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

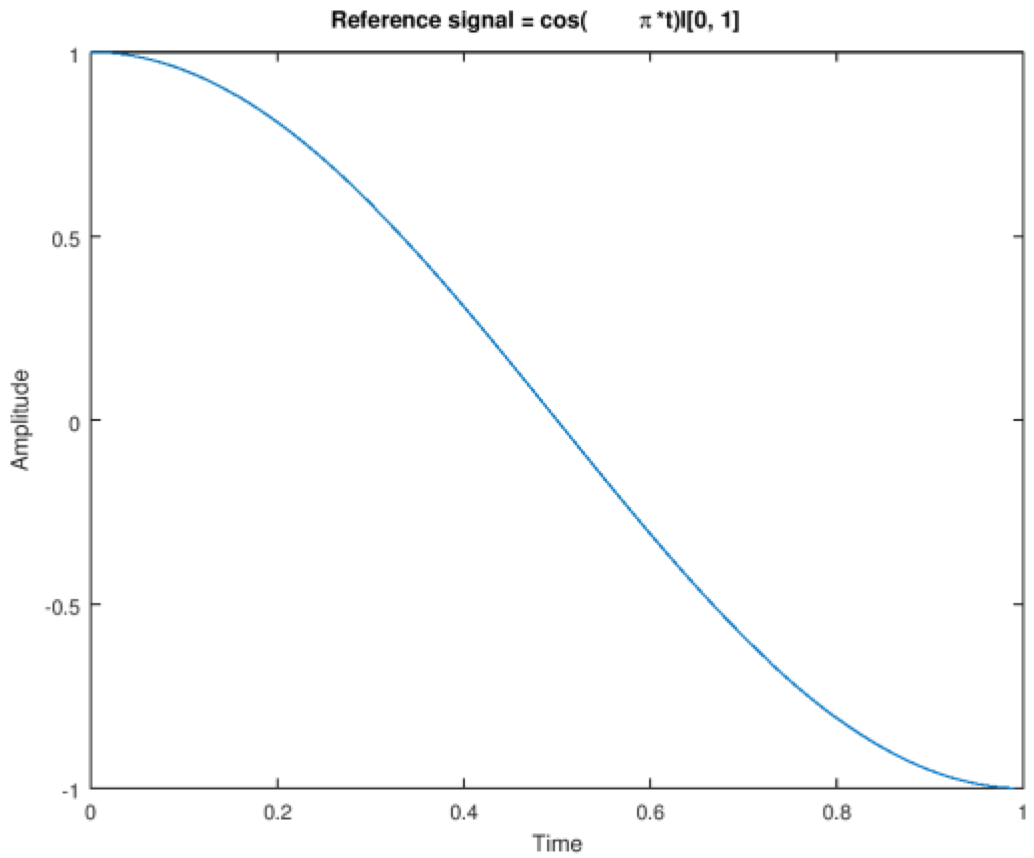

Fig. 1a – Reference signal

Message signal:

