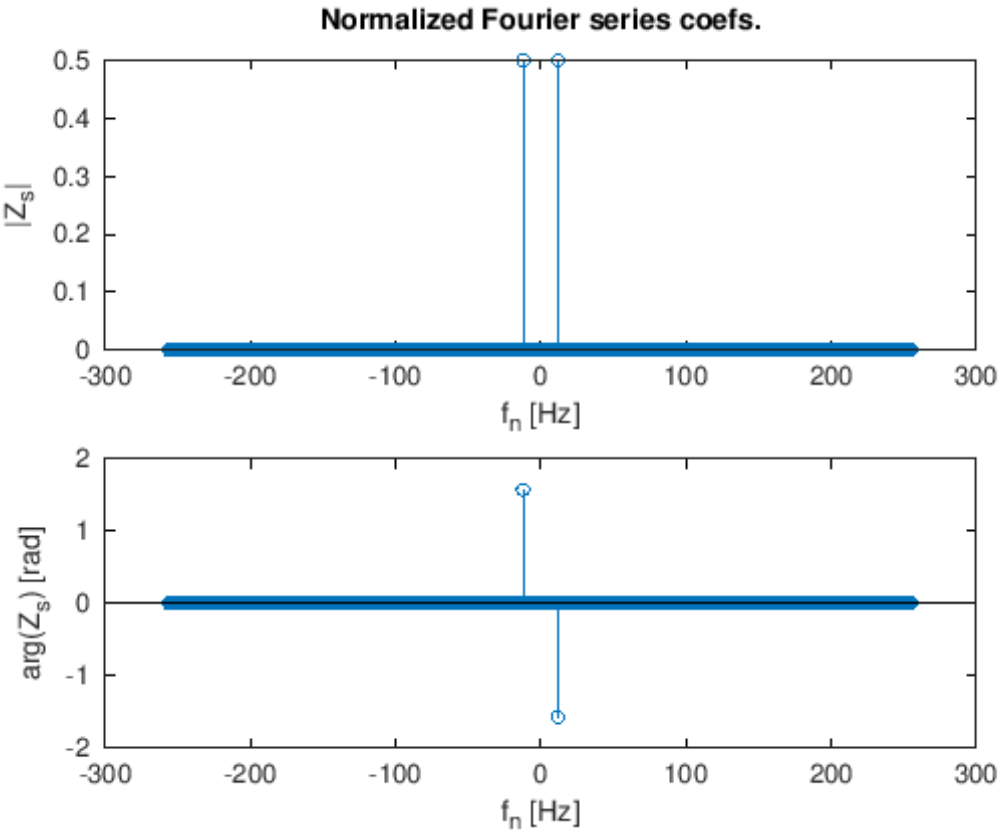
Illustrative code:

[13]:
```
# Parameters
N  = 512;  # number of samples
Tw = 1;   # windowing duration in natural units, i.e. seconds

f1= 12;   # frequency of the cosine waveform signal in Hz

#In-built variables
Ts = Tw/N; # sampling period
fs = 1/Ts; # sampling frequency
df = fs/N; # spectral resolution

# support vectors
t_s= -Tw/2:Ts:Tw/2-Ts; # time support, of duration Tw
f_n= -fs/2:df:fs/2-df; # bilateral frequency support


##### Building the test discrete time sampled signal
z_s = sin(2*pi*f1*t_s); # a simple sine waveform signal, of frequency f1


##### DFT calculation
```

[13]:
```
# 2-fft instruction
Z_s_FSC = fftshift(fft(z_s)/N);  # fft-based normalized Fourier series␣
 ↪coefficients (FSC) to be represented over a bilateral spectrum

zero_phase_criterion = 1e-8;     # arbitrary threshold w.r.t which the absolute␣
 ↪value of the imaginary part of the DFT will be compared

Phi_Z_s_FSC= angle(Z_s_FSC).*(abs(Z_s_FSC)>zero_phase_criterion); # the phase is␣
 ↪computed by the "angle" instruction, for each Fourier series coefficient␣
 ↪sample, but it is multiplied by our zero phase criterion that either outputs␣
 ↪0, or 1.
Phi_Z_s_FSC= Phi_Z_s_FSC.*(abs(Phi_Z_s_FSC) > zero_phase_criterion); # the␣
 ↪resulting phase is forced to be 0 if it is smaller than the␣
 ↪zero_phase_criterion


##### Displays
figure
subplot(2,1,1), stem(f_n, abs(Z_s_FSC)), xlabel('f_n [Hz]'), ylabel('|Z_s|'),␣
 ↪title('Normalized Fourier series coefs.')
subplot(2,1,2), stem(f_n, Phi_Z_s_FSC),  xlabel('f_n [Hz]'), ylabel('arg(Z_s)␣
 ↪[rad]')

figure
plot(t_s, z_s, 'o-'), xlabel('time [s]'), ylabel('Signal z_s(t)'),␣
 ↪title('Temporal representation of the signal')
```


Temporal representation of the signal
Signal $z_s(t)$
time [s]
1
0.5
0
-0.5
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8

[13]:



The spectrum is consistent with the one of the causal signal, as expected owing to the periodicity of the signal.

Signal 2: Let us now look at the situation where, instead of a 1 s-windowing, the windowing is now defined as $T_w = T_1 = 1/f_1$, symetrically around 0. In other words:

$$z(t) = \sin(2\pi f_1 t), \forall t \in \left[-\frac{T_1}{2}; \frac{T_1}{2}\right[\tag{47}$$

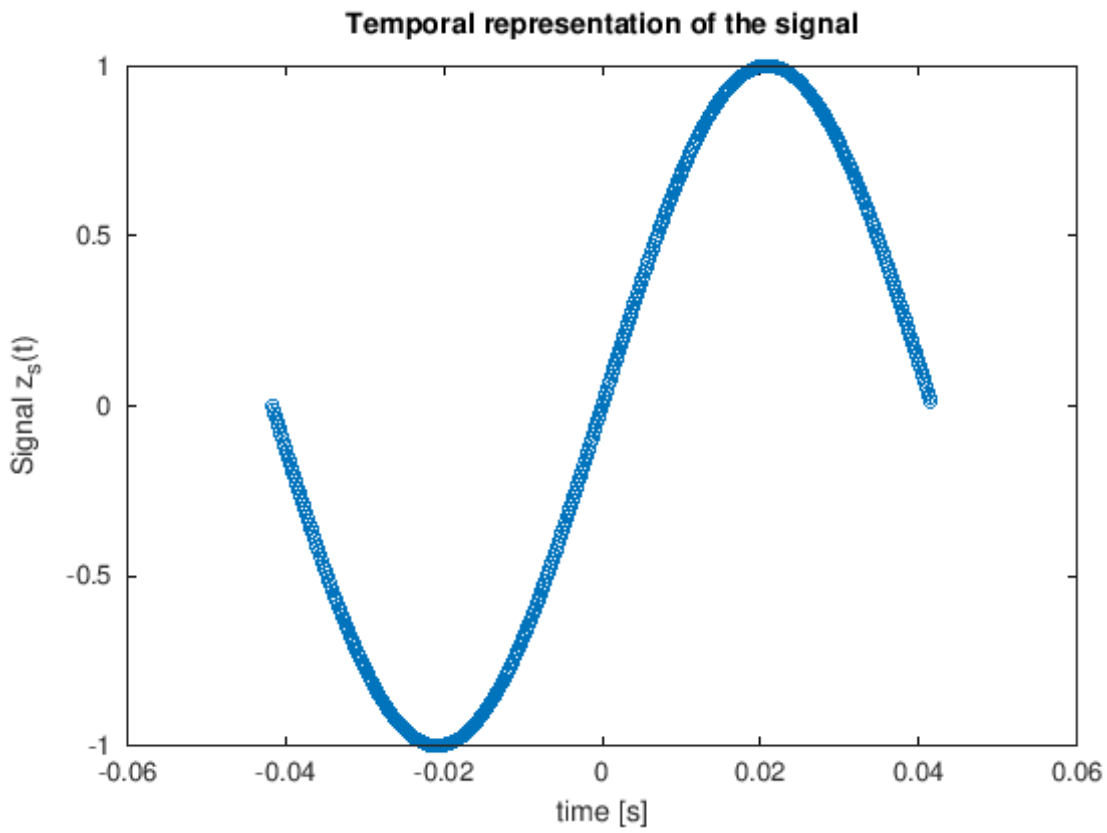
The code below exemplifies that.

