Introduction
to spectral analysis

Honza Černocký, ÚPGM

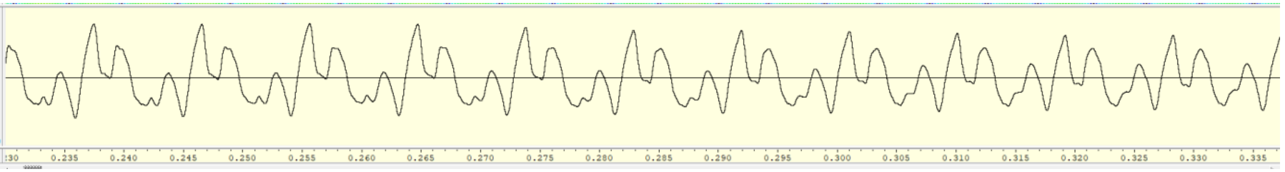
Please open Python notebook `02_spectral`.

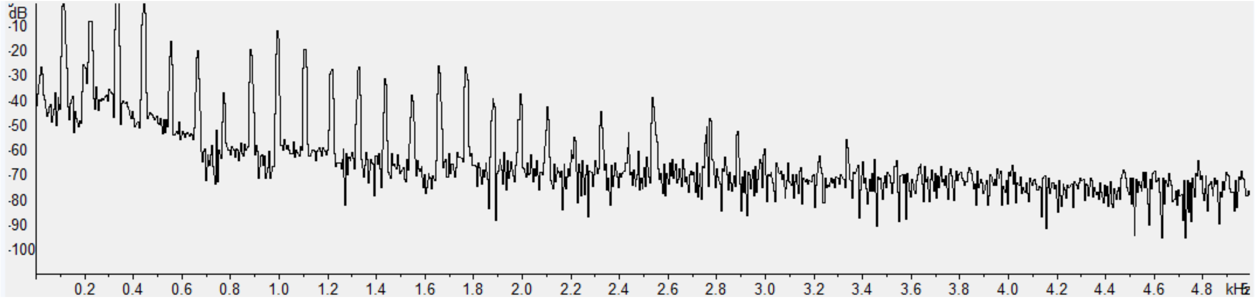
The goal of spectral analysis

- Take a complicated signal and try to see how it is composed from frequency components
- What do we want to know about them
  - At which frequencies they are
  - How strong they are
  - How they are shifted in time
- We can perform it for a single segment of a signal -> **spectrum**
- Or we can see the evaluation of spectrum during the time -> **spectrogram**

