Lab 6
Aravind Reddy V
IMT 2015 524

<u>Question 1</u>

I and Q component of FM signal, kf = 0.25

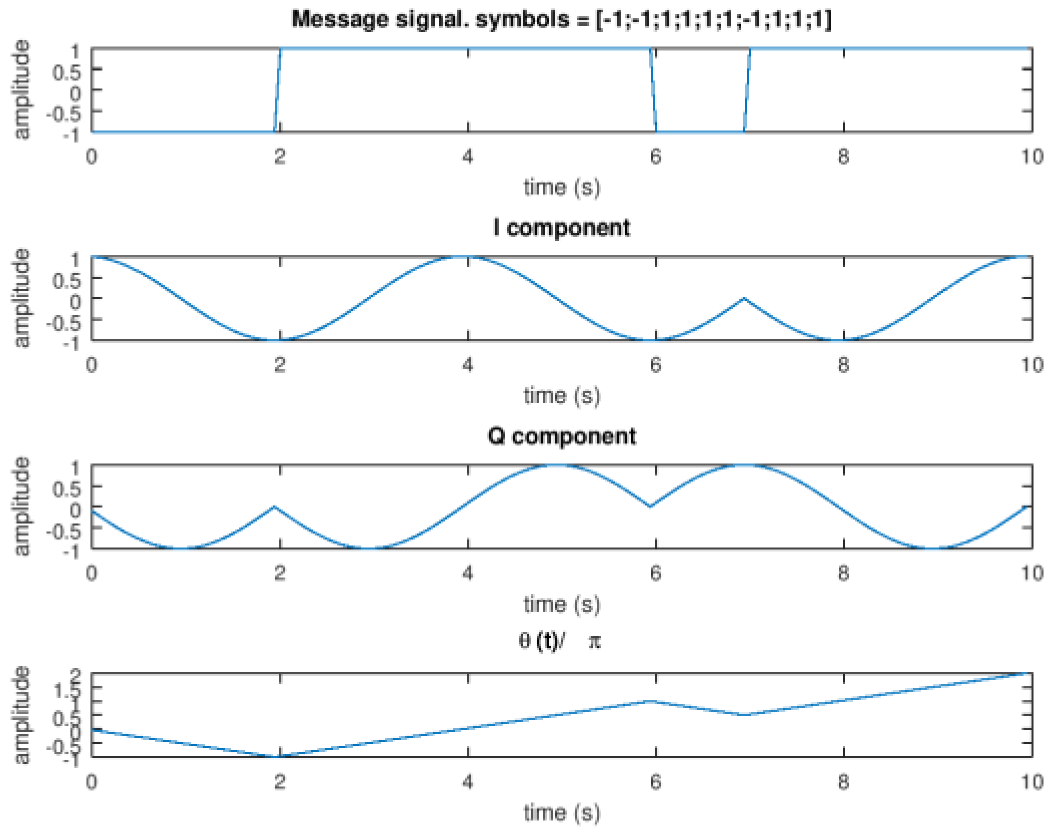

Fig. 1 – I and Q components, theta/pi of FM signal

Phase decreases when message symbol is negative and increases when it is positive

When the message signal changes, there is a phase change in either I or Q component.

