Applications of signal processors

ADVANCED DSP ALGORITHMS

in Telecommunication

Author: Grzegorz Szwoch

Gdańsk University of Technology, Department of Multimedia Systems

Introduction

Four algorithms selected from more advanced methods of digital signal processing, for practical application in telecommunication systems:

- signal resampling, decimation and interpolation;
- removal of changing distortion – adaptive filtering;
- frequency detection – autocorrelation;
- signal demodulation and envelope detection
  - analytic signal and Hilbert transformer.

Signal resampling

PROBLEM #1: sampling frequency 48 kHz is too low for our application.

- Oversampling – processing the signal with higher sampling frequency than the target one.
- In audio processing, signals are often oversampled 4-times, sampling frequency is 192 kHz.
- Drawback: more operations to perform; in signal oversampled by a factor of 4, DSP has to perform 4 times as much operations during the same time.

Decimation

- At the output, we need a signal sampled with 48 kHz.
- If we have *fs* = 192 kHz, we have 4 times too much samples.
- Let's take every 4^{th} sample. Now, we have the number of samples we need.
- Before taking the samples, we need to process the oversampled signal with a low-pass decimation filter, with cut-off frequency equal to the target Nyquist frequency (24 kHz in our example).
- If we don't apply the filter, aliasing occurs.

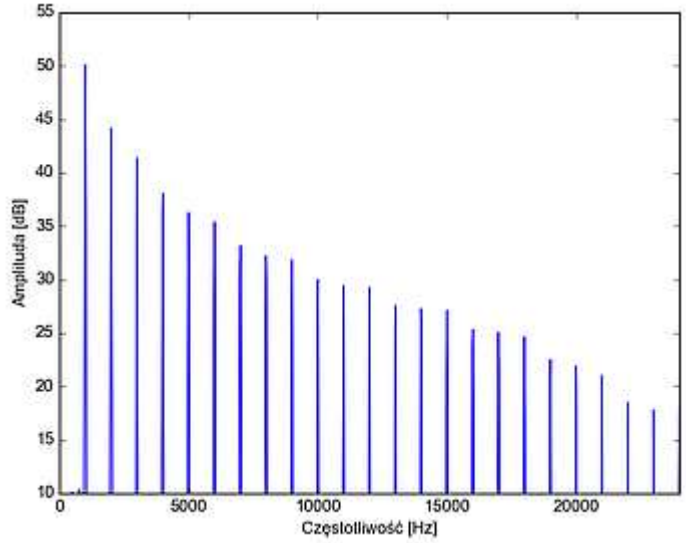
Decimation

Decimation from *fs1* to *fs2* = *fs1* / *D*:

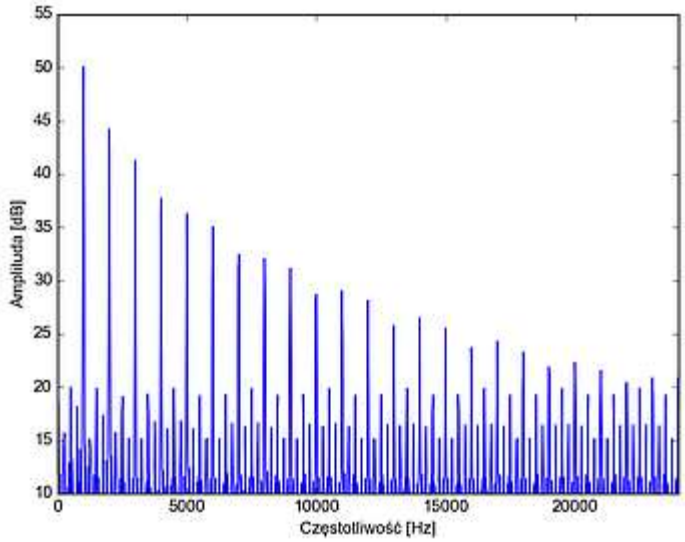
- low-pass filter with *fc* = *fs2* / 2,
- then take every *D*-th sample.

