Norwegian University of Science and Technology
Department of Electronics and Telecommunications

TTT4120 Digital Signal Processing
Problem Set 7

This problem set deals with stochastic processes and their statistical properties. Relevant chapters from the book are 12.1,12.2, 14.1. The maximum score for each problem is given in parentheses. Note that the total number of points for this exercise is 13.

Problem 1 (4 points)

Given three different types of white noise with *unit variance*:

- White binary noise (each sample takes the value -1 or 1 with the same probability)
- White Gaussian noise (each sample has a normal distribution)
- White uniform noise

(a) For each noise type, generate and plot one realization of length 100 samples. (Useful Matlab functions: `rand` and `randn`)

- What do these noises have in common?
- How do they differ?

(b) For each noise type

- write an expression for the probability distribution
- compute the mean value, autocorrelation function and power density spectrum

(c) We would now like to estimate the statistical properties of the three noise types based on a noise segment.

- Generate a segment of 20000 samples for each noise type.

- Compute mean value estimates and compare to the theoretical values computed in (b).
- Compute estimates of the autocorrelation function. Plot them on the interval [-10,10], and compare to the theoretical values computed in (b).

(Useful Matlab functions: `mean`, `xcorr`)

Problem 2 (3 points)

A random signal $x(n)$ is generated by filtering white Gaussian noise $w(n)$ with variance $\sigma_w^2 = \frac{3}{4}$ by a causal filter with transfer function

$$H(z) = \frac{1}{1 + \frac{1}{2}z^{-1}}.$$

(a) Calculate the following statistic properties for $x(n)$:

- mean
- autocorrelation function
- power density spectrum
- power

(b) Write the expressions for the estimators of the statistical properties in (a) based on a signal segment of length N.

(c) Generate a segment of length N=20000 samples of the signal $x(n)$.

- Use the estimators from (b) to compute the estimates based on this signal segment.
- Compare the result to the theoretical values computed in a). (You should plot both theoretical and estimated values of the autocorrelation function and power density spectrum in order to compare them. The autocorrelation function should be plotted on the interval $[-10, 10]$.)

(Useful Matlab functions: `randn`, `filter`, `mean`, `xcorr`, `fft`)

(d) Write a Matlab program that estimates the power density spectrum using Bartlett method.

- Use the program to compute the estimate of the power density spectrum based on K=10 and K=100 nonoverlaping segments of the data generated in (c).
- Compare the estimate with the theoretical value and with the periodogram estimate.

(e) Repeat (c) and (d) several times to see how the obtained estimates vary when the different signal segments are used for the estimation.

Problem 3 (3 points)

This problem deals with the statistical properties of the mean estimator $\hat{m}_x$.

(a) Use the realization of signal $x(n)$ generated in Problem 2c), and compute 200 mean estimates based on nonoverlapping segments of length $K = 20$.

(b) Use the Matlab function `hist` to plot the histogram of the mean estimates. Use 20 histogram bins.

(c) Use the Matlab functions `mean` and `var` to estimate the mean and variance of $\hat{m}_x$.

(d) Repeate the procedure for $K = 40$, and $K = 100$.

(e) Compare the results obtained with different values of K. Do they agree with the theoretical results on the statistical properties of the mean estimator $\hat{m}_x$? Explain!

Appendix

Required Functions:

```
#1-randn()
import numpy as np
np.random.randn(N)

#2-sqrt()
import numpy as np
np.sqrt(x)

#3-power(x,y)
import numpy as np
np.power(x,y) #x^y

#4-normal(mean, std, size = num_samples) #generating white noise
#Example:
import numpy
mean = 0
std = 1
num_samples = 1000
WN = numpy.random.normal(mean, std, size=num_samples

#5-plt.hist(samples, num_bins)
import matplotlib.pyplot as plt
plt.hist(samples, num_bins)
plt.show()
#or
```

```
import seaborn as sns
sns.distplot(samples, bins = num_samples,
↪  hist_kws={'edgecolor':'black'})

#6-acf(x) #autocorrelations
import statsmodels.api as sm
#calculate autocorrelations
sm.tsa.acf(x)
#We can also specify the number of lags to use with the nlags
↪  argument:
sm.tsa.acf(x, nlags=5)


#7-fft() %Fast Fourier Transform (FFT)
from scipy.fft import fft
y = fft(x)

#8-spectrogram(signal)and welch(signal)
from scipy import signal
from matplotlib import pyplot as plt

# plot the spectrogram, compute and plot the power spectral
↪  density (PSD):
# The spectrum of the signal on consecutive time windows
freqs, times, spectrogram = signal.spectrogram(sig)
plt.figure()
plt.imshow(spectrogram, aspect='auto', cmap='hot_r',
↪  origin='lower')
plt.title('Spectrogram')
plt.ylabel('Frequency band')
plt.xlabel('Time window')
plt.tight_layout()

# Compute and plot the power spectral density (PSD)
# The power of the signal per frequency band
freqs, psd = signal.welch(sig, noverlap = None)
#noverlapint, optional. Number of points to overlap between
↪  segments. If None, noverlap = nperseg

plt.figure()
plt.semilogx(freqs, psd)
plt.title('PSD: power spectral density')
plt.xlabel('Frequency')
plt.ylabel('Power')
plt.tight_layout()
plt.show

# Compute and plot the Bartlett power spectral density (PSD)
#If noverlap is 0, this method is equivalent to Bartlett's method
freqs, psd = signal.welch(sig,  noverlap = 0)
plt.figure()
plt.semilogx(freqs, psd)
```

```
73 plt.title('PSD: power spectral density')
74 plt.xlabel('Frequency')
75 plt.ylabel('Power')
76 plt.tight_layout()
77 plt.show()
```