Short Time Fourier Transform. Spectrograms.

Mathematical Tools for ITS (11MAI)

Mathematical tools, 2021

Jan Přikryl

11MAI, lecture 4

Monday, October 11, 2021

version: 2021-10-19 09:36

Department of Applied Mathematics, CTU FTS

Application of DFT

Computer session 1

DFT of Non-stationary Signals

Windowing and Localization

Short-time Fourier Transform

Spectrograms

Assignment

Matlab Session 4.1

Consider the analog signal

$$x(t) = 2.0\cos(2\pi\,5t) + 0.8\sin(2\pi\,12t) + 0.3\cos(2\pi\,47t)$$

on the interval $t \in [0, 1)$. Sample this signal with period $T = 1/128\,\mathrm{s}$ and obtain sample vector $\mathbf{x} = (x_0, x_1, x_2, \ldots, x_{127})$.

a) Create a MATLAB m-file which plots signals $x(t)$ and $\mathbf{x}$

b) Using the definition of the DFT from the last lecture find $\mathbf{X}$.

c) Use the MATLAB function `fft(x)` to compute DFT of $\mathbf{X}$.

d) Make a MATLAB m-file which computes the DFT of $\mathbf{x}$ and plots the signal and its spectrum.

e) Compute the IDFT of the $\mathbf{X}$ and compare it with the original signal $x(t)$.
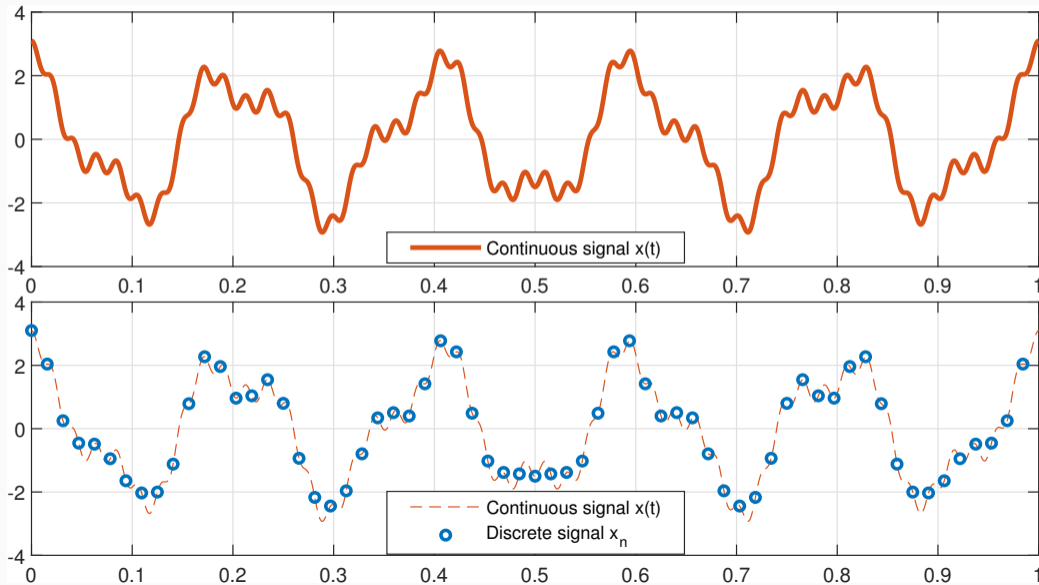
```matlab
clear
% The "continuous" original signal
t = linspace(0,1,1001);
x = 2.0*cos(2*pi*5*t) + 0.8*sin(2*pi*12*t) + 0.3*cos(2*pi*47*t);
% The sampled signal
N = 128;  % number of samples
ts = linspace(0,1,N+1);  % the last sample is at t=1
ts(end) = [];  % now we have N time samples
xs = 2.0*cos(2*pi*5*ts) + 0.8*sin(2*pi*12*ts) + 0.3*cos(2*pi*47*ts);
```

```matlab
figure(1);
subplot(2,1,1);
plot(t,x,'LineWidth',2.5,'Color',[1 0 0]);
grid on;
subplot(2,1,2);
plot(ts, xs,'o','LineWidth',2.0,'Color',[0 0 1]);
hold on;
plot(t,x,'--','Color',[1 0 0]);
grid on;
legend('Discrete signal x(n)','Continuous signal x(t)');
hold off;
pause
```

Computing **X** using the definition of the DFT is not so complicated:

```
X = zeros(N,1);
for k=0:(N-1)
  % Sum of N basis function samples forms X_k
  xk = 0;
  for m=0:(N-1)  % Note: 'm' goes from 0, but MATLAB indexes from 1
    % Note: -1j denotes the imaginary unit
    xk = xk + xs(m+1)*exp(-1j*2*pi*k*m/N);  % Matlab indexes start at 1
  end
  X(k+1) = xk;
end
```

Q: How many times will the `xk` update code be executed?

