Matlab assignment 2

Name: Ainhoa Arnaiz **Student ID:** H00301693

Question 1

When we numerically calculate the Fourier coefficients for a periodic signal, we have to take into account that:

$$x(t) = \underbrace{a_0}_{dc} + \underbrace{\sum_{n=1}^{\infty}(a_n \cos n\omega_0 t + b_n \sin n\omega_0 t)}_{ac \ \text{components}}$$

$$a_0 = \frac{1}{T}\int_0^T x(t)dt \quad a_n = \frac{2}{T}\int_0^T x(t)\cos(n\omega_0 t)dt \quad b_n = \frac{2}{T}\int_0^T x(t)\sin(n\omega_0 t)dt$$

Thus, if we use Matlab numerical integration routines, this is the result we get for the signal example in Fourier_series_real.m:

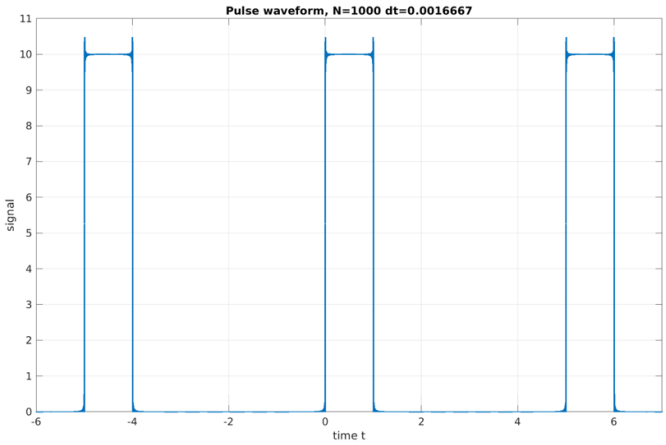

Figure 1 Pulse waveform

```
%Trigonometric Fourier series W7 >
clear all; close all; clc;

T = 5; % period
w0 = 2*pi/T;

F = @(s) [10.*(0<=s & s<1) + 0.*(1<=s & s<=5)];

N = 1000; % 2*N+1 is number of terms in the sum
B = N/T; % The highest frequency is B=N/T
dt = 1/(3*B); % the Nyquist rate is 1/(2*B)

a0 = 1/T*integral(F,0,T);
An = zeros(N); phase = zeros(N);

t = -6:dt:7;

sum = a0*ones(size(t));
for n=1:N
   an = 2/T*integral(@(t) F(t).*cos(n*w0*t), 0, T);
   bn = 2/T*integral(@(t) F(t).*sin(n*w0*t), 0, T);
   sum = sum + an*cos(n*w0*t) + bn*sin(n*w0*t);
end

figure,
plot(t,sum,'linewidth',1.5)
axis([-6 7 0 11])
grid
xlabel('time t')
ylabel('signal')
title(['Pulse waveform, N=',num2str(N),' dt=',num2str(dt)])
```

Question 2

For demodulation, we need to generate a local carrier at the receiver in frequency and phase coherence (synchronism) with the carrier used at the modulator. Both phase and frequency synchronism are extremely critical. Therefore, when there is no phase or frequency error the demodulated signal is almost equal to the messaged one.

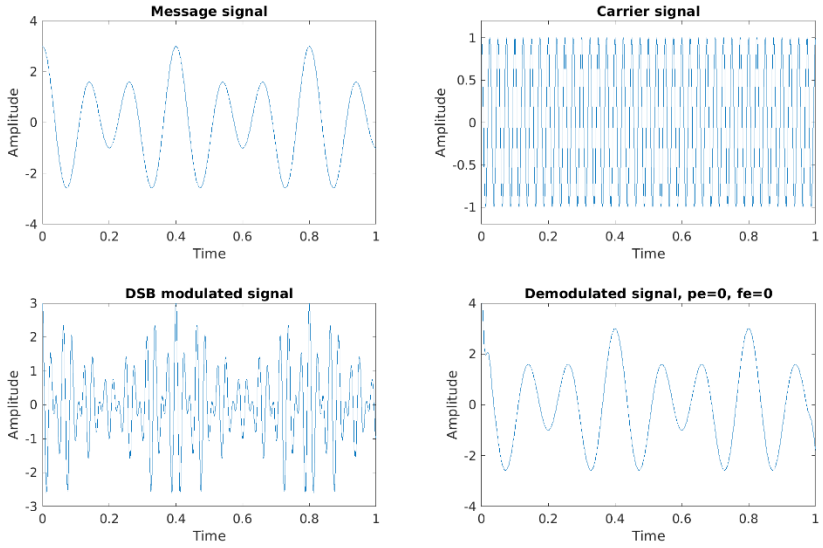

Figure 2 Demodulated signal without errors

In this ideal case, after applying LPF and removing the parts outside the spectrum, we get