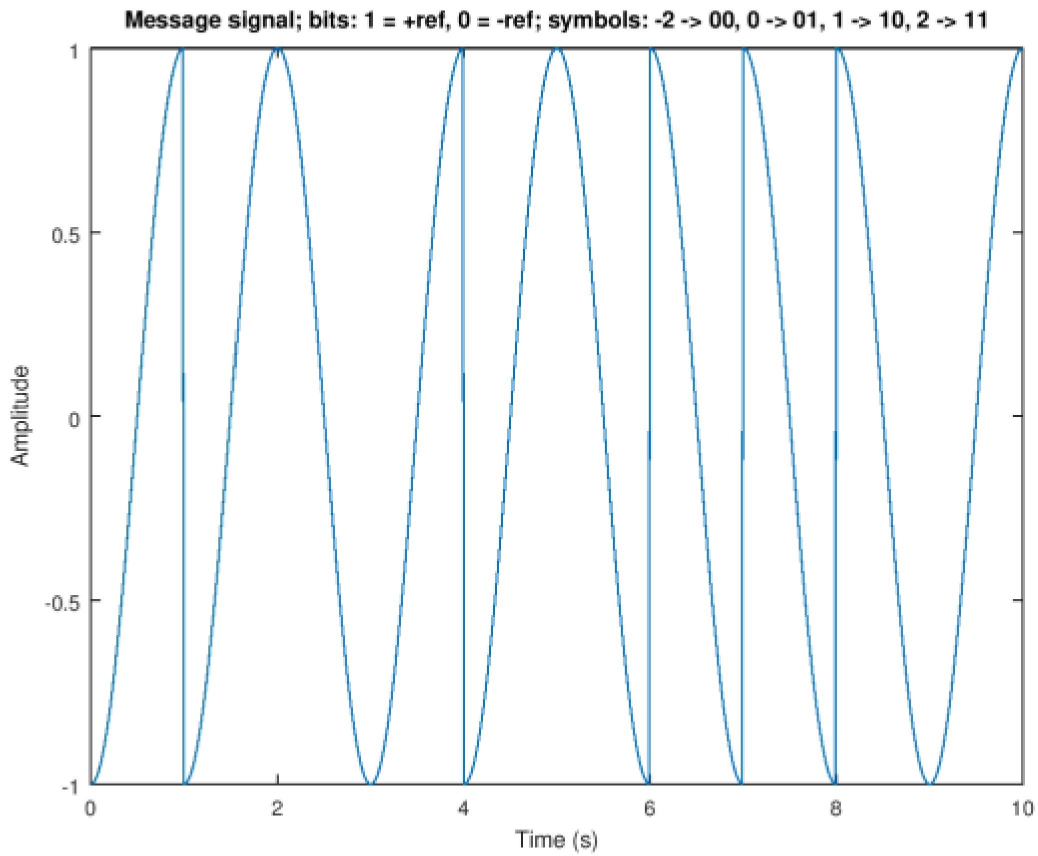

Fig. 1b – Message signal

Question 1 b&c

Using carrier frequency = 25 Hz.

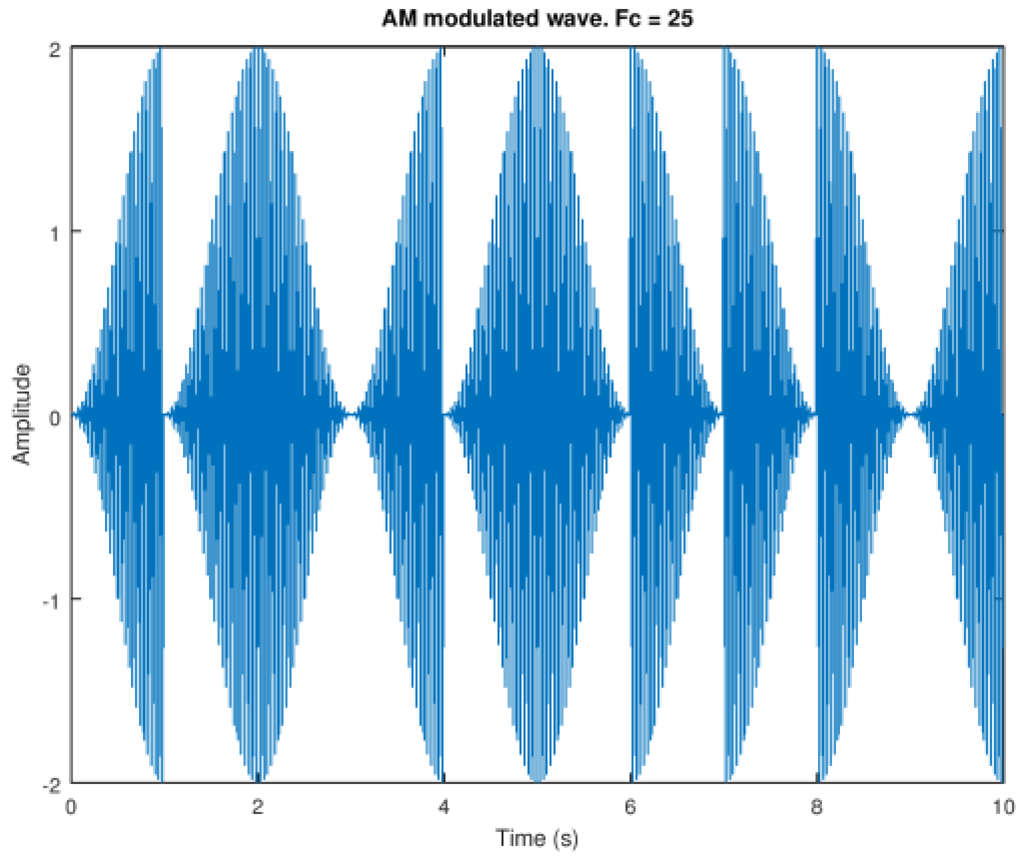

Fig.2 – AM signal

Question 1d

Carrier frequency = 25 Hz.