In its original sense, (non)stationarity is a property of stochastic processes:

Definition (Stationarity)

A stochastic process is said to be stationary if its unconditional joint probability distribution does not change when shifted in time. Consequently, its parameters such as mean and variance do not change over time.

A signal is an observation of events that correspond to a result of some process. If the properties of the process that generates the events do not change in time, then the process is stationary.

In such a case we (not quite correctly) say that the signal is stationary.

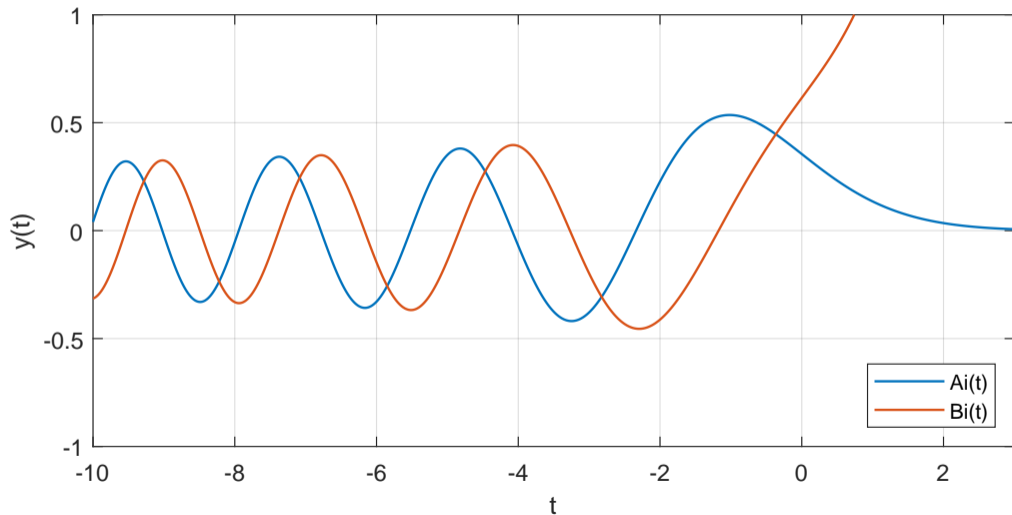Non-stationary systems $\Leftrightarrow$ differential/difference equations with time-varying coefficients, e.g.,

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}y(t) - t\,y(t) = 0.$$

Solution for this particular case is Airy's functions

$$\mathrm{Ai}(t) = \frac{1}{\pi}\int_0^\infty \cos\left(\frac{\tau^3}{3} + t\tau\right)\,\mathrm{d}\tau$$

$$\mathrm{Bi}(t) = \frac{1}{\pi}\int_0^\infty \left[\exp\left(-\frac{\tau^3}{3} + t\tau\right) + \sin\left(\frac{\tau^3}{3} + t\tau\right)\right]\,\mathrm{d}\tau$$

Signal $y(t)$ is an output of a non-stationary process/system $\Rightarrow$ we say that the signal is non-stationary.
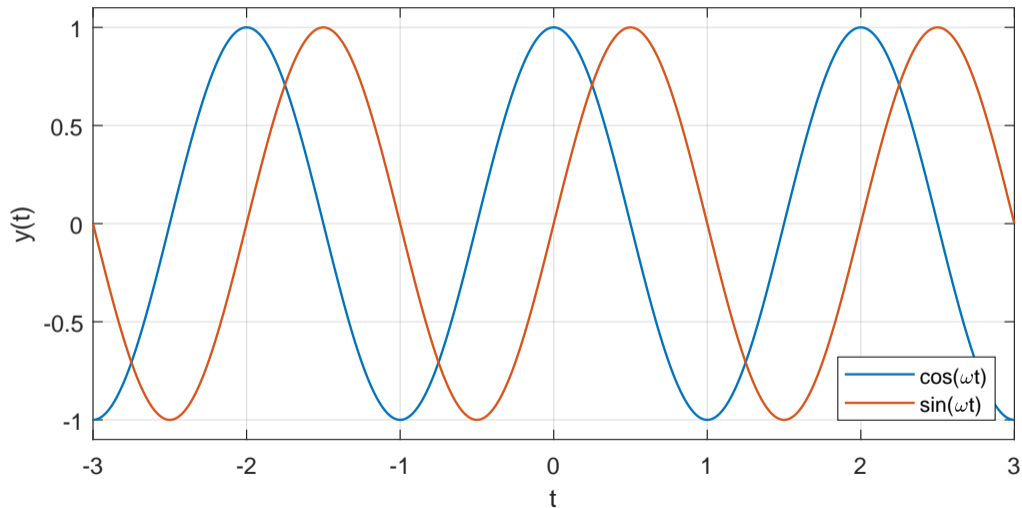
Stationary systems $\Leftrightarrow$ differential/difference equations with constant coefficients, e.g.,