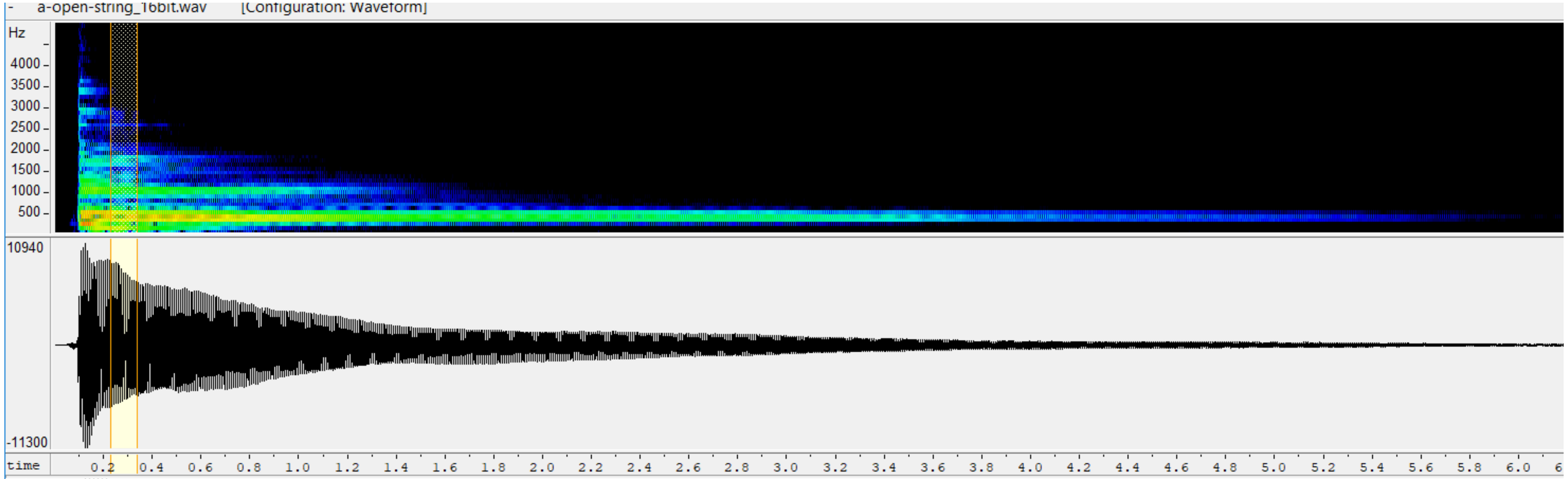
Guitar

- Piece of signal and its spectrum

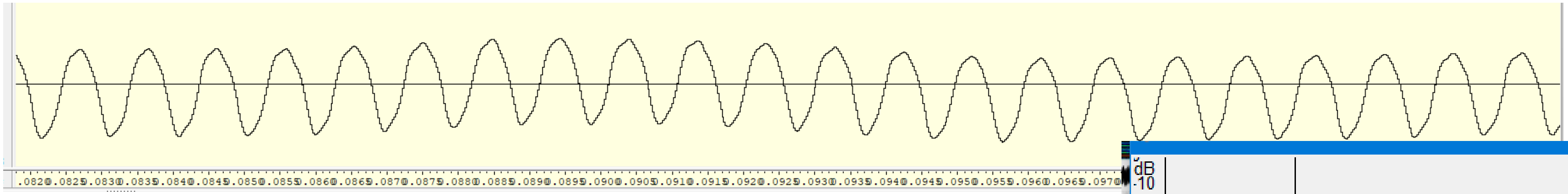


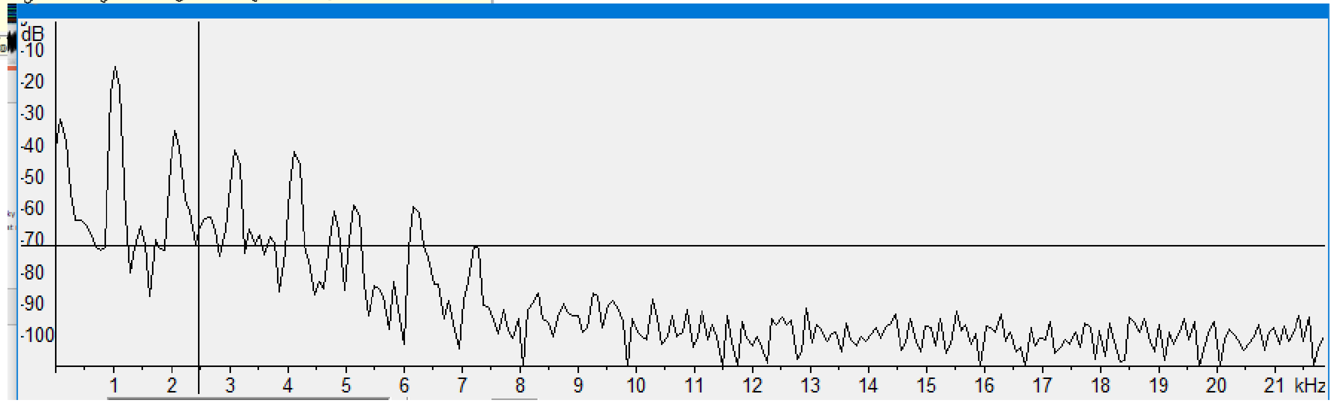


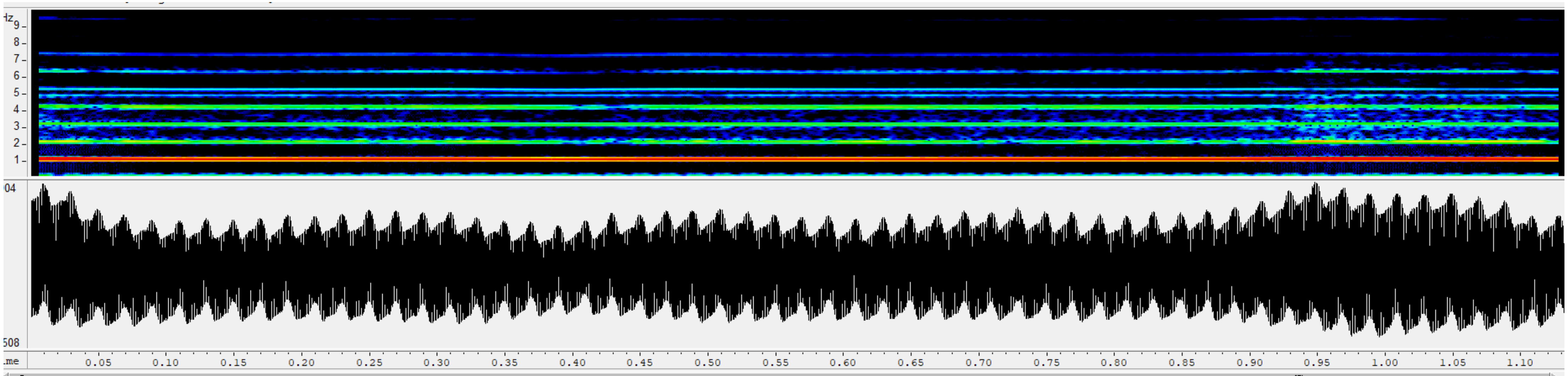
- Whole signal and its spectrogram



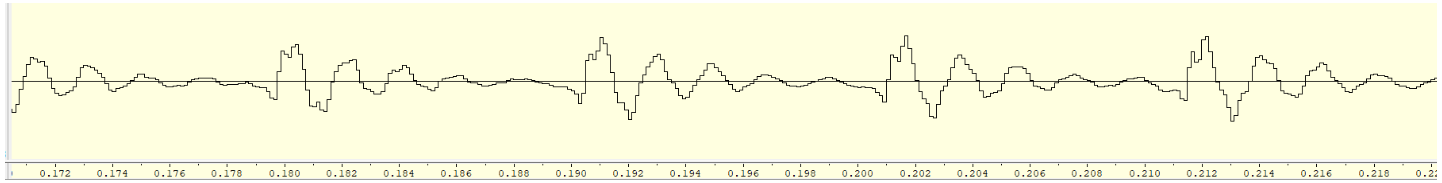
Recorder (zobcová flétna)

