From Fourier Series to Analysis of Non-stationary Signals – VI

Miroslav Vlček, Jan Přikryl

November 11, 2019

Department of Applied Mathematics, CTU FTS

Contents

Revision of sampled signals

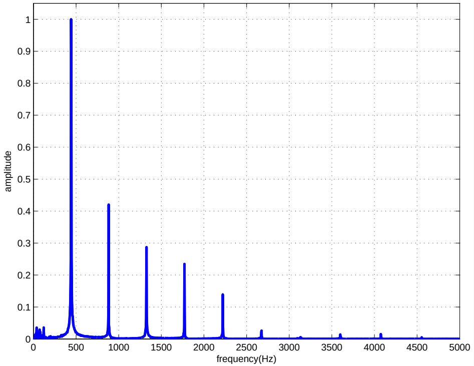
Definition (Nyquist-Shannon Sampling Theorem, 1927)

It is possible precisely to reconstruct a continuous-time signal from its samples, given that

1. the signal is bandlimited;
2. the sampling frequency is greater than twice the signal bandwidth.

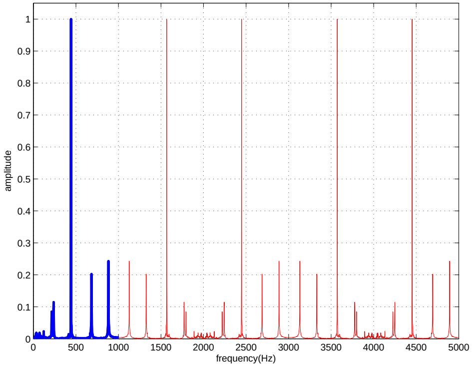
Aliasing in Audio

- The initial sound is a numerically synthesized piano-tone at 440Hz. The sampling frequency is of 44.1kHz (CD-quality).
- The harmonic frequencies at multiple of the fundamental tone (440Hz) are clearly visible.



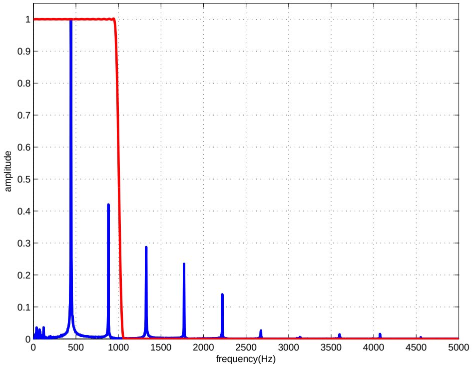
Aliasing in Audio

- The sound will be resampled at 2 kHz, without precautions against aliasing. The tone sounds rather strange.
- The aliasing is visible on the graphs as a “warping” of the frequencies against a “mirror” at the Nyquist frequency 1 kHz.



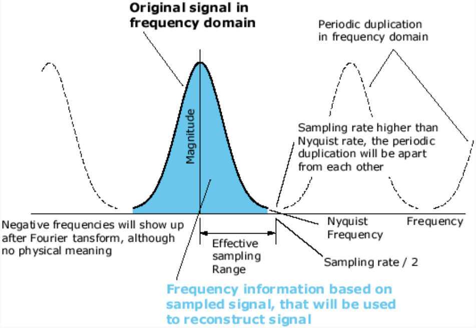
Aliasing in Audio

- In order to avoid aliasing, the spectrum of the signal should be zero at frequencies higher than the Nyquist frequency before resampling. A low-pass filter is used to achieve this



...for a digital signal processing with DFT there are limits:

- The signal must be band-limited. This means there is a frequency above which the signal is zero.
- Hence the maximum useable frequency in the DFT is $fs/2$ - the Nyquist [1] frequency!