Illustrative code:

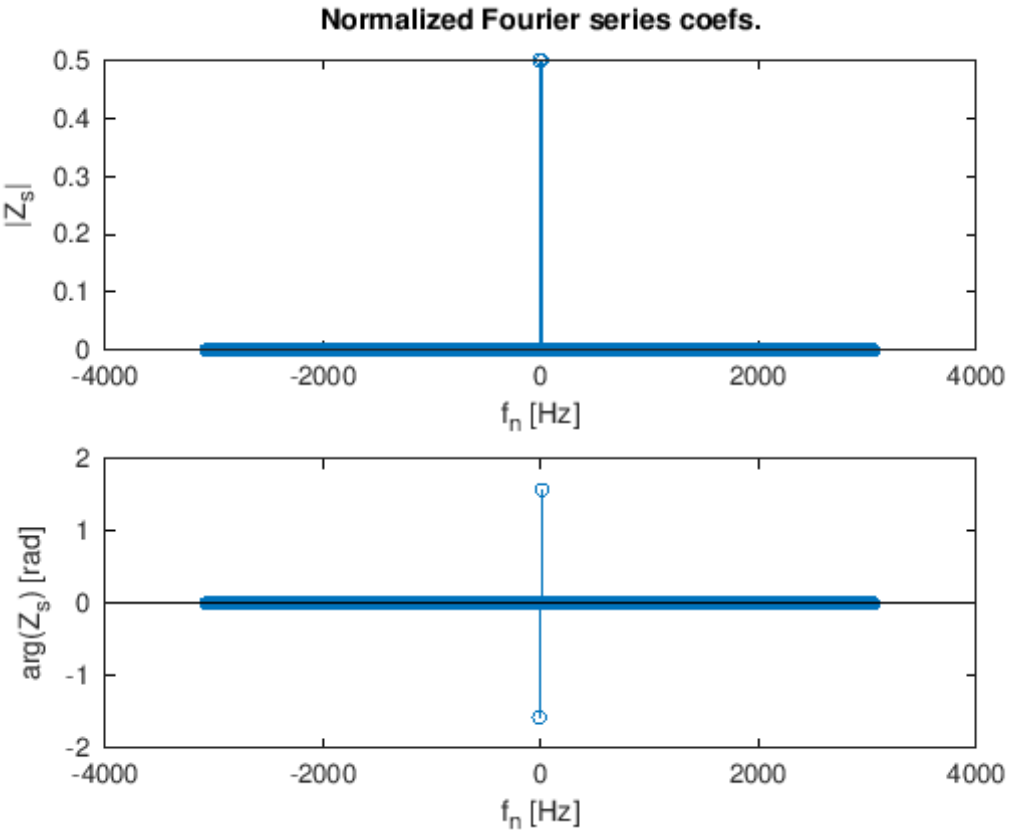
[10]:
```
# Parameters
N  = 512;  # number of samples

f1= 12;    # frequency of the cosine waveform signal in Hz
Tw = 1/f1; # windowing duration now defined over one period only

#In-built variables
Ts = Tw/N; # sampling period
fs = 1/Ts; # sampling frequency
```

```
df = fs/N; # spectral resolution

# support vectors
t_s= -Tw/2:Ts:Tw/2-Ts; # time support, of duration Tw
f_n= -fs/2:df:fs/2-df; # bilateral frequency support


##### Building the test discrete time sampled signal
z_s = sin(2*pi*f1*t_s); # a simple sine waveform signal, of frequency f1


##### DFT calculation
# 2-fft instruction
Z_s_FSC = fftshift(fft(z_s)/N);  # fft-based normalized Fourier series␣
 ↪coefficients (FSC) to be represented over a bilateral spectrum

zero_phase_criterion = 1e-8;

Phi_Z_s_FSC= angle(Z_s_FSC).*(abs(Z_s_FSC)>zero_phase_criterion); # the phase is␣
 ↪computed by the "angle" instruction, for each Fourier series coefficient␣
 ↪sample, but it is multiplied by our zero phase criterion that either outputs␣
 ↪0, or 1.
Phi_Z_s_FSC= Phi_Z_s_FSC.*(abs(Phi_Z_s_FSC) > zero_phase_criterion); # the␣
 ↪resulting phase is forced to be 0 if it is smaller than the␣
 ↪zero_phase_criterion


##### Displays
figure
subplot(2,1,1), stem(f_n, abs(Z_s_FSC)), xlabel('f_n [Hz]'), ylabel('|Z_s|'),␣
 ↪axis([-15, 15,-0.1, 0.5]), title('Zoom in')
subplot(2,1,2), stem(f_n, Phi_Z_s_FSC),  xlabel('f_n [Hz]'), ylabel('arg(Z_s)␣
 ↪[rad]'), axis([-15, 15, -pi, pi])

figure
subplot(2,1,1), stem(f_n, abs(Z_s_FSC)), xlabel('f_n [Hz]'), ylabel('|Z_s|'),␣
 ↪title('Normalized Fourier series coefs.')
subplot(2,1,2), stem(f_n, Phi_Z_s_FSC),  xlabel('f_n [Hz]'), ylabel('arg(Z_s)␣
 ↪[rad]')

figure
plot(t_s, z_s, 'o-'), xlabel('time [s]'), ylabel('Signal z_s(t)'),␣
 ↪title('Temporal representation of the signal')
```

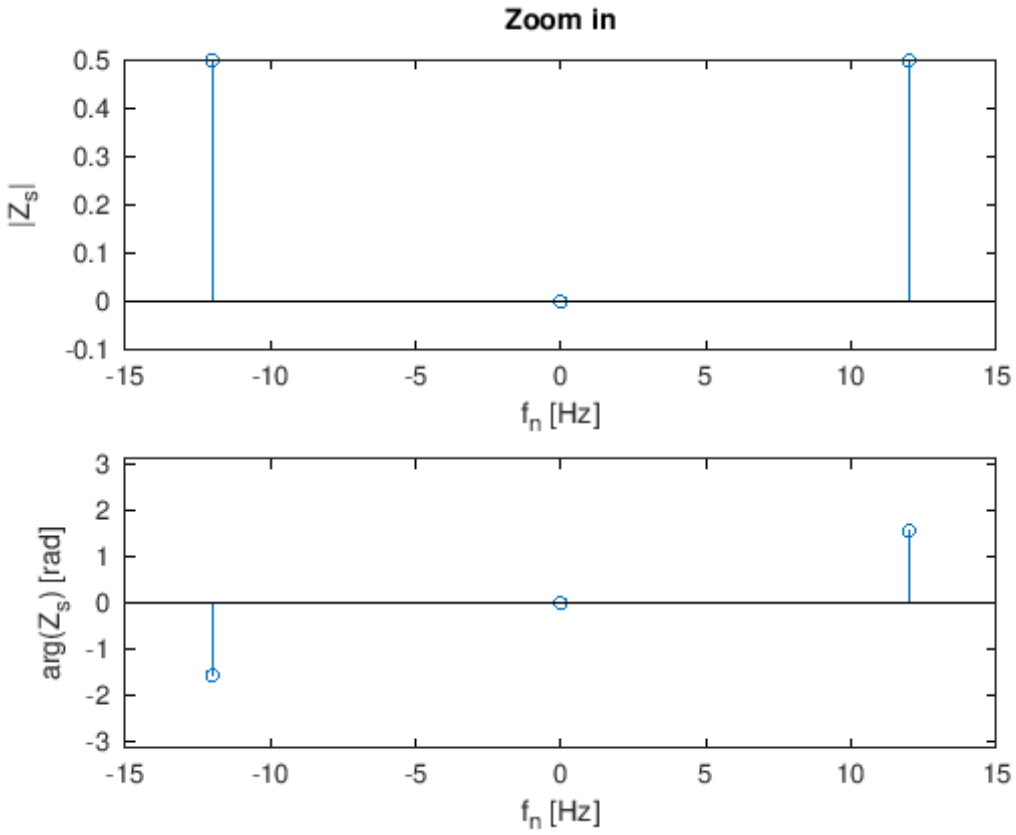
[10]:


Temporal representation of the signal
Signal $z_s(t)$
time [s]
1
0.5
0
-0.5
-1
-0.06
-0.04
-0.02
0
0.02
0.04
0.06

[10]:


Normalized Fourier series coefs.
0.5
0.4
0.3
0.2
0.1
0
|Zs|
-4000
-2000
0
2000
4000
fn [Hz]
2
1
0
-1
-2
arg(Zs) [rad]
-4000
-2000
0
2000
4000
fn [Hz]

[10]:



We have added a zoom in the $[-15; +15]$ Hz range, where three samples only are visible due to the frequency resolution of the problem $\delta f = 1/T_w = 1/T_1 = f_1 = 12$ Hz. The magnitudes of the modulii of the peaks at $\pm f_1$ are 0.5 as expected, but the phase peaks are surprinsigly reversed compared to the former calculation of the same signal. Owing to its periodicity, the spectrum should be identical, though.