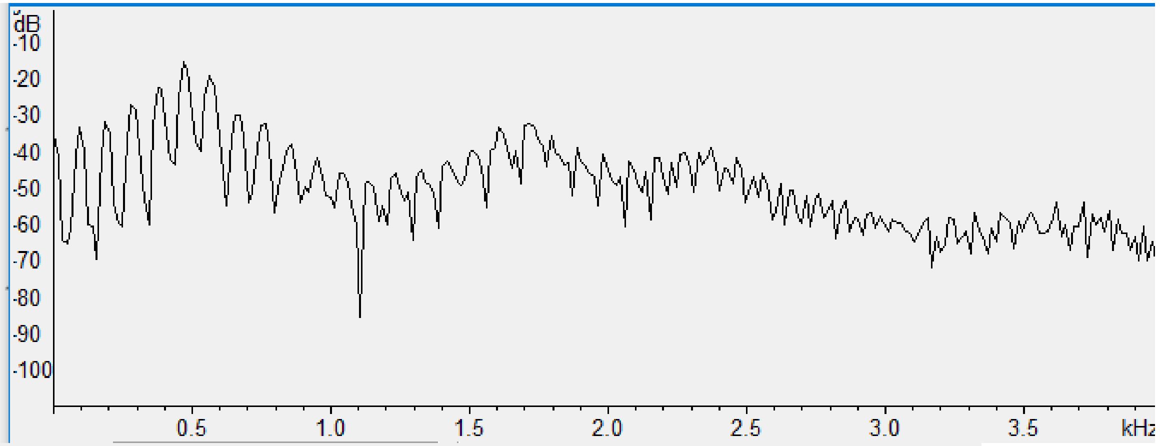


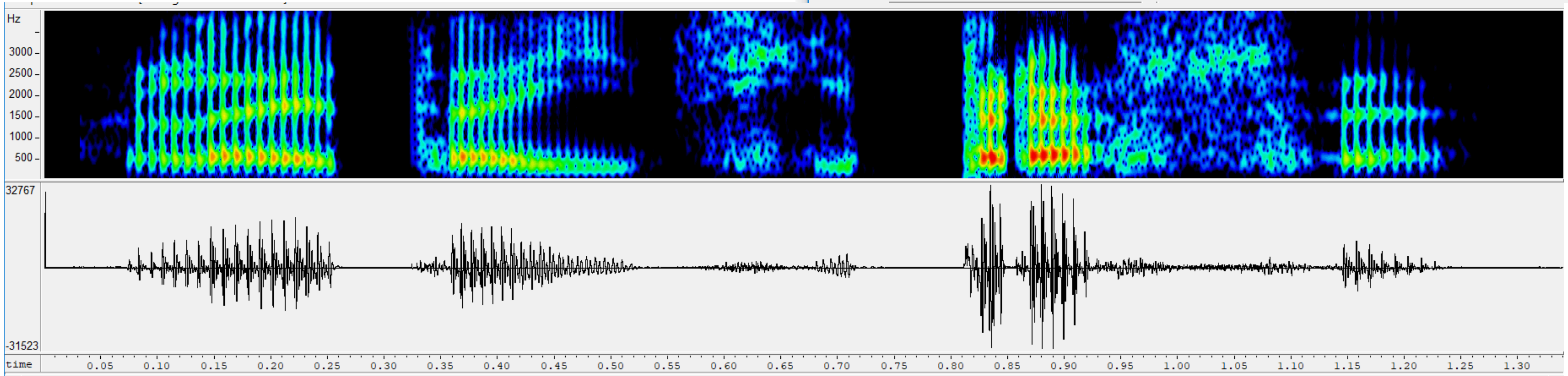




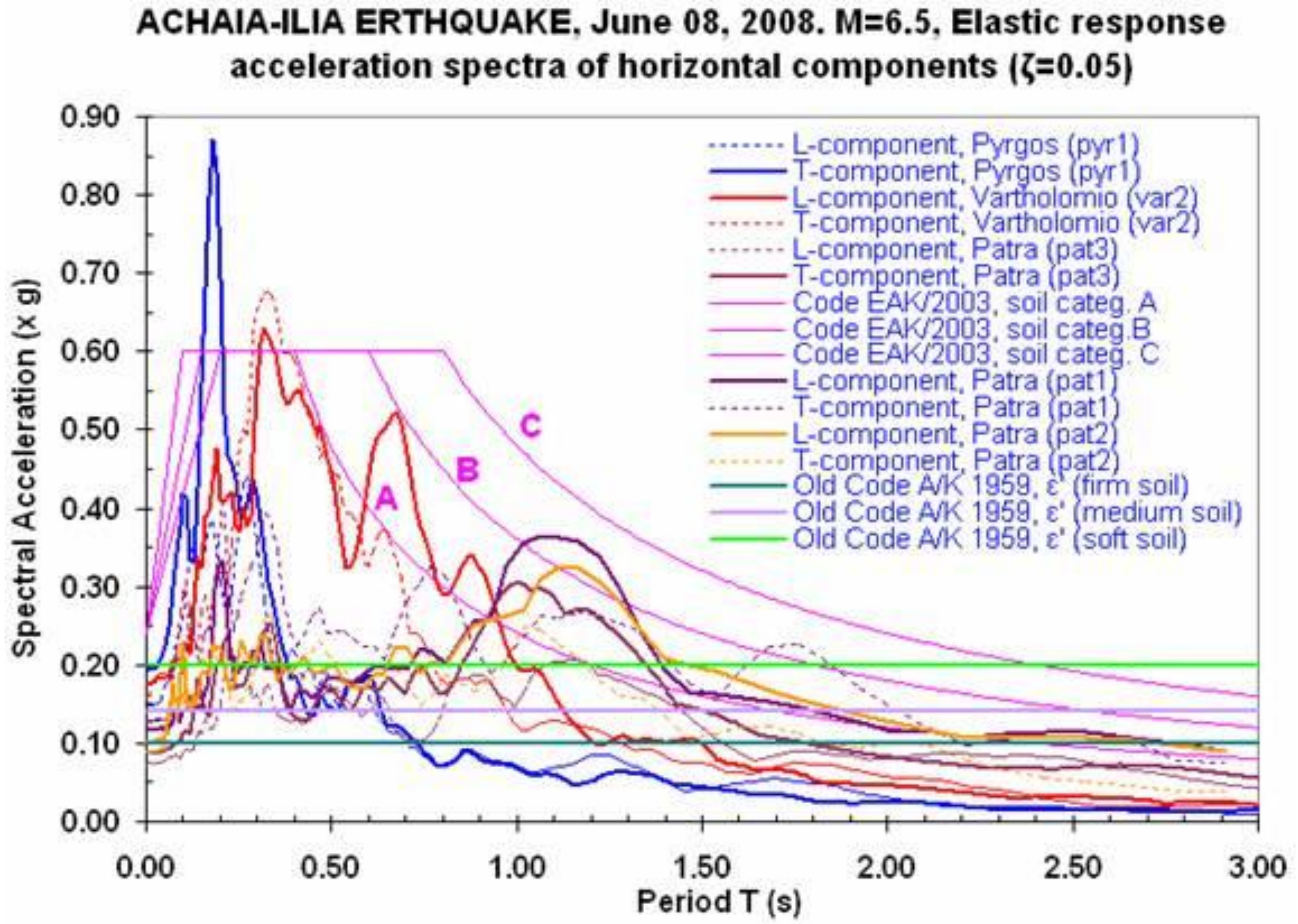
Human voice (lidský hlas)







Not just audio … seismology, vibration analysis



Why are the signals around us complex ?

Because they are created by complicated physical processes, not by a cosine generator !

- Strings: several vibration modes together, see for example https://www.youtube.com/watch?v=BSIw5SgUirg
- Flutes (tubes): dtto, https://www.youtube.com/watch?v=KZ7intMz2Y4
- Human voice: vocal chords (hlasivky) do everything but smooth movements, and this creates lots of frequencies: https://www.youtube.com/watch?v=y2okeYVclQo

Harmonically related signals

- Most of frequency analysis involves a **fundamental frequency** (základní frekvence, fundamentální frekvence)
- and its multiples – **harmonically related frequencies** (harmonicky vztažené frekvence) or simply **harmonics** (harmonické).
  - Musicians might have heard about it in the music theory classes – aliquots (alikvotní tóny)

Our spectral analysis will follow the same principles – **fundamental frequency and its mutliples.**

Spectral analysis

- Correlation
- Determination of similarity
- Projection to bases

The same !