[1]Harry Nyquist 1889-1976

Windowing and Localization

Example (Frequency hop)

Consider two different periodic signals $f(t)$ and $g(t)$ defined on $0 \leq t < 1$ with frequencies $f_1 = 96\,\text{Hz}$ and $f_2 = 235\,\text{Hz}$ as follows:

- $f(t) = 0.5\,\sin(2\pi f_1 t) + 0.5\,\sin(2\pi f_2 t)$
- $g(t) = \begin{cases} \sin(2\pi f_1 t) & \text{for } 0 \leq t < 0.5, \\ \sin(2\pi f_2 t) & \text{for } 0.5 \leq t < 1.0. \end{cases}$

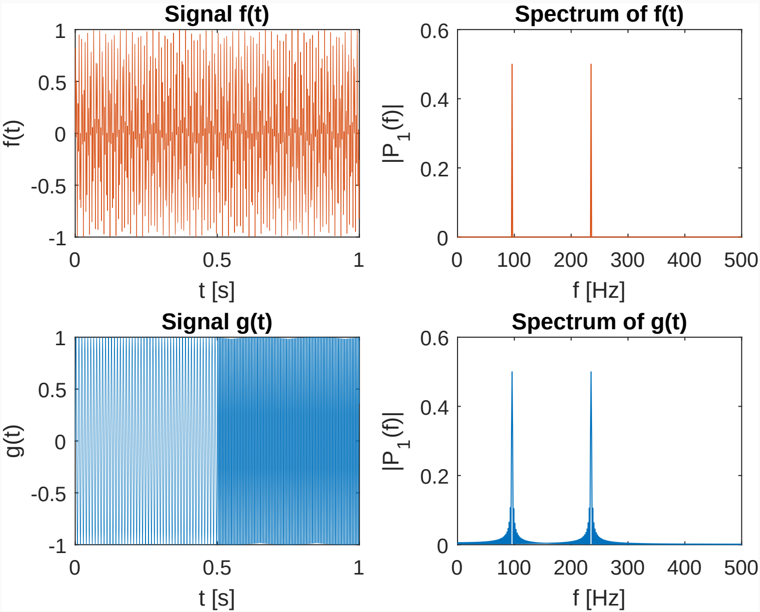
Use the sampling frequency $f_s = 1000\,\text{Hz}$ to produce sample vectors $\mathbf{f}$ and $\mathbf{g}$. Compute the DFT of each sampled signal.

Nonlocality of DFT

Two different signals $f(t)$ and $g(t)$ are constructed with Matlab commands

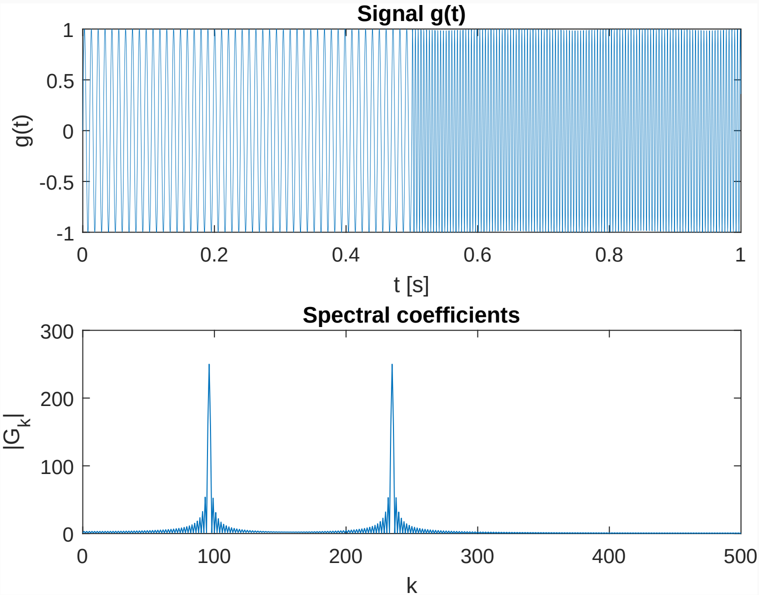
```
Fs = 1000; % sampling frequency
f1 = 96;
f2 = 235;
t1 = (0:499)/Fs; % time samples for 'g1'
t2 = (500:999)/Fs; % time samples for 'g2'
t = [t1 t2]; % time samples for 'f'
f = 0.5*sin(2*pi*f1*t)+0.5*sin(2*pi*f2*t);
g1 = [sin(2*pi*f1*t1) zeros(1,500)];
g2 = [zeros(1,500) sin(2*pi*f2*t2)];
g = g1+g2;
```

