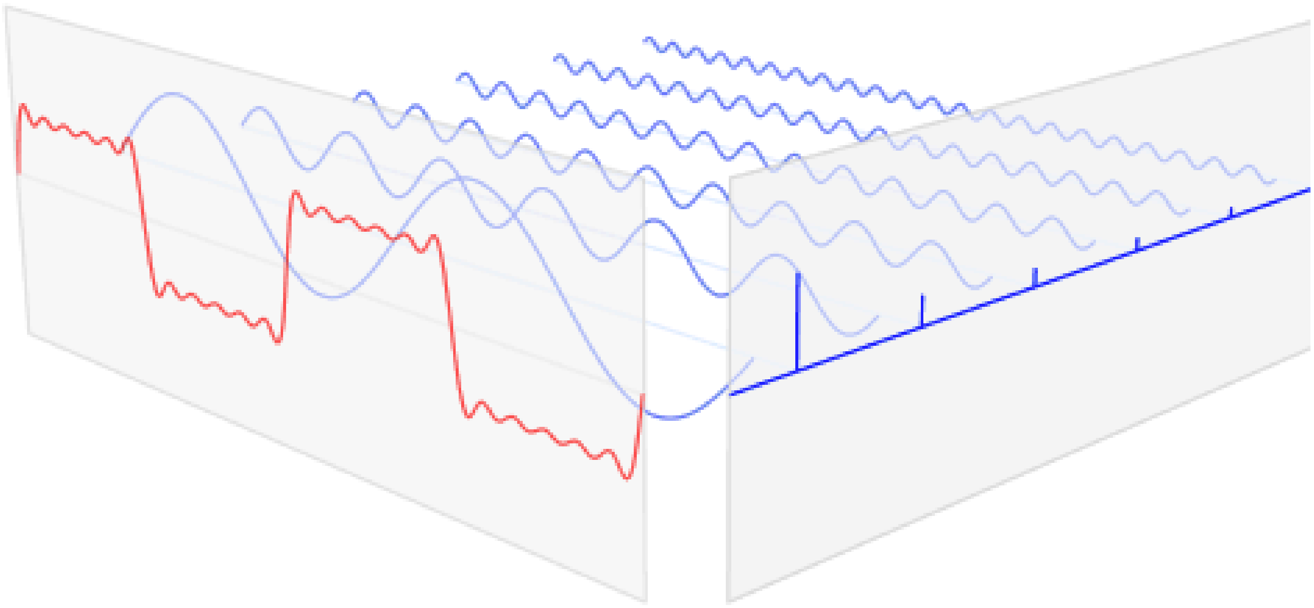
DTU Health Technology, Hearing Systems section
Building 352, DTU, 2800 Lyngby
Phone: 45 25 39 30, Fax: 45 88 05 77 www.hea.healthtech.dtu.dk/

Course 22051: Signals and Linear Systems in Discrete Time

Cloudy...

Bastian Epp: 45 25 39 53 (office), e-mail: bepp@dtu.dk



Fourier analysis of digital signals

1 Time to frequency and frequency to time

In the first part you will analyze signals generated by yourself and some recorded signals. For each of the signals, provide a plot of the time signal and the different spectra (real part, imaginary part, magnitude). If it makes sense, also plot the phase spectrum. For acoustical signals, it makes sense to plot the spectra on a decibel (dB) scale. To do so, convert your vector holding the spectrum from linear amplitude to 'dB re 1' via:

```
signal_dB = 20*log10(abs(vector_with_spectrum));
```

1.1 Toolbox: Spectrum

To make coding easier, you can pack operations that you use frequently into a function, such that you don't have to write the whole code, but just call the function with the appropriate input parameters. Since there will be quite some spectra involved, we start with a function that takes a signal and its sampling frequency, and returns the corresponding complex spectrum along with the correct frequency vector.

- Write a function that returns the complex spectrum of a signal:

```
function [Y, freq] = make_spectrum(signal, fs)
% Here goes the help message
% ...

% compute spectrum (note: it will be complex-valued).
Y = fft(signal);

% The FFT needs to be scaled in order to give a physically plausible scaling.
Y = Y/(length(Y));
  % NOTE: If you do an IFFT, this scaling must NOT be done.
  % We'll get to this in the lecture. If you are only interested
  % in the positive frequencies, you need to scale by <length(Y)/2>.

% frequency vector
delta_f = ...
freq = 0:delta_f: ...
  % NOTE: The first element that comes out of the FFT is the DC offset
  % (i.e., frequency 0). Each subsequent
  % bin is spaced by the frequency resolution <delta_f> which you can
  % calculate from the properties of the inut signal. Remember the highest
  % frequency you can get in a digitized signal....

% ...
% convert into column vector (if required)
Y = Y(:);
freq = freq(:);

% eof
end
```

- Generate a sinusoid of frequency 500 Hz, duration of 0.5 seconds, and amplitude of 1. Pass the signal to your new function. Plot the result over the frequency and make sure the peak is on the correct frequency.