- Notation
  - *x[n]* is the unknown signal
  - *a[n]* is a known (generated) analysis signal
  - *c* is the resulting coefficient quantifying correlation / similarity / strength of projection

$$c = \sum_{n=0}^{N-1} x[n]a[n]$$

Implementation of this simple equation

- *x[n]* and *a[n]* are stored in row vectors.
- `np.sum(a * x)` – straightforward implementation
- `np.dot(a, x.T)` – dot product (skalární součin) of 2 vectors (the 2nd one must be column)
- `np.matmul(a, x.T)` – the same using a function for matrix multiplication
- `np.matmul(A, x.T)`
  - more bases stored in the rows of matrix **A**
  - we'll get whole vector of coefficients
  - we'll see this all the time ! $\mathbf{c} = \mathbf{A}\mathbf{x}^\top$

#computing_projection

Examples of analysis – use also your **intuition** !

- "Unknown" signal has *N=128* samples, let's begin with a **D.C. signal** …
- We'll always **show the product** *x[n] a[n]*
- Analysis signals will be
  - another DC signal — big similarity
  - Cosine – 1 period in *N* samples — no similarity
  - Cosine – 2 periods in *N* samples — no similarity

#dc_signal_analysis

- “Unknown” signal is **1 period of cosine**
- Analysis signals will be
  - another DC signal — no similarity
  - Cosine – 1 period in *N* samples — big similarity
  - Cosine – 2 periods in *N* samples — no similarity

#1cos_analysis

- “Unknown” signal are **2 periods of cosine**
- Analysis signals will be
  - another DC signal — no similarity
  - Cosine – 1 period in *N* samples — no similarity
  - Cosine – 2 periods in *N* samples — big similarity

#2cos_analysis

- “Unknown” signal are **2 periods of cosine + some D.C. component**
- Analysis signals will be
  - another DC signal — some similarity
  - Cosine – 1 period in *N* samples — no similarity
  - Cosine – 2 periods in *N* samples — some similarity

Wow, the projection manages to separate the “strengths” of D.C. and cosine !

#2cos_dc_analysis

- “Unknown” signal are **2 periods of cosine + some noise**
- Analysis signals will be
  - another DC signal   not showing - no similarity
  - Cosine – 1 period in *N* samples   not showing - no similarity
  - Cosine – 2 periods in *N* samples   some similarity – same as for clean one !

Wow, analysis by cosines seems to be robust !

#2cos_noise_analysis

Cosines seem to work !

- **Big positive** – correlation, similarity, this frequency IS in the analyzed signal.
- **Big negative** – anti-correlation, similar, but in the inverse sense, the frequency IS in the analyzed signal with minus sign.
- **Small / zero** – no correlation, no similarity, the frequency is not there or just a little.

Massive spectral analysis with cosines !

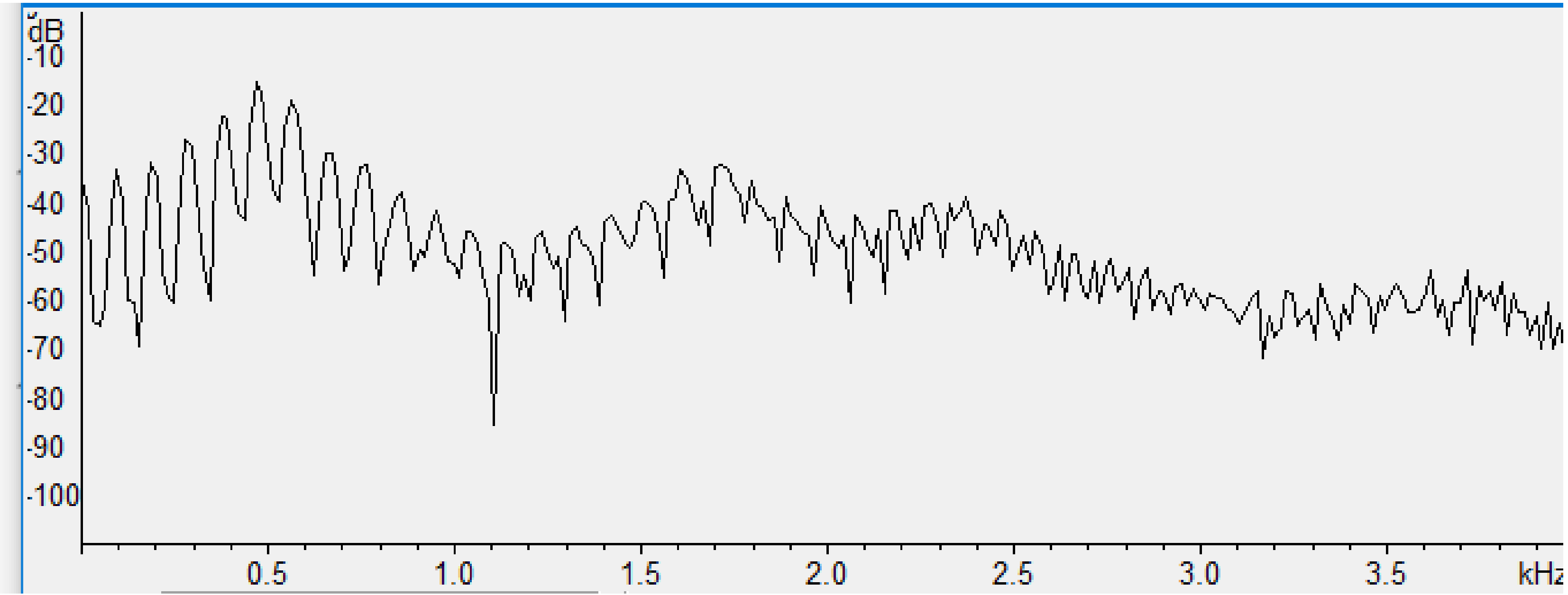
- Working with cosines seems to work perfectly, let's run a massive spectral analysis with many of them.
- We'll get many cosines $cos(2\pi\ k\ n\ /\ N\ )$ … but **how many** ?
  - k = 0 is the D.C. : $cos(2\pi\ k\ n\ /\ N\ ) = cos(0) = 1$
  - $k = 1$ is one period in N samples
  - $k = 2$ are two periods in N samples
  - …
  - $k = N/2$ is the fastest cosine we can generate: $cos(2\pi\ (N/2)\ n\ /\ N\ ) = cos(\pi\ n)$ generates +1, -1, +1, -1 … changing polarity every sample.