Magnitude of DFT for $f(t)$ and $g(t)$



- It is obvious that each signal contains dominant frequencies close to 96 Hz and 235 Hz and the magnitudes are fairly similar.
- But: The signals $f(t)$ and $g(t)$ are quite different in the time domain!
- The example illustrates one of the shortcomings of traditional Fourier transform: nonlocality or global nature of the basis vectors $\mathbf{W}_N$ or its constituting analog waveforms $\mathrm{e}^{\mathrm{j}2\pi kt/T}$.

Detail of signal $g(t)$



- Discontinuities are particularly troublesome.
- The signal $g(t)$ consists of two sinusoids only, but the excitation of several G_ks in frequency domain around the dominant frequencies gives the impression that the entire signal is more oscillatory.
- We would like to have possibility to localize the frequency analysis to smaller portions for the signal.
- These requirements led to development of windowed Fourier transform or short time Fourier transform – STFT.

Windowing

Consider a sampled signal $\mathbf{x} \in \mathbb{C}^N$, indexed from 0 to $N-1$. We wish to analyse the frequencies present in $\mathbf{x}$, but only within a certain time range. We choose integers $m \geq 0$ and M such that $m + M \leq N$ and define a vector $\mathbf{w} \in \mathbb{C}^N$ as

$$w[k] = \begin{cases} 1 & \text{for } m \leq k \leq m + M - 1 \\ 0 & \text{otherwise} \end{cases}$$

We use $\mathbf{w}$ to define a new vector $\mathbf{y}$ with components

$$y[k] = w[k]x[k] \quad \text{for } 0 \leq k \leq N-1.$$

We use notation $\mathbf{y} = \mathbf{wx}$ and refer to the vector $\mathbf{w}$ as the (rectangular) window.

Proposition

Let $\mathbf{x}$ *and* $\mathbf{w}$ *be vectors in* $\mathbb{C}^N$ *with discrete Fourier transforms* $\mathbf{X}$ *and* $\mathbf{W}$, *respectively. Let* $\mathbf{y} = \mathbf{w}\mathbf{x}$ *have DFT* $\mathbf{Y}$. *Then*

$$\mathbf{Y} = \frac{1}{N} \mathbf{X} * \mathbf{W},$$

where $*$ *is circular convolution in* $\mathbb{C}^N$.

Definition (Circular convolution)

The n-th element of an N-point circular convolution of N-periodic vectors $\mathbf{X}$ and $\mathbf{W}$ is

$$Y[n] = \frac{1}{N} \sum_{m=0}^{N-1} X[m] W[(n-m) \bmod N].$$

Windowing

When processing a non-stationary signal we assume that the signal is short-time stationary and we perform a Fourier transform on these small blocks — we multiple the signal by a window function that is zero outside the defined "short-time" range.

Definition (Rectangular window)

The rectangular window is defined as:

$$w(n) = \begin{cases} 1 & \text{for } 0 \leq n < N \\ 0 & \text{otherwise} \end{cases}$$

The Matlab command `rectwin(N)` produces the N-point rectangular window.

Definition (Hamming window)

The most common windowing function in speech analysis is the Hamming window:

$$w(n) = \begin{cases} 0.54 + 0.46 \cos\left(\frac{2\pi n}{N-1}\right) & \text{for } 0 \leq n < N \\ 0 & \text{otherwise} \end{cases}$$

Matlab command `hamming(N)` produces the N-point Hamming window.