$$d(t) = \frac{1}{2} m(t)$$

However, if the receiver doesn't know the carrier signal exactly, the demodulated signal varies from the messaged one. If, for instance, we introduce a phase error (pe = 1.2, for example) using the same demodulation scheme, the power of the demodulated signal is going to decrease by cos(pe).

$$d(t) = \frac{1}{2} m(t) \cos\varphi$$

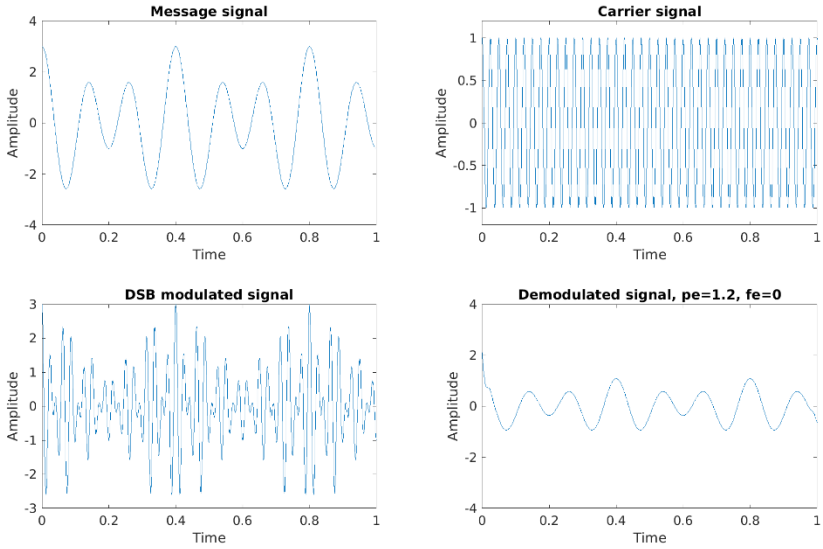

Figure 3 Demodulated signal with phase error

In addition, if the transmitter and the receiver carrier frequencies differ just by 1 Hz, the output will be the modulated signal m(t) multiplied by a time-varying signal whose gain goes from the maximum

to 0 every half-second (the beat-effect). Even less than a 1 Hz error causes a distortion that is very difficult to repair (see Figure 4).

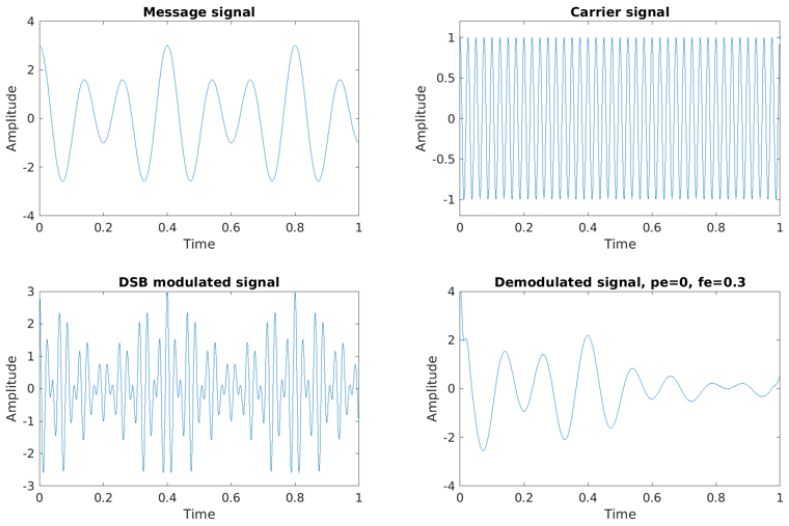

Figure 4 Demodulated signal with freq. error

Question 3

The results of my Matlab-simulated QAM demodulation experiments:

