Scores	48-50	45-47	42-44	39-41	36-38	33-35	30-32	27-29	24-26	21-23	0-20
Grade	1.0	1.3	1.7	2.0	2.3	2.7	3.0	3.3	3.7	4.0	5.0

21 July 2022

Final Examination in Stochastic Signals and Systems

Do NOT use red pens! Do NOT use pencils! Solutions written with pencil are VOID. Duration 120 minutes.
Allowed means: 1 page (size A4, single-sided), handwritten only. Participants at the university: Fill your solutions in the answer form only. Online participants: Write on a blank sheet.

Problem #1: Moments and correlation functions (20 scores)

Write MATLAB code that solves the following tasks. Use the given variable names.

a) Create a sine signal named "sin1" within the timeframe 0 seconds until 5 seconds. The frequency of the sine signal shall be 30 Hz, the amplitude is 4. The sampling frequency is 500 Hz (i.e. 500 samples per second). (2 scores)
b) Add pseudo-random noise to "sine1" thus creating a random signal. Call the new signal "noisysine1". (2 scores)
c) Calculate the autocorrelation function of "noisysine1". (1 score)
d) Create a ramp function named "ramp" within the timeframe of -2 seconds to +3 seconds. The function values shall continuously grow from 0 at left end up to 5 at the right end of the ramp. Sampling frequency 500Hz. (2 scores)
e) Calculate the cross-correlation of "noisysine1" and "ramp". Pay attention on the timeframe. (1 score)
f) Calculate the sum of "noisysine1" and "ramp". Call the result "noisyrampsine". Pay attention on the timeframe. (2 scores)
g) Calculate the power spectral density of "noisyrampsine" by implementing the Wiener-Khinchine theorem. Use a Hamming window. Do not make use of the psd()-function. (5 scores)
h) A discrete random variable named "ranvar" takes the values 1, 2, 3 and 4 with the probabilities 0.2, 0.3, 0.1 and 0.4 respectively. Calculate the first order moment, the second order moment and the variance. (5 scores)

Answer form for problem #1

a)

```
% Define parameters
duration = 5;      % Timeframe in seconds
Fs = 500;          % Sampling frequency in Hz
f1 = 30;           % Frequency of the sine signal in Hz
A = 4;             % Amplitude of the sine signal
% Time axis
t = 0:(1/Fs):duration-(1/Fs); % Generate time vector from 0 to 5 seconds with Fs sampling rate
% Create the sine signal
sin1 = A * sin(2*pi*f1*t);
```

b)

```
noise_amplitude = 0.5;
% Generate pseudo-random noise
noise = noise_amplitude * randn(size(t)); % randn generates normally distributed random numbers
% Add noise to the sine signal to create the random signal random_signal = sin1 + noise;
```

c)

d)

e)

f)	
g)	
h)	

```
%% Task a
% Define parameters
duration = 5;      % Timeframe in seconds
Fs = 500;          % Sampling frequency in Hz
f1 = 30;           % Frequency of the sine signal in Hz
A = 4;             % Amplitude of the sine signal
% Time axis
t = 0:(1/Fs):duration-(1/Fs); % Generate time vector from 0 to 5 seconds with Fs sampling rate
% Create the sine signal
sin1 = A * sin(2*pi*f1*t);
%% Task b
% Generate pseudo-random noise
noise_amplitude = 0.5; % Amplitude of the pseudo-random noise
noise = noise_amplitude * randn(size(t)); % randn generates normally distributed random numbers
% Add noise to the sine signal to create the random signal
noisysine1 = sin1 + noise;
%% Task c
% Calculate the autocorrelation function
[r1, lags1] = xcorr(noisysine1, 'biased');
%% Task d
% Create the ramp function
ramp_t = (-2):(1/Fs):(3-1/Fs); % Generate time vector from -2 to 3 seconds with Fs sampling rate
ramp = 5 * ramp_t; % Linearly increasing values from 0 to 5
%% Task e
% Calculate the cross-correlation between noisysine1 and ramp
[c1, lags2] = xcorr(noisysine1, ramp, 'biased');
%% Task f
% Calculate the sum of noisysine1 and ramp
noisyrampsine = noisysine1;
start_index = find(t >= -2, 1); % Find the index corresponding to t = -2
end_index = find(t <= 3, 1, 'last'); % Find the index corresponding to t = 3
noisyrampsine(start_index:end_index) = noisysine1(start_index:end_index) + ramp;
%% Task g
% Calculate the power spectral density using the Wiener-Khinchine theorem with Hamming window
hammingWindow = hamming(length(noisyrampsine));
hammedSignal = noisyrampsine .* hammingWindow;
% Calculate the autocorrelation of the hammedSignal
[r_ham, lags_ham] = xcorr(hammedSignal, 'biased');
tau_ham = lags_ham / Fs;
% Calculate the power spectral density
Rxxdft_ham = abs(fftshift(fft(r_ham))) / length(r_ham);
freq_ham = -Fs/2 : Fs/length(r_ham) : Fs/2 - (Fs/length(r_ham));
%% Task h
% Define the random variable and probabilities
ranvar_values = [1, 2, 3, 4];
probabilities = [0.2, 0.3, 0.1, 0.4];
% Calculate the first order moment (mean)
mean_value = sum(ranvar_values .* probabilities);
% Calculate the second order moment (variance)
variance = sum((ranvar_values - mean_value).^2 .* probabilities);
% Calculate the variance
second_order_moment = variance + mean_value^2;
```