#show_range_of_cosines

The input signal and reference spectrum …

- **256** samples from sound ‘e’ from my favorite test signal “Létající prase”.
- the reference spectrum should look like this

#prase_signal



Prepare the battery of cosines and run the show!

$$a_0[n] = \cos(2\pi \frac{0}{N} n)$$

$$a_1[n] = \cos(2\pi \frac{1}{N} n)$$

$$a_2[n] = \cos(2\pi \frac{2}{N} n)$$

$$\cdots$$

$$a_{\frac{N}{2}}[n] = \cos(2\pi \frac{\frac{N}{2}}{N} n)$$

$$c_0 = \sum_{n=0}^{N-1} a_0[n]x[n]$$

$$c_1 = \sum_{n=0}^{N-1} a_1[n]x[n]$$

$$c_2 = \sum_{n=0}^{N-1} a_2[n]x[n]$$

$$\cdots$$

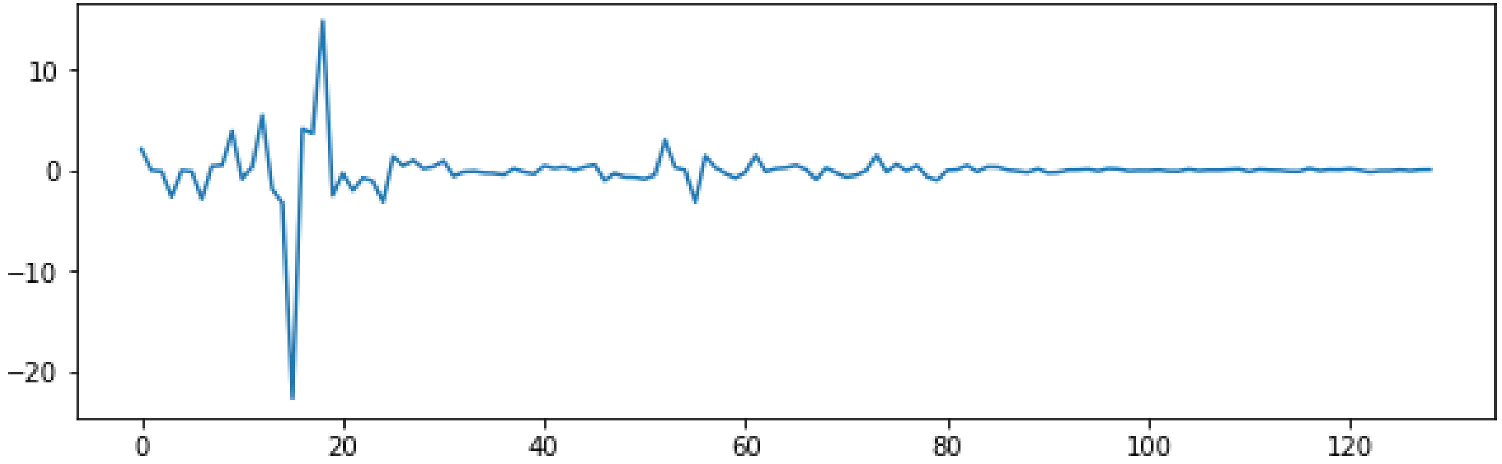
$$c_{\frac{N}{2}} = \sum_{n=0}^{N-1} a_{\frac{N}{2}}[n]x[n]$$

$$\mathbf{c} = \mathbf{A}\mathbf{x}^\top$$

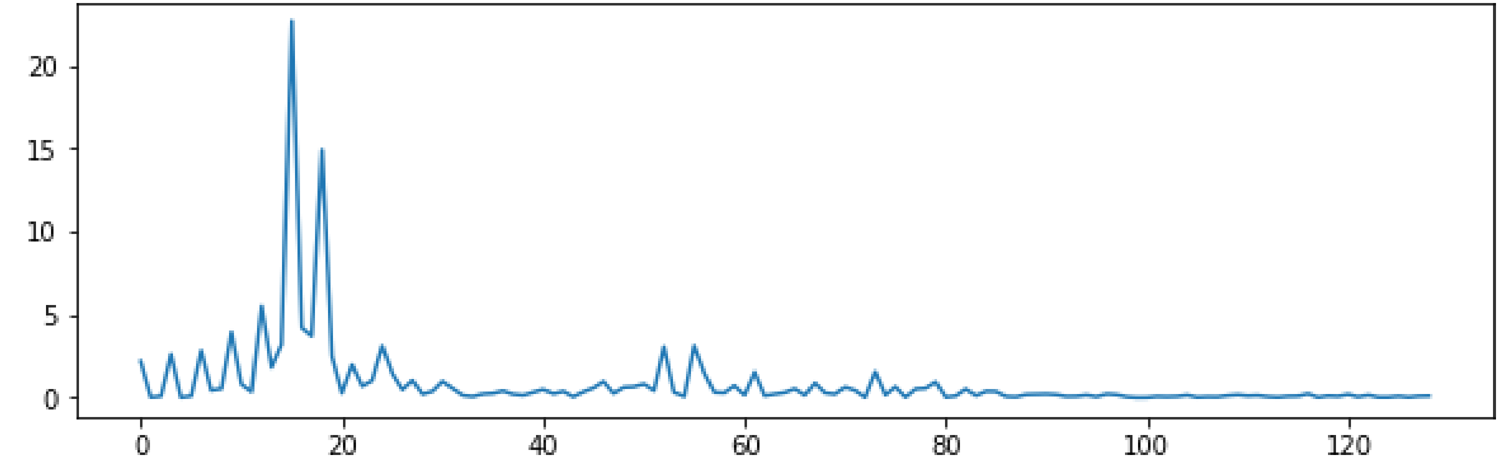
#full_cos_anal

Result of analysis

- Coefficients c_k



- Coefficients c_k
  in absolute values $|c_k|$



Is this OK ???