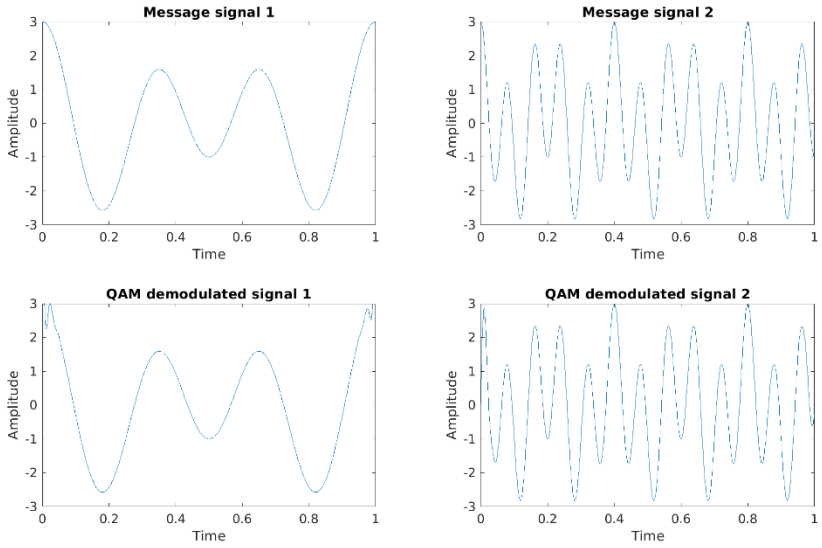

Figure 5 QAM demodulated signals

```
% Quadrature Amplitude Modulation and Demodulation W8 > 51-54
clc; clear all; close all;

fc = 40; % Carrier frequency in Hz
fm1 = 2; % Modulating frequency in Hz
fm2 = 5 % Modulating frequency in Hz
Fs = 1000; % Sampling frequency in Hz

t=0:1/Fs:1;
m1=cos(2*pi*fm1*t)+2*cos(3*pi*fm1*t); % Message signal 1
m2=cos(2*pi*fm2*t)+2*cos(5*pi*fm2*t); % Message signal 2

c1=cos(2*pi*fc*t); % In-phase carrier signal
c2=sin(2*pi*fc*t); % Quadrature-phase carrier signal

% Modulation
x1=m1.*c1; % Modulated signal 1
x2=m2.*c2; % Modulated signal 2
x=x1+x2;

```

```
% Demodulation
y1 = x.*c1;
y2 = x.*c2;
[b,a]=butter(5,2*fc/Fs);
y1 =filtfilt(b,a,y1)*2;
y2 =filtfilt(b,a,y2)*2;

% plots
figure(1),
subplot(221); plot (t,m1)
ylabel('Amplitude'); xlabel('Time');
title('Message signal 1');

subplot(222); plot (t,m2)
ylabel('Amplitude'); xlabel('Time');
title('Message signal 2');

subplot(223); plot (t,y1)
axis([0 1 -3 3]);
ylabel('Amplitude'); xlabel('Time');
title('QAM demodulated signal 1');

subplot(224); plot (t,y2)
ylabel('Amplitude'); xlabel('Time');
title('QAM demodulated signal 2');
```

Question 4

When we frequency modulate a signal, the modulator combines the carrier with the message signal to get the transmitted signal:

$$\begin{aligned} y(t) &= A_c \cos\left(2\pi \int_0^t f(\tau)d\tau\right) \\ &= A_c \cos\left(2\pi \int_0^t \left[f_c + f_\Delta x_m(\tau)\right] d\tau\right) \\ &= A_c \cos\left(2\pi f_c t + 2\pi f_\Delta \int_0^t x_m(\tau)d\tau\right) \end{aligned}$$

However, in a sinusoidal baseband signal:

$$\int_0^t x_m(\tau)d\tau = \frac{\sin(2\pi f_m t)}{2\pi f_m}$$

Therefore,

$$y(t) = A_c \cos\left(2\pi f_c t + \frac{f_\Delta}{f_m}\sin(2\pi f_m t)\right)$$

Knowing this, the result in Figure 6 is the output we get if we apply FM in angle_modulation.m