Question 2

I and Q component of FM signal, kf = 4

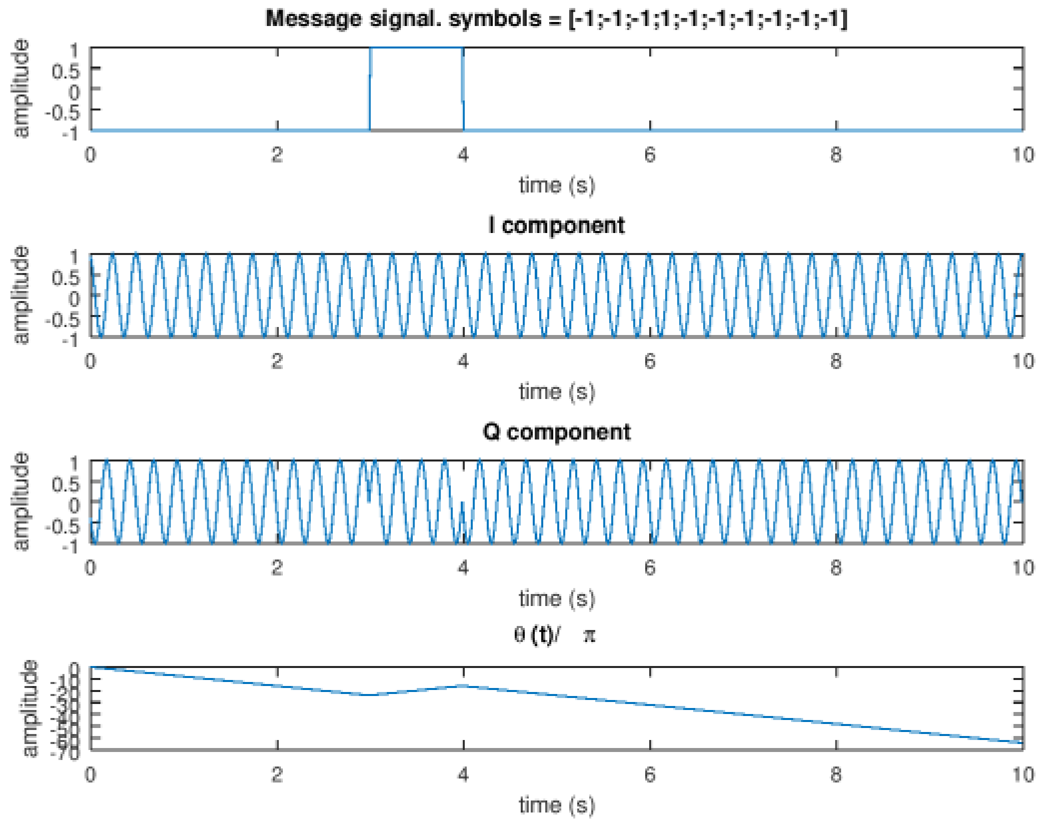

Fig.2 – kf = 4

Patterns in I and Q are more difficult to observe now
Phase, however, can be read with the same ease.

Question 3

$$\frac{d}{dt}\left(\tan^{-1} x\right) = \frac{1}{1+x^2}$$

$$\theta(t) = \tan^{-1}\left(\frac{y_s(t)}{y_c(t)}\right) \Rightarrow \theta'(t) = \frac{y_c^2(t)}{y_s^2(t)+y_c^2(t)} \cdot \left(\frac{y_s(t)}{y_c(t)}\right)'$$

$$= \frac{\cancel{y_c^2(t)}}{y_s^2(t)+y_c^2(t)} \cdot \frac{y_c(t)\cdot y_s'(t) - y_s(t)\cdot y_c'(t)}{\cancel{[y_c(t)]^2}}$$