VI-1-b Questions & answers *So, what is the problem?*

Answer:

The DFT does not make any assumption on the causal, or non-causal, character of the sampled signal $z_s(t_s)$. It is the user who decides whether the signal is to be plotted against a causal, or a non-causal, time interval. The DFT does not consider the actual time support $\{nT_s\}$ but only the (time) index n, which always begins at $n = 0$ (Python) or $n = 1$ (Octave). Therefore, **the structural nature of the DFT is to assume that the signal $z_s(t_{sw})$ is causal**, *i.e.* spreading over a fully positive time interval (cf. section III-5).

In the example above, the DFT algorithm therefore interprets the signal as starting from a "false reference time" $t_{frt} = 0$ (the signal "goes down") and lasting a duration $T_w = T_1$. In this case, doing so, the operation falsifies the phase of the signal at the "true reference time" $t_{trt} = 0$, as defined by the user for the temporal representation of the signal (the signal "goes up"). If the signal is interpreted as starting from $t_{frt} = 0$, it is immediately seen that the regular expression

for it is $z(t) = -\sin(2\pi f_1 t)$, and NOT $z(t) = +\sin(2\pi f_1 t)$, as we initially believed. Hence the inversion of the phase peaks in the spectrum by a factor of π.

But, why did this effect not occur when considering the intial time interval $t \in [-1/2; +1/2]$?

Answer:

Having a look to that signal, it is readily seen that, over that windowing, the phases of the signal at $t_{frt} = 0$ and $t_{trt} = 0$ are similar. Hence, the consistent calculation of the phase of the DFT.

Is there a way to treat non-causal signals without having to care about the way the windowing is performed?

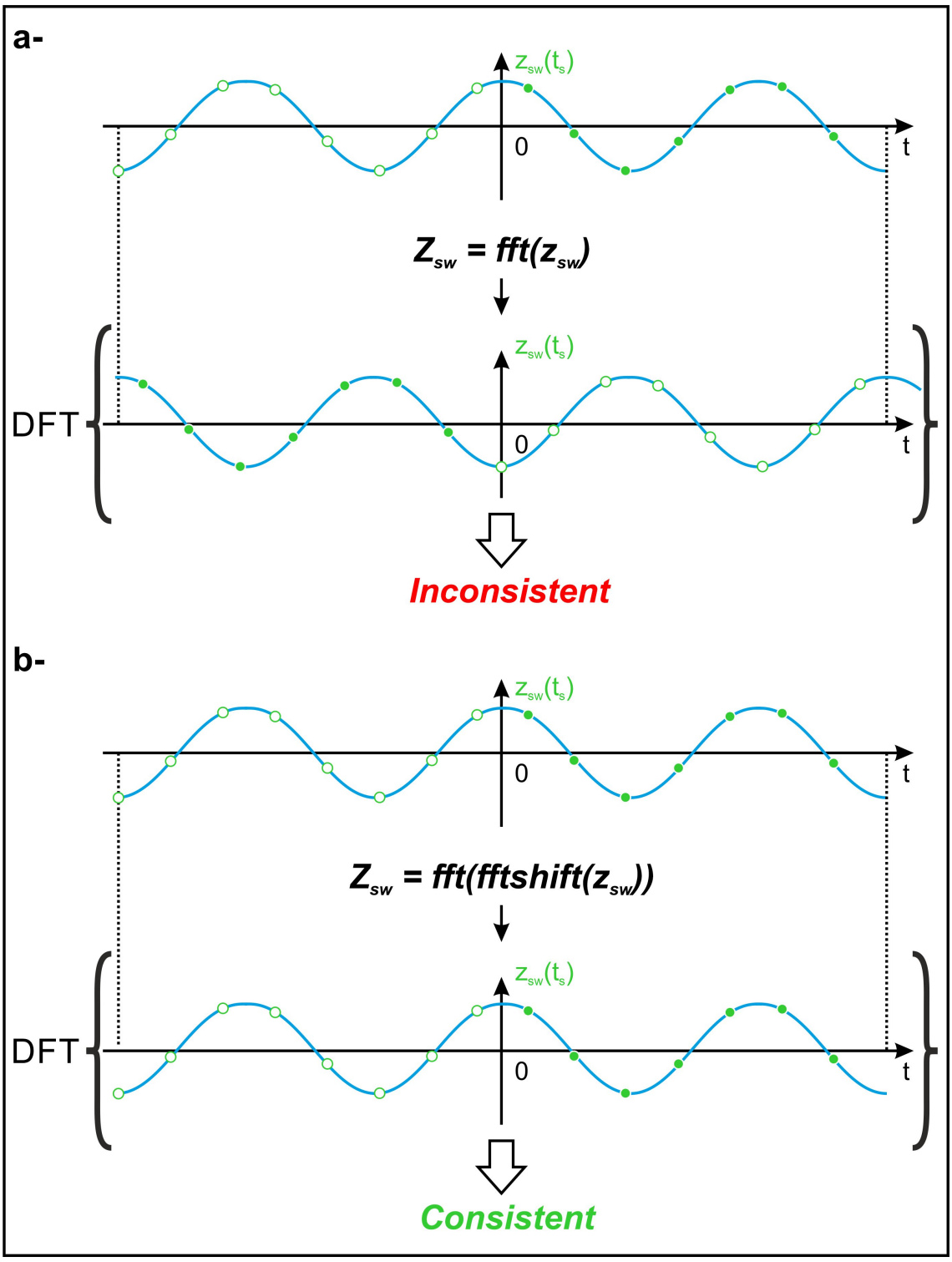
Answer:

Yes, if we know the signal is truly non-causal and that the way its DFT phase is calculated matters, we can force the non-causal character of the signal to be taken into account, as demonstrated below.

VI-1-c Forcing the non-causality to be considered in the DFT spectrum

One forces the first half of the signal $z_{sw}(t_s)$ (first $N/2$ samples) standing for the negative part of the time support to be placed at the end of signal vector. Doing so, the "false reference time" of the DFT is forced to match the "true reference time" of the signal, and the DFT can be computed consistently.

This is achieved by using again the instruction *"fftshift"*, but that is now applied to the signal $z_{sw}(t_s)$ itself, before the computation of the DFT. The figure and the code below exemplify that.



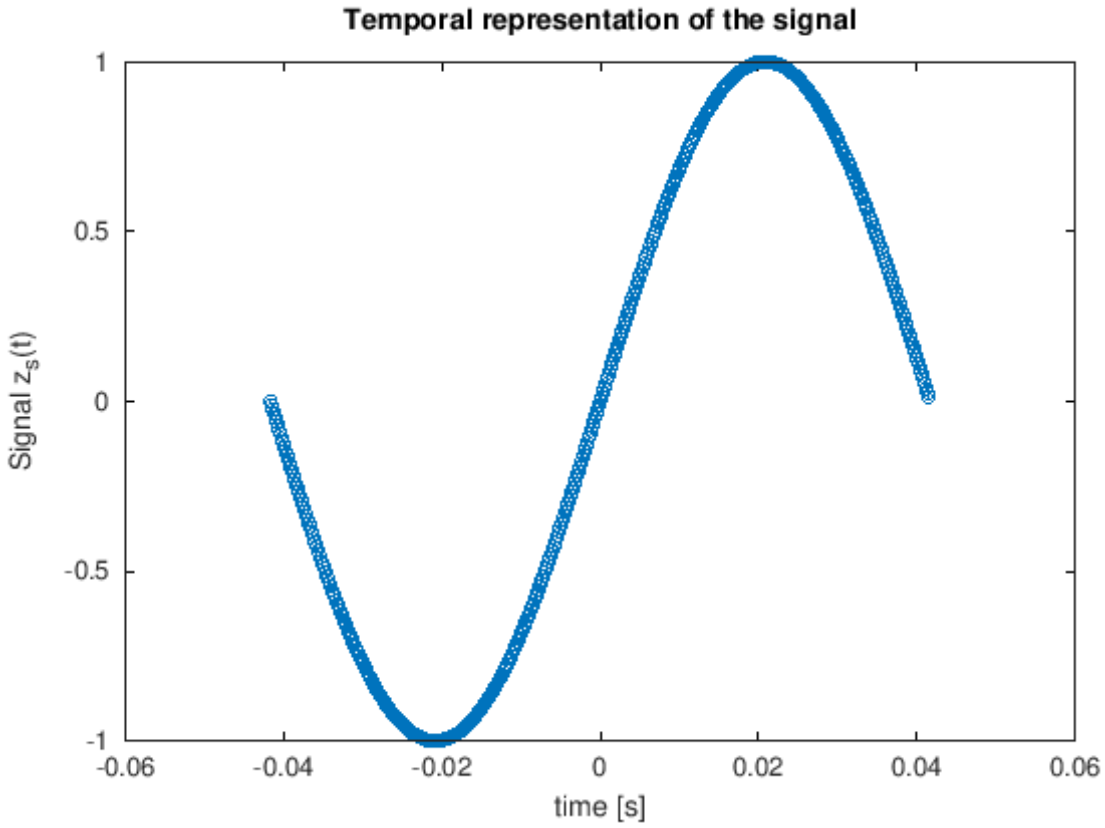
Illustrative code:

[12]:
```
# Parameters
N  = 512;  # number of samples
```

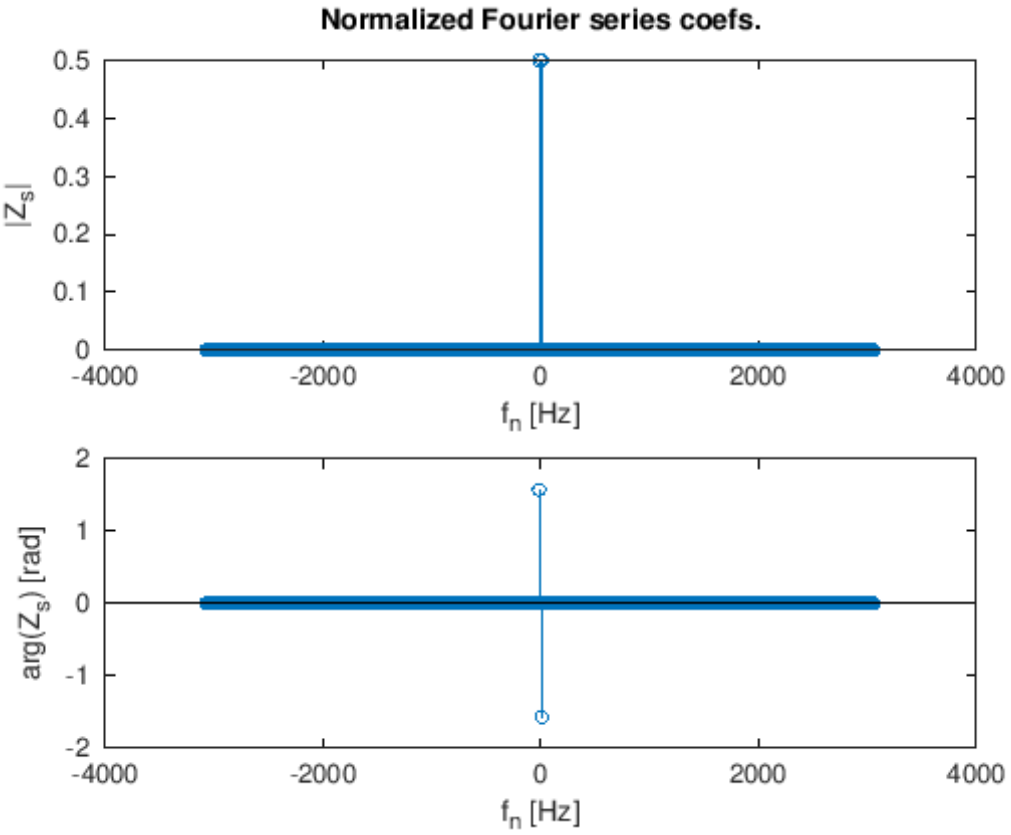
```
f1= 12;    # frequency of the cosine waveform signal in Hz
Tw = 1/f1; # windowing duration now defined over one period only

#In-built variables
Ts = Tw/N; # sampling period
fs = 1/Ts; # sampling frequency
df = fs/N; # spectral resolution

# support vectors
t_s= -Tw/2:Ts:Tw/2-Ts; # time support, of duration Tw
f_n= -fs/2:df:fs/2-df; # bilateral frequency support


##### Building the test discrete time sampled signal
z_s = sin(2*pi*f1*t_s); # a simple sine waveform signal, of frequency f1


##### DFT calculation
# 2-fft instruction
Z_s_FSC = fftshift(fft(fftshift(z_s))/N);  # fft-based normalized Fourier series␣
 ↪coefficients (FSC) to be represented over a bilateral spectrum. NOTE THE USE␣
 ↪OF FFTSHIFT ON THE SIGNAL TO ACCOUNT FOR THE NON-CAUSALITY OF THE SIGNAL

zero_phase_criterion = 1e-8;

Phi_Z_s_FSC= angle(Z_s_FSC).*(abs(Z_s_FSC)>zero_phase_criterion); # the phase is␣
 ↪computed by the "angle" instruction, for each Fourier series coefficient␣
 ↪sample, but it is multiplied by our zero phase criterion that either outputs␣
 ↪0, or 1.
Phi_Z_s_FSC= Phi_Z_s_FSC.*(abs(Phi_Z_s_FSC) > zero_phase_criterion); # the␣
 ↪resulting phase is forced to be 0 if it is smaller than the␣
 ↪zero_phase_criterion


##### Displays
figure
subplot(2,1,1), stem(f_n, abs(Z_s_FSC)), xlabel('f_n [Hz]'), ylabel('|Z_s|'),␣
 ↪axis([-15, 15,-0.1, 0.5]), title('Zoom in')
subplot(2,1,2), stem(f_n, Phi_Z_s_FSC),  xlabel('f_n [Hz]'), ylabel('arg(Z_s)␣
 ↪[rad]'), axis([-15, 15, -pi, pi])

figure
subplot(2,1,1), stem(f_n, abs(Z_s_FSC)), xlabel('f_n [Hz]'), ylabel('|Z_s|'),␣
 ↪title('Normalized Fourier series coefs.')
subplot(2,1,2), stem(f_n, Phi_Z_s_FSC),  xlabel('f_n [Hz]'), ylabel('arg(Z_s)␣
 ↪[rad]')
```

```
figure
plot(t_s, z_s, 'o-'), xlabel('time [s]'), ylabel('Signal z_s(t)'),␣
 ↪title('Temporal representation of the signal')
```

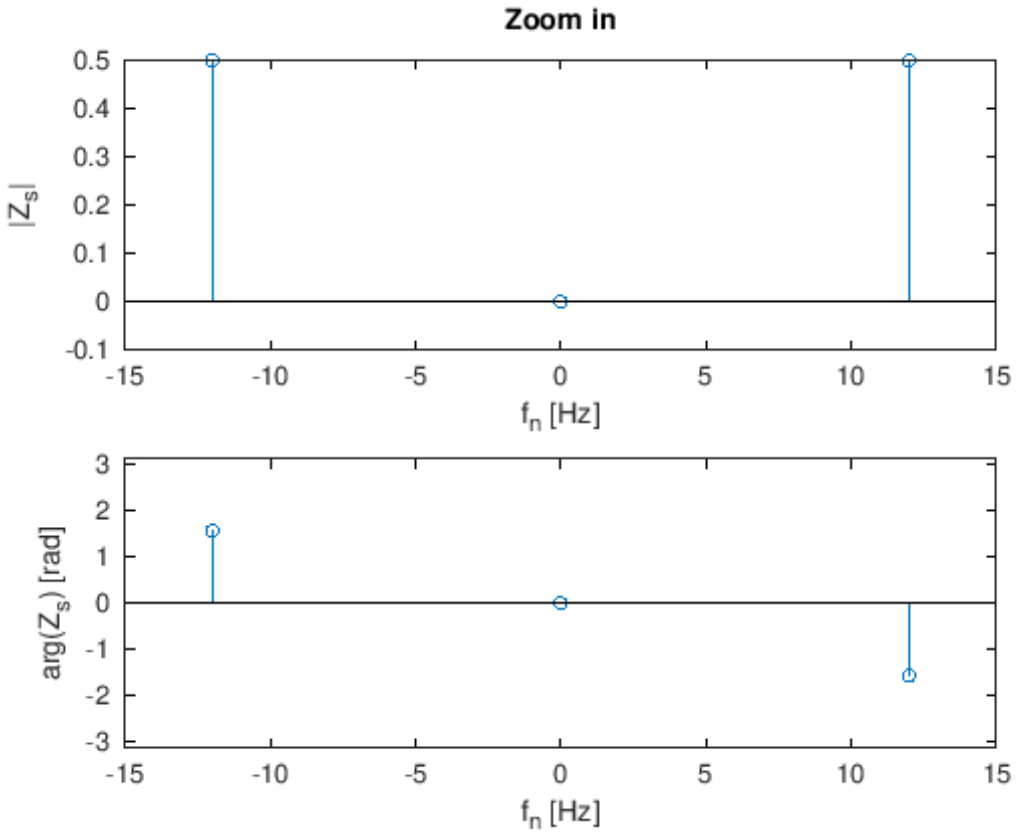
[12]:



[12]:


Normalized Fourier series coefs.
0.5
0.4
0.3
0.2
0.1
0
|Zs|
-4000
-2000
0
2000
4000
fn [Hz]
2
1
0
-1
-2
arg(Zs) [rad]
-4000
-2000
0
2000
4000
fn [Hz]

[12]:



The phase is now represented consistently.