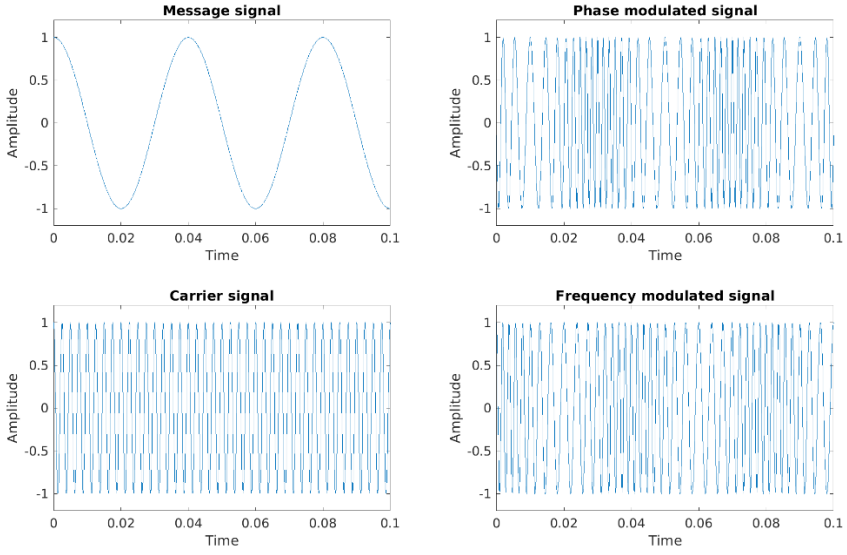

Figure 6 FM signal

```
% Frequency and phase modulation W8 > 55-58
clc; clear all; close all;

fc = 400; % carrier frequency in Hz
fm = 25; % modulating frequency in Hz
kf = 150;
kp = 8;

t=0:0.0001:0.1;
m=cos(2*pi*fm*t); % Message signal
c=cos(2*pi*fc*t); % Carrier signal

% https://en.m.wikipedia.org/wiki/Frequency_modulation

% PM signal
x_PM = cos(2*pi*fc*t+kp*m);
% FM signal
x_FM = cos(2*pi*fc*t + (kf/fm)*sin(2*pi*fm*t)); % assuming Am = 1, thus deltaF = kf

% plots
```

Question 5

In a phase-shift keyed (PSK) digital communication system a binary digit is communicated to a receiver by sending either s1(t)= Acos(ωt) to represent “1” or s0(t)= Acos(ωt+π) to represent “0”. The input to the receiver is the noise corrupted signal x(t) = si(t) + w(t), where w(t) represents the channel noise. The received signal x(t) = si(t) + w(t) is demodulated by multiplying it by and applying a LPF. The result can be represented by a random variable:

$$\xi = \begin{cases} -A/2 + W & \text{for a } 0 \\ A/2 + W & \text{for a } 1 \end{cases}$$

where W is a Gaussian random variable. If we examine this closely, we can see how the error depends on the signal amplitude A. Consider the case of a 1 having been transmitted. Intuitively, if A is large enough, then the chance that the noise will cause an error or equivalently, $\xi < 0$, should be small. This probability, termed the probability of error, is given by $Pe = P(A/2+W < 0)$.

Therefore, in errorPSK.m, when A = [0.1:0.1:4] the probability of error is never less than T = 0.001.

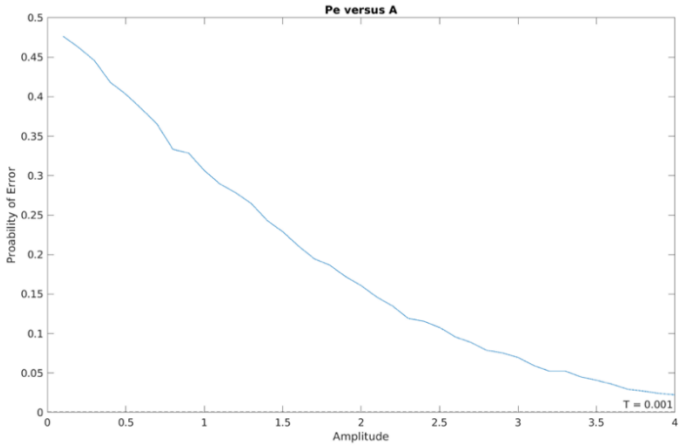

Figure 7 Probability of error versus Amplitude [0.1:0.1:4]

However, if we increase the amplitude to, for example, A = [5:0.1:10], the probability of error decreases drastically, becoming smaller than T.

