COMM.SYS.300 COMMUNICATION THEORY

Matlab Exercise #2

Random variables and random processes

1 RANDOM VARIABLES

1.1 ROLLING A FAIR 6-FACED DICE (DISCRETE VALIABLE)

- Generate random samples for rolling a fair 6-faced dice for N times (i.e. there's an equal probability for each value of the dice).
    - First, roll the dice for N=10 times (help randi)

```
N_faces = 6; %Number of faces in the dice
N_trials = 10; %Number of trials (how many times the dice is rolled)
trials = randi(N_faces,1,N_trials); % Getting random integers 1...6
```

    - Plot the histogram of the outcome values (help histogram, help hist, help bar, help histcounts)
        - histogram-function was introduced in MATLAB R2014b

```
x_histogram_centers = 1:6; %x-axis for the bin center coordinates

% Before version R2014b ------------------------------------------

hist_count = hist(trials,x_histogram_centers);


figure
bar(x_histogram_centers,hist_count)
xlabel('Rolled number')
ylabel('occurrence number')
grid on
% Histograms can be plotted directly as "hist(trials,x_histogram_centers)",
% but the plotting parameters are more difficult to handle.

% Version R2014b or after-----------------------------------------

% x-axis for the bin edge coordinates (last edge is for the right edge)
histogram_edges = 0.5:1:6.5;
% Below the ~ sign means that we discard that specific output argument
[hist_count, ~]= histcounts(trials,histogram_edges);

figure
bar(x_histogram_centers,hist_count)
xlabel('Rolled number')
ylabel('occurrence number')
```

```
grid on % Set a grid in the plotted figure
title('Dice rolling histogram')

% Histograms can plotted directly as "h = histogram(trials,'BinLimits',[0.5
% 6.5],'BinMethod','integers')", where h is the histogram object.
```

- Normalize the histogram values with N_trials (number of trials) so that we get the experimental probability density function (pdf) for the dice. Hence, after the normalization the sum of the histogram bin area ("integral") is equal to one, as required with pdfs (actually when talking about discrete pdf, we often refer to pmf (probability mass function)).

```
pdf_experimental = hist_count/N_trials;
```

  - Plot the normalized histogram, and define the true (analytic) pdf for the fair dice and plot it on top of the experimental pdf (help hold)

```
figure
bar(x_histogram_centers,pdf_experimental)
xlabel('Rolled number')
ylabel('pdf')
grid on
title('Dice rolling normalized histogram')

one_face_probability = 1/N_faces; % probability of one face of the dice
hold on % avoids removing the previous plot
% plotting true pdf (a line between two points)
plot([0.5 6.5],[one_face_probability one_face_probability],'r')
hold off
% Using legend we can name different data curves in the plot (in order of
% appearance)
legend('Experimental pdf','True pdf')
```

  - Are the true pdf and the experimental pdf perfectly equal?
- Repeat the process for a few of times and compare the outcomes:
  - Is the experimental pdf varying between simulations?
- Increase the number of trials to N=100000 and repeat the process for a few times
  - Is the experimental pdf varying between simulations now?

1.2 NORMAL/GAUSSIAN DISTRIBUTED RANDOM VARIABLE

- Generate 1000 samples for a normally distributed random variable $X \sim (\mu, \sigma^2)$, where the mean and variance are given as $\mu=3$, and $\sigma^2=4$ (help randn)

```
N_samples = 1000;
mu = 3;
sigma = sqrt(4); % the variance is 4 (i.e. sigma^2=4)
```

  - Calculate the following statistics of the observed samples
    - Mean (or average, or expected value) (help mean)
    - Standard deviation (help std)
    - Variance (help var)
- Plot the experimental pdf (i.e. the normalized histogram so that the sum of histogram bin area is equal to one)
  - Define the histogram bin center x-coordinates as bin_centers = -7:bin_width:13, where the bin_width = 0.5 defines the width of one bin

```
bin_width = 0.5; %bin width (in the x-axis)
```

```
bin_centers = -7:bin_width:13; %x-axis for the bin center coordinates

% x-axis for the bin edge coordinates (last edge is for the right edge):
% (Three dots "…" continues the command in the following line)
bin_edges = (bin_centers(1)-…
bin_width/2):bin_width:(bin_centers(end)+bin_width/2);

% ~ means that we discard that output argument
[hist_count, ~]= histcounts(X,bin_edges);

pdf_experimental = hist_count/sum(hist_count*bin_width);

figure
bar(bin_centers,pdf_experimental,1)
% and so on with the titles, xlabels,…
```