- Change the frequency of the signal to 499 Hz and plot the result in the same graphs as above. Please comment on the result.

1.2 Fourier transform of a synthetic signal

Synthesize the following signal of duration `T = 4` seconds with a suitable sampling frequency:

$$s(n) = \sum_{k=0}^{4} \cos(2\pi \cdot 2^k f_0 \cdot t + k \cdot \frac{\pi}{3}) \quad f_0 = 25; \tag{1.1}$$

- Plot the time signal in the time window from [0.8 s, 0.9 s].

- Compute and plot the spectrum of the whole signal with Hz on the x-axis and linear amplitude on the y-axis. Plot 1) Magnitude and phase for the positive frequencies in two separate panels (magnitude on top of phase) and 2) the real part and imaginary part in two panels (imaginary part on top of real part), including the DC component (0 Hz). Use Euler's formula to explain why the spectrum looks as it does. What would it look like if the signal was composed of sin() rather than cos()?

  `useful command:  subplot`

- Plot the magnitude spectrum with Hz on a log scale on the x-axis and dB on the y-axis. Mark the first peak in the spectrum with a circle.

  `useful command:  semilogx`

- Save the signal as a .wav file with a bit depth of 16 bits and load it again. Make sure the signal is not distorted by saving into a .wav file. Plot the loaded signal on top of the synthetic time signal using a different color in the time interval [0.85 s,0.925 s]. Provide comments on your coding.

1.3 Fourier transform of a recorded signal

Load the sound 'piano.wav' (see DTU LEARN) into Matlab. It is the sound of a piano playing a single note. Make sure you extract the sampling frequency as well.

- Plot the time signal in the time window from 0 s to 1 s.

- Compute and plot the spectrum of the whole signal (whole duration) with Hz on the x-axis and dB on the y-axis. Plot only the positive frequencies, including the DC component (0 Hz).

- Use the plot of the power spectrum (absolute magnitude) to estimate the fundamental frequency of the signal (approximate with a precision of 0.5 Hz, make sure you have the correct frequency vector).

Cool. Now we know how that piano signal is composed. It definitely is far away from being a pure tone. It might happen, that you would need a piano, but do not have access to it. In this case, you might choose to synthesize the sound (i.e., "create it artificially"). Synthesis in the frequency domain is essentially like a Fourier transform, only that we don't start in the time domain, but in the frequency domain. This means that we provide a vector of a spectrum that we want to have, and transform this spectrum via `IFFT` into the time domain. That's what we are going to do now:

As you can see from the spectrum, the signal consists of harmonics of a fundamental frequency f_0. Synthesize the signal with a length of 2 s in the frequency domain by approximating it via a harmonic series.

$$s[t] = \sum_{k=1}^{N} a_k \sin(2\pi \cdot k \cdot f_0 \cdot t) \tag{1.2}$$

Generate the spectrum using N components (you choose a N, pick the one you think contribute most) in the frequency domain and use `ifft` to get back to the time domain. The way to do it is to generate a vector in MatLab and to assign values to the correct frequency bins. If you did it correctly, the time signal after the `ifft` should be purely real (i.e., the imaginary part is vanishingly small compared to the real part). Listen to the sound and compare it to the original recording.

In the above example, all components had the same phase. Now we consider the phase by assigning complex numbers to the frequency bins:

- Use the the complex spectrum (coding magnitude and phase) to synthesize the signal using the same components as in the previous step, but this time include amplitude and phase.

$$s[t] = \sum_{k=1}^{N} r_k \cdot \left(e^{j\varphi_k} + e^{-j\varphi_k}\right) \tag{1.3}$$

NOTE: Each Fourier coefficient is a complex number, i.e., magnitude and phase are the absolute magnitude and the angle of that complex number, respectively. Recall that there is an anti-symmetry between positive and negative frequencies.

2 Some analytical work

Show that the Fourier transform of a *rect*-function in the time domain corresponds to a *sinc* function in the frequency domain and vice versa. Provide an expression of the zero-crossings of the sinc function as a function of the width W:

$$\mathrm{rect}_W(x) = \begin{cases} \frac{1}{W} & \text{if} \quad 0 \leq x < W \\ 0 & \text{if} \quad x \geqq W \end{cases} \tag{2.1}$$