Let’s try to re-synthesize the signal …

- Multiply each basis by the respective coefficient and sum it all

$$x_s[n] = c_0 + c_1 a_1[n] + c_2 a_2[n] + \ldots + c_{\frac{N}{2}} a_{\frac{N}{2}}[n]$$

- For our cosines

$$x_s[n] = c_0 + c_1 \cos(2\pi \frac{1}{N} n) + c_2 \cos(2\pi \frac{2}{N} n) + \ldots + c_{\frac{N}{2}} \cos(2\pi \frac{\frac{N}{2}}{N} n)$$

- Also nicely doable as vector-matrix multiplication: $\boldsymbol{x}_s = \boldsymbol{c}^T \boldsymbol{A}$

#full_cos_synt

Failure – WHY ???

Phase is the problem !

- “Unknown” signal are **2 periods of a sine**
- Analysis signals will be
  - Cosine – 2 periods in *N* samples **zero, that’s bad !!!**

#2sin_analysis

$\Rightarrow$ **we’ll need to analyze the signals by both cosine and sine !**

#2cos_sin_analysis

Analysis with whole groups of cosines and sines

$$a_0[n] = \cos(2\pi \frac{0}{N} n) \quad b_0[n] = \sin(2\pi \frac{0}{N} n)$$

$$a_1[n] = \cos(2\pi \frac{1}{N} n) \quad b_1[n] = \sin(2\pi \frac{1}{N} n)$$

$$a_2[n] = \cos(2\pi \frac{2}{N} n) \quad b_2[n] = \sin(2\pi \frac{2}{N} n)$$

$$\ldots \qquad\qquad \ldots$$

$$a_{\frac{N}{2}}[n] = \cos(2\pi \frac{\frac{N}{2}}{N} n) \quad b_{\frac{N}{2}}[n] = \sin(2\pi \frac{\frac{N}{2}}{N} n)$$

- How will the analysis signals for limit values look like ?

Let's go

$$c_0 = \sum_{n=0}^{N-1} a_0[n]x[n] \qquad d_0 = \sum_{n=0}^{N-1} b_0[n]x[n]$$

$$c_1 = \sum_{n=0}^{N-1} a_1[n]x[n] \qquad d_1 = \sum_{n=0}^{N-1} b_1[n]x[n]$$

$$c_2 = \sum_{n=0}^{N-1} a_2[n]x[n] \qquad d_2 = \sum_{n=0}^{N-1} b_2[n]x[n]$$

$$\ldots \qquad \ldots$$

$$c_{\frac{N}{2}} = \sum_{n=0}^{N-1} a_{\frac{N}{2}}[n]x[n] \qquad d_{\frac{N}{2}} = \sum_{n=0}^{N-1} b_{\frac{N}{2}}[n]x[n]$$

$$\mathbf{c} = \mathbf{A}\mathbf{x}^\top$$

$$\mathbf{d} = \mathbf{B}\mathbf{x}^\top$$

#full_cos_sin_anal

Looks pretty good, how about re-synthesis ?

- Again, multiply each basis by the respective coefficient and sum it all

$$
\begin{aligned}
xs[n] \quad = c_0 \quad & + c_1 \cos(2\pi \frac{1}{N} n) + c_2 \cos(2\pi \frac{2}{N} n) + \ldots + c_{\frac{N}{2}} \cos(2\pi \frac{\frac{N}{2}}{N} n) \\
& + d_1 \sin(2\pi \frac{1}{N} n) + d_2 \sin(2\pi \frac{2}{N} n) + \ldots + d_{\frac{N}{2}} \sin(2\pi \frac{\frac{N}{2}}{N} n)
\end{aligned}
$$

- Also nicely doable as vector-matrix multiplication: $\boldsymbol{x}_s = \boldsymbol{c}^T \boldsymbol{A} + \boldsymbol{d}^T \boldsymbol{B}$

#full_cos_sin_synt

Works ! However, keeping track of cosines and sines is a bit complicated …

The ultimate basis for spectral analysis – a complex exponential

- Complex exponential contains **both cosine and sine in one function**

$$a[n] = e^{j2\pi \frac{k}{N} n} = \cos 2\pi \frac{k}{N} n + \sin 2\pi \frac{k}{N} n$$

Complex exponentials are also harmonically related: the first has normalized frequency *1/N*, the next *2/N*, etc …

#complex_exp

- Special cases
  - k = 0 – analysis of the D.C. component (always 1)
  - k = N/2 – analysis of the fastest function – each sample changes polarity
  - Both are **real** !

Analysis by complex exponentials

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi \frac{k}{N} n}$$

- Why the **minus** ???
- Projection to real numbers: in case |a| = 1, it is enough to multiply with the base and we obtain a good coefficient for reconstruction:
  x = 3, b = 1: projection *c = xb = 3*, reconstruction: *cb = 3 . 1 = 3* **OK**
- Projection to a complex number: *x = 3,* $b = \frac{1}{\sqrt{2}}(1+j)$
  projection: $c = x \times b = 3 \times \frac{1}{\sqrt{2}}(1+j) = \frac{1}{\sqrt{2}}(3+3j)$
  reconstruction: $x = c \times b = (3+3j) \times \frac{1}{\sqrt{2}}(1+j) = 3j$ **not OK**

Analysis involves conjugating the complex basis

- Fixing this problem - take a complex conjugate (komplexní sdružení) of the basis:

$$c = x \times b^{\star} = 3 \times \frac{1}{\sqrt{2}}(1-j) = \frac{1}{\sqrt{2}}(3-3j)$$

$$c \times b = 6.$$

- More generally: for analysis signal *a[n]*, that is complex, take it's complex conjugate $a^{*}[n]$. For our bases

$$a[n] = e^{j2\pi\frac{k}{N}n} \qquad a^{\star}[n] = e^{-j2\pi\frac{k}{N}n}$$

- The same complex exponential, but turning in opposite direction

#plus_minus_complex_exp

Discrete Fourier transform (Diskrétní Fourierova transformace) (DFT)

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi \frac{k}{N} n}$$

$$a_0[n] = e^{j2\pi \frac{0}{N} n} \qquad X[0] = \sum_{n=0}^{N-1} a_0^\star[n] x[n]$$

$$a_1[n] = e^{j2\pi \frac{1}{N} n} \qquad X[1] = \sum_{n=0}^{N-1} a_1^\star[n] x[n]$$

$$a_2[n] = e^{j2\pi \frac{2}{N} n} \qquad X[2] = \sum_{n=0}^{N-1} a_2^\star[n] x[n]$$

$$\dots$$

$$a_{\frac{N}{2}}[n] = e^{j2\pi \frac{\frac{N}{2}}{N} n} \qquad X[\frac{N}{2}] = \sum_{n=0}^{N-1} a_{\frac{N}{2}}^\star[n] x[n]$$

$$\mathbf{X} = \mathbf{A}^\star \mathbf{x}^\top$$

- What does it tell us ?
  - Index k tells us **which frequency**
  - Magnitude (absolutní hodnota, modul) $|X[k]|$ tells us **how much**
  - Phase (fáze, úhel, argument, fázový posuv) arg $X[k]$ tells us **how shifted**