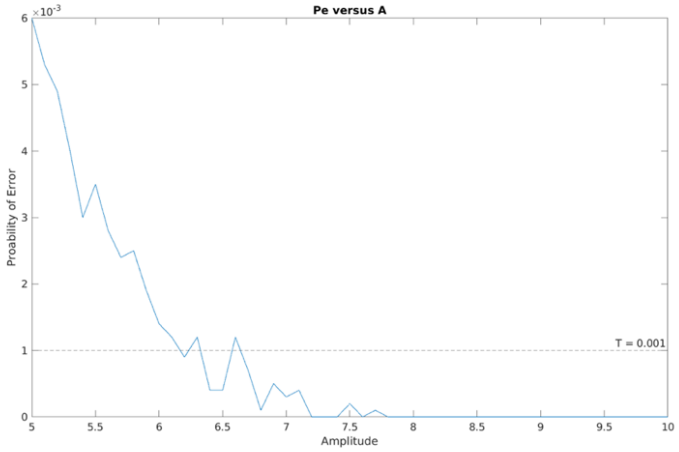

Figure 8 Probability of error versus Amplitude [5:0.1:10]

Question 6

The Matlab script correlation_and_psd.m visualises two signals x(t)=3sin(6πt)+sin(20πt)and y(t)=x(t)+2w(t), where w(t) is a zero mean noise. My script plots the graphs of the auto-correlation of y(t) and cross-correlation of x(t) and y(t) (see Figure 9) using the Matlab function xcorr(). In addition, I use fft() to compute the power spectral densities of x(t) and y(t) and plot them.

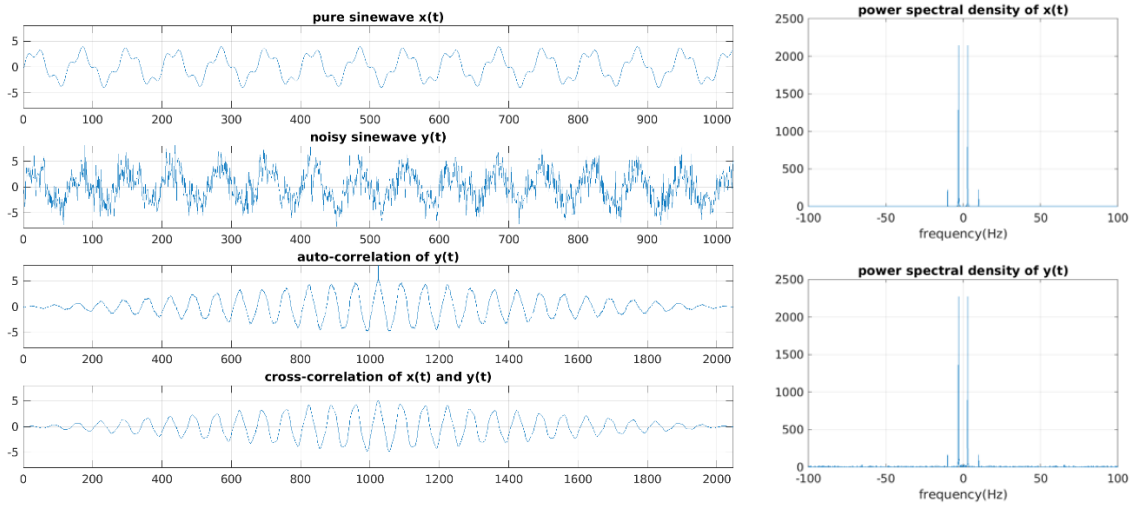

Figure 9 correlations and PSD of x(t) and y(t)

```
% correlation of two signals W9 > 6,7,11 (How PSD is used for signal denoising)
close all; clear all; clc;
N = 1024; % number of samples to generate
f1 = 3; % frequency of the sinewave
f2 = 10; % frequency of the sinewave
fs = 200; % sampling frequency
n = 0:N-1; % sampling index
x = 3*sin(2*pi*f1*n/fs)+sin(2*pi*f2*n/fs); % generating x[n]
y = x+2*randn(1,N); % generating y[n]=x[n]+w[n]

%auto-correlation and PSD of x(t) and y(t)
xc = xcorr(x,'biased');
xcdft = abs(fftshift(fft(xc)));
freqx = -fs/2:fs/length(xc):fs/2-(fs/length(xc));

yc = xcorr(y,'biased');
ycdft = abs(fftshift(fft(yc)));
freqy = -fs/2:fs/length(yc):fs/2-(fs/length(yc));

%cross-correlation of x(t) and y(t)
r = xcorr(x,y,'biased');