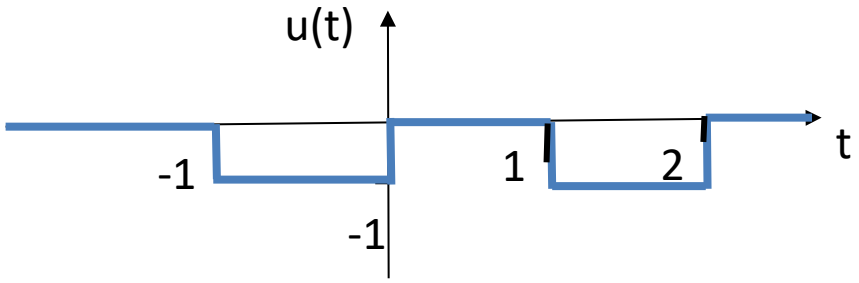
Problem #3: Random processes and correlation functions (15 scores)
Let

$$s_{xx}(\tau) = d^2 \cdot e^{-8\pi \cdot |\tau| \cdot \cos(\omega\tau)}$$

the autocorrelation function (ACF) of the stationary random process x(ζ,t).

Moreover let a stationary Gaussian random process $y(\zeta, t)$.

a) Calculate the first moment of the random process x(ζ,t). (2 scores)
b) Calculate the standard deviation of the random process x(ζ,t). (2 scores)
c) Write down the PDF of the random process $y(\zeta, t)$. (3 scores)
d) Assume that the random process $y(\zeta, t)$ is input to a linear time invariant system which is characterized by the transfer function $H(j\omega)$. Give a statement that describes the output process as precise as possible. (3 scores)
e) If your task would be to find the maximum value of the ACF of the stationary random process $y(\zeta, t)$, which value of τ would you choose for the calculation? (2 scores)
f) Sketch the ACF of this function (3 scores):



Answer form for problem #3
Fill in the final result only

a)	
b)	
c)	
d)	
e)	
f)	

Problem #2: Autocorrelation (5 scores)

Write MATLAB code that solves the following task. Only MATLAB code, no sketches or other calculations. Use the given variable names.

Create a time signal named “sumOFsines” as the sum of three sine signals. All sine signals have the same frequency of f=50 Hz and the same amplitude a=2. The phase angles of the sine signals are 0, $\pi/4$ and $\pi/3$ respectively. The sampling frequency is 1000 Hz. Calculate the centered FFT of “sumOFsines” and plot it in the frequency range from 0 Hz until 250 Hz.