- Plot the true (analytic) pdf of $X \sim (\mu, \sigma^2)$ on top of the experimental pdf

```
pdf_true = 1/(sqrt(2*pi)*sigma)*exp(-(mu-bin_edges).^2/(2*sigma^2));
hold on
plot(bin_edges,pdf_true,'r','LineWidth',3) % defines a specific line width
% … and so on with the titles, xlabels,… (remember the legend also)
```

- Repeat the experiment with different number of samples: N=100 and N=100000
    - See the difference in the fitting between the experimental pdf and the analytic pdf as the number of samples changes
- Based on the histogram with N=100000 samples, define the probability $\mathrm{P}(X>5.25)$
    - Integrate (i.e. sum) histogram bins, whose x-axis center values are larger than 5.25
        - Note that actually 5.25 is exactly on edge between two histogram bins

```
b = 5.25;
indices_with_bin_center_larger_than_b = bin_centers > b;
considered_bin_values = …
pdf_experimental(indices_with_bin_center_larger_than_b);

%area of the considered bins
probability_X_larger_than_b = sum(considered_bin_values*bin_width)
```

    - Check your answer with the Q-function $\mathrm{P}(X>\mathrm{b}) = Q((b-\mu)/\sigma)$ (help qfunc)

```
analytic_probability = qfunc((b-mu)/sigma)
```

2 RANDOM PROCESSES

2.1 WHITE NOISE VS. COLORED NOISE

- Generate a zero mean white Gaussian noise signal with variance $\sigma^2=3$

```
N = 10000; % Number of generated samples
noise_var = 3; % Desired noise variance
noise = sqrt(noise_var)*randn(1,N); % Noise signal generation
```

    - Plot the noise signal

```
figure
plot(noise)
xlabel('sample index')
ylabel('noise amplitude')
title('White noise')
xlim([0 100]) %define the x-axis limits
```

- Plot the histogram of the noise signal to see that it is Gaussian distributed

```
figure
histogram(noise,40)
xlabel('noise amplitude')
ylabel('histogram count')
title('White noise histogram')
```

- Implement a low pass FIR filter (help firpm, recap from the previous exercise)
  - Use filter order of 60 and a transition band from 0.1*Fs/2 to 0.2*Fs/2

```
N_filter = 60; %even number
h = firpm(N_filter,[0 0.1 0.2 1],[1 1 0 0]);
```

  - Plot the impulse response and amplitude (frequency) response of the filter
    - You can define the amplitude response as a function of normalized frequency (i.e. –Fs/2…Fs/2 → -1…1)

```
N_freq =  length(noise);
freq_vec_filter = -1:2/N_freq:(1-2/N_freq);
%frequency vector values normalized between -1 and 1

figure,plot(freq_vec_filter,10*log10(fftshift(abs(fft(h,N_freq)))))
xlabel('Normalized frequency (F_s/2=1)')
ylabel('Amplitude')
title('Amplitude response of the filter')
```

- Filter the noise signal using the above filter (help filter)

```
% Filter the noise signal:
filtered_noise = filter(h,1,noise);
```

- Plot the white noise and filtered noise signals in the same figure and compare
  - Remember the filter delay in order to align the signals properly in time

```
filtered_noise = filtered_noise(N_filter/2+1:end); %remove the delay

figure
plot(noise(1:N_samples_plot))
hold on
plot(filtered_noise(1:N_samples_plot),'r')
legend('White noise','Colored noise')
xlabel('sample index')
ylabel('noise amplitude')
title('White noise and filtered (colored) noise')
```

- Plot the histogram of the filtered noise signal
  - How it is distributed? (recap: What is the distribution of the output signal of an LTI system, if the input signal is Gaussian distributed?)
- Compute and plot the autocorrelation function of the original white noise signal (help xcorr)

```
[corr_fun, lags] = xcorr(noise);
% we normalize the max-value to 1 and use stem-function in order to emphasize
% the impulse-like nature of the outcome
figure,stem(lags,corr_fun/max(corr_fun))
xlabel('\tau')
% \tau gives the Greek tau-letter (\ works generally for the Greek alphabet)
ylabel('R(\tau)')
title('Autocorrelation of white noise')
xlim([-30 30])
```

  - What type of autocorrelation function should the white noise have?