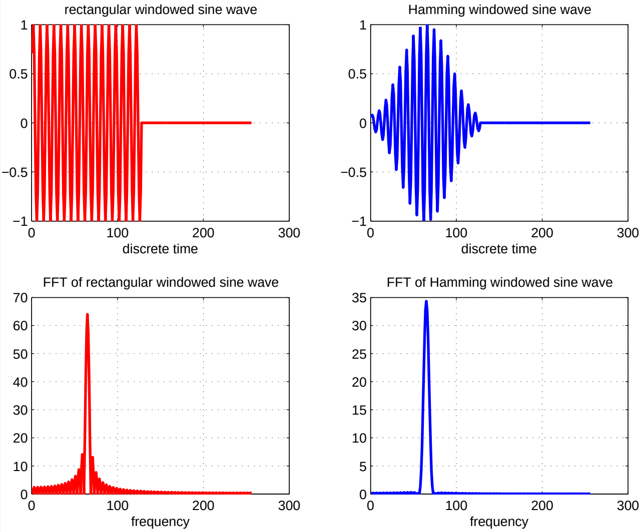
Definition (Blackman window)

Another common type of window is the Blackman window:

$$w(n) = \begin{cases} 0.42 + 0.5\cos\left(\dfrac{2\pi n}{N-1}\right) & \\ \quad +0.08\cos\left(\dfrac{4\pi n}{N-1}\right) & \text{for } 0 \leq n < N \\ 0 & \text{otherwise} \end{cases}$$

Use `blackman(N)` to produce the N-point Blackman window.

Windowing result



Matlab project

Consider signal $f(t) = \sin(2\pi f_1 t) + 0.4\ \sin(2\pi f_2 t)$ defined on $0 \leq t \leq 1$ with frequencies $f_1 = 137\,\mathrm{Hz}$ and $f_2 = 147\,\mathrm{Hz}$:

a) Use Matlab to sample $f(t)$ at $N = 1000$ points $t_k = \{k/f_s\}_{k=0}^{N}$ with sampling frequency $f_s = 1000\,\mathrm{Hz}$

```
N = 1000; % number of samples
Fs = 1000; % sampling frequency
f1 = 137; % 1. frequency
f2 = 147; % 2. frequency
tk = (0:(N-1))/Fs; % sampling times
f = sin(2*pi*f1*tk) + ...
    0.4*sin(2*pi*f2*tk); % sampled signal
```

ČVUT
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

b) Compute the DFT of the signal with `F=fft(f)` resp. `F=fft(f,N)`.
   Consult the Matlab documentation and explain the difference!
c) Display the magnitude of the Fourier transform with `plot(abs(F(0:501))`
d) Construct a rectangular windowed version of $f(n)$ for window length 200 with

```
fwa = f;
fwa(201:1000) = 0.0;
```

e) Compute the DFT of `fwa` and display the magnitude of the first 501 components.
f) Can you distinguish the two constituent frequencies?
   Be careful: is it really obvious that the second frequency is not a side lobe leakage?

g) Construct a windowed version of $f(n)$ of length 200 with

```
fwb = f(1:200);
```

h) Compute the DFT and display the magnitude of the first 101 components.

i) Can you distinguish the two constituent frequencies? Compare the plot of `fwb` with the DFT of `fwa`.

j) Repeat the parts d–h using other window lengths such as 300, 100 or 50. How short can the time window be and still allow resolution of the two constituent frequencies?

k) Does it matter whether we treat the windowed signal as a vector of length 1000 as in part 4 or shorter vector as in part 7? Does the side lobe energy confuse the results?

Homework

Homework

1. Repeat the parts a)–k) from the lecture projkect, but this time using a triangular window.
2. A triangular window vector **w** of length $L = 201$ can be constructed using

```
L = 201;
w = triang(L);
```

3. Construct a windowed signal of the length 1000 as

```
fwc = zeros(size(f));
fwc(1:L)=f(1:L).*w;
```

and compute its spectrum using `fft(fwc)`.

4. Try varying the window length L. What is the shortest window that allows you to distinguish the two frequencies?
5. Repeat the previous parts 1–10 for the Hamming window.
6. Submit the answers for the several questions raised in parts 1–16 as a written Report on Window Functions by November 20, 2019.