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2} y(t) + \omega_0^2 \, y(t) = 0.$$

For this particular case the solution is a harmonic wave

$$\cos \omega_0 t, \ \sin \omega_0 t$$

Signal $y(t)$ is an output of a stationary process/system $\Rightarrow$ we say that the signal is stationary.

A deterministic signal is said to be stationary if it can be written as a discrete sum of cosine waves or exponentials:

$$x(t) = A_0 + \sum_{k=1}^{N} A_k \cos(\omega_k t + \Phi_k)$$

$$x(t) = \sum_{k=-N}^{N} C_k e^{j\omega_k t + \Phi_k}$$

i.e. as a sum of elements which have constant instantaneous amplitude and instantaneous frequency.

In the discrete case, a sequence $(x_n)_n$ assumed to be sampled from an output of a random process is said to be wide-sense stationary (or stationary up to the second order) if its variance is independent of time:

$$\forall m : \text{var}(X_{(m:m+M)}) = E[(x - \mu)^2] = \frac{1}{M-1} \sum_{k=m}^{M} (x_k - \mu)^2 = \sigma^2$$

Here, the population mean μ is always computed over the corresponding slice $m : m + M$.

A signal is said to be non-stationary if one of these fundamental assumptions is no longer valid, e.g. either variance σ^2, or autocorrelation $\varrho_{xx}(n, n, m)$, or both are time-varying.

For example:

- a finite duration signal, and in particular a transient signal (for which the length is short compared to the observation duration), is non-stationary.
- speech and EEG are non-stationary signals.
- however, in specific situations may short time sections of EEG be considered stationary.

DFT assumes the signal is stationary. It cannot detect local frequency or phase changes.

Example (Frequency hop)

Consider two different periodic signals $f(t)$ and $g(t)$ defined on $0 \leq t < 1$ with frequencies $f_1 = 96$ Hz and $f_2 = 235$ 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 $f_s = 1000$ Hz to produce sample vectors $\mathbf{f}$ and $\mathbf{g}$.
Compute the DFT of each sampled signal.

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

```matlab
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;
```
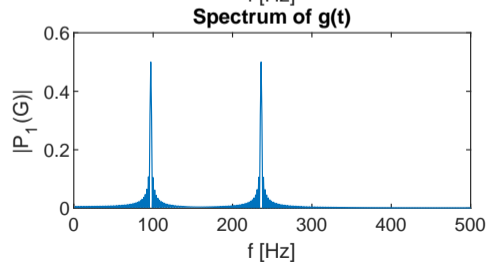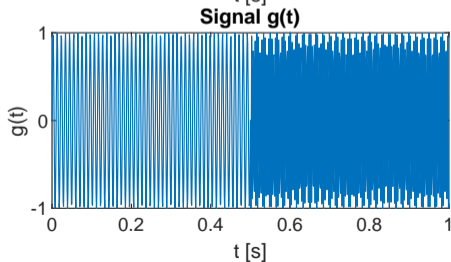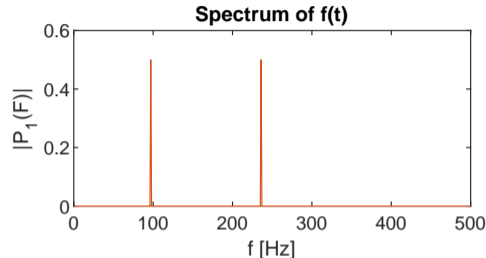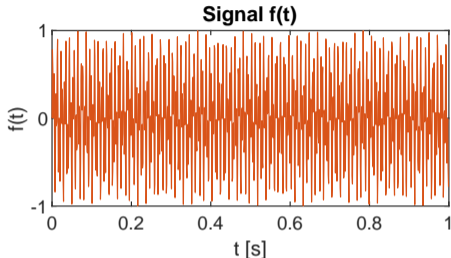
- It is obvious that each signal contains dominant frequencies close to 96 Hz and 235 Hz and the magnitudes are fairly similar.
- But: The signals $f(t)$ and $g(t)$ are quite different in the time domain!
- The example illustrates one of the shortcomings of traditional Fourier transform: nonlocality or global nature of the basis vectors $\mathbf{w}_N$ or its constituting analog waveforms $e^{j2\pi kt/T}$.
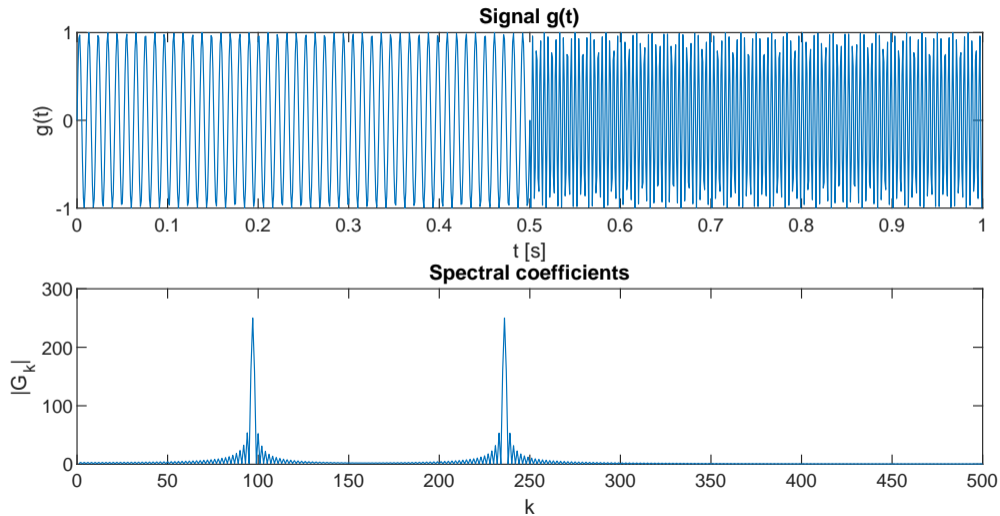
Summary:

- 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 make the frequency analysis more local — e.g. by analysing smaller portions of the signal.

Consider basis functions $\sin \omega_k t$, $\cos \omega_k t$ and $\delta(t)$ and their support:

support region	in time	in frequency
$\sin \omega_k t$	$(-\infty, \infty)$	0
$\cos \omega_k t$	$(-\infty, \infty)$	0
$\delta(t)$	0	$(-\infty, \infty)$

- The basis functions $\sin \omega_k t$ and $\cos \omega_k t$ cannot localize time!
- The $\delta(t)$ cannot localize frequency!

To localize changes in the signal in time domain by FFT we need to look at shorter parts of the signal — time windows.

Computing DFT of long signals is unfeasible:

- When sampling an audio signal at a sampling rate 44.1 kHz, 1 hour of stereophonic music would be $44\,100 \times 2 \times 60 \times 60 = 317\,520\,000$ samples!
- If we want to compute DFT, the closest power-of-two FFT is $2^{28} = 268\,435\,456$ per channel.
- A better approach is to break the long signal into small segments and analyze each one with FFT

These requirements led to development of windowed version of Fourier transform — the short-time Fourier transform, STFT.

Application of DFT
DFT of Non-stationary Signals
Windowing and Localization
    Computer session 2
Short-time Fourier Transform
Spectrograms
Assignment

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 $\mathbf{y} = \mathbf{w}\mathbf{x}$ 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

$$Y_n = \frac{1}{N}\sum_{m=0}^{N-1} X_m W_{(n-m)\,\mathrm{mod}\,N}.$$

Definition

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.

Main problem: spectral leakage

Competing properties:

- high resolution — ability to distinguish close frequencies
- high dynamic range — ability to distinguish frequencies with different amplitudes

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.

High resolution $\times$ low dynamic range: good separation of similar amplitudes for similar frequencies, poor at distinguishing far away frequencies of different amplitudes.

Definition (Hamming window)

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

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

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

A frequently used form of Hann window, better dynamic range at the cost of some resolution.
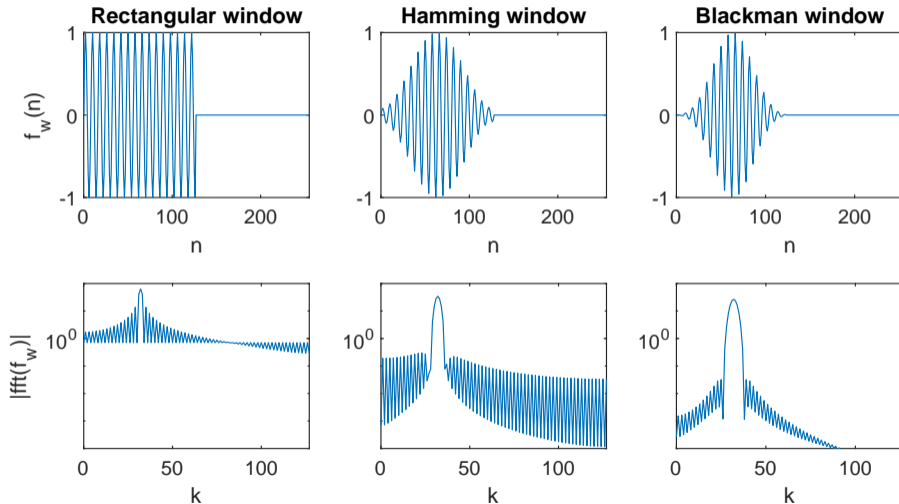
Definition (Blackman window)