Answer form for problem #2

```
% Given parameters
f = 50;         % Frequency of the sine signals in Hz
a = 2;          % Amplitude of the sine signals
Fs = 1000;      % Sampling frequency in Hz
% Time axis
duration = 1;   % 1 second duration
t = (0:(1/Fs):(duration-(1/Fs))); % Generate time vector
% Create the three sine signals
sine1 = a * sin(2*pi*f*t);                  % Phase angle = 0
sine2 = a * sin(2*pi*f*t + pi/4);           % Phase angle = pi/4
sine3 = a * sin(2*pi*f*t + pi/3);           % Phase angle = pi/3
% Sum the three sine signals to create "sumOFsines"
sumOFsines = sine1 + sine2 + sine3;
% Calculate centered FFT of "sumOFsines"
fft_sumOFsines = fftshift(fft(sumOFsines));
% Frequency axis
N = length(sumOFsines);
freq = (-Fs/2):(Fs/N):(Fs/2 - Fs/N);
% Plot the centered FFT in the frequency range from 0 Hz to 250 Hz
figure;
plot(freq, abs(fft_sumOFsines));
title('Centered FFT of "sumOFsines"');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
xlim([0, 250]);
grid on;
```

Problem #4: Viterbi Algorithm (5 scores)

Given is a state table of a transversal filter of order M=2 with x(e,k) at the input and w(e,k) at the output Given is a state table of a transversal filter of order M=2 with x(e,k) at the input and w(e,k) at the output. Draw the corresponding state diagram.

$x(e,k)$	S_{1k}	S_{2k}	$S_{1(k+1)}$	$S_{2(k+1)}$	$w(e,k)$
0	0	0	0	0	0
0	1	0	0	1	1
0	0	1	0	0	0.3
0	1	1	0	1	1.3
1	0	0	1	0	0.6
1	1	0	1	1	1.6
1	0	1	1	0	0.9
1	1	1	1	1	1.9

Answer form for problem #4

Problem #5: Optimum Systems (5 scores)

a) Assume a perfectly working Matched Filter. Which signal shape would you observe at the output of that filter if the signal shape at the input signal would be a rectangular impulse with a length of 2ms? Which would be the length of the output signal? (2 scores)

b) Give reasons why EKF needs Jacobian matrices. (3 scores).

Answer form for problem #5

Be brief!

a)	If the input signal is a rectangular impulse with a length of 2 ms, the matched filter's output signal shape would be a matched version of the input signal. The matched filter output would be a sinc function, which is the ideal response for detecting the rectangular impulse in the presence of noise. The length of the output signal would depend on the length of the matched filter's impulse response. The impulse response of the matched filter is typically longer than the input signal, and its length determines the duration of the output signal. If the matched filter has a duration of, for example, 5 ms, the output signal would have a length of 5 ms. The output signal duration is generally determined by the length of the matched filter's impulse response.
b)	Linearization of Nonlinear Models: To apply the Kalman Filter framework to nonlinear systems, the EKF linearizes the nonlinear system and measurement models around the current estimated state. The Jacobian matrices are used to perform this linearization process. They capture the partial derivatives of the nonlinear equations with respect to the state variables, allowing the system to be approximated as a linear system in the local vicinity of the current estimate. Propagation of Covariance: In the prediction step of the EKF, the covariance matrix of the state estimate is propagated through the nonlinear system model. To linearize the covariance propagation process, the Jacobian matrix of the system dynamics is used. This enables the EKF to approximate the error covariance propagation while still considering the nonlinearities in the system. Measurement Update: In the update step of the EKF, the nonlinear measurement model is used to update the state estimate based on the actual measurements. The Jacobian matrix of the measurement model is required to linearize this nonlinear function and update the state estimate while incorporating the measurement uncertainties. Improving Efficiency: While the EKF is an approximation to handle nonlinear systems, it is computationally more efficient than other nonlinear filtering methods like the Unscented Kalman Filter or the Particle Filter. The use of Jacobian matrices allows the EKF to leverage the well-established Kalman Filter algorithm while still dealing with nonlinearities to some extent. Handling Non-Gaussian Noise: The EKF assumes that the process and measurement noise are Gaussian. While this assumption might not hold for all nonlinear systems, the EKF can provide reasonable estimates even in the presence of non-Gaussian noise. The Jacobian matrices help to propagate the Gaussianity assumptions through the nonlinear transformations. In summary, Jacobian matrices are fundamental to the Extended Kalman Filter as they enable the linearization of nonlinear system and measurement models, allowing the EKF to estimate state variables in systems with nonlinear dynamics more efficiently than other nonlinear filtering methods. However, it is essential to be cautious with the linearization approximation, as it may lead to suboptimal performance or even divergence if the nonlinearities are severe or the initial state estimate is far from the true state.