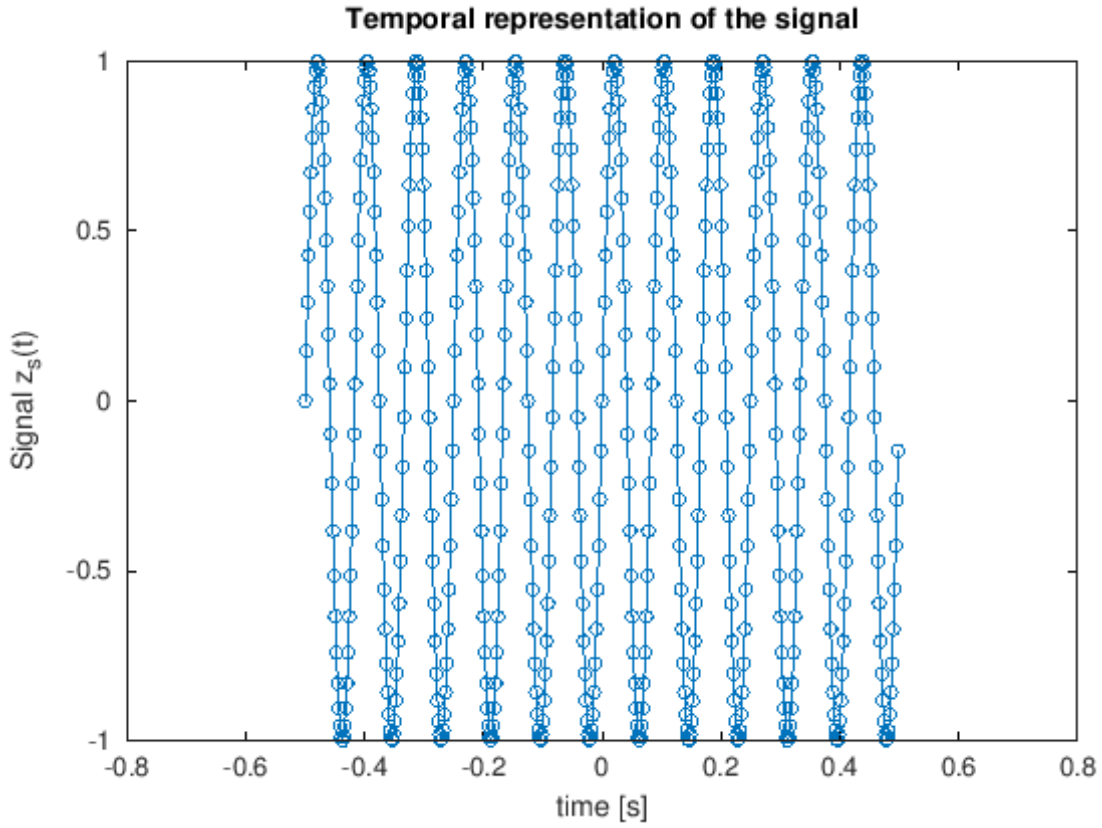
Obviously, that use of *"fftshift"* on a well-windowed non-causal signal, as initially defined (equ.43), does not modify its DFT, as shown in the code below.

Illustrative code:

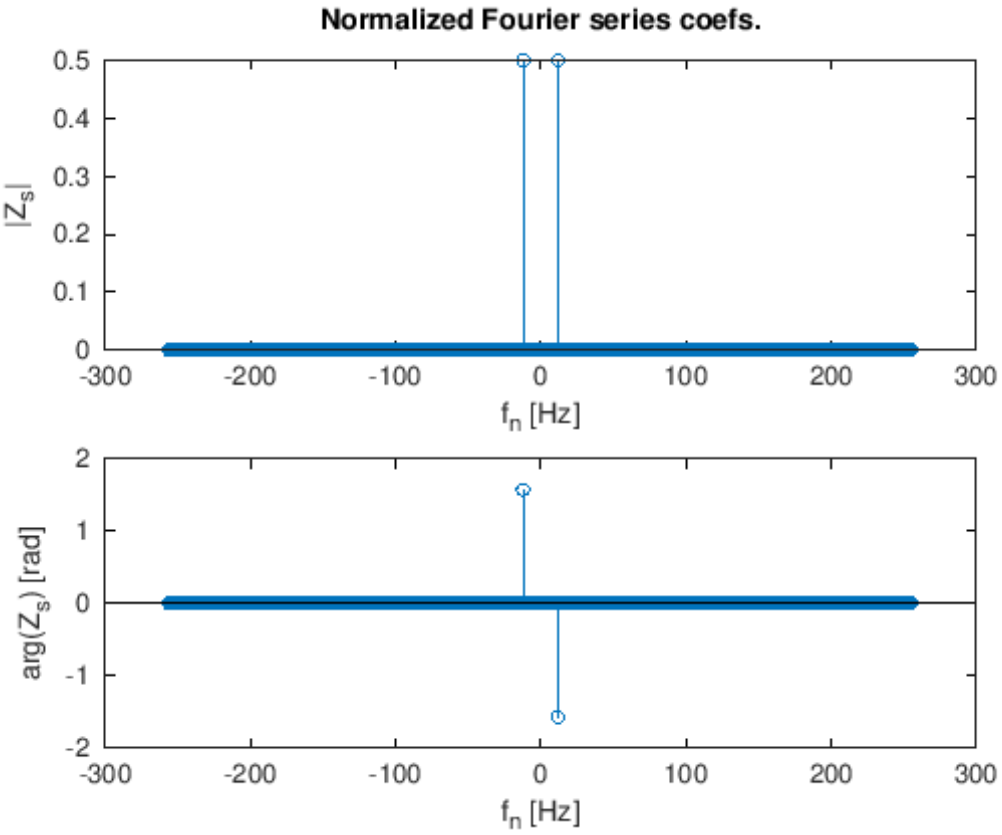
[5]:
```
# Parameters
N  = 512;  # number of samples
Tw = 1;    # windowing duration in natural units, i.e. seconds

f1= 12;    # frequency of the cosine waveform signal in Hz

#In-built variables
Ts = Tw/N; # sampling period
fs = 1/Ts; # sampling frequency
df = fs/N; # spectral resolution

# support vectors
t_s= -Tw/2:Ts:Tw/2-Ts; # time support, of duration Tw
f_n= -fs/2:df:fs/2-df; # bilateral frequency support
```

```
##### Building the test discrete time sampled signal
z_s = sin(2*pi*f1*t_s); # a simple sine waveform signal, of frequency f1


##### DFT calculation
# 2-fft instruction
Z_s_FSC = fftshift(fft(fftshift(z_s))/N);  # fft-based normalized Fourier series␣
 ↪coefficients (FSC) to be represented over a bilateral spectrum. NOTE THE USE␣
 ↪OF FFTSHIFT ON THE SIGNAL TO ACCOUNT FOR THE NON-CAUSALITY OF THE SIGNAL

zero_phase_criterion = 1e-8;     # arbitrary threshold w.r.t which the absolute␣
 ↪value of the imaginary part of the DFT will be compared

Phi_Z_s_FSC= angle(Z_s_FSC).*(abs(Z_s_FSC)>zero_phase_criterion); # the phase is␣
 ↪computed by the "angle" instruction, for each Fourier series coefficient␣
 ↪sample, but it is multiplied by our zero phase criterion that either outputs␣
 ↪0, or 1.
Phi_Z_s_FSC= Phi_Z_s_FSC.*(abs(Phi_Z_s_FSC) > zero_phase_criterion); # the␣
 ↪resulting phase is forced to be 0 if it is smaller than the␣
 ↪zero_phase_criterion


##### Displays
figure
subplot(2,1,1), stem(f_n, abs(Z_s_FSC)), xlabel('f_n [Hz]'), ylabel('|Z_s|'),␣
 ↪title('Normalized Fourier series coefs.')
subplot(2,1,2), stem(f_n, Phi_Z_s_FSC),  xlabel('f_n [Hz]'), ylabel('arg(Z_s)␣
 ↪[rad]')

figure
plot(t_s, z_s, 'o-'), xlabel('time [s]'), ylabel('Signal z_s(t)'),␣
 ↪title('Temporal representation of the signal')
```

[5]:


Temporal representation of the signal
Signal $z_s(t)$
time [s]
1
0.5
0
-0.5
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8

[5]:



VI-2 DFT of a non-periodic signal

We now consider the more generic case of non-periodic signals, hence the use of the FT formalism.

