Lab 6

Aravind Reddy V

IMT 2015 524

Question 1a

Bn = [-2, 1, 0, 2, -1]

4 different symbols, using 2 bits to represent each symbol

Using cos(pi\*t)I[0, 1] as reference symbol

|  |
| --- |
| a_ref.png  Fig. 1a – Reference signal |

Message signal:

|  |
| --- |
| a.png  Fig. 1b – Message signal |

Question 1 b&c

Using carrier frequency = 25 Hz.

|  |
| --- |
| 2_am.png  Fig.2 – AM signal |

Question 1d

Carrier frequency = 25 Hz.

|  |
| --- |
| d_freq.png  Fig.3a – Frequency response of message and AM |

Question 1e

Demodulated wave closely resembles the actual message signal. Symbols can be read off from eyeballing.

|  |
| --- |
| e_dem.png  Fig. 4a – Diode response, demodulated wave and message signal |

|  |
| --- |
| resp.png  Fig. 4b – Filter response. Rc = |

Code:

%% Octave

symbols = [-2, 1, 0, 2, -1]

dt = 0.01;

t\_ref = 0:dt:1-dt;

ref = cos(pi\*t\_ref);

plot(t\_ref, ref);

xlabel("Time");

ylabel("Amplitude");

title("Reference signal = cos(\\pi\*t)I[0, 1]");

print -dpng a\_ref.png

% a

% symbols are -2, 1, 0, 2, 1

% There are four different symbols, using 2 bits per symbol

% -2 : 00

% 0 : 01

% 1 : 10

% 2 : 11

% bits = 00 | 10 | 01 | 11 | 10

% 0 = -1

% 1 = +1

% bit zero bit one

b\_zero = ref.\*-1;

b\_one = ref.\*1;

% message -2, 0, 1, 2

m\_minus\_two = [b\_zero'; b\_zero'];

m\_zero = [b\_zero'; b\_one'];

m\_one = [b\_one'; b\_zero'];

m\_two = [b\_one'; b\_one'];

t\_message = 0:dt:1\*10 - dt; % 10 bits

message = [ m\_minus\_two; m\_one; m\_zero; m\_two; m\_one]';

plot(t\_message, message);

title("Message signal; bits: 1 = +ref, 0 = -ref; symbols: -2 -> 00, 0 -> 01, 1 -> 10, 2 -> 11");

xlabel("Time (s)");

ylabel("Amplitude");

print -dpng a.png

%%%%%%%%%%%%%%%%%%%%%%%%%%

% b & c

% mo = |-1|

% Ac = 1

% amod = 1 => A = 1

Fc = 25;

t\_am = t\_message;

carrier = cos(2\*pi\*Fc\*t\_am);

am = (message.\*carrier).+carrier;

plot(t\_am, am);

title(["AM modulated wave. Fc = ", num2str(Fc)]);

xlabel("Time (s)");

ylabel("Amplitude");

print -dpng 2\_am.png

%%%%%%%%%%%%%%%%%%%%%%%%%%

% d

[freq\_message, ampl\_message] = getFFT(message, dt);

[freq\_am, ampl\_am] = getFFT(am, dt);

subplot(2, 1, 1);

plot(freq\_message, ampl\_message);

xlabel("Frequency (Hz)");

ylabel("Magnitude");

title("Frequency Response of message signal");

subplot(2, 1, 2);

plot(freq\_am, ampl\_am);

xlabel("Frequency (Hz)");

ylabel("Magnitude");

title(["Frequency Response of DSB AM signal. Fc = ", num2str(Fc)]);

print -dpng d\_freq.png

%%%%%%%%%%%%%%%%%%%%%%%%%%

% e

time\_diode = t\_am;

diode = diodeFilter(am);

subplot(3, 1, 1);

plot(time\_diode, diode);

title("Diode Response");

xlabel("Time (s)");

ylabel("Amplitude");

RC = 0.1;

[dem\_time, dem] = RCfilter(time\_diode, diode, RC);

% [dem\_time, dem] = DCblock(dem\_time, dem);

subplot(3, 1, 2);

plot(dem\_time, dem);

title("Demodulated wave");

xlabel("Time (s)");

ylabel("Amplitude");

xlim([0, 25]);

subplot(3, 1, 3);

plot(t\_message, message);

title("Message signal; bits: 1 = +ref, 0 = -ref; symbols: -2 -> 00, 0 -> 01, 1 -> 10, 2 -> 11");

xlabel("Time (s)");

ylabel("Amplitude");

print dpng e\_dem.png

%%%%%%%%%%%%%%% Filter response

% H(s) = 1/(1 + sRC);