#dft_anal

Synthesis from all this …

- We have done synthesis as $x_s[n] = c_0 + c_1 a_1[n] + c_2 a_2[n] + \ldots + c_{\frac{N}{2}} a_{\frac{N}{2}}[n]$
- So here it should be

$$x_s[n] = X[0] + X[1] e^{j2\pi \frac{1}{N} n} + X[2] e^{j2\pi \frac{2}{N} n} + \ldots + c_{\frac{N}{2}} e^{j2\pi \frac{\frac{N}{2}}{N} n}$$

- Ooops, we'd get something complex ☹
- We could extract the magnitudes and phases from complex coefficients and do something like this

$$x_s[n] = X[0] + \sum_{k=0}^{\frac{N}{2}} 2|X[k]| \cos\left(2\pi \frac{k}{N} n + \arg X[k]\right)$$

- But it would be complicated as a hell, and we love our complex exponentials, so …

Complex exponentials only …

- We use the old trick with a conjugate complex exponential turning in the opposite direction
  - Definition: $$\cos\alpha = \frac{e^{j\alpha} + e^{-j\alpha}}{2}$$
  - Our case $$X[k]e^{j2\pi\frac{k}{N}n} + X^{\star}[k]e^{-j2\pi\frac{k}{N}n} = 2|X[k]|\cos\left(2\pi\frac{k}{N}n + \arg X[k]\right)$$

#discrete_cos_decomposition (reproduced from the last lecture)

- But we absolutely want to stay in **positive** indices *k*, a bit of math will help us to go from –*k* (negative) to *N-k* (positive):

$$e^{-j2\pi\frac{k}{N}n} = e^{-j(2\pi\frac{k}{N}-2\pi)n} = e^{j2\pi\frac{N-k}{N}n}$$

DFT with all *N* output coefficients

- We perform the analysis $X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi \frac{k}{N} n}$ for the full range $k = 0 \dots N-1$ knowing that there will be symmetry between
  - The coefficients: $X[k] = X^*[N-k]$, this means $|X[k]| = |X[N-k]|$ and arg $X[k]$ = - arg $X[k]$
  - The complex exponentials: $\left(e^{j2\pi \frac{k}{N} n}\right)^\star = e^{-j2\pi \frac{k}{N} n}$
- We'll perform the synthesis $x_s[n] = \sum_{k=0}^{N-1} X[k] e^{+j2\pi \frac{k}{N} n}$ again with the full range of coefficients $k = 0 \dots N-1$ knowing that
  - $k = 0$ will produce a D.C. value (real signal)
  - $k = N/2$ will produce a real signal as well *(+1,-1,+1,-1, …)*
  - For other *k*'s, pairs *k* and *N-k* will produce two complex exponentials running in the opposite direction, summing up to a cosine:

$$X[k]e^{j2\pi \frac{k}{N} n} + X[-k]e^{-j2\pi \frac{k}{N} n} = 2|X[k]| \cos\left(2\pi \frac{k}{N} n + \arg X[k]\right)$$

Full DFT spectrum and how many values do we have ?

#full_dft_anal

- Let's count the floats:
  - $X[0]$ – real, 1 float
  - $X[1 \ldots N/2-1]$ – complex, 2 floats each: $2(N/2 - 1)$
  - $X[N/2]$ – real, 1 float
  - $X[N/2+1 \ldots N-1]$ – no floats needed, we already have them: $X[k] = X^*[N-k]$
- All together: $1 + 2(N/2 - 1) + 1 = N$, **OK, DFT preserves all the information**

DFT synthesis – inverse Discrete Fourier Transform (inverzní diskrétní Fourierova transformace)

$$x_s[n] = \sum_{k=0}^{N-1} X[k] e^{+j2\pi \frac{k}{N} n}$$

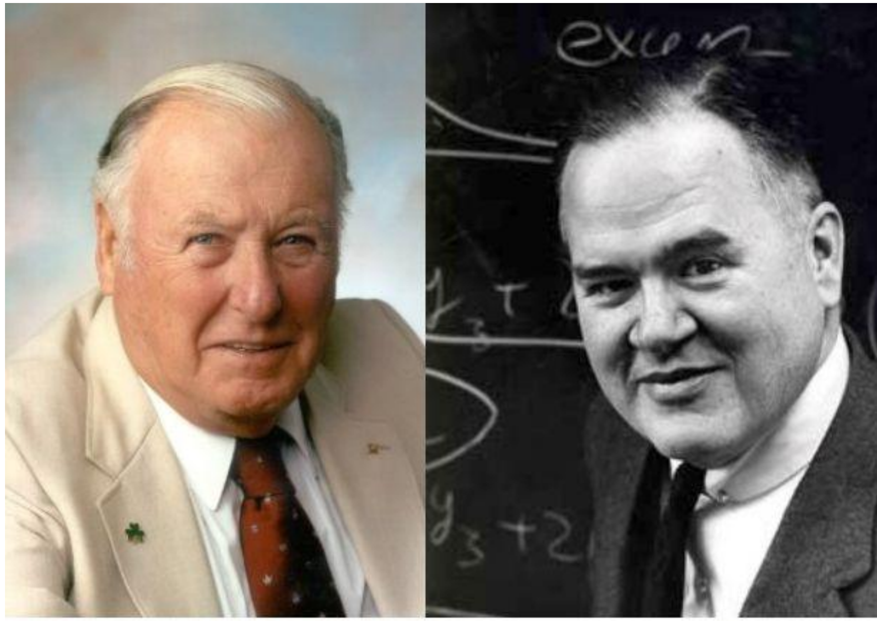
#full_dft_synt

- Very nice, except for the dynamic range … max = 0.57 for the original signal and 147 for the synthesized one ☹
- … it has something to do with the “normality” of DFT bases (see next lecture).
- Now, we just apply a correction term, so the ultimate definition of DFT and IDFT is

$$\boxed{X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi \frac{k}{N} n}, \qquad x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{+j2\pi \frac{k}{N} n}}$$

