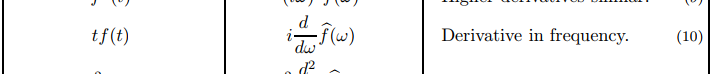
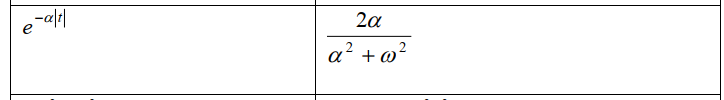
Mitchenkov Dmitriy

Problem 1. Compute (derive math formulas) the Fourier transform

a.

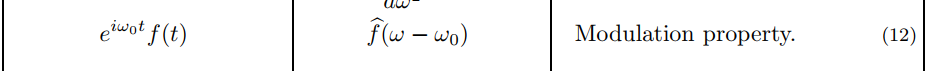


**

*Answer:*

b.

*So we need to find Fourier transform from 2 signals:*

**

**

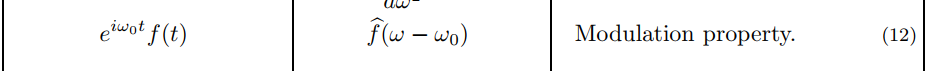
*Replace the variables:*

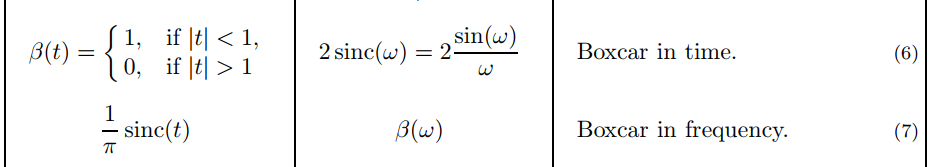
1. *The same for*

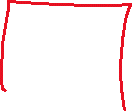
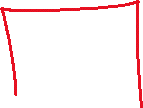
*Final result:*

*Answer:*

Problem 2.Compute (derive math formulas) the spectrum ofthe sine-filled sinc function