Example – generating a sawtooth wave *f* = 1 kHz:

4x oversampling with decimation

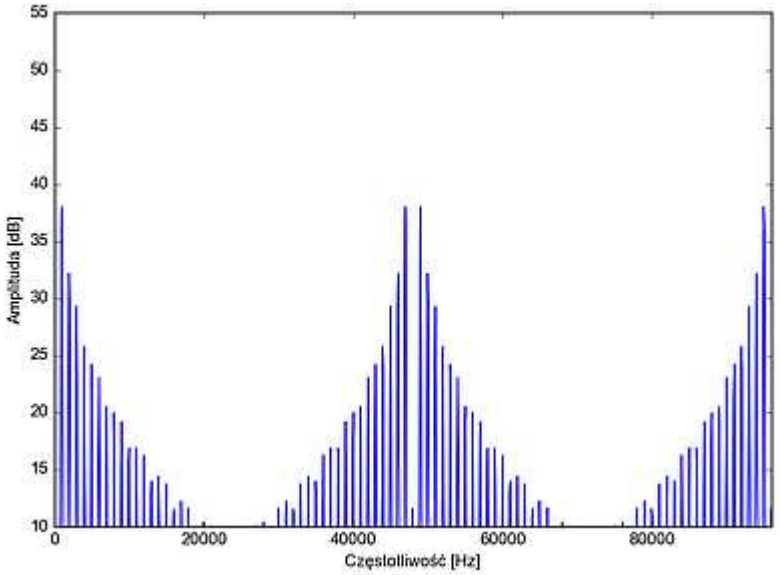


No oversampling



Interpolation

- We have a signal sampled with 48 kHz.
  How do we upsample it to 192 kHz?
- Let's insert 3 zeros in between each pair of samples.
- We have the sufficient number of samples.
- Spectrum – almost correct. Almost.

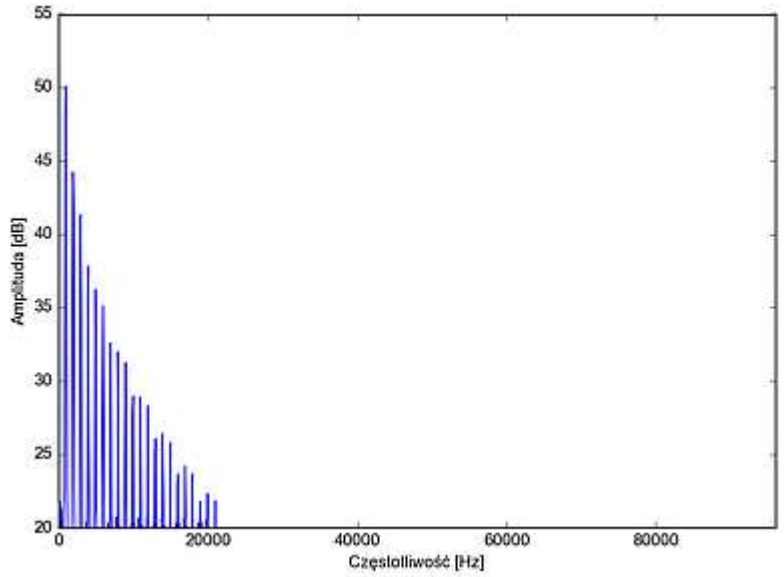


Interpolation

Interpolation from *fs1* to *fs2* = *fs1* * *L*:

- insert (*L*-1) zeros in between each pair of samples,
- process the signal with a low-pass filter with *fc* = *fs1* / 2.

Example – sawtooth wave *f* = 1 kHz:



Resampling

- Problem #1A: we have a signal with *fs1* = 44 100 Hz, but we need *fs2* = 48 000 Hz.
- Resampling: changing the sampling frequency with a factor of (*L*/*D*) :
  - insert (*L*-1) zeros in between each pair of samples,
  - process with a low-pass filter (only one),
  - take every *D*-th sample.
- *L* / *D* = 48000 / 44100 = 160 / 147
  therefore: interpolation with *L* = 160
  and decimation with *D* = 147.