Using DFT in your code

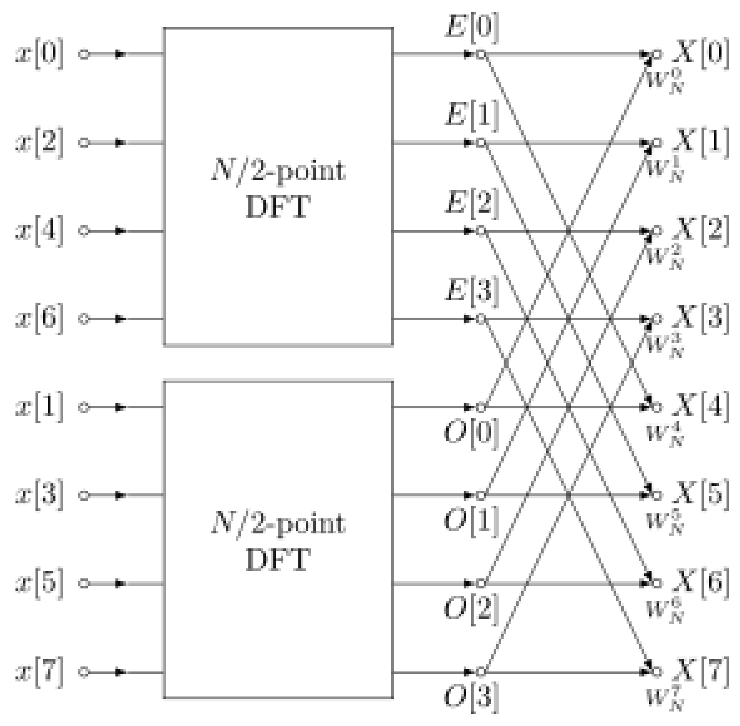
- DFT is computation hungry
  - Computation of each output coefficient involves *N* complex mutliplications and *N* complex additions.
  - And there are *N* output coefficients, co that the complexity is $2N^2$
  - Quadratic complexity is bad even with modern computers with GPUs and was even worse in the 60's where a computer occupying whole room had computing power smaller than your smart watch.
- Developing FFT in the 60's completely revolutionized the signal processing



James William Cooley (1926-)2016

John Wilder Tukey (1915-2000)

Fast Fourier transform (rychlá Fourierova transformace) - FFT



- Works only for powers of 2: $N = 2^b$
- Works in b stages and uses the symmetries of complex numbers, each stage needs only N operations, the graph looks like butterflies => butterly algorithm
- The complexity goes from N^2 to $N \log_2 N$
- **fft** is part of all numerical libraries in all possible languages.
- Sorry, no time to derive it thoroughly, but look at one of many sources (incl. Wikipedia)

Remember FFT is not a new transform, it is a fast implementation of DFT !

#fft_anal_synt

Showing and interpreting the result of DFT - frequency

- Your boss/colleague/customer won't like you for just the index *k* on x-axis
- **Normalized frequency (normovaná frekvence)** k / N is better
- **Regular frequency** in **Hertz** is even better
  - Need the **sampling frequency (vzorkovací frekvence)** F_s [Hz] – number of samples per second. Sampling period **(vzorkovací perioda)** $T_s = 1 / F_s$ is in seconds.
- many ways to derive the conversion but let's do it this simple way:
  - The period of a cos (or sine or complex exp.) with normalized frequency of $1/N$ is N samples.
  - The period in time is $N T_s$. Therefore the frequency is $1/(N T_s) = F_s / N$.

Normalizing and de-normalizing frequencies involves just by division and multiplication by the sampling frequency F_s.

#dft_freq_axes

Showing and interpreting the result of DFT – going to F_s or just $F_s/2$?

- We usually show only the **left** part of the DFT spectrum as the right one is not informative.
  - indices $0 \dots N/2$, attention, this is $N/2+1$ coefficients, not $N/2$!!! For $N=256$, we'll need to keep 129 values !!!
  - normalized frequencies $0 \dots 1/2$,
  - regular frequencies $0 \dots F_s/2$

#dft_final_visualization

Attention: the halves are symmetrical only for real input signals. Beware in case you have to process anything complex (digital radio, microphone arrays, …)

Frequency resolution of DFT

- Interval 0 … F_s is divided into N samples, therefore the frequency resolution is F_s / N.

14	A♯,/B♭,	A♯1/B♭1	58.2705
13	A,	A1	55.0000
12	G♯,/A♭,	G♯1/A♭1	51.9131
11	G,	G1	48.9994
10	F♯,/G♭,	F♯1/G♭1	46.2493
9	F,	F1	43.6535

- This is not much, consider an example:
  - F_s = 48000 Hz.
  - Trying to tune low tones on a piano
  - Performing DFT with N = 256 samples.
  - The resoluition is 48000 / 256 = 187 Hz – quite bad if we need units of Hz !

Increasing the resolution – option I.

- Increase the number of samples, in case we have F_s = 48000 Hz, let's take N = 65536 and run FFT
- This will take lots of computation !
- The missing samples can be either taken from the signal (if we have them) or filled by zeros – **zero padding** (doplňování nulami)

#zero_padding

With zero padding, the result looks nicer, but there is no new information !

or … Discrete time Fourier Transform (Fourierova transformace s diskrétním časem) - DTFT

- DFT sets the normalized frequency points to multiples of *1 / N*
- DTFT can work with anything – usually defined with normalized angular frequency $\omega = 2\pi \frac{f}{F_s}$

$$\boxed{\tilde{X}(e^{j\omega}) = \sum_{n=0}^{N-1} x[n] e^{-j\omega n}}$$

- We can set the range of frequencies to anything we want.
- However, needs to compute by definition and therefore much slower than FFT.
- Let us show analysis of our speech spectrum with better precision around the maximum

#dtft study carefully, you might need it in the project ☺

SUMMARY

- We analyze by multiplying and summing vectors
- Difficult signals are analyzed by harmonically related functions
  - Cosines – not enough
  - Cosines and sines – too complicated
  - Ultimate solution: complex exponentials => DFT
- The original signal can be fully re-synthesized – IDFT (making use of properties of complex numbers to get a real signal)
- The results are there for N discrete frequencies from 0 till almost F_s
  - Of these, only N/2+1 are worth showing
  - with a nice frequency axis !
- Frequency resolution is limited but can be improved
  - by zero padding
  - By switching to DTFT