**



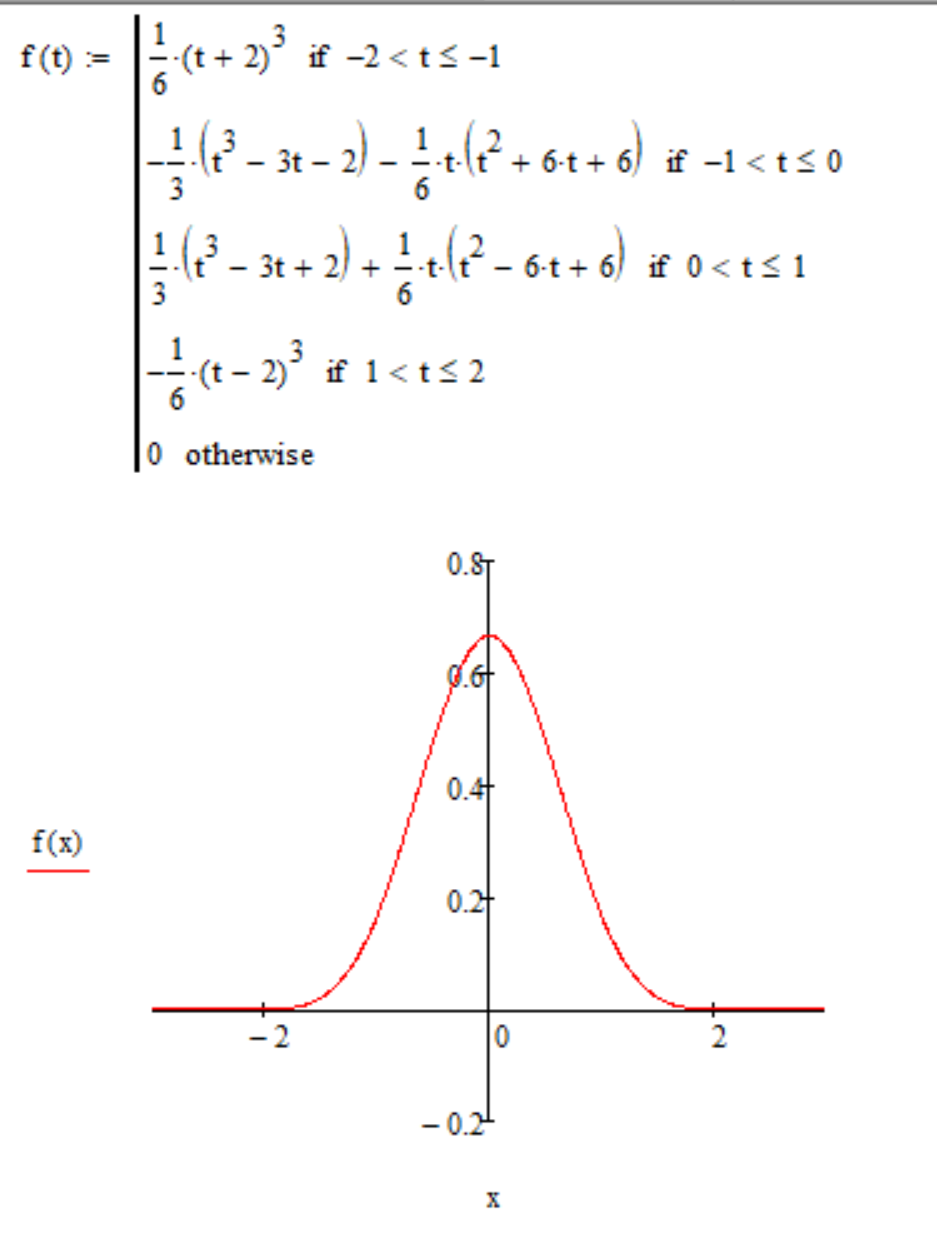


Problem 3.Compute (derive math formulas) the autocorrelation of the triangular pulse

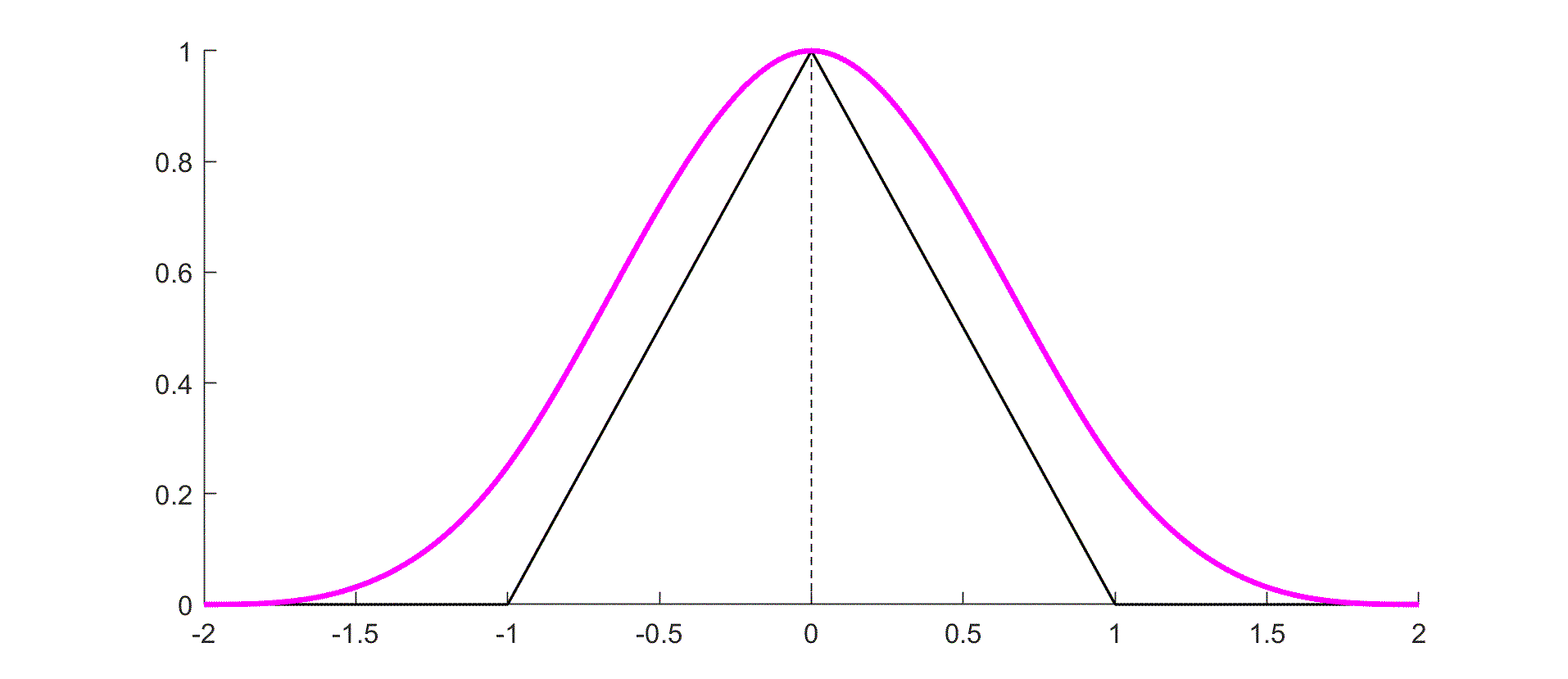
1. Compute (derive math formulas) the autocorrelation of the triangular pulse



