Coloring and whitening

Learning objectives

- Generate colored Gaussian noise using the Weiner-Khinchin theorem
- Estimate Power Spectral Density using Welch’s method
- Learn how to whiten given data

Colored Gaussian Noise

- See **NOISE/colGaussNoiseDemo.m.**
- In this script, we use the Wiener-Khinchin theorem

$$S_{out}(f) = S_{in}(f)|T(f)|^2$$

with

$$S_{in}(f) = const.$$ (White noise)

and a filter with transfer function

$$T(f) = \sqrt{S_{out}(f)}$$

to generate colored noise having a PSD that approximates a target PSD

- The target PSD is

$$S_{out}(f) = \begin{cases} (f-100)(300-f), f \in [100,300] \\ 0, \quad otherwise \end{cases}$$

- The script generates 16384 samples of colored Gaussian noise with a sampling frequency of 1024 Hz

```
%% Demo for colored Gaussian noise generation
%Sampling frequency for noise realization
sampFreq = 1024; %Hz
%Number of samples to generate
nSamples = 16384;
```

Setting the sampling frequency and number of samples

```
%Target PSD given by the inline function handle
targetPSD = @(f) (f>=100 & f<=300).*(f-100).*(300-f)/10000;
freqVec = 0:0.1:512;
psdVec = targetPSD(freqVec);
```

Target PSD is a quadratic function of frequency in [100,300] Hz and zero outside this interval. Note that the highest frequency specified for generating the PSD is half the sampling frequency

```
fltrOrdr = 500;
outNoise = statgaussnoisegen(nSamples,[freqVec(:),psdVec(:)],fltrOrdr,sampFreq);
```

```
% Design FIR filter with T(f)= square root of target PSD
freqVec = psdVals(:,1);
sqrtPSD = sqrt(psdVals(:,2));
b = fir2(fltrOrdr,freqVec/(sampFreq/2),sqrtPSD);
```

Designing the filter with a transfer function: $T(f) = \sqrt{S_{out}(f)}$

```
inNoise = randn(1,nSamples);
outNoise = sqrt(sampFreq)*fftfilt(b,inNoise);
```

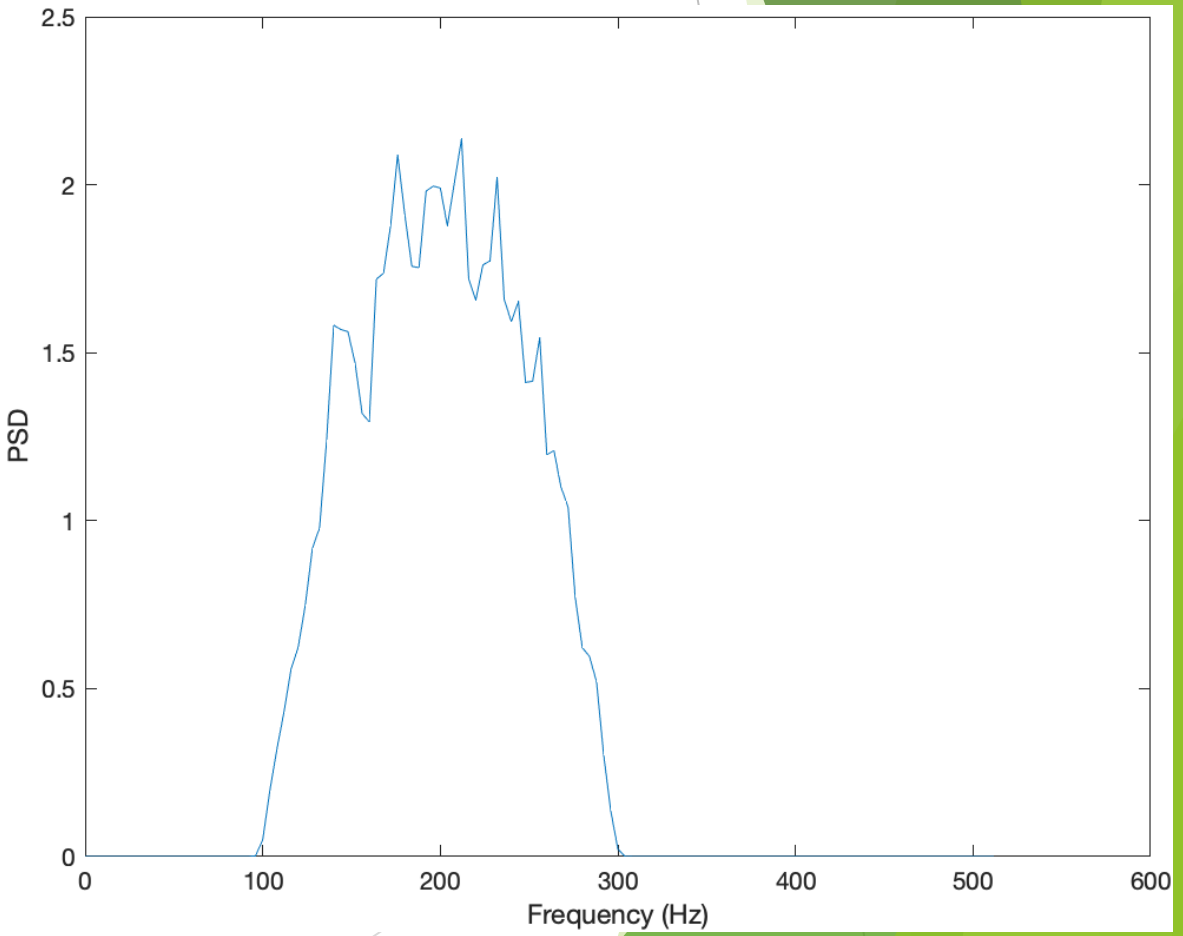
Running the filter on iid Gaussian noise to generate colored noise