$$= \frac{y_c(t)\cdot y_s'(t) - y_s(t)\cdot y_c'(t)}{y_s^2(t)+y_c^2(t)}$$

Fig.3 - derivation

Question 4

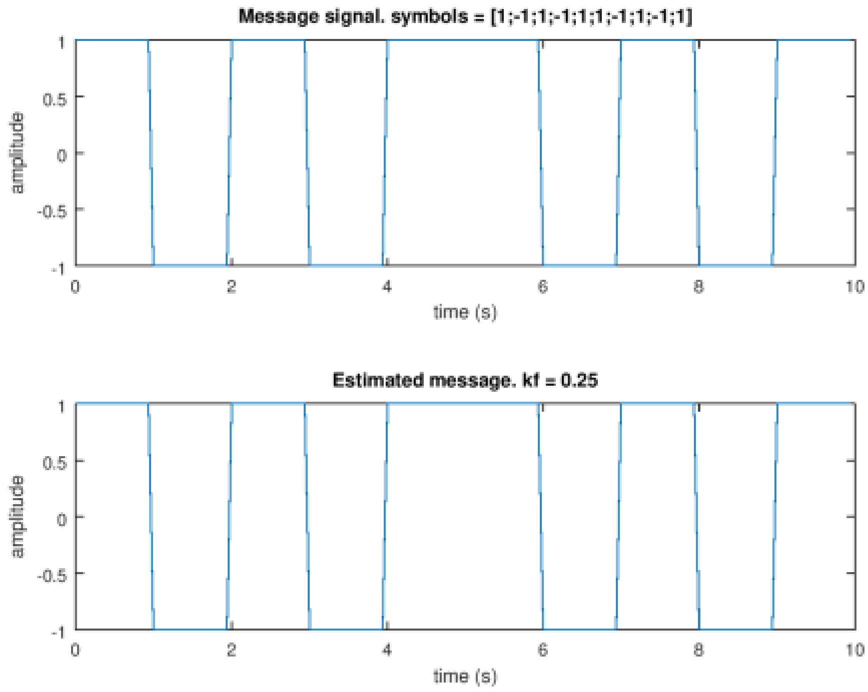

Fig. 4a – Estimated signal, kf = 0.25

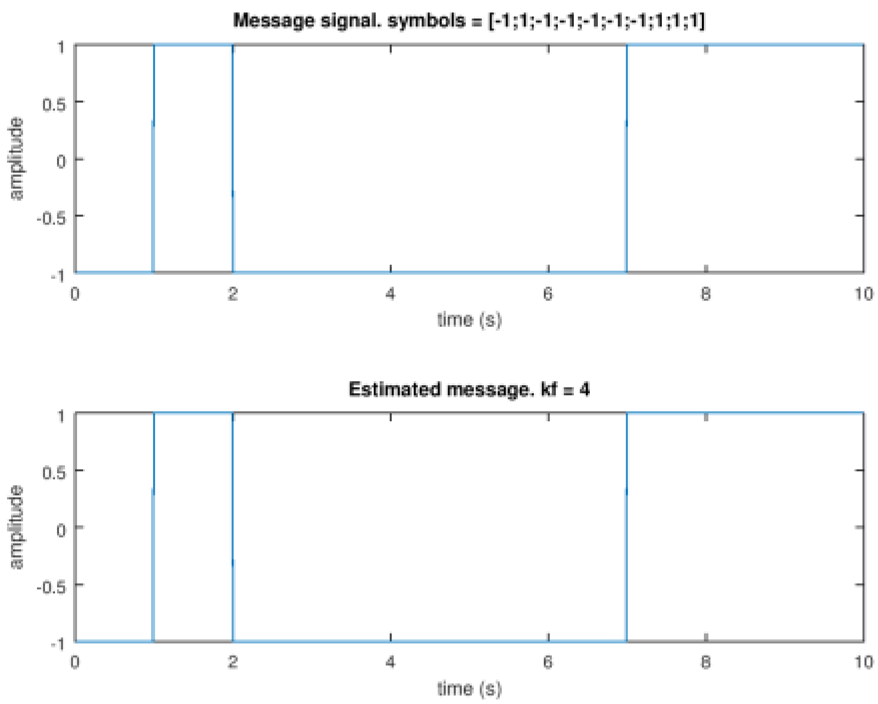

Fig. 4b – Estimated signal, kf = 4

From eyeballing, these are good estimates of the original message signal.