|  |  |  |
| --- | --- | --- |
| 1 |  |  |
| 2 |  |  |
| 3 |  | |
| 4 |  | |
| 5 |  |  |
| 6 |  |  |



b)



  
Code listing:

clear all

close all

f = @(x) (1-abs(x)).\*(abs(x)<1);

fs = 100; ts = 1/fs; t\_0 = -2;

t\_x = t\_0:ts:2;

y = f(t\_x);

t\_c = -4:ts:4;

corr\_data = xcorr(y,'normalized');

figure(2);hold on;

plot(t\_x,y);plot(t\_c,corr\_data);

xlim([-2 2])

cr = @(t) corr\_data(round((t-(t\_0\*2))\*fs+1));

fps = 60;

T = 1/fps;

figure(1);hold on;

% подготовка матрицы и градиента

frm = getframe(gcf);

set(gcf,'color','w');

[im,map] = rgb2ind(frm.cdata,16);

im(1,1,1,length(t\_x)) = 0;

map = [double(de2bi(0:7)); 1 0.5 0.5; 0.5 0 0; 1 1 0.5; 0.5 0.5 0; 0.5 0.5 1; 0 0 0.5; 0.5 1 0.5; 0 0.5 0;];

start = tic;

frame = tic;

for tau = (2\*t\_0):2\*ts:(-2\*t\_0)

clf,hold on;set(gcf,'color','w');

t = t\_0:ts:tau;

i = round((tau-(2\*t\_0))\*(fs/2)+1);

if and(-2<=tau,tau<=-1)

% yellow

lim = -1:ts:(tau+1+(ts/2));

area(lim,max(f(lim),f(lim-tau)),'LineStyle','none','FaceColor','#FFFF80');

area(lim,min(f(lim),f(lim-tau)),'LineStyle','none','FaceColor','#808000');

end

if and(-1<=tau,tau<=0)

% red

lim = -1:ts:(tau+(ts/2));

area(lim,max(f(lim),f(lim-tau)),'LineStyle','none','FaceColor','#FF8080');

area(lim,min(f(lim),f(lim-tau)),'LineStyle','none','FaceColor','#800000');

% yellow

lim = tau:ts:(0+(ts/2));

area(lim,max(f(lim),f(lim-tau)),'LineStyle','none','FaceColor','#FFFF80');

area(lim,min(f(lim),f(lim-tau)),'LineStyle','none','FaceColor','#808000');

% blue

lim = 0:ts:(tau+1+(ts/2));

area(lim,max(f(lim),f(lim-tau)),'LineStyle','none','FaceColor','#8080FF');

area(lim,min(f(lim),f(lim-tau)),'LineStyle','none','FaceColor','#000080');

end

if and(0<=tau,tau<=1)

% red

lim = (tau-1):ts:(0+(ts/2));

area(lim,max(f(lim),f(lim-tau)),'LineStyle','none','FaceColor','#FF8080');

area(lim,min(f(lim),f(lim-tau)),'LineStyle','none','FaceColor','#800000');

% green

lim = 0:ts:(tau+(ts/2));

area(lim,max(f(lim),f(lim-tau)),'LineStyle','none','FaceColor','#80FF80');

area(lim,min(f(lim),f(lim-tau)),'LineStyle','none','FaceColor','#008000');