DSPLIB functions

- Decimating filter:

firdec	*Decimating FIR Filter*
Function	ushort oflag = firdec (DATA *x, DATA *h, DATA *r, DATA *dbuffer , ushort nh, ushort nx, ushort D)

- Interpolating filter:

firinterp	*Interpolating FIR Filter*
Function	ushort oflag = firinterp (DATA *x, DATA *h, DATA *r, DATA *dbuffer , ushort nh, ushort nx, ushort I)

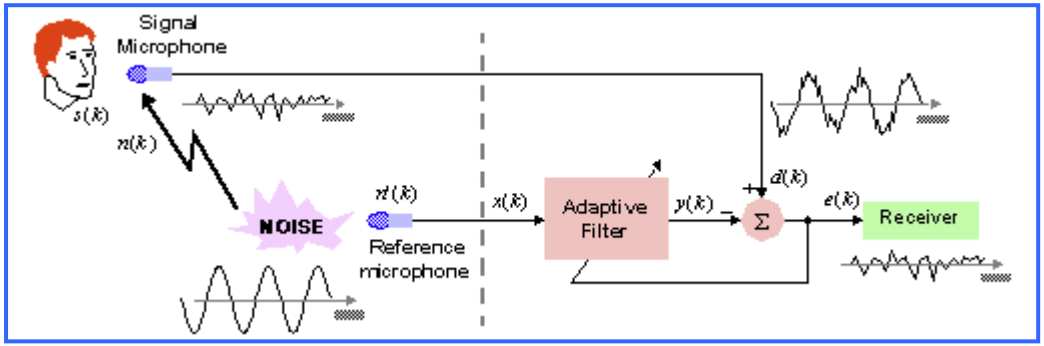
Noise removal

PROBLEM #2

- A hands-free microphone/speaker system in a car.
- A microphone collects speech and noise from the car.
- If the noise was stationary, we would use a normal filter to remove noise.
- It won't work, because the noise is constantly changing.
- We need a filter that adapts to the changing noise.

Adaptive filters

- Adaptive filters – filters with coefficients calculated by an algorithm.
- Reference signal: from the main microphone.
- Filtered signal: from another microphone that collects only noise, not the speech.
- Error signal: difference between the filtered and the reference signals.
- Adaptation of filter coefficients to minimize the filtering error.



LMS algorithm

- LMS – Least mean squares.
- The algorithm tries to minimize energy of the error signal.
- The higher the error, the larger change to the filter coefficients is needed.
- Adaptation of filter coefficients:

$$w_k(n+1) = w_k(n) + \mu\, x(n-k)\, e(n)$$

(μ - adaptation step – speed of adaptation)

Adaptive filters in DSPLIB

dlms *Adaptive Delayed LMS Filter*

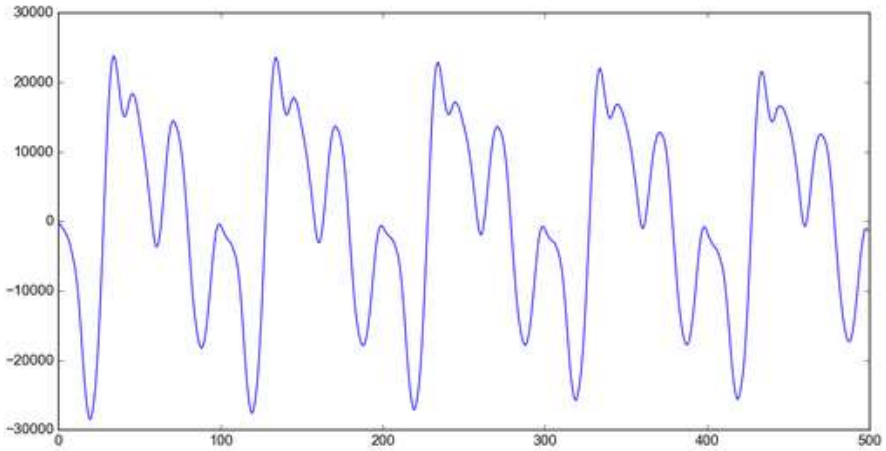
Function ushort oflag = dlms (DATA *x, DATA *h, DATA *r, DATA *des, DATA *dbuffer, DATA step, ushort nh, ushort nx)

- *x* – input signal
- *h* – filter coefficients
- *r* – buffer for the filtered signal
- *des* – reference (desired) signal
- *step* – adaptation speed
- *nh* – number of filter coefficients
- *nx* – number of processed samples