Question 5

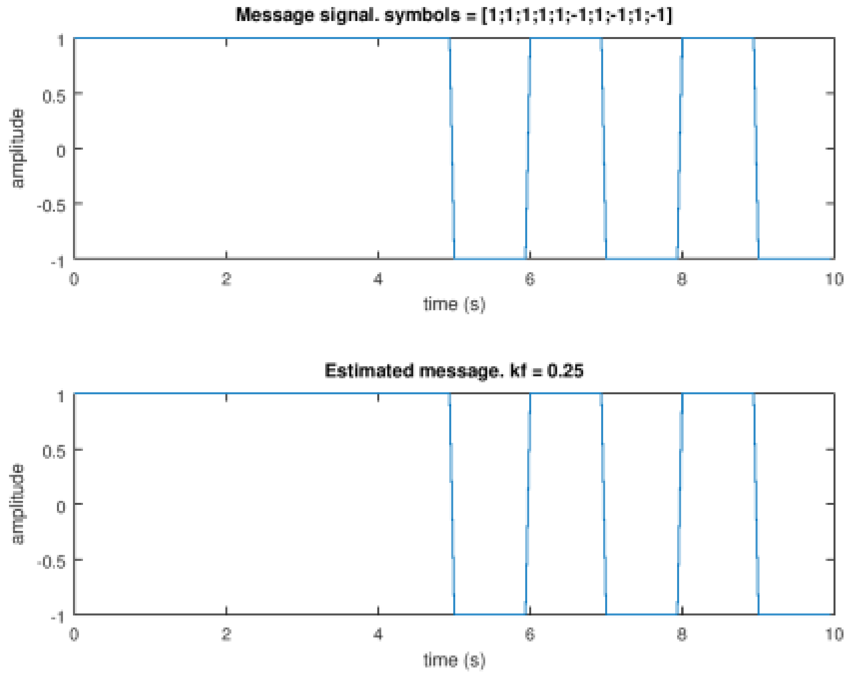

Fig. 5a – Random phase added to complex envelope – Kf = 0.25

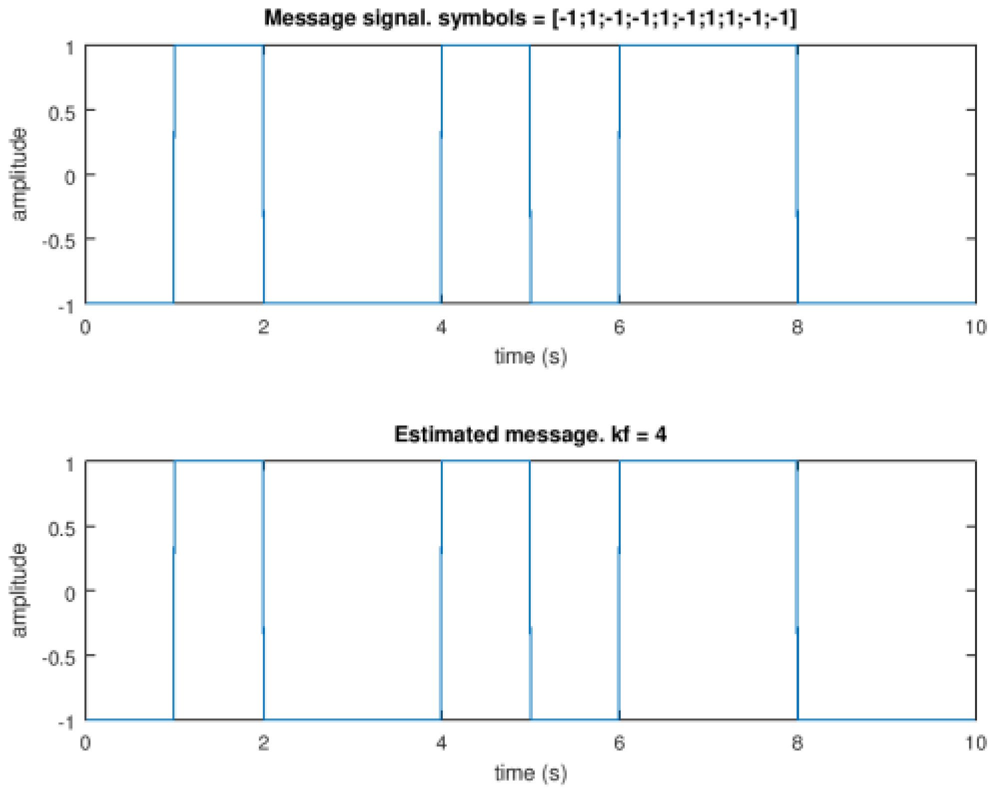

Fig. 5b – Random phase added to complex envelope – Kf = 4

Addition of phase does not produce any distortion.