Another common type of window is the Blackman window:

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

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

Better dynamic range than Hamming at the cost of some resolution.

Matlab Session 4.2

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 $f_1 = 137\,\text{Hz}$ and $f_2 = 147\,\text{Hz}$:

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

```matlab
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
```

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 d) or shorter vector as in part g)? Does the side lobe energy confuse the results?

- Assume that $(x_n)_{n=0}^{\infty}$ is an infinitely long sequence
- In order to localize energy in both time and frequency we segment the signal into short-time pieces and calculate DFT of each segment
- Sampled STFT for a window $(w_m)_{m=0}^{M-1}$ defined in the region $0 \leq m \leq M-1$ is given by

$$X_{k,\ell} = \sum_{m=0}^{M-1} w_m \cdot x_{\ell-m} \, e^{-j2\pi\frac{km}{N}}$$

Time-frequency analysis using STFT
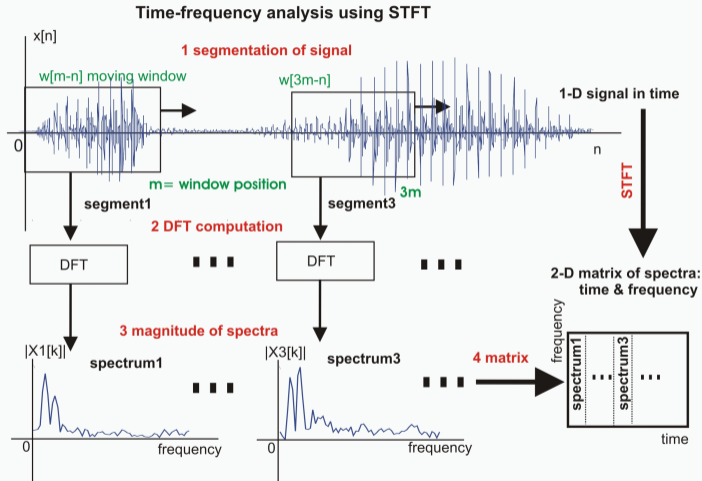
STFT = signal segmentation using window & repeatedly used FT (or DFT)
visualisation of matrix: surface or spectrogram (next slide)
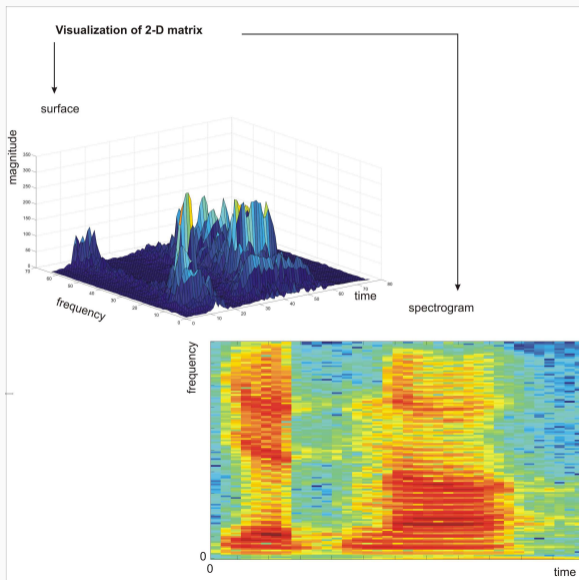
Application of DFT

DFT of Non-stationary Signals

Windowing and Localization

Short-time Fourier Transform

Spectrograms

Assignment

In MATLAB the command

```
spectrogram(x,window,noverlap,nfft,fs,'yaxis')
```

performs short-time Fourier transform and plots a 2D frequency-time diagram, where

- `x` is the signal specified by vector x.
- if `window` is an integer, `x` is divided into segments of length equal to that integer value
- otherwise, `window` is a Hamming window of length `nfft`
- `noverlap` is the number of samples each segment of `x` overlaps

43

In MATLAB the command

```
spectrogram(x,window,noverlap,nfft,fs,'yaxis')
```

performs short-time Fourier transform and plots a 2D frequency-time diagram, where