Determining the sound pitch

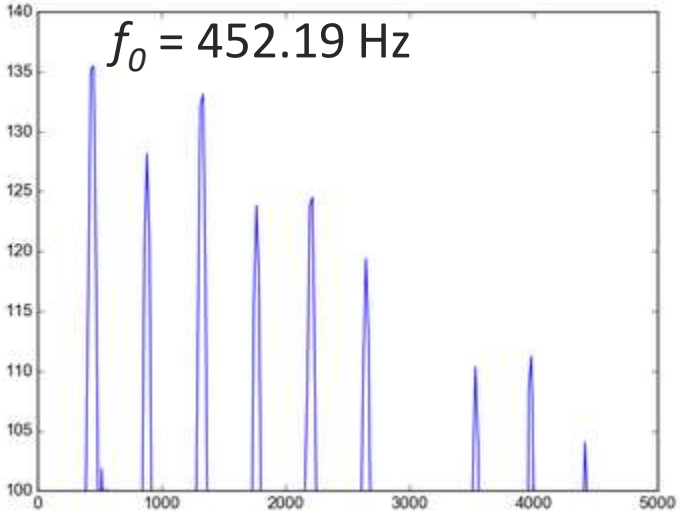
PROBLEM #3:

- We have a recording of a musical instrument, such as trumpet. We want to find its pitch on a musical scale.
- Musical sounds are quasi-periodic and harmonic.
- Quasi-period – duration of the shortest repeating fragment.
- Fundamental frequency – a reciprocal of a quasi-period, determines the sound pitch on the musical scale, e.g., A4.



Determining the sound pitch with FFT

- A most obvious approach: we do the FFT and we look for the first maximum.
- This method is not very reliable:
  - it's not always easy to find the base component (it may not be the strongest one),
  - low accuracy – depends on the frequency resolution of FFT,
  - we can fit a parabola to the peak, as described in an earlier lecture.