Estimating PSD

- The script **NOISE/colGaussNoiseDemo.m** also plots an *estimate* of the PSD of the colored noise using the `pwelch` function in Matlab
- Note: pwelch produces a one-sided PSD while we designed the filter using a two-sided PSD
- Hence, there is a factor of 2 difference between the target and estimated PSDs
- The normalization in **NOISE/ statgaussnoisegen.m** produces the correct two-sided PSD

```
inNoise = randn(1,nSamples);
outNoise = sqrt(sampFreq)*fftfilt(b,inNoise);
```

```
% Estimate the PSD
% (Pwelch plots in dB (= 10*log10(x)); plot on a linear scale)
[pxx,f]=pwelch(outNoise, 256,[],[],sampFreq);
figure;
plot(f,pxx);
xlabel('Frequency (Hz)');
ylabel('PSD');
```



Colored Gaussian Noise

- Run the **NOISE/colGaussNoiseDemo.m** script
- Examine the target and estimated PSDs: Apart from overall normalization, the shapes should look similar
  - Note that the estimated PSD is obtained from a noise realization and, hence, has fluctuations in it
  - Increase the number of samples by factors of 2 and 4: examine the figures again
  - Why does the estimated PSD become smoother?
  - Enhance the script by putting axes labels, plot titles etc.
- Examine the noise time series by zooming in: Does it look like a WGN realization? How does it differ?
- Plot the histogram of the noise realization: Is it still a Normal PDF?

Tasks

- You have been provided a plain text file: “testData.txt” in the **NOISE** folder:
  - First column: sampling times
  - Second column: data values
- The data contains:
  - A realization of colored Gaussian noise plus …
  - A mystery signal added after $t = 5.0$ sec
- You can load the data file using “load(‘testData.txt’)”: Matlab uses the file extension ‘.txt’ to recognize that this is a plain text file.
- Use the signal-free part of the data to:
  - Estimate the noise PSD
  - Use the estimated PSD and emulate the code in **NOISE/ statgaussnoisegen.m** to design a whitening filter
- Then,
  - Whiten the data
- Plot the spectrograms of the data before and after whitening
- Plot the data time series before and after whitening
  - Is the presence of the signal clearer in the data after whitening?