Delft University of Technology
Faculty of Electrical Engineering, Mathematics, and Computer Science
Circuits and Systems Group

EE4C03 STATISTICAL DIGITAL SIGNAL PROCESSING AND MODELING

Matlab exercise, 14 September 2020, 15:30–16:30

1. Generation and analysis of random noise signals

Background [Hayes, Sec. 3.3.7]:

A white noise signal $v(n)$ is characterized by an impulse autocovariance function

$$c_v(k) = \sigma_v^2 \delta(k)$$

and a flat power spectral density (PSD)

$$P_v(e^{j\omega}) = \sigma_v^2$$

- In matlab, generate and plot 1000 samples of zero mean unit variance noise

```
v = randn(1000,1)
```

- Use the function `xcov` to calculate and plot the sample autocovariance function for lags $|k| < 100$.

  (An alternative function is `xcorr` which computes the autocorrelation function (ACF), i.e. not correcting for the mean.)

- Use FFT on the sample ACF to calculate and plot the PSD. Is this what you expect?

The matlab `help xcov` does not say how the correlation function is actually computed. As we will see later in class, there are various ways.

2. Generation and analysis of harmonic signals

Background [Hayes, Examples 3.3.1 and 3.3.3]:

A sum of sinusoids with uniformly distributed random phases

$$x(n) = \sum_{m=1}^{M} A_m \sin(n\omega_m + \phi_m)$$

has an ACF that is also a sum of sinusoids

$$r_x(k,l) = \frac{1}{2} \sum_{m=1}^{M} A_m^2 \cos[(k-l)\omega_m]$$

and a PSD consisting of a sum of impulse functions

$$P_x(e^{j\omega}) = \frac{1}{2}\sum_{m=1}^{M} A_m^2[u_0(\omega - \omega_m) + u_0(\omega + \omega - m)]$$

Background [Hayes, Sec. 3.3.5]:

The autocorrelation matrix of a wide-sense stationary random process is a Hermitian Toeplitz matrix containing the different ACF values

$$\mathbf{R}_x = \text{Toep}\{r_x(0),\, r_x(1),\, \cdots,\, r_x(p)\}$$

- Generate and plot 2048 samples of a sum of $M = 10$ sinusoids with unit amplitudes, uniformly distributed random phases, and frequencies $\omega_m = m(2\pi)/64$.
- Calculate and plot the sample ACF for lags $|k| < 128$
- Calculate and plot the PSD
- Construct the 128×128 autocorrelation matrix $\mathbf{R}_x$ and plot its eigenvalues

  You can use `toeplitz` to make the Toeplitz matrix.

---

3. Generation and analysis of nonstationary signals

Background:

Most of the theory on random processes is based on the assumption of wide-sense stationary signals:

1. mean does not depend on time: $m_x(n) = m_x$,
2. ACF does not depend on time, only on lag: $r_x(k, l) = r_x(k - l)$,
3. variance is finite: $c_x(0) < \infty$, a technical condition

However, many physical signals are nonstationary by nature. The *short-time Fourier transform (STFT)* is a mathematical tool to estimate the time-varying spectrum of nonstationary signals. The STFT is obtained by splitting a signal into shorter overlapping segments, and calculating a discrete Fourier transform (DFT) for each segment. The STFT can be visualized in a *spectrogram.*

- Read the sound file `speech_dft.wav` (from the Simulink DSP Blockset, but included as a file with this exercise) into a vector in the MATLAB workspace, and determine the sampling rate.

  Use `wavread` for this.

- Plot and play back the time-domain signal

  Use `soundsc` for this.

- Plot the spectrogram, using the following parameters:

- length of segments = 256 samples
- overlap of segments = 128 samples
- length of segment DFT = 256 samples
- visible frequency range = half sampling rate
- time on x-axis, frequency on y-axis

Use `spectrogram` for this.

How do you interprete the spectrogram? If the signal is stationary, what would you expect to see?