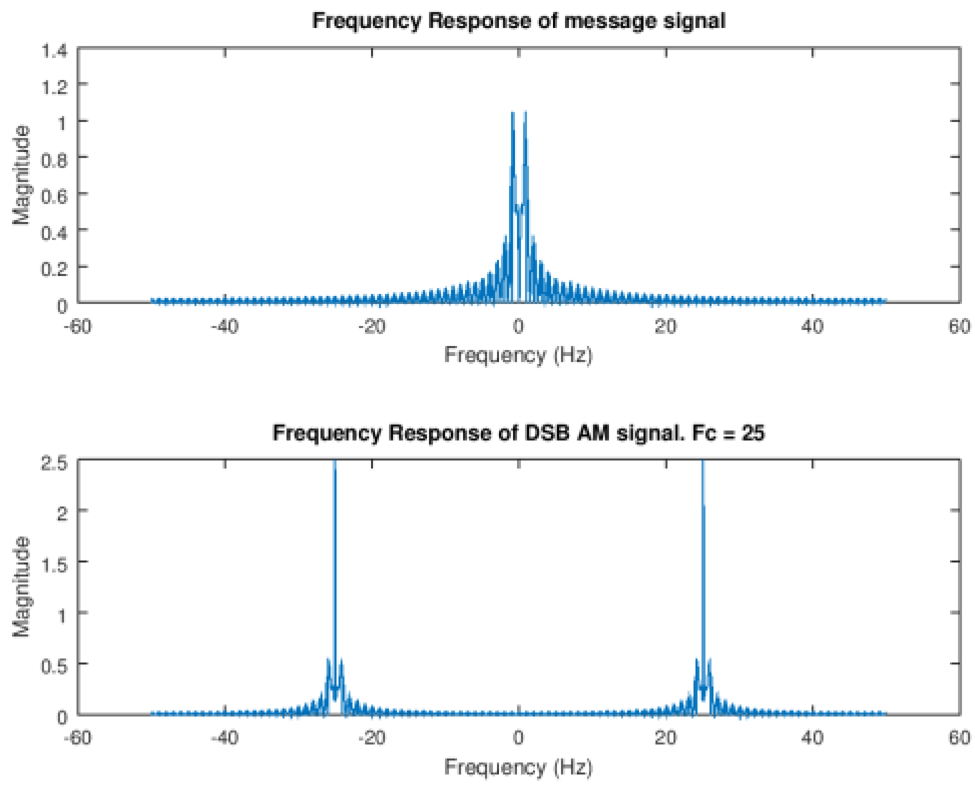

Fig.3a – Frequency response of message and AM

Question 1e

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

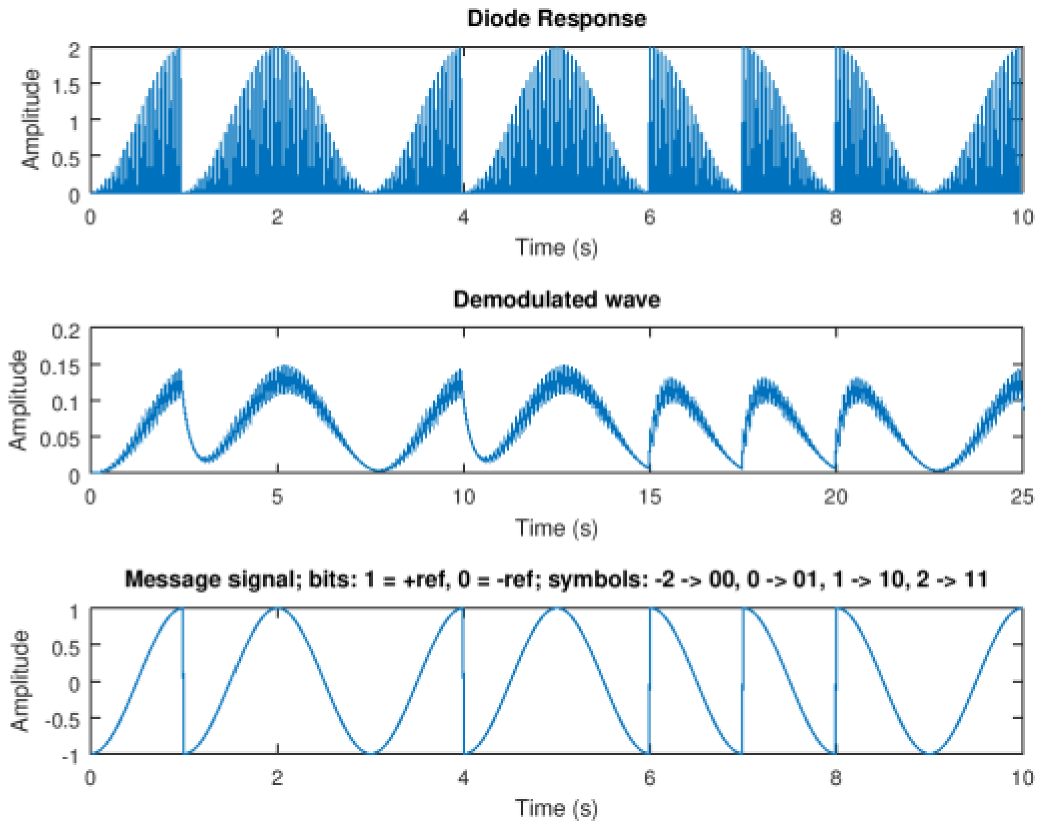