```

```
%plot
figure(1);
subplot(4,1,1),plot(x),axis([0 1024 -8 8]),title('pure sinewave x(t)'),grid;
subplot(4,1,2),plot(y),axis([0 1024 -8 8]),title('noisy sinewave y(t)'),grid;
subplot(4,1,3),plot(yc),axis([0 2048 -8 8]),title('auto-correlation of y(t)'),grid;
subplot(4,1,4),plot(r),axis([0 2048 -8 8]),title('cross-correlation of x(t) and y(t)'),grid;

figure(2);
subplot(2,2,1),plot(x),axis([0 500 -8 8]),title('pure sinewave x(t)'),grid;
subplot(2,2,3),plot(y),axis([0 500 -8 8]),title('noisy sinewave y(t)'),grid;
subplot(2,2,2),plot(freqx,xcdft),xlabel('frequency(Hz)'),title('power spectral density of x(t)'),grid;
subplot(2,2,4),plot(freqy,ycdft),xlabel('frequency(Hz)'),title('power spectral density of y(t)'),grid;
```

PSD can be a powerful tool for signal denoising. In Matlab you can use FFT to find the PSD or the power spectrum (power per freq.), and then find the frequencies with larger power (and zero out all others) and apply inverse FFT to output a filtered time signal. It is a simple and effective way of cleaning those signals that have been corrupted.

Question 7

a) Use U = rand(1,n) with n = 10 to generate 10 Rayleigh random numbers.

```
Generated Rayleigh random numbers are
3.6164      2.5948      3.8955      2.4253      1.4302
3.4246      1.2721     0.88641      4.2871      1.6091
```

b) Use U = rand(1,n) with n = 10000 and hist(R,50) to generate a histogram of Rayleigh random numbers. However, I have used histogram(R,'normalization','pdf') because I find it more suitable for the desired output.

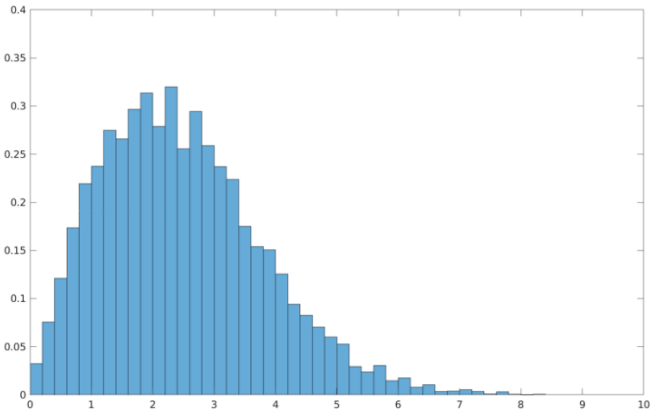

Figure 10 Rayleigh histogram when n = 10000

c) Plot the graph of Rayleigh pdf and compare with the histogram you generated.