Question 6

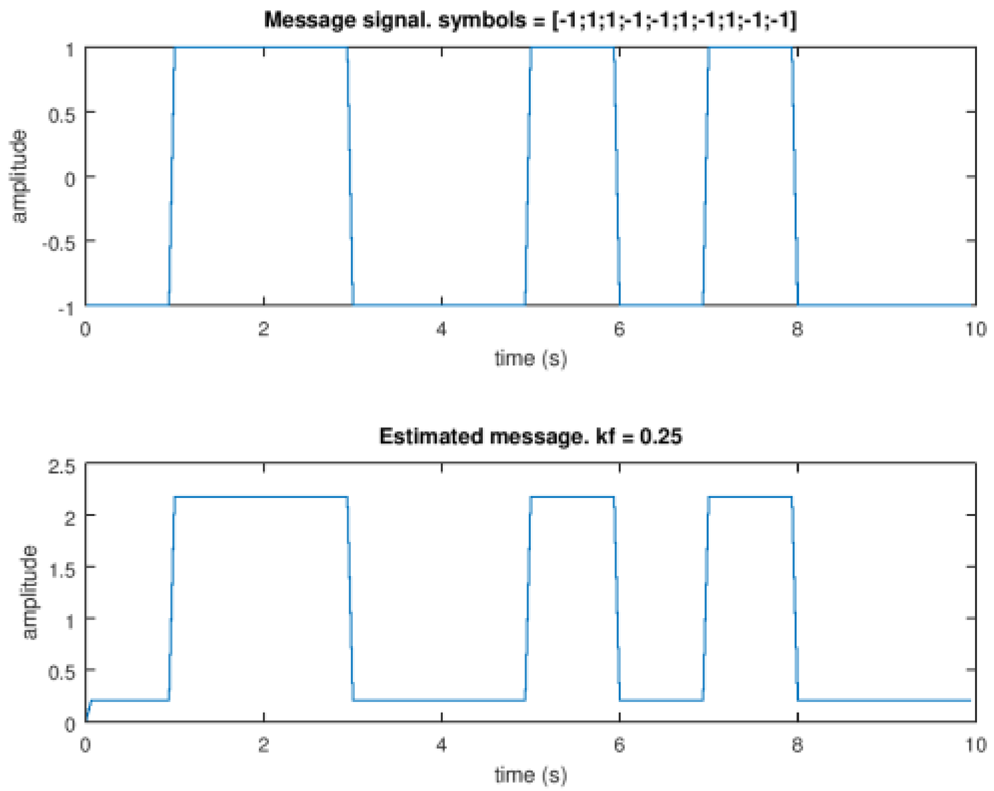

Fig. 6a – Random phase and frequency offset – Kf = 0.25

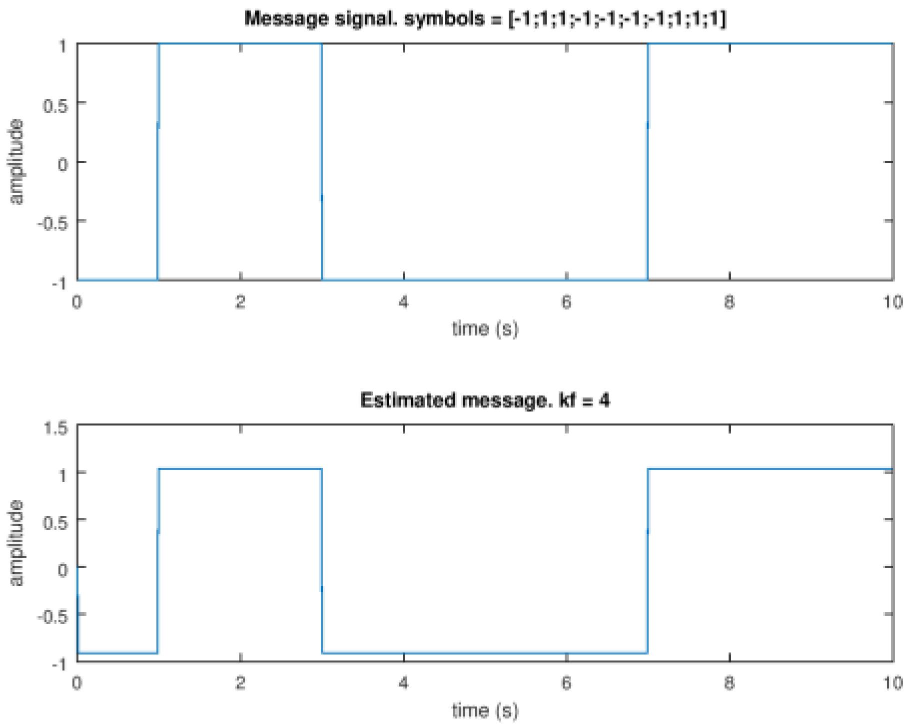

Fig. 6b - Random phase and frequency offset – Kf = 4

Frequency offset produces a DC shift in the estimated message