- x is the signal specified by vector x.
- if window is an integer, x is divided into segments of length equal to that integer value
- otherwise, window is a Hamming window of length nfft
- noverlap is the number of samples each segment of x overlaps

In MATLAB the command

```
spectrogram(x,window,noverlap,nfft,fs,'yaxis')
```

performs short-time Fourier transform and plots a 2D frequency-time diagram, where

- `x` is the signal specified by vector x.
- if `window` is an integer, `x` is divided into segments of length equal to that integer value
- otherwise, `window` is a Hamming window of length `nfft`
- `noverlap` is the number of samples each segment of `x` overlaps

In MATLAB the command

```
spectrogram(x,window,noverlap,nfft,fs,'yaxis')
```

performs short-time Fourier transform and plots a 2D frequency-time diagram, where

- `x` is the signal specified by vector x.
- if `window` is an integer, `x` is divided into segments of length equal to that integer value
- otherwise, `window` is a Hamming window of length `nfft`
- `noverlap` is the number of samples each segment of `x` overlaps

In MATLAB the command

```
spectrogram(x,window,noverlap,nfft,fs,'yaxis')
```

performs short-time Fourier transform and plots a 2D frequency-time diagram, where

- `nfft` is the FFT length and is the maximum of 256 or the next power of 2 greater than the length of each segment of `x`
- `fs` is the sampling frequency, which defaults to normalized frequency
- using `'yaxis'` displays frequency on the *y*-axis and time on the *x*-axis

In MATLAB the command

```
spectrogram(x,window,noverlap,nfft,fs,'yaxis')
```

performs short-time Fourier transform and plots a 2D frequency-time diagram, where

- `nfft` is the FFT length and is the maximum of 256 or the next power of 2 greater than the length of each segment of `x`
- `fs` is the sampling frequency, which defaults to normalized frequency
- using `'yaxis'` displays frequency on the y-axis and time on the x-axis

In MATLAB the command

```
spectrogram(x,window,noverlap,nfft,fs,'yaxis')
```

performs short-time Fourier transform and plots a 2D frequency-time diagram, where
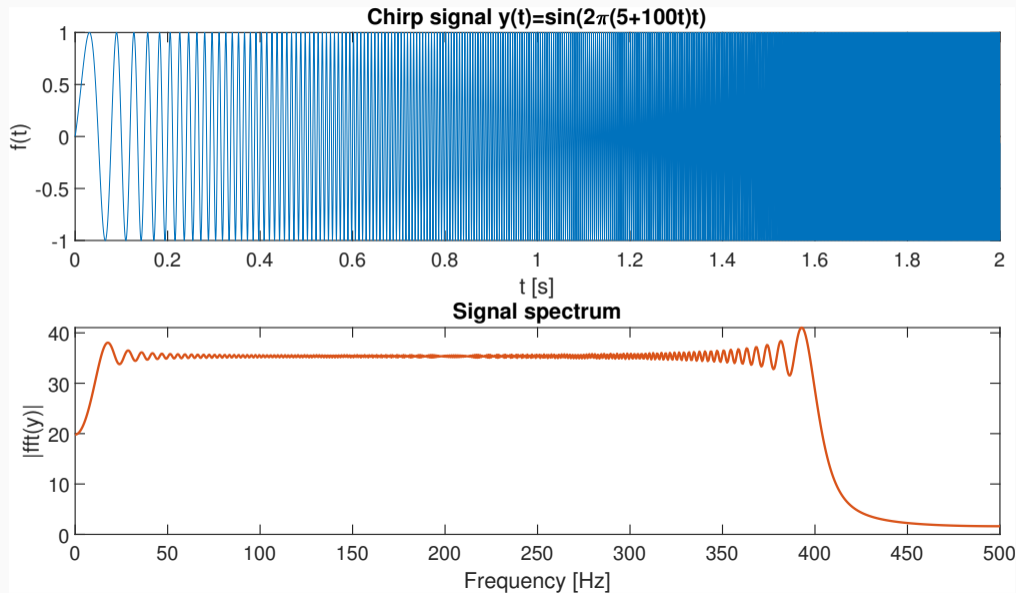
- `nfft` is the FFT length and is the maximum of 256 or the next power of 2 greater than the length of each segment of `x`
- `fs` is the sampling frequency, which defaults to normalized frequency
- using `'yaxis'` displays frequency on the *y*-axis and time on the *x*-axis
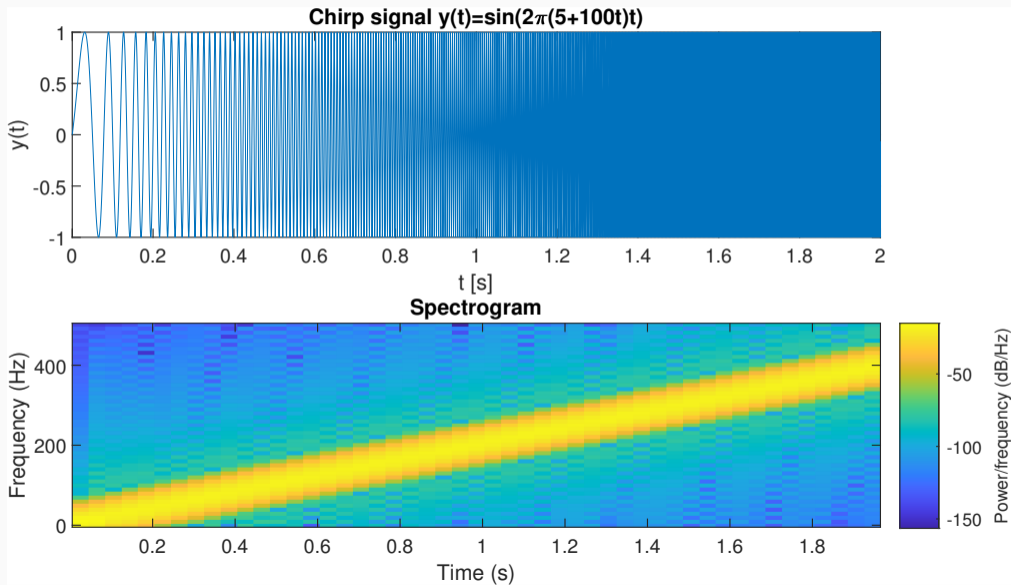
In MATLAB the command

```
spectrogram(x,window,noverlap,nfft,fs,'yaxis')
```