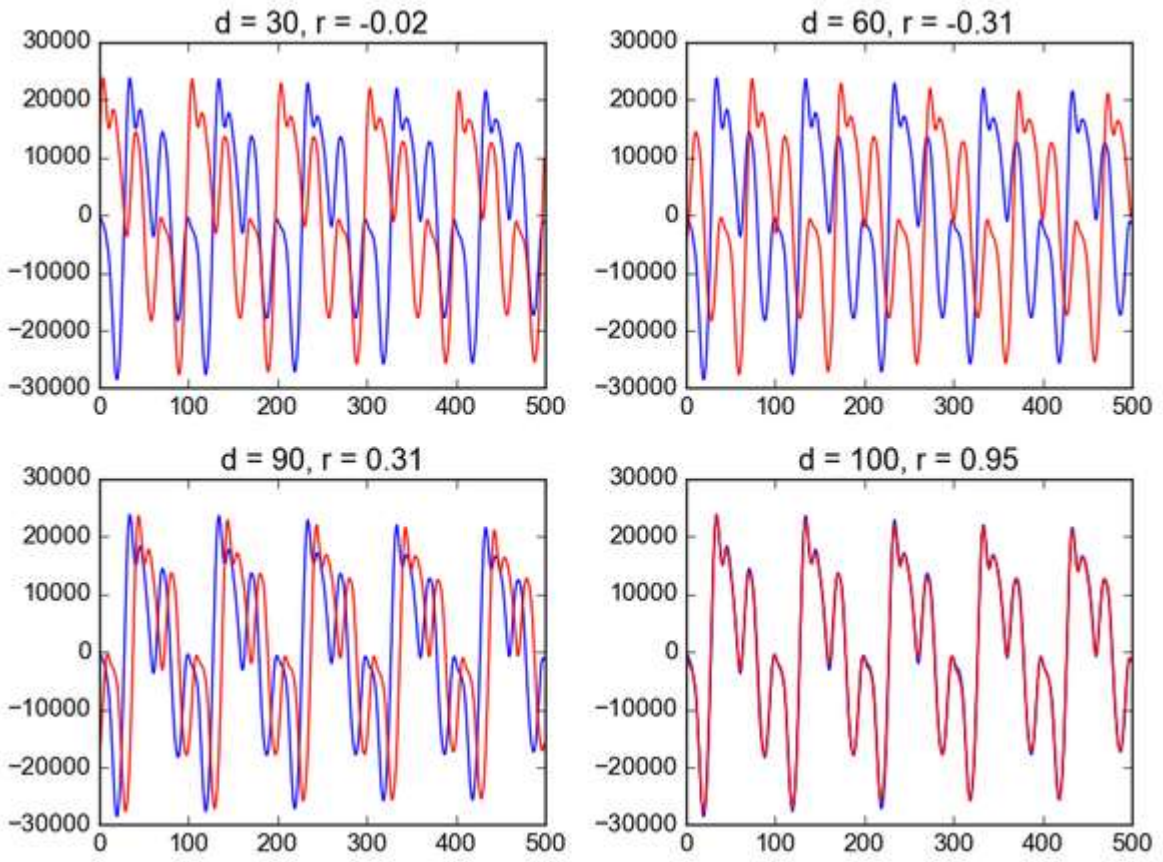


Autocorrelation

- Autocorrelation coefficient: a measure of similarity between the signal and its copy shifted in time. Values [-1, 1], value of zero means complete dissimilarity.
- Autocorrelation function: values of the coefficient calculated for various time shifts (lags).
- We compute the autocorrelation coefficients, shifting the signal by an increasing number of samples.
- Maximum of the function indicates the highest similarity for a given lag.
- Therefore, the first non-zero maximum determines the quasi-period, and thus the fundamental frequency.

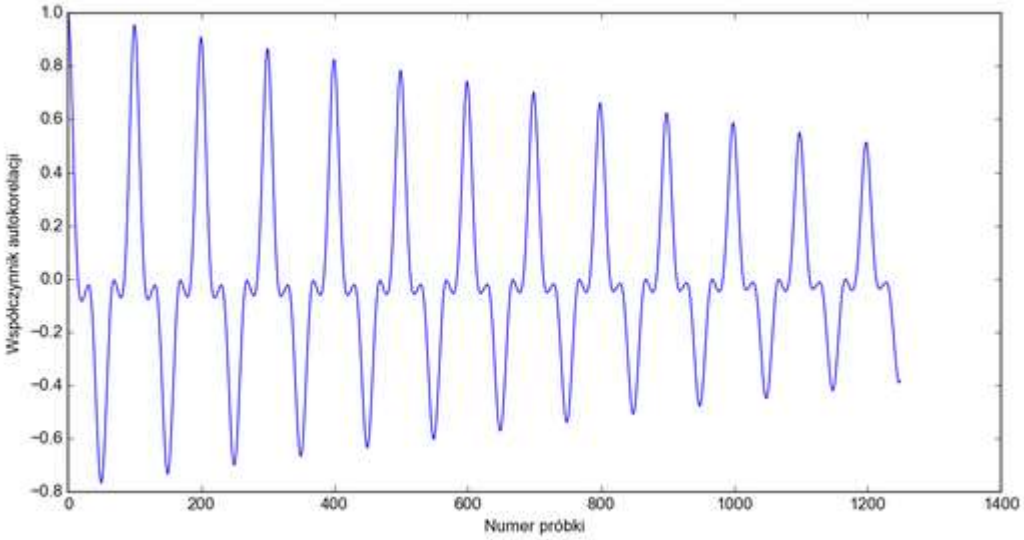
Autocorrelation

Autocorrelation coefficient *r* for various lags *d*



Sound pitch with autocorrelation

- The autocorrelation function plot shows maxima for multiplies of the quasi-period.
- The first maximum at d = 100, therefore:
  f_0 = 44100 / 100 = 441 Hz



Autocorrelation

Another example: determining presence of speech in the signal.

- Speech is quasi-periodic, so there will be maxima in the autocorrelation function.
- A noise is always uncorrelated – no maxima.
- Signal gating: we check how many autocorrelation values in the buffer exceeds a threshold, and we decide: “speech / no speech”.

Autocorrelation in DSPLIB

acorr	*Autocorrelation*
Function	ushort oflag = acorr (DATA *x, DATA *r, ushort nx, ushort nr, type)

Autocorrelation can be also computed with FFT:

- we compute the spectrum (FFT),
- then we get a squared module of the spectrum,
- and we compute an inverse FFT (IFFT) from the squared module which gives us the autocorrelation function.

