A.A. 2023-2024 - Prof. D. Dardari, Prof. G. Pasolini

Activity 1: Sine wave and noise generation, filtering

- **Generation of a sine wave**

  Open the **Editor** and write the MATLAB function `[t,x,N]=SinusoidalSource_2023(A, f0,T,fs)` that returns a vector `x` containing the samples of a sinusoidal tone (sine wave) with amplitude `A`, frequency `f0`, duration `T` seconds and sampling rate `fs`. In particular, the output vector `t` represents the sampled time axis in the interval [0, `T`] and `x` contains the samples of the sinusoid $x(t)$ taken in the time instants defined in `t`. As third output parameter, `N`, the function shall return the number of generated samples (size of the vector).

```
function [t,x,N]=SinusoidalSource_2023(A, f0,T,fs)

% A: amplitude of the sinusoid
% f0: frequency of the sinusoid
% T: duration (sec)
% fs: sampling frequency (fs>2*f0)

Ts=1/fs;  % Sampling time
t=0:Ts:T; %Sampled time axis
x=A*cos(2*pi*f0*t); % generation of the sinusoid samples
N=length(x); % Number of samples of the signal
end
```

  Leave the **Editor** and move to the **Command Window** to test the function:

```
A=1 % amplitude of the sinusoid
f0=50; % frequency of the sinusoid
T=1; % duration of the signal
fs=5000; % sampling frequency
[t,x,N]=SinusoidalSource_2023(A, f0,T,fs);
figure
plot(t,x) % plot in the time domain
grid on
xlabel('t [s]')
ylabel ('x(t) [V]')
title('Sinusoid')
figure
PlotSpectrum_2023(x,fs); % plot the power spectrum
```

- **White Gaussian noise**

  The thermal noise affecting communications is dubbed *white Gaussian noise* (WGN): the mean value is zero, all samples are uncorrelated, hence the power spectral density (PSD) is frequency flat, and the probability density function (pdf) is

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

Generation of WGN samples: open the Editor and write the following MATLAB script.

```
% White Gaussian noise generator
close all
clear all
clc

N=1000; % number of samples
sigma=2; % standard deviation

rng(2); % generates the same set of random numbers each time
x=sigma*randn(N,1);

figure
plot(x)
xlabel('n')
ylabel('x[n]')
grid

figure
histogram(x)
xlabel('x')
ylabel('number of outcomes')
title('WGN')
```

- **Filtering example using a finite response filter (FIR)**

  Open the MATLAB **Live Editor** and write the following code, also introducing images when appropriate.

```
close all % close all plots
clear all % clear all variables
clc % clear the screen

%% Generation of the sinusoid and plot
A=1; % Amplitude of the sinusoid [V]
f0=1000; % frequency of the sinusoid
fs=20000; % sampling frequency
duration=0.05; % signal duration in seconds
[t,x,N]=SinusoidalSource_2023(A, f0,duration,fs); % generation of the signal

plot(t,x,'r')
xlabel('t [s]')
ylabel ('x(t) [V]')
title('Original signal x(t)')
axis([min(t) max(t) 1.2*min(x) 1.2*max(x)])
signal_power=0.5*A^2;
fprintf('sinusoid power [V^2]=%f', signal_power)

%% *Noise generation*
```

```
sigma=0.6; % std deviation of the noise. The noise power is sigma^2 [V^2]
noise=sigma*randn(1,N); % generation of the noise
fprintf('Noise power[V^2]=%f', sigma^2)
x_noisy=x+noise; % add Gaussian noise to the sinusoid

signal_to_noise_ratio_dB=10*log10(signal_power/(sigma^2));
fprintf('SNR [dB]=%f', signal_to_noise_ratio_dB)

figure
plot(t,x_noisy,'k')
hold on
plot(t,x,'r')
legend ('x(t)+noise','x(t)')
title('Signals');

figure
PlotSpectrum_2023(x_noisy,fs);
title('Spectrum of the signal+noise before filtering');

Nf=400; % number of FIR filter taps
%% *Lowpass filter design*
Fpass=2000; % 3 dB  cut frequency
h_lowpass=fir1(Nf, Fpass/(0.5*fs)); %filter design

%% *Filter impulse response and frequency response (transfer function)*
stem(h_lowpass) % filter taps (coefficients), that is, filter impulse response
title('Lowpass filter: impulse response')
freqz(h_lowpass,1,[],fs); % plot the frequency response
title('Lowpass filter: frequency response')

%% *Passband filter design*
Fpass1=800; % low cut frequency of the filter
Fpass2=1200; % high cut frequency of the filter
h_bandpass=fir1(Nf, [Fpass1/(0.5*fs) Fpass2/(0.5*fs)],'bandpass'); % filter design

%% *Filter impulse response and frequency response (transfer function)*
stem(h_bandpass) %filter taps (coefficients), that is, filter impulse response
title('Bandpass filter: impulse response')
freqz(h_bandpass,1,[],fs); % plot the frequency response
title('Bandpass filter: frequency response')

%% *Filtering*
y=conv(x_noisy,h_bandpass,'same'); % filter the signal with the bandpass filter
%y=conv(x_noisy,h_lowpass,'same'); % filter the signal with the lowpass filter

%% *Plots*
figure
PlotSpectrum_2023(y,fs);
title('Spectrum of the signal after filtering');
```

```
figure
plot(t,x,'r')
hold on
plot(t,y,'b')
xlabel('t [s]')
legend ('x(t): original sinusoid','y(t): output of the filter')
title('Signals');

figure
plot(t,y,'b')
hold on
plot(t,x_noisy,'k')
xlabel('t [s]')
legend ('y(t): output of the filter','x(t)+noise: input of the filter')
title('Signals');
```

Utility function

```
function [pxx,f] = PlotSpectrum_2023(x,fs)
% Plots the power spectrum of signal x
% x: vector containing the samples of the signal
% fs: sampling rate

[pxx,f] = pwelch(x,[],[],[],fs,'centered');
plot(f,10*log10(pxx))
xlabel('f (Hz)');
ylabel('dB_{{V^2}/Hz}');
grid;
end
```

- Run the filtering example adopting different setups:
  - compare the outputs when adopting the lowpass filter or the bandpass filter;
  - compare the outputs for different powers of the sine wave;
  - for a given power of the sine wave, increase the standard deviation of the noise and observe the spectra of the signals at the input and at the output of the filter;
  - for the sine wave, choose a frequency that does not fall within the passband of the filter and observe the outputs.