VI-2-a Causal signal

Signal 3:

For that application, we consider a causal, time-limited but non-periodic (because of the random noise) signal that is built as follows. We consider the realistic situation of a relevant electrical signal, perturbed by an ambient electrical noise. The relevant signal is supposed to be a sinusoidal signal of frequency 17 Hz. The noise results from two components: a sinusoidal oscillation, whose frequency and amplitude are arbitrarily set to 50 Hz and 0.1 V, respectively, and a uniform random background (white noise), whose magnitude equals that of the 50 Hz oscillation. The amplitude of the relevant signal is 5 times smaller than that of the 50 Hz oscillation. Hence, the relevant signal is "drowned in noise".

The signal is windowed over $T_w = 1$ s. The code below shows how powerfull the DFT is to detect the spectral components of the noisy signal.

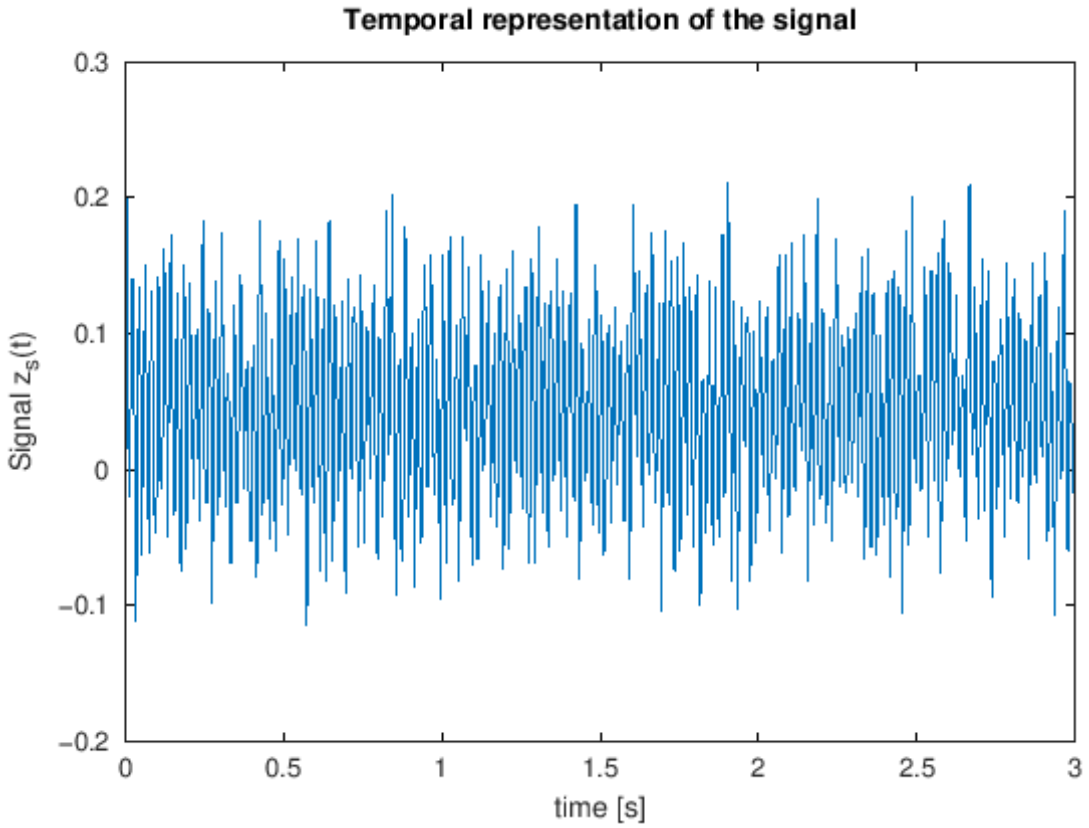
Illustrative code:

[4]:
```
# Parameters
N  = 512;  # number of samples
Tw = 3;    # windowing duration now defined over one period only

f1 = 50;    # frequency of the 1st signal component in Hz
f2 = 17;    # frequency of the 2nd signal component in Hz
mag_s1 = 0.1;      # magnitude of the 1st signal component, in V
mag_s2 = mag_s1/5; # magnitude of the 2nd signal component, in V

#In-built variables
Ts = Tw/N; # sampling period
fs = 1/Ts; # sampling frequency
df = fs/N; # spectral resolution

noise=rand(1,N)*mag_s1;

# support vectors
n   = 0:N-1;
t_s = n*Ts; # time support, of duration Tw
f_n = -fs/2:df:fs/2-df; # bilateral frequency support
tau = Tw/2;

##### Building the test discrete time sampled signal
z_s = mag_s2*sin(2*pi*f2*t_s)+mag_s1*sin(2*pi*f1*t_s)+noise; # noisy signal



##### DFT calculation
# 2-fft instruction
Z_s_DFT = fftshift(fft(z_s))/N;  # fft-based normalized DFT to be represented␣
 ↪over a bilateral spectrum.
disp(max(abs(Z_s_DFT)))   # JMT

zero_phase_criterion = 1e-8;

Phi_Z_s_DFT= angle(Z_s_DFT).*(abs(Z_s_DFT)>zero_phase_criterion); # Idem for DFT
Phi_Z_s_DFT= Phi_Z_s_DFT.*(abs(Phi_Z_s_DFT) > zero_phase_criterion);


##### Displays
figure
subplot(2,1,1), semilogy(f_n, abs(Z_s_DFT)), xlabel('f_n [Hz]'),␣
 ↪ylabel('|Z_s|'), title('Normalized DFT')
subplot(2,1,2), plot(f_n, Phi_Z_s_DFT),  xlabel('f_n [Hz]'), ylabel('arg(Z_s)␣
 ↪[rad]')

figure
```

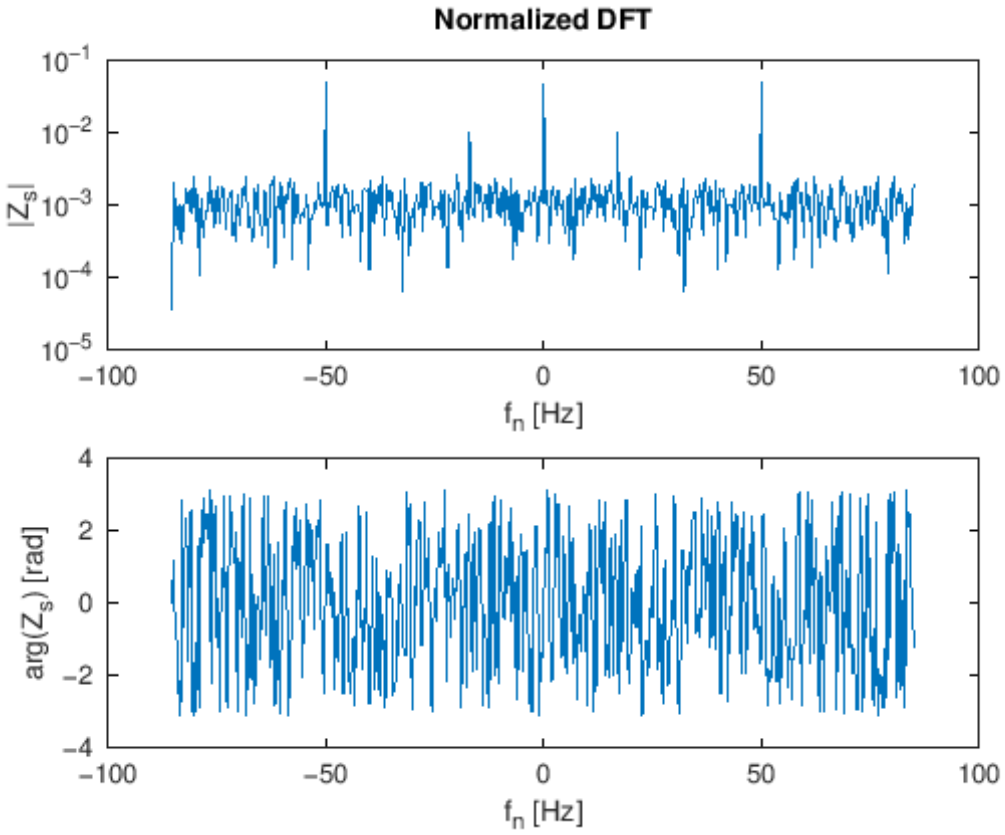
```
plot(t_s, z_s), xlabel('time [s]'), ylabel('Signal z_s(t)'), title('Temporal␣
 ↪representation of the signal')
```

```
0.050418
```

[4]:



[4]:



In the bilateral spectrum of the modulus of the DFT, the 50 Hz components of the noise are well-visible, but the 17 Hz components of the actual signal too, whereas these are not really identifiable in the temporal representation of the signal (50 Hz only).