Fig. 4a – Diode response, demodulated wave and message signal

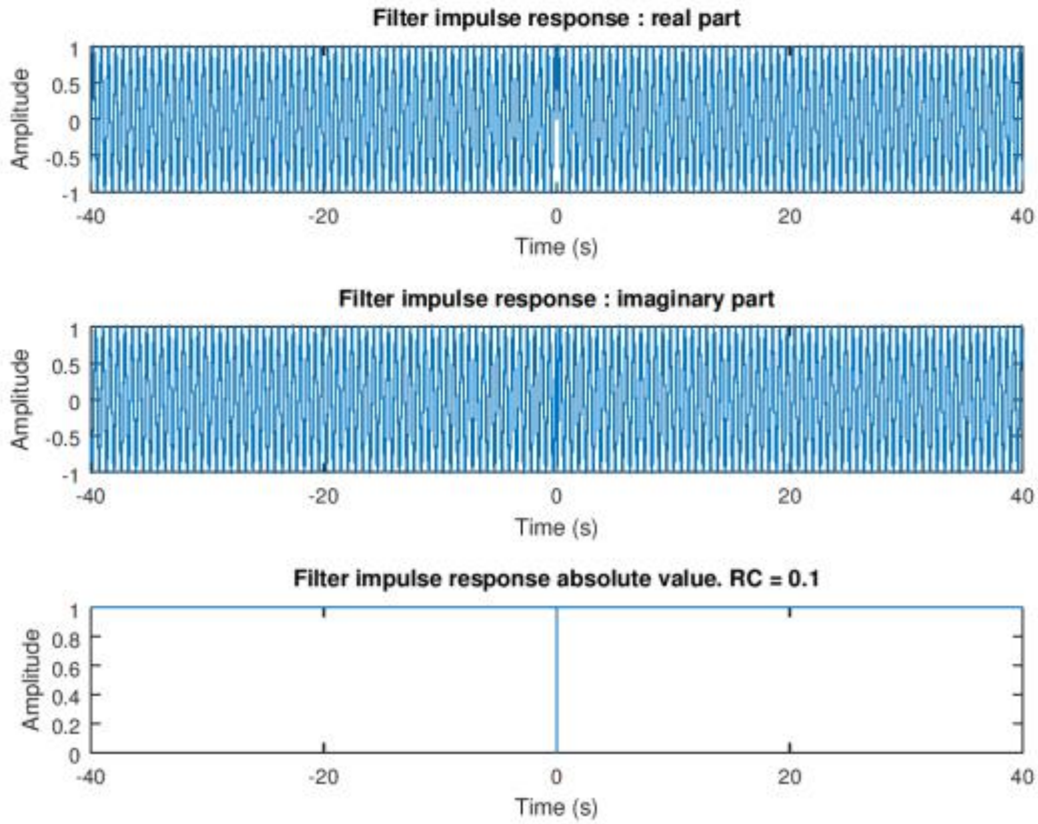

Fig. 4b – Filter response. Rc =

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
```