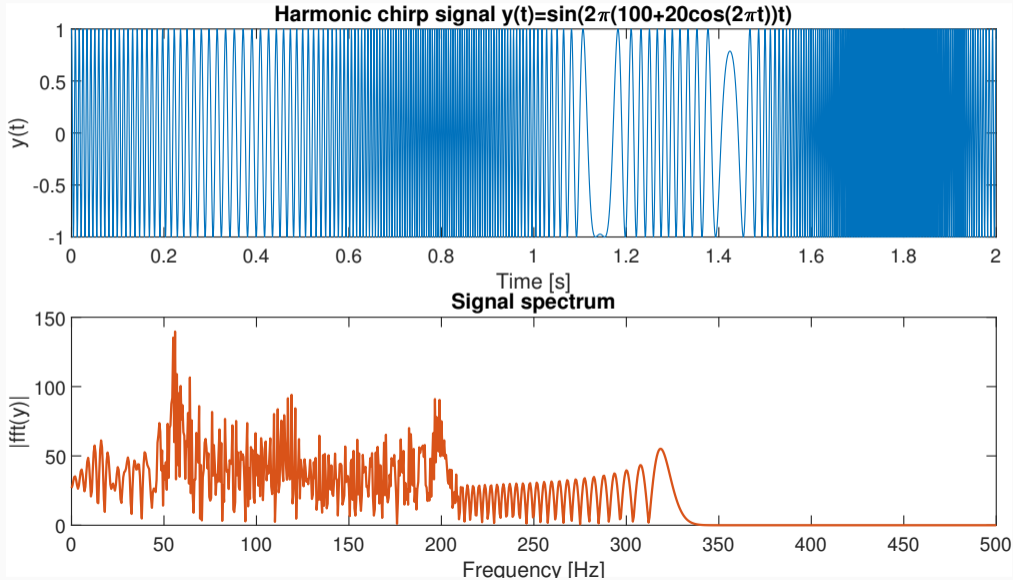performs short-time Fourier transform and plots a 2D frequency-time diagram.