VI-2-b Non-causal signal

Signal 4:

In this application, we consider the important case of the rectangle function, a typical example of non-causal signal.

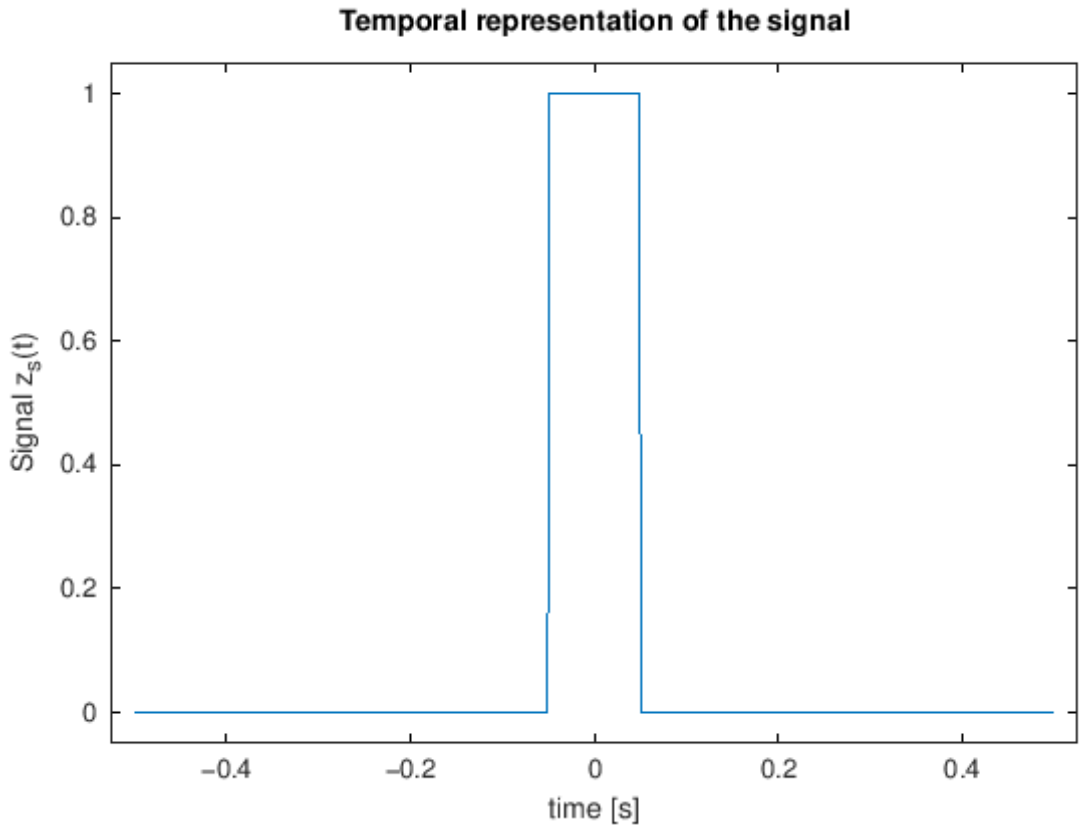
Illustrative code:

[1]:
```
# Parameters
N  = 512;  # number of samples
Tw = 1;    # windowing duration

width = Tw/10;
tau   = 0;
A     = 1;

#In-built variables
Ts = Tw/N; # sampling period
```

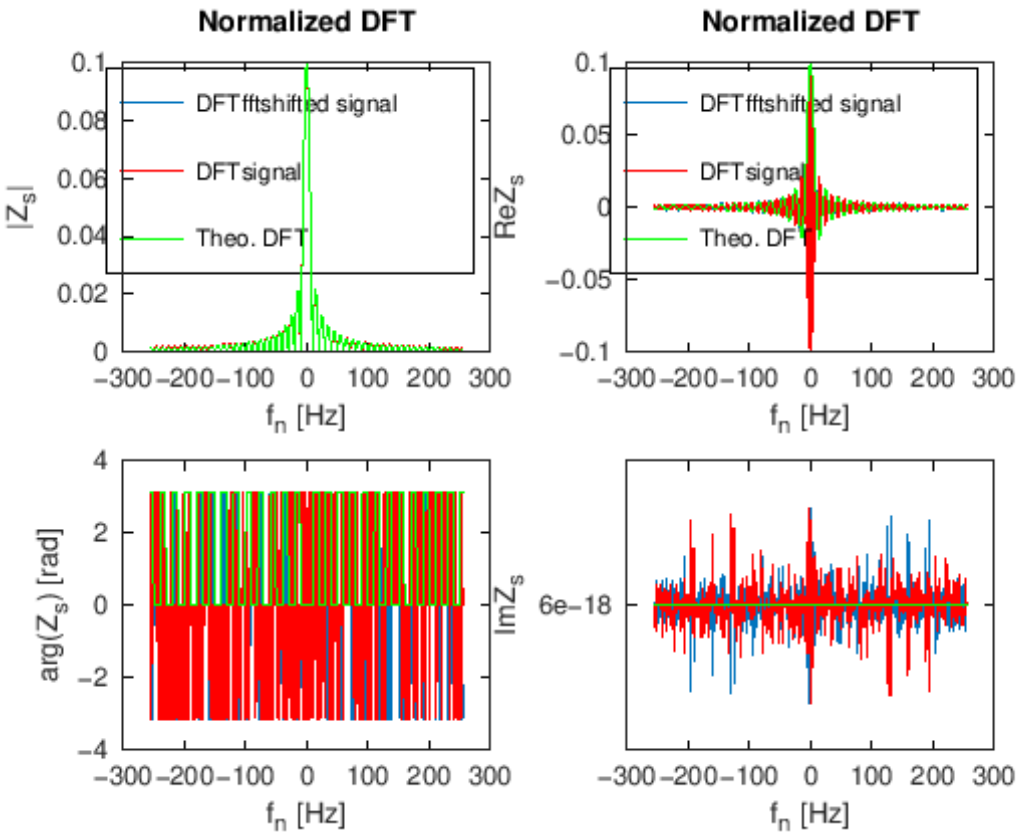
```
fs = 1/Ts; # sampling frequency
df = fs/N; # spectral resolution


# support vectors
t_s = -Tw/2:Ts:Tw/2-Ts; # time support, of duration Tw
f_n = -fs/2:df:fs/2-df; # bilateral frequency support


##### Building the test discrete time sampled signal
z_s = A*(abs(t_s-tau)<width/2);


##### REQUIRED CARDINAL SINE FUNCTION TO COMPARE DFTs to theory
function [y] = mysinc(x,Amp)
    y(find(x))    = Amp*sin(pi*x(find(x)))./(pi*x(find(x)));
    y(find(x==0)) = Amp;
end


##### DFT calculation
# 2-fft instruction
Z_s_DFT      = fftshift(fft(fftshift(z_s)))*Ts;  # fft-based normalized DFT of␣
 ↪the properly fftshifted non-causal signal.
Z_s_DFT_ns   = fftshift(fft(z_s))*Ts;            # fft-based normalized DFT of␣
 ↪not fftshifted signal.
Z_s_DFT_theo = mysinc(f_n*width,A*width);        # Cardinal sine theoretical DFT

zero_phase_criterion = 1e-8;

Phi_Z_s_DFT     = angle(Z_s_DFT).*(abs(Z_s_DFT)>zero_phase_criterion);
Phi_Z_s_DFT     = Phi_Z_s_DFT.*(abs(Phi_Z_s_DFT) > zero_phase_criterion);
Phi_Z_s_DFT_ns  = angle(Z_s_DFT_ns).*(abs(Z_s_DFT_ns)>zero_phase_criterion);
Phi_Z_s_DFT_ns  = Phi_Z_s_DFT_ns.*(abs(Phi_Z_s_DFT_ns) > zero_phase_criterion);
Phi_Z_s_DFT_theo= angle(Z_s_DFT_theo).*(abs(Z_s_DFT_theo)>zero_phase_criterion);
Phi_Z_s_DFT_theo= Phi_Z_s_DFT_theo.*(abs(Phi_Z_s_DFT_theo) >␣
 ↪zero_phase_criterion);


##### Displays
figure
fmin= -40;
fmax= 40;
subplot(2,2,1), plot(f_n, abs(Z_s_DFT)), xlabel('f_n [Hz]'), ylabel('|Z_s|'),␣
 ↪title('Zoom in')
hold on, plot(f_n, abs(Z_s_DFT_ns),'r'), plot(f_n,abs(Z_s_DFT_theo),'g')
axis([fmin fmax 0 A*width])
```

```
legend('DFT\{fftshifted signal\}','DFT\{signal\}','Theo. DFT')
subplot(2,2,3), plot(f_n, Phi_Z_s_DFT),  xlabel('f_n [Hz]'), ylabel('arg(Z_s)␣
 ↪[rad]')
hold on, plot(f_n, Phi_Z_s_DFT_ns,'r'), plot(f_n, Phi_Z_s_DFT_theo,'g')
axis([fmin fmax -pi pi])

subplot(2,2,2), plot(f_n, real(Z_s_DFT)), xlabel('f_n [Hz]'),␣
 ↪ylabel('Re\{Z_s\}'), title('Zoom in')
hold on, plot(f_n, real(Z_s_DFT_ns),'r'), plot(f_n,real(Z_s_DFT_theo),'g')
legend('DFT\{fftshifted signal\}','DFT\{signal\}','Theo. DFT')
axis([fmin fmax -A*width A*width])
subplot(2,2,4), plot(f_n, imag(Z_s_DFT)),  xlabel('f_n [Hz]'),␣
 ↪ylabel('Im\{Z_s\}')
hold on, plot(f_n, imag(Z_s_DFT_ns),'r'), plot(f_n,imag(Z_s_DFT_theo),'g')
axis([fmin fmax -1e-18 1e-18])

figure
subplot(2,2,1), plot(f_n, abs(Z_s_DFT)), xlabel('f_n [Hz]'), ylabel('|Z_s|'),␣
 ↪title('Normalized DFT')
hold on, plot(f_n, abs(Z_s_DFT_ns),'r'), plot(f_n,abs(Z_s_DFT_theo),'g')
legend('DFT\{fftshifted signal\}','DFT\{signal\}','Theo. DFT')
subplot(2,2,3), plot(f_n, Phi_Z_s_DFT),  xlabel('f_n [Hz]'), ylabel('arg(Z_s)␣
 ↪[rad]')
hold on, plot(f_n, Phi_Z_s_DFT_ns,'r'), plot(f_n, Phi_Z_s_DFT_theo,'g')

subplot(2,2,2), plot(f_n, real(Z_s_DFT)), xlabel('f_n [Hz]'),␣
 ↪ylabel('Re\{Z_s\}'), title('Normalized DFT')
hold on, plot(f_n, real(Z_s_DFT_ns),'r'), plot(f_n,real(Z_s_DFT_theo),'g')
legend('DFT\{fftshifted signal\}','DFT\{signal\}','Theo. DFT')
subplot(2,2,4), plot(f_n, imag(Z_s_DFT)),  xlabel('f_n [Hz]'),␣
 ↪ylabel('Im\{Z_s\}')
hold on, plot(f_n, imag(Z_s_DFT_ns),'r'), plot(f_n,imag(Z_s_DFT_theo),'g')


figure
plot(t_s, z_s),     xlabel('time [s]'), ylabel('Signal z_s(t)'), title('Temporal␣
 ↪representation of the signal')
axis([-1.05*Tw/2 1.05*Tw/2 -0.05 1.05])
```