% Inverse fourier transform for h(s)

% http://www.wolframalpha.com/input/?i=InverseFourier(1%2F(1%2Bas))

% RC and t are positive

resp\_time = -40:dt:40;

resp1 = sign(resp\_time).\*sin(resp\_time./RC);

resp2 = sign(resp\_time).\*cos(resp\_time./RC).\*-i;

coeff = (1/RC)\*(sqrt(pi/2));

resp = resp1.+resp2;

% resp = resp.\*coeff;

subplot(3, 1, 1);

plot(resp\_time, real(resp));

title("Filter impulse response : real part");

xlabel("Time (s)");

ylabel("Amplitude");

subplot(3, 1, 2);

plot(resp\_time, imag(resp));

title("Filter impulse response : imaginary part");

xlabel("Time (s)");

ylabel("Amplitude");

subplot(3, 1, 3);

plot(resp\_time, abs(resp));

title(["Filter impulse response absolute value. RC = ", num2str(RC)]);

xlabel("Time (s)");

ylabel("Amplitude");

print -dpng resp.png

% getFFT.m

%% getFFT: function description

function [freq, ampl] = getFFT(wave, dt)

WAVE = fft(wave);

fs = 1/dt;

freq = -fs/2:fs/(length(WAVE) - 1):fs/2;

ampl = fftshift(abs(WAVE))\*dt/2;

end

% DCblock.m

%% DBblock: function description

function [time\_dcblock, signal\_dcblock] = DCblock(time, signal)

meanSignal = mean(signal)

time\_dcblock = time;

signal\_dcblock = signal.-meanSignal;

end

% diodeFilter.m

%% diodeFilter: makes all values less than zero 0

function [result] = diodeFilter(vector)

vector(vector < 0) = 0;

result = vector;

end

% RCfilter.m

%% RCFilter: function description

function [time\_f, signal\_f] = RCfilter(time, signal, RC = 0.383)

% t\_response = 0:ns/length(time):ns;

% 1/fc < RC < 1/b

% b = 1.5 KHz

t\_response = time;

dt = 1/40;

u\_response = ones(length(signal), 1);

% RC = 3.833 / 10;

% RC = 1 / 1.5;

temp\_t = t\_response./RC;

temp\_t = temp\_t.\*-1;

temp\_t\_exp = arrayfun( @(x) exp(x), temp\_t);

u\_response = u\_response.\*temp\_t\_exp;

u\_response = u\_response(1,:);

size(t\_response)

size(u\_response)

[time\_f, signal\_f] = contconv(signal, u\_response, time(1), t\_response(1), dt);

end

% contconv.m

function [time, convolution] = contconv (x1, x2, t1, t2, dt)

Tstart1 = t1;

Tstop1 = t1 + length(x1)\*dt - dt;

Tstart2 = t2;

Tstop2 = t2 + length(x2)\*dt - dt;

startTime = Tstart1 + Tstart2;

endTime = Tstop1 + Tstop2;

time = startTime:dt:endTime;

t = 1

convolution = conv(x1,x2).\*dt;

endfunction

contFT.m

function [X,f,df] = contFT(x,tstart,dt,df\_desired)

%Use Matlab DFT for approximate computation of continuous time Fourier %transform

%INPUTS

%x = vector of time domain samples, assumed uniformly spaced

%tstart= time at which first sample is taken

%dt = spacing between samples

%df\_desired = desired frequency resolution

%OUTPUTS

%X=vector of samples of Fourier transform

%f=corresponding vector of frequencies at which samples are obtained %df=freq resolution attained (redundant--already available from %difference of consecutive entries of f)

%%%%%%%%%

%minimum FFT size determined by desired freq

Nmin=max(ceil(1/(df\_desired\*dt)),length(x));

%choose FFT size to be the next power of 2

Nfft = 2^(nextpow2(Nmin));

%compute Fourier transform, centering around

X=dt\*fftshift(fft(x,Nfft));

%achieved frequency resolution

df=1/(Nfft\*dt);

%range of frequencies covered

f = ((0:Nfft-1)-Nfft/2)\*df; %same as f=-1/(2\*dt):df:1/(2\*dt) - df

%phase shift associated with start time

X=X.\*exp(-j\*2\*pi\*f\*tstart);

End