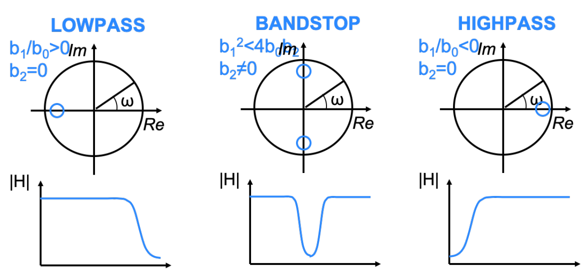
---

4. Design and analysis of elementary digital filters

Background [Hayes, Sec. 3.6]:

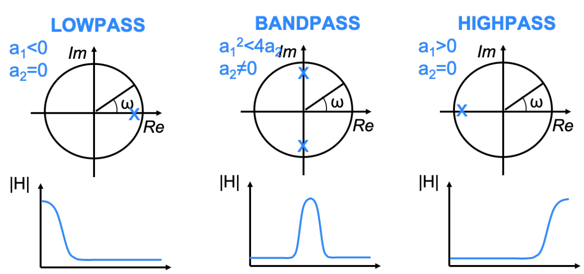
First-and second-order all-zero filters (FIR filters):

$$H(z) = b_0 + b_1 z^{-1} + b_2 z^{-2}$$



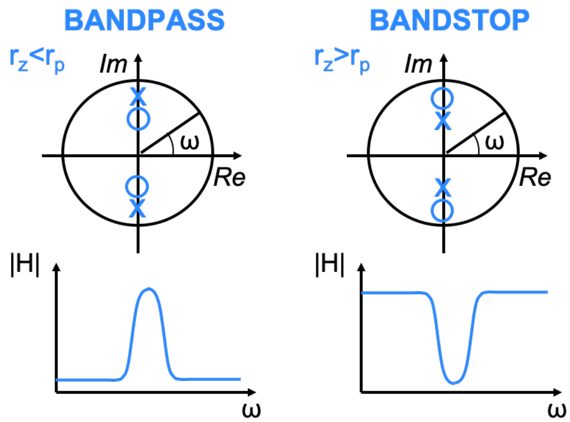
First-and second-order all-pole filters:

$$H(z) = \frac{b_0}{1 + a_1 z^{-1} + a_2 z^{-2}}$$



Biquadratic filters:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$



- Design a highpass filter with one zero at $z = 0.9$
- Plot the pole-zero diagram in the complex plane

  Use `zplane` for this.
- Plot the frequency response (magnitude & phase)

  Use `freqz` for this.

- Design a bandpass filter with central frequency $\omega_c = 1$ rad, $r_z = 0.8$, $r_p = 0.9$

  This means: the zeros are $z_k = r_z e^{\pm j\omega_c}$, and similarly for the poles.
- Plot the pole-zero diagram in the complex plane.
- Plot the frequency response (magnitude & phase).

5. Filtering and analysis of random noise signals

Background [Hayes, Sec. 3.4]:

Filtering a signal $x(n)$ using a filter with impulse response $h(n)$ yields

$$y(n) = x(x) * h(n) = \sum_{k=0}^{\infty} h(k)x(n-k)$$

In the frequency domain, this is equivalent to

$$P_y(e^{j\omega}) = P_x(e^{j\omega})\,|H(e^{j\omega})|^2$$

- Generate and plot 1000 samples of zero mean unit variance white noise
- Calculate and plot the PSD

- Design a highpass filter with one zero at $z = 0.9$.
- Plot the frequency magnitude response
- Filter the white noise signal using the highpass filter

- Calculate and plot the PSD of the resulting output signal

- Design a bandpass filter with central frequency $\omega_c = 1$ rad, $r_z = 0.8$, $r_p = 0.9$
- Plot the frequency magnitude response
- Filter the white noise signal using the bandpass filter
- Calculate and plot the PSD of the resulting output signal

We expect to see the shape of the filter amplitude spectrum back in the PSD.