In current Matlab versions, the `colorbar` command is automatically issued to append a color scale to the current axes.
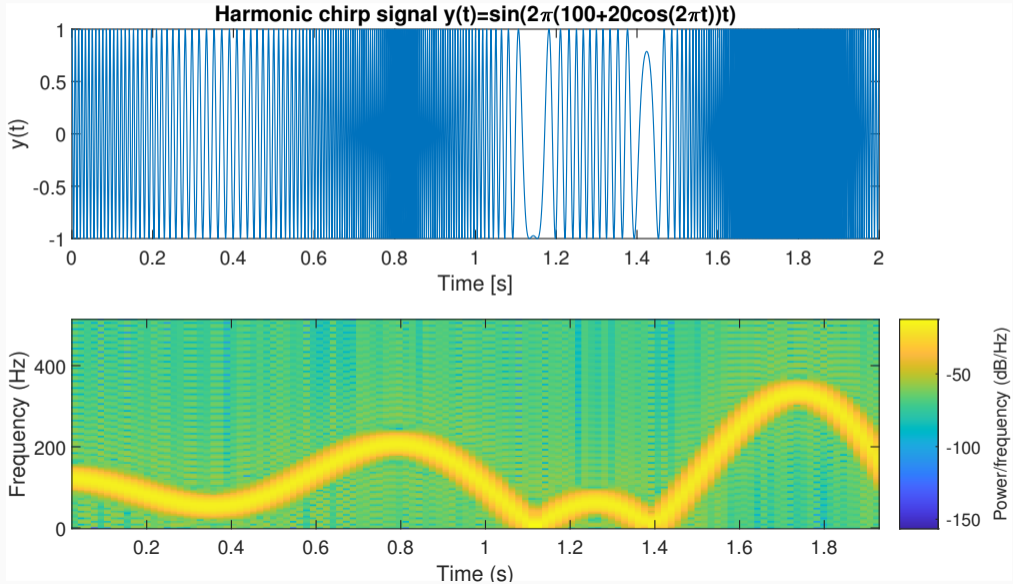
Chirp signal $y(t)=\sin(2\pi(5+100t)t)$

Signal spectrum

44

Chirp signal $y(t)=\sin(2\pi(5+100t)t)$

Spectrogram

Harmonic chirp signal y(t)=sin(2π(100+20cos(2πt))t)

Signal spectrum

- Start MATLAB. Load in the "ding" audio signal with command
  `y=wavread('ding.wav');`. In case that the WAV file is not present on your
  computer, a copy is available for download from 11MAI website.
  The audio signal is stereo one and can be decoupled into two channels by
  `y1=y(:,1); y2=y(:,2);`. The sampling rate is 22 050 Hz, and the signal contains
  20 191 samples. If we consider this signal as sampled on an interval $[0, T)$, then
  $T = 20191/22050 \approx 0.9157$ seconds.

- Compute the DFT of the signal with `Y1=fft(y1);` and `Y2=fft(y2);`. Display the
  magnitude of the Fourier transform with `plot(abs(Y1))` or `plot(abs(Y2))`. The
  DFT is of length 20 191 and symmetric about center.

- Use the data cursor button on plot window to pick out the frequency and amplitude of the two (obviously) largest components in the spectrum. Compute the actual value of each significant frequency in Hertz.

- Note:
  - Since MATLAB indexes from 1, the DFT coefficient Y_k is actually `Y(k+1)` in MATLAB! Also Y_k corresponds to frequency $f_k = k/T = k/0.9157$ and so `Y(k)` corresponds to $f_{k-1} = (k-1)/T = (k-1)/0.9157$.
  - The DFT of real-valued signal is symmetrical around $f_s/2$. You can plot only the first half of the DFT with `plot(abs(Y1(1:6441)))` or `plot(abs(Y2(1:6441)))`.

- Let f_1, f_2 denote these two largest frequencies in Hertz, and let A_1, A_2 denote the corresponding amplitudes from the plot. Define these variables in MATLAB as `f1`, `f2`, and `A1`, `A2`.
- Generate a new signal using only these frequencies, sampled at $22\,050\,\text{Hz}$ on the interval $[0, 1)$ with

```
t = [0:1/22050:1];
y12 = (A1*sin(2*pi*f1*t) + A2*sin(2*pi*f2*t))/(A1+A2)
```

- Play the original sound with `sound(y1)` and the synthesized version `sound(y12)`. Repeat the experiment with sound of the second channel `sound(y2)`. Note that our synthesis does not take into account the phase information at these frequencies.
- Does the artificial generated signal reproduce `ding.wav` correctly? Compare the quality!

a) Repeat the parts a)–k) from the Matlab Session 4.2, but this time using a triangular window.

b) A triangular window vector **w** of length $L = 201$ can be constructed using

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

c) 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)`.

d) Try varying the window length L. What is the shortest window that allows you to distinguish the two requencies?

e) Repeat the parts a)–k) from the Matlab Session 4.2 for the Hamming window.

f) Summarise the answers to the questions raised in Matlab Session 4.2 for rectangular, triangular, and Hamming windows as a part of your report.

Submit your report by Friday, October 29, 2021 using the web page
`http://zolotarev.fd.cvut.cz/mni`

Solution report should be formally correct (structuring, grammar).

Images should be vectors, not bitmaps.

Only `.pdf` files are acceptable. Handwritten solutions and `.doc` and `.docx` files ill not be accepted.

Solutions written in TEX (using LyX, Overleaf, whatever) may receive small bonification.