Question 7

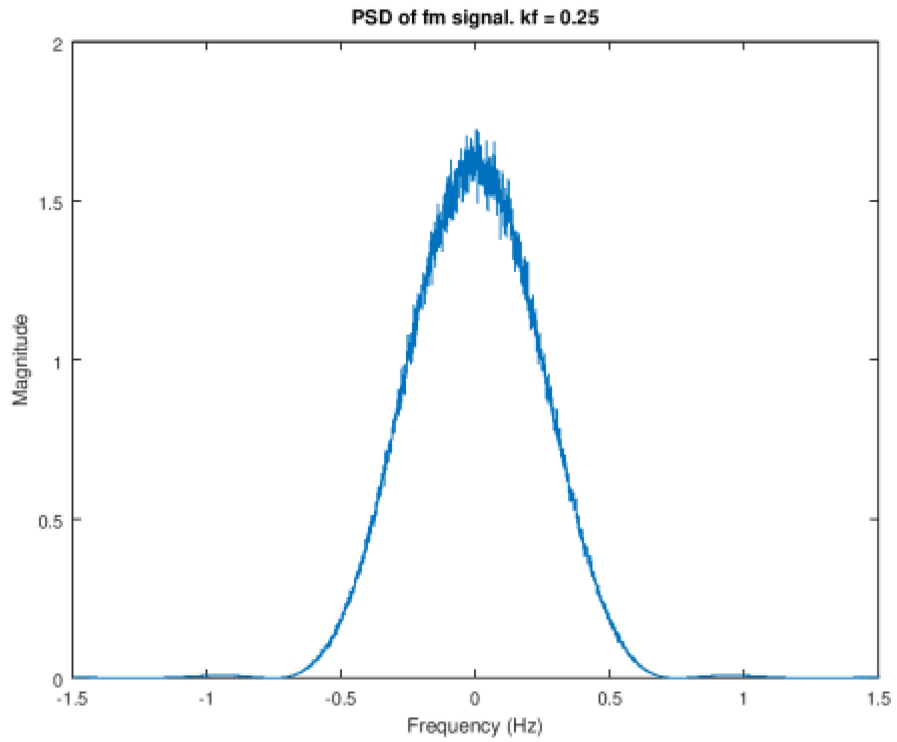

Fig. 7a – PSD of FM signal. Kf = 0.25

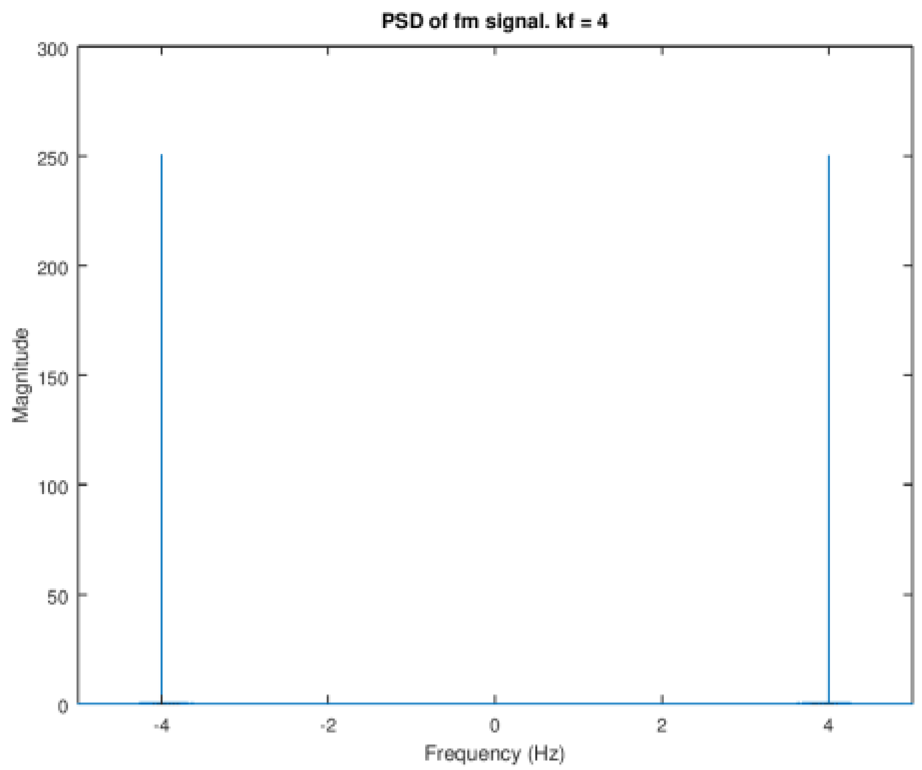

Fig. 7b – PSD of FM signal. Kf = 4

The peak starts splitting as K increases, at kf = 0.5, the peak is completely split.