Analysis of a compound signal

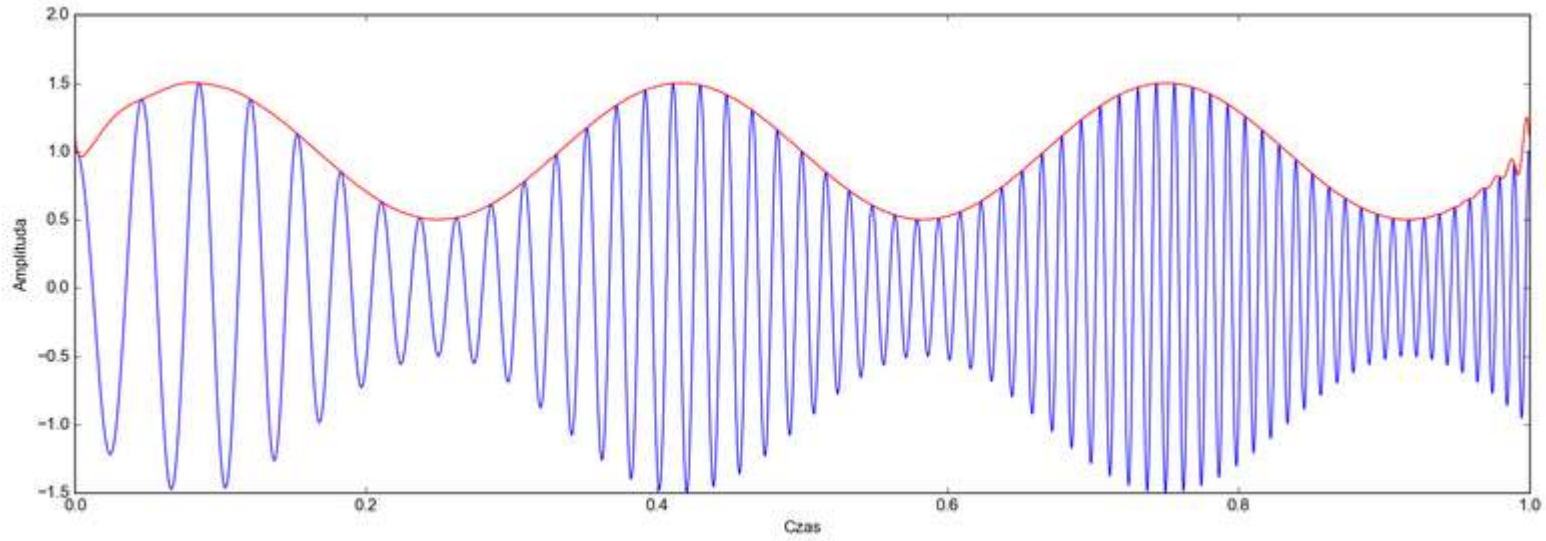
PROBLEM #4

- We have the following signal:
  - a chirp (sweep) with linearly increasing frequency from 20 Hz to 100 Hz,
  - amplitude modulated: signal amplitude is multiplied by a sine with f = 3 Hz.
- For any moment, we need to find:
  - value of the signal envelope (the modulating sine),
  - current frequency of the signal.

Analysis of a compound signal

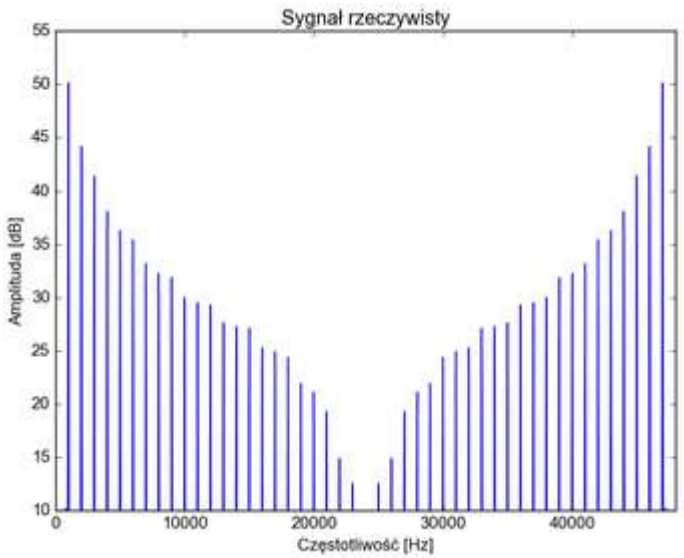
This is how our signal looks:

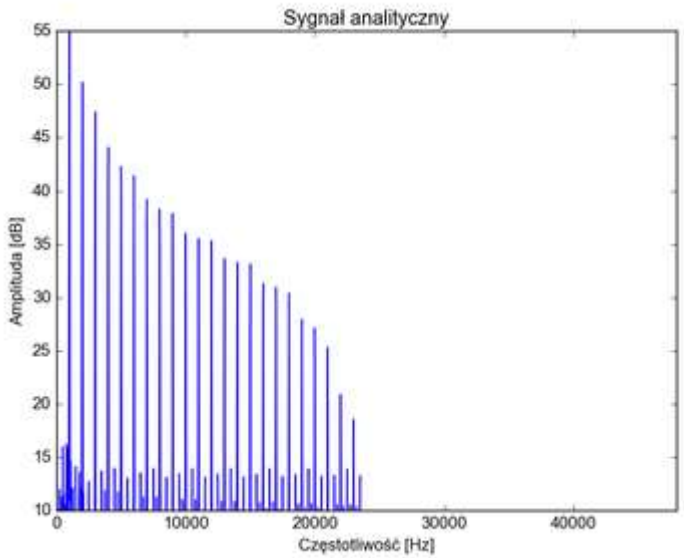
- blue color – values of the signal,
- red color – signal envelope (the modulating sine 3 Hz).



Analytic signal

- Real signal has two copies of the spectrum from 0 to *fs*.
- Let's remove the second copy, from *fs*/2 to *fs*.
- We have created a complex-valued signal, called the analytic signal.





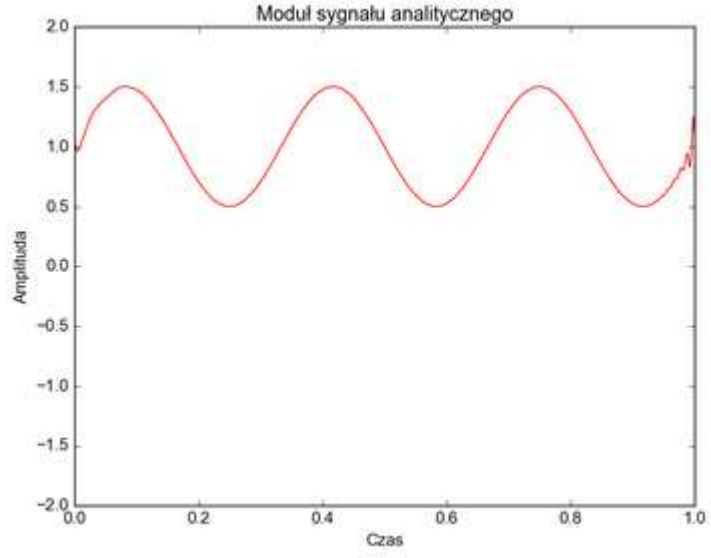
Signal envelope

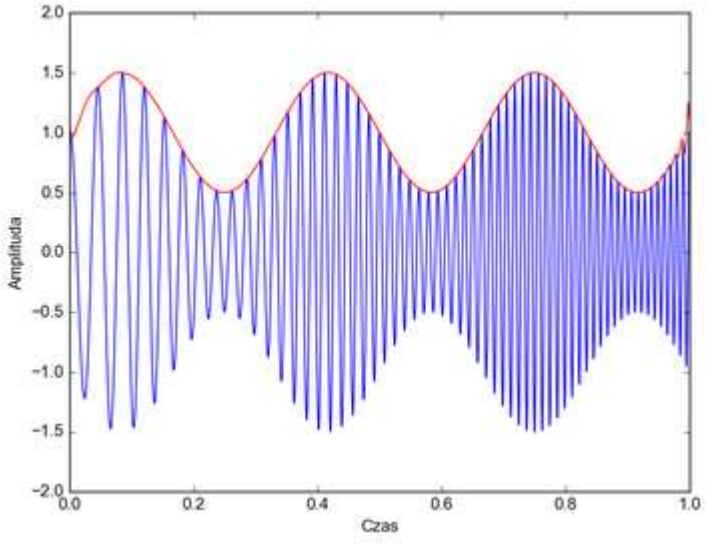
What do we get from computing the analytic signal:

- absolute value of the analytic signal $r(n)$:

$$y(n) = \sqrt{\mathrm{Re}(r(n))^2 + \mathrm{Im}(r(n))^2}$$

is the envelope of the original signal
(only if the signal is symmetric).