% blue

lim = tau:ts:(1+(ts/2));

area(lim,max(f(lim),f(lim-tau)),'LineStyle','none','FaceColor','#8080FF');

area(lim,min(f(lim),f(lim-tau)),'LineStyle','none','FaceColor','#000080');

end

if and(1<=tau,tau<=2)

% green

lim = (tau-1):ts:(1+(ts/2));

area(lim,max(f(lim),f(lim-tau)),'LineStyle','none','FaceColor','#80FF80');

area(lim,min(f(lim),f(lim-tau)),'LineStyle','none','FaceColor','#008000');

end

plot(t\_x,f(t\_x),'k','LineWidth',1,'DisplayName','triangular func.');

plot([tau tau],[0 1],'k--');plot(tau,cr(tau),'ko');

plot(t\_x+tau,f(t\_x),'k','LineWidth',1);

plot([0 0],[0 1],'k--');

plot(t,cr(t),'m','LineWidth',2,'DisplayName','autocorrelation func.');

xlim([-2 2])

ylim([0 1])

time = toc(frame);

frame = tic;

% pause(T-time);

frm = getframe(gcf);

img = frame2im(frm);

im(:,:,1,i) = rgb2ind(img,map);

% delete (ar);

% delete (point);

end

im(:,:,1,1) = rgb2ind(img,map);

imwrite(im,map,'exp\_4.gif','DelayTime',T,'LoopCount',inf)

% avg\_fps = length(x)/toc(start)

Problem 4. Compute (by hands) the convolution of the following signals:a. h = [2 3 6 8]; x = [1 2 10 1]; b. h = [5 1 3 10]; x = [9 6 10 1];

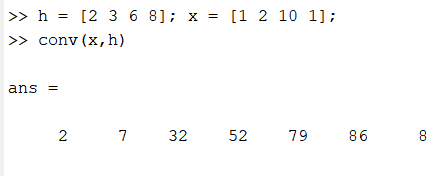
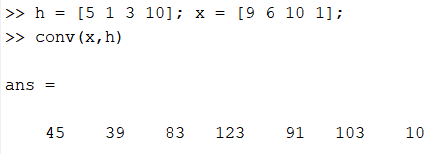
Step 1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | *x* | | | |  |  |  | *x* | | | |
|  |  | *1* | *2* | *10* | *1* |  |  |  | *9* | *6* | *10* | *1* |
| *h* | *2* | 2 | 4 | 20 | 2 |  | *h* | *5* | 45 | 30 | 50 | 5 |
| *3* | 3 | 6 | 30 | 3 |  | *1* | 9 | 6 | 10 | 1 |
| *6* | 6 | 12 | 60 | 6 |  | *3* | 27 | 18 | 30 | 3 |
| *8* | 8 | 16 | 80 | 8 |  | *10* | 90 | 60 | 100 | 10 |

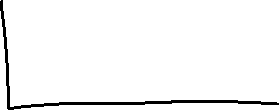
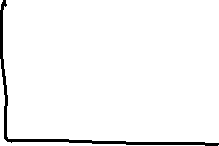
Step 2.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | *x* | | | |  |  |  |  |  |  |  | *x* | | | |  |  |  |
|  |  | *1* | *2* | *10* | *1* |  |  |  |  |  |  |  | *9* | *6* | *10* | *1* |  |  |  |
| *h* | *2* | 2 | 4 | 20 | 2 | 0 | 0 | 0 |  |  | *h* | *5* | 45 | 30 | 50 | 5 | 0 | 0 | 0 |
| *3* | 0 | 3 | 6 | 30 | 3 | 0 | 0 |  |  | *1* | 0 | 9 | 6 | 10 | 1 | 0 | 0 |
| *6* | 0 | 0 | 6 | 12 | 60 | 6 | 0 |  |  | *3* | 0 | 0 | 27 | 18 | 30 | 3 | 0 |
| *8* | 0 | 0 | 0 | 8 | 16 | 80 | 8 |  |  | *10* | 0 | 0 | 0 | 90 | 60 | 100 | 10 |
|  |  | **2** | **7** | **32** | **52** | **79** | **86** | **8** |  |  |  |  | **45** | **39** | **83** | **123** | **91** | **103** | **10** |

Step 3.