Question 8

No, they are not.

As seen from plots in the previous question,
for k = 0.25, bandwidth is approximately 0.75 Hz
for k = 0.40, bandwidth is approximately 4.00 Hz

From Carson's formula,
for k = 0.25, bandwidth is approximately 2.50 Hz
for k = 0.40, bandwidth is approximately 10.0 Hz

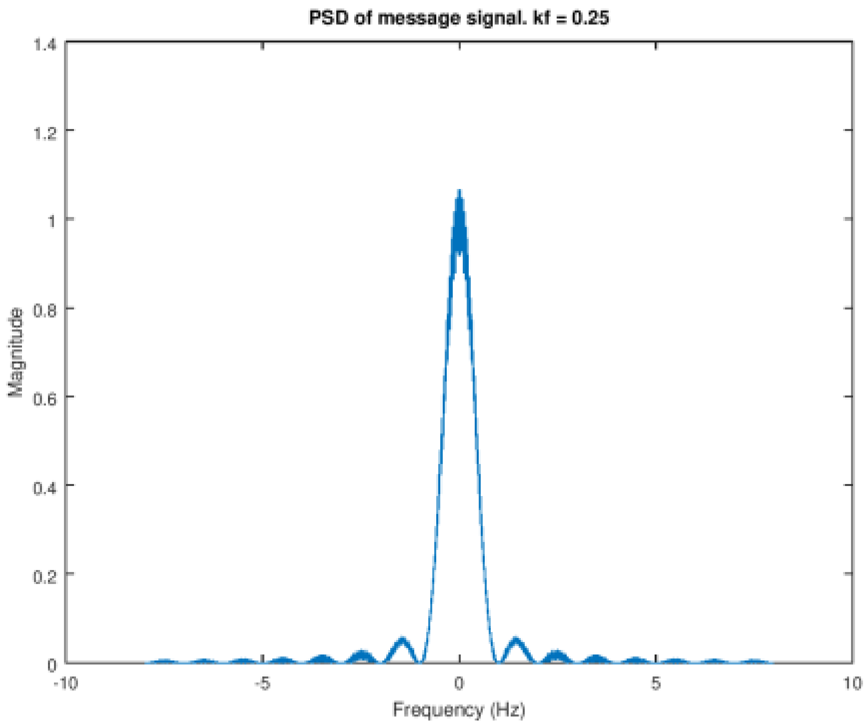

Fig. 8a – PSD of message signal, kf = 0.25 – bandwidth = 1 Hz