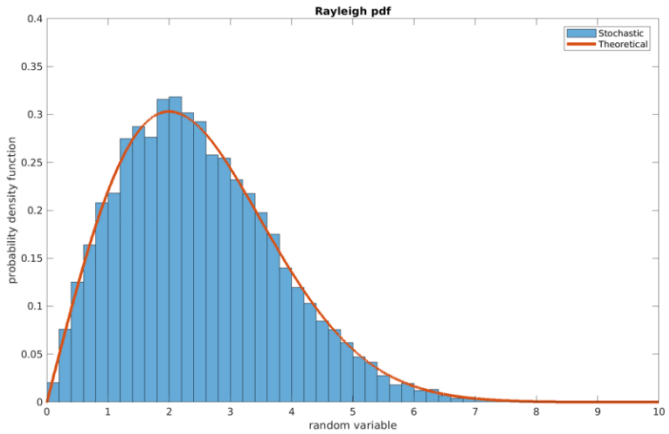

Figure 11 Comparing Rayleigh histogram and pdf

```
% Generation of Rayleigh random variable
clc; clear all; close all;

m=0; % mean
sgma=2;
n=10000; % number of random numbers
U=rand(1,n); % n uniform r.v.

% Rayleigh r.v. from U
R = sgma*sqrt(2*log(1./(1-U)));
%disp('Generated Rayleigh random numbers are');
%disp(num2str(R(:).'));

% plot histogram of Rayleigh random (use only when n=10000)
histogram(R,'normalization','pdf');

% plot the pdf
x=(m-5*sgma):0.01:(m+5*sgma); % define x-axis
y = x/sgma^2.* exp((-x.^2)/(2*sgma^2)); % Rayleigh pdf

hold on
plot (x,y,'LineWidth',3);
axis([0 10 0 0.4]);
ylabel('probability density function'); xlabel('random variable');
title('Rayleigh pdf');
legend('Stochastic','Theoretical')
```

Question 8

First, we need to apply nodal analysis to the s-domain circuit to get a system of linear equations.

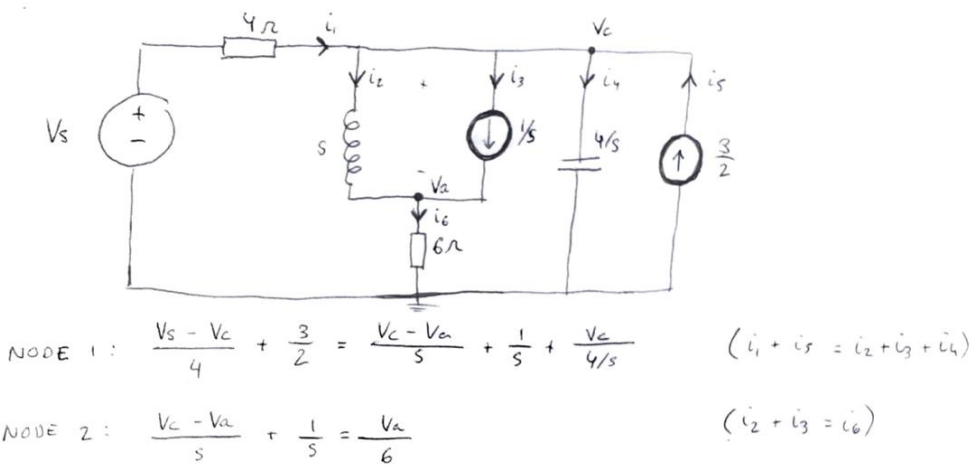

Figure 12 nodal analysis of the s-domain circuit

We know that Vs(t) = 6exp(-3t)u(t) V. Applying the Laplace transform we can get Vs(s). If we then replace it in the equations and solve the system, we can get Vc(s) and Va(s). Finally, if we apply the inverse Laplace transform to Vc(s) and Va(s), we will get Vc(t) and Va(t).

I have used Matlab to do all the calculations, and this is the result:

```
Vs(s) =
6/(s + 3)


va_s =

(6*(s^2 + 10*s + 27))/((s + 3)*(s^2 + 7*s + 10))


vc_s =

(2*(3*s^2 + 28*s + 66))/((s + 3)*(s^2 + 7*s + 10))

Va(t) =
22*exp(-2*t) - 18*exp(-3*t) + 2*exp(-5*t)

Vc(t) =
(44*exp(-2*t))/3 - 9*exp(-3*t) + exp(-5*t)/3
```

```
syms vs s t va vc

%Laplace transform of Vs(t)
vs = 6*exp(-3*t)*heaviside(t);
vs = laplace(vs,t,s);
disp('Vs(s) = ');
disp(vs);

%Solving system of linear equations (two unknowns in the s-domain)
eqn1 = (vs-vc)/4 + 3/2 == (vc-va)/s + 1/s + vc/(4/s) ;
eqn2 = (vc-va)/s + 1/s == va/6;
sol = solve([eqn1, eqn2], [va, vc]);
va_s = sol.va
vc_s = sol.vc
%fprintf('x = %d and y = %d \n',xSol,ySol);

%Applying inverse Laplace to find Va(t) and Vc(t)
va_t = ilaplace(va_s);
disp('Va(t) =');
disp(va_t);

vc_t = ilaplace(vc_s);
disp('Vc(t) =');
disp(vc_t);
```