Problem 5. The sinusoidal signal with the frequency 6kHz ….a. … is sampled with the frequency 10 kHz. Compute the apparent frequency after the signal reconstruction.





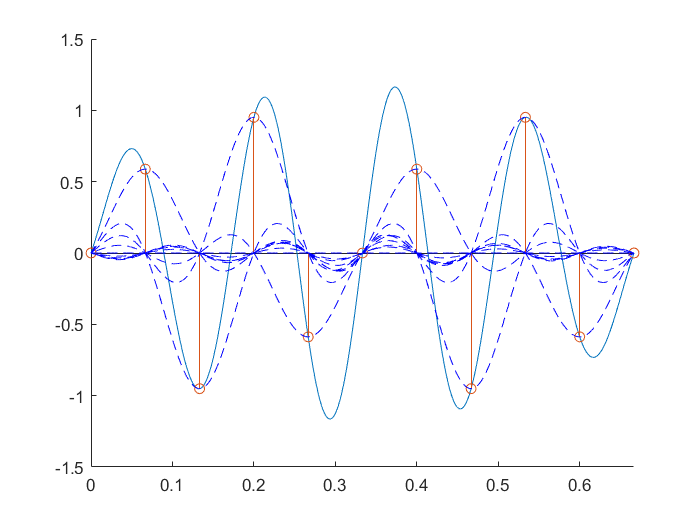
The signals in time domain



The signals in frequency domain

Answer: The reconstructed with Nyquist filter signal will have the frequency equal to 4kHz.

b. … is represented by 4 signal periods sampled with the sample clock frequency 15 kHz. Are these samples enough to reconstruct the initial signal correctly (without ANY error)? Plot the original and the reconstructed signals. Explain the difference.



The reconstruction of signal using Nyquist filter



The reconstructed signal

Answer: this quantity of samples is not enough to reconstruct the limited in time signal. With incrasing a number of periods error will decrease

MATLAB code for task №5:

close all

clear all

fss = 15e3; % sampling frequency for modeling

fs = 15; % sampling frequency

f\_signal = 6; % signal frequency

w =2\*pi\*f\_signal; % signal angular frequency

Num\_of\_periods = 4; % number of periods of signal

t = Num\_of\_periods/f\_signal; % modeling time: 0..t

time\_signal = 0:(1/fss):t; % time counts for modeling

time\_sampled = 0:(1/fs):t; % time counts for discretization

signal = sin(w\*time\_signal); % raw signal

discrete = sin(w\*time\_sampled); % sample counts

figure(1);plot(time\_signal,signal);

hold on;

% plot(time\_sampled,discrete,'ks')

stem(time\_sampled,discrete,'ks--');

% попытка восстановить сигнал с использованием синков

tt = (-t:(1/fss):t); % time counts for filter (based on modeling frequency)

Nyquist\_filter = sinc(fs\*tt); % Nyquist filter in time domain

% plot(tt,Nyquist\_filter);

upsampled = upsample(discrete,fss/fs); % upsample discrete signal to the modeling frequency (new samples is equal to zero)

upsampled(end+1-(fss/fs-1):end) = []; % remove last unused zeros: S000S000S[000] (S = sample)

restored = conv(upsampled,Nyquist\_filter,'same'); % restored signal: signal -> sample -> filter

ttt = linspace(0,t,length(restored));

figure(2);hold on;

plot(ttt,restored);stem(time\_sampled,discrete);plot(time\_signal,signal);

legend('restored','sampled','original');

figure(4);hold on;

fft\_restored = abs(fft(restored)/length(restored));

plot(linspace(0,fss,length(restored)),fft\_restored)

fft\_sampled = abs(fft(upsampled)/length(upsampled)).\*fss/fs; %energy loss compensation

plot(linspace(0,fss,length(upsampled)),fft\_sampled)

fft\_signal = abs(fft(signal)/length(signal));

plot(linspace(0,fss,length(signal)),fft\_signal)

legend('restored','sampled','original');xlim([0, 2\*fs]);set(gca, 'YScale', 'log')

% return

[rId, cId] = find(upsampled);

figure(3);hold on;

plot(ttt,restored);stem(time\_sampled,discrete);

for i = cId

plot(tt+((i-1)\*1/fss),upsampled(i)\*Nyquist\_filter,'b--');

end

xlim([0 t]);