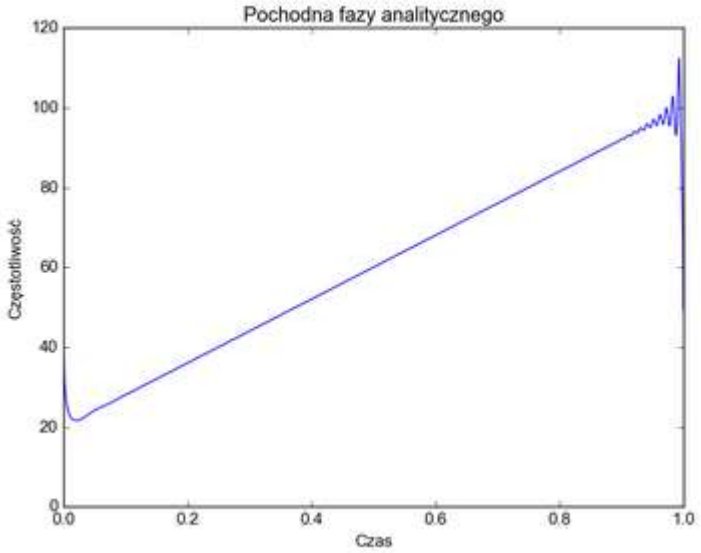
Instantaneous frequency

- Phase of the analytic signal:

$$\varphi(n) = \arctan\left(\frac{\mathrm{Im}(r(n))}{\mathrm{Re}(r(n))}\right)$$

- Instantaneous frequency = derivative of the phase:

$$f(n) = \frac{f_s}{2\pi}\left(\varphi(n) - \varphi(n-1)\right)$$



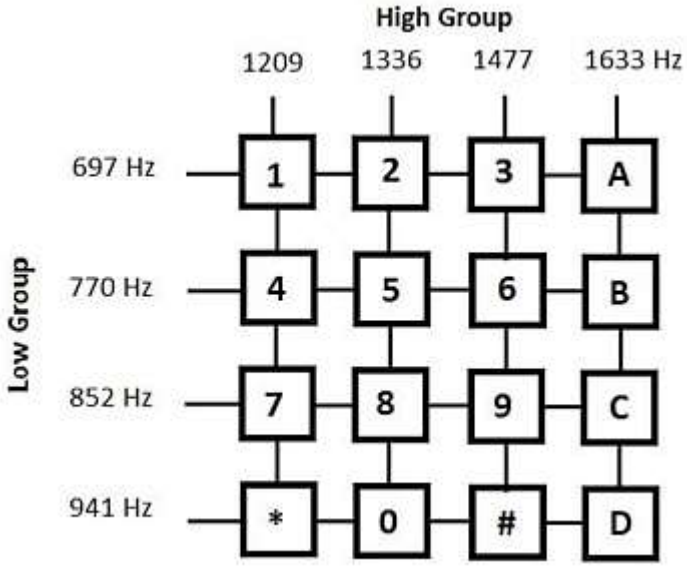
Hilbert transformer in DSPLIB

- Algorithm that transforms the real signal into the analytic signal is called a Hilbert transformer.
- It can be realized as a FIR filter, or with spectral processing (FFT, removing one copy, IFFT).
- In DSPLIB:

hilb16	*FIR Hilbert Transformer*
Function	ushort oflag = hilb16 (DATA *x, DATA *h, DATA *r, DATA *dbuffer, ushort nx, ushort nh)

Bonus - DTMF

- DTMF – *Dual-Tone Multi Frequency*
- A method of encoding digits, used in Telecommunication.
- Every sign is represented as a duotone – two sines selected from 8 possible frequencies.
- For example, digit “6”:
  770 Hz + 1477 Hz
- Used e.g. for selecting the number in a phone call.



Bonus - DTMF

„Homework” – please think about how to solve this problem.

We need to create a DTMF detector with DSP.

- Detecting the beginning and the end of each symbol:
  - how to detect a digit, exactly once per symbol?
- Detection of the duotone frequencies:
  - filters? FIR or IIR?
  - FFT?
  - maybe another method?

Similar, but more difficult problem: how to create a Morse code detector with DSP?