- Compute and plot the autocorrelation function of the filtered (i.e., colored) noise signal
    - Use the above statements, but with “plot” instead of “stem”:

```
[corr_fun, lags] = xcorr(noise);
figure,plot(lags,corr_fun/max(corr_fun))
%...and so on
```

    - Compare this with the previously plotted impulse response of the filter
- In the end, plot the power spectra (i.e., amplitude spectrum squared) of the two noise signals in the same figure (use the decibel scale 20*log10(abs(.)))

```
noise_abs_spec = 20*log10(abs(fft(noise(1:length(filtered_noise)))));
filtered_noise_abs_spec = 20*log10(abs(fft(filtered_noise)));

%Define the frequency vector values (normalized between -1 and 1):
freq_vec = -1:2/length(noise_abs_spec):1-2/length(noise_abs_spec);

figure
plot(freq_vec,fftshift(noise_abs_spec))
hold on
plot(freq_vec,fftshift(filtered_noise_abs_spec),'r')
hold off
xlabel('Normalized frequency (F_s/2=1)')
ylabel('power [dB]')
title('Noise spectra')
legend('White noise','Filtered (coloured) noise')
```

    - Notice that both signals are still Gaussian distributed (see the previous histograms), but the power spectra are different.
        - Try to remember what is the connection between the correlation function and the power spectral density function?
    - Make sure you understand the difference between the following concepts: Gaussian noise and white noise
        - I.e., here both noise signals are Gaussian, but only the other one is white…

2.2 RANDOM WALK MODEL (EXAMPLE FROM THE CLASSROOM EXERCISES)

Let’s consider a random sequence $X[n]$, the so called *random walk* model, as

$$X[n] = \sum_{i=1}^{n} W[i]$$

where $W[i]$ are binary i.i.d. (independent and identically distributed) random variables with probabilities $\mathrm{P}[W[i] = s] = p$ and $\mathrm{P}[W[i] = -s] = 1 - p$

- Generate a random process of 2000 samples and 5000 realizations (=ensemble size)
    - Use first p=0.5 and s=1, but write the code so that you can conveniently test the process with other values too

```
N_samples = 2000; %Number of samples for each realization
N_ensemble = 5000; %Number of signal realizations (i.e., the size of ensemble)

%Step probability and step size:
p = 0.5; % P(Wi=s) = p, P(Wi=-s) = 1-p
s = 1; %step length

n = 1:N_samples; % vector of samples indices

% Generating matrix of randomly generated steps:
```

```
W = rand(N_ensemble,N_samples);
% (i.e. uniformly distributed random values between 0 and 1)

indices_with_positive_s = W<p; % find out steps going "up"

W(indices_with_positive_s) = s; % Define steps for going "up"
W(~indices_with_positive_s) = -s; % Define steps for going "down"


% The overall "random walk" is achieved by taking the cumulative sum over the
% steps:
X = cumsum(W,2);
% (Notice that now each row describes one random walk realization, so the sum
% is taken over the 2nd dimension)
```

- Plot five example realizations of the random walk process (help for)
    - Use subplots inside one figure (help subplot)

```
figure
for ind = 1:5
    subplot(5,1,ind)
    plot(n,X(ind,:))
    xlabel('n')
    ylabel('X(n)')
    grid on
    title(['Realization #' num2str(ind)])
    %num2str converts a numerical value into a character value
end

% Here is a handy way to get a full screen figure
% (otherwise the figure might be to unclear):
set(gcf,'units','normalized','outerposition',[0 0 1 1])
```

- Calculate the ensemble mean and the ensemble variance of the process
    - Plot the calculated ensemble mean $\mathrm{E}[X[n]]$ and variance $\mathrm{E}[(X[n]- \mathrm{E}[X[n]])^2]$ in the same figure with the theoretical values given as
        - Theoretical mean: $\mathrm{E}[X[n]] = ns(2p\text{-}1)$
        - Theoretical variance: $\mathrm{E}[(X[n]- \mathrm{E}[X[n]])^2] = np(2s)^2(1\text{-}p)$
    - Notice that the ensemble mean and variance are now functions of the sequence index n (“=time”), so there will be a specific mean and variance for each sequence index

```
mean_theory = n*s*(2*p-1); % Theoretical mean
var_theory = n*(2*s)^2*p*(1-p); % Theoretical variance

mean_observed = mean(X); % Empirical mean (i.e., what we observe)
var_observed = var(X); % Empirical variance (i.e., what we observe)

figure
plot(n,mean_observed,'b','LineWidth',3)
hold on
plot(n,mean_theory,'r:','LineWidth',2)
hold off
legend('observed mean','theoretical mean')
ylim([-2 2]) % set the axis limits in y-direction only
xlabel('n')
ylabel('Mean')
title('Mean over the sample index')
%
% and the same for the variance…
```

- Re-run the process with different values of p, s, and try different number of realizations
    - Due to memory issues, do not try too many realizations at the same time (stay below 100000 realizations and 2000 samples)
    - What if the number of realizations is very low?
- Is the process stationary (in wide sense)? Why?/Why not?