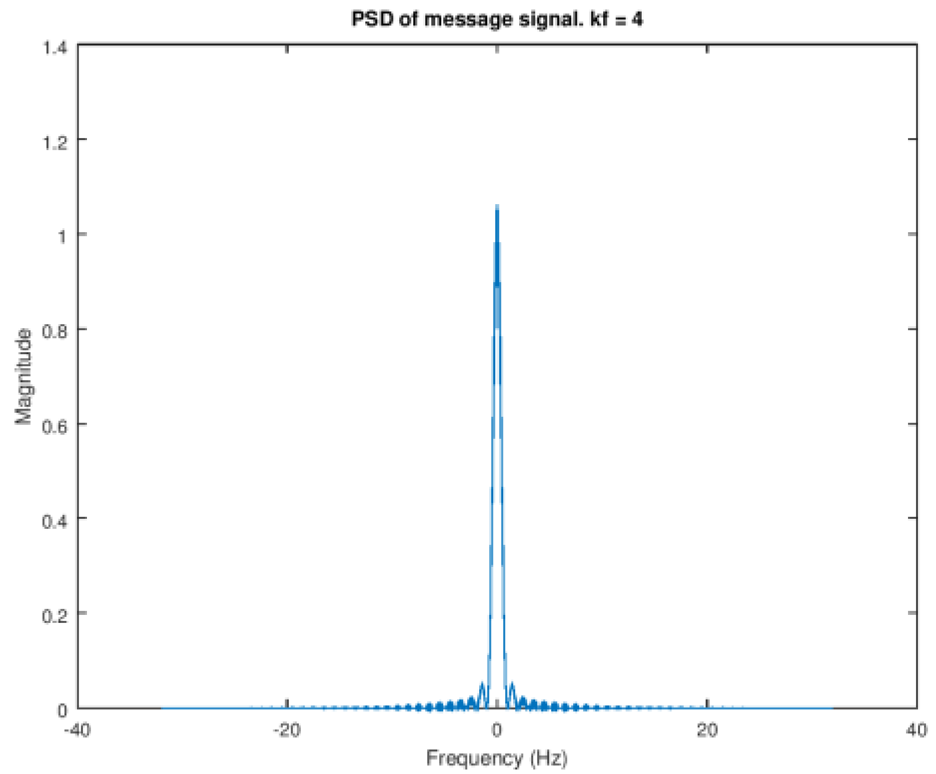

Fig. 8a – PSD of message signal, kf = 4 – bandwidth = 1 Hz

Bandwidth calculation via Carson's formula:
Message bandwidth B = 1 Hz
Beta = kf * B
Bandwidth = 2B(1 + Beta)

So, for Kf = 0.25, B = 1 Hz, Beta = 0.25 and Bandwidth = 2.5 Hz
And for Kf = 4, B = 1 Hz, Beta = 4 and Bandwidth = 10 Hz

Question 9

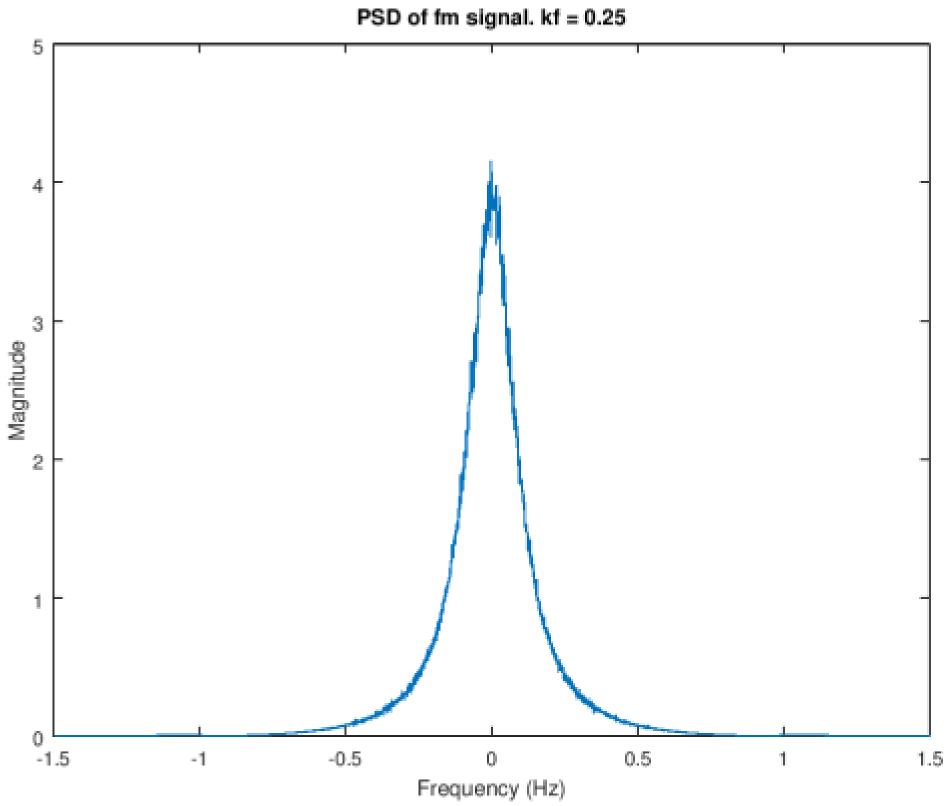

Fig. 9a – PSD of FM for kf = 0.25