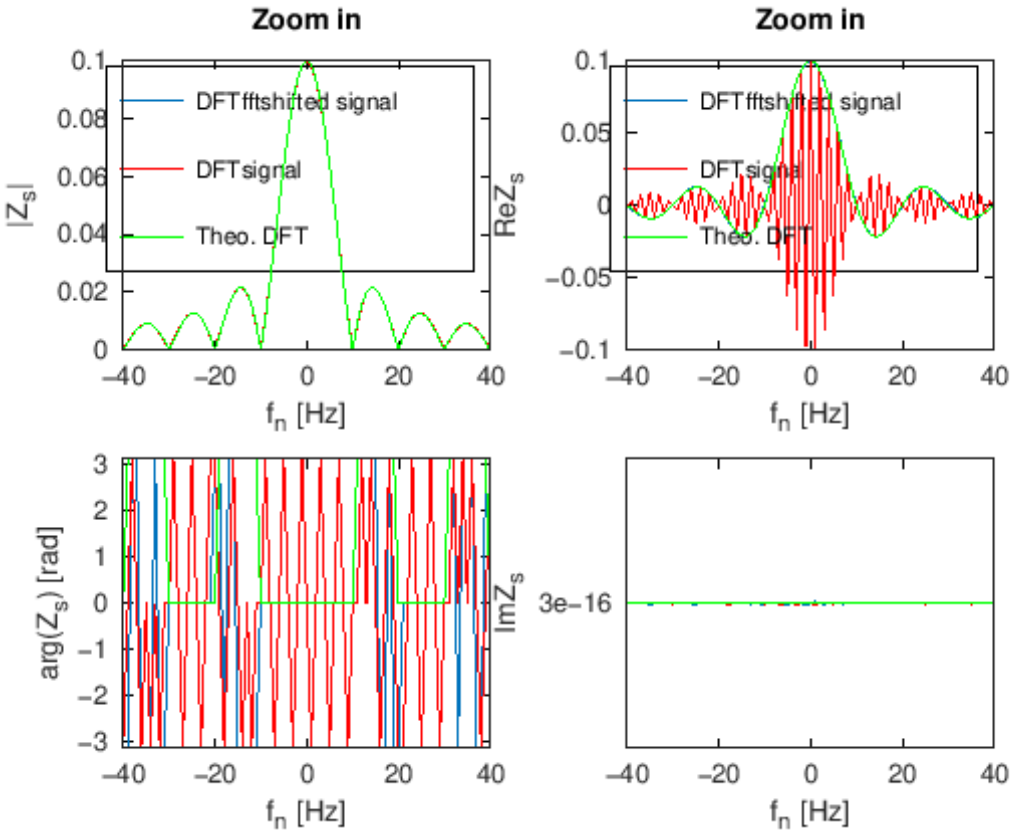
[1]:


Temporal representation of the signal
Signal $z_s(t)$
time [s]
1
0.8
0.6
0.4
0.2
0
−0.4
−0.2
0
0.2
0.4

[1]:


Normalized DFT
Normalized DFT
DFTfftshifted signal
DFTsignal
Theo. DFT
|Z_s|
ReZ_s
arg(Z_s) [rad]
ImZ_s
f_n [Hz]
6e−18

[1]:


Zoom in
Zoom in
DFTfftshifted signal
DFTsignal
Theo. DFT
|Z_s|
ReZ_s
arg(Z_s) [rad]
ImZ_s
3e−16
f_n [Hz]

VII- DFT implementation using Octave: summary

The table below gives the summary of the Octave syntax to compute the mono-, or bilateral, the DFT spectrum of a windowed discrete time signal $z_{sw}(t_s)$ consisting of N samples, whether the signal is periodic of not.

		Signal: N samples **Sampling period:** $T_s = 1/f_s$ **Windowing duration:** $T_w = Nt_s$	**Sampling frequency:** f_s **Spectral resolution:** $\delta f = 1/T_w$
		T-periodic signal, or finite-duration T signal (Fourier series formalism)	**Infinite duration signal (Fourier Transform formalism)**
Samples: ***Freq. support :***		**N/2 samples** **f = 0 : δf : f_s/2-δf;**	**N/2 samples** **f = 0 : δf : f_s/2-δf;**
Monolateral spectrum	***Causal signal***	**Z_{sw} = fft(z_{sw}) / N;** **Z_{sw} = Z_{sw}(1:N/2);** **Z_{sw} = [Z_{sw}(1) 2*Z_{sw}(2:N/2)];**	**Z_{sw} = fft(z_{sw}) * T_s;** **Z_{sw} = Z_{sw}(1:N/2);** **Z_{sw} = [Z_{sw}(1) 2*Z_{sw}(2:N/2)];**
	Non-causal signal	**Z_{sw} = fft(fftshift(z_{sw})) / N;** **Z_{sw} = Z_{sw}(1:N/2);** **Z_{sw} = [Z_{sw}(1) 2*Z_{sw}(2:N/2)];**	**Z_{sw} = fft(fftshift(z_{sw})) * T_s;** **Z_{sw} = Z_{sw}(1:N/2);** **Z_{sw} = [Z_{sw}(1) 2*Z_{sw}(2:N/2)];**
Samples: ***Freq. support :***		**N samples** **f = -f_s/2 : δf : f_s/2-δf;**	**N samples** **f = -f_s/2 : δf : f_s/2-δf;**
Bilateral spectrum	***Causal signal***	**Z_{sw} = fftshift(fft(z_{sw})) / N;**	**Z_{sw} = fftshift(fft(z_{sw})) * T_s;**
	Non-causal signal	**Z_{sw} = fftshift(fft(fftshift(z_{sw})))/N;**	**Z_{sw} = fftshift(fft(fftshift(z_{sw})))*T_s;**

VIII- References

1 - The Fast Fourier Transform and Its Applications, Book by E. Oran Brigham (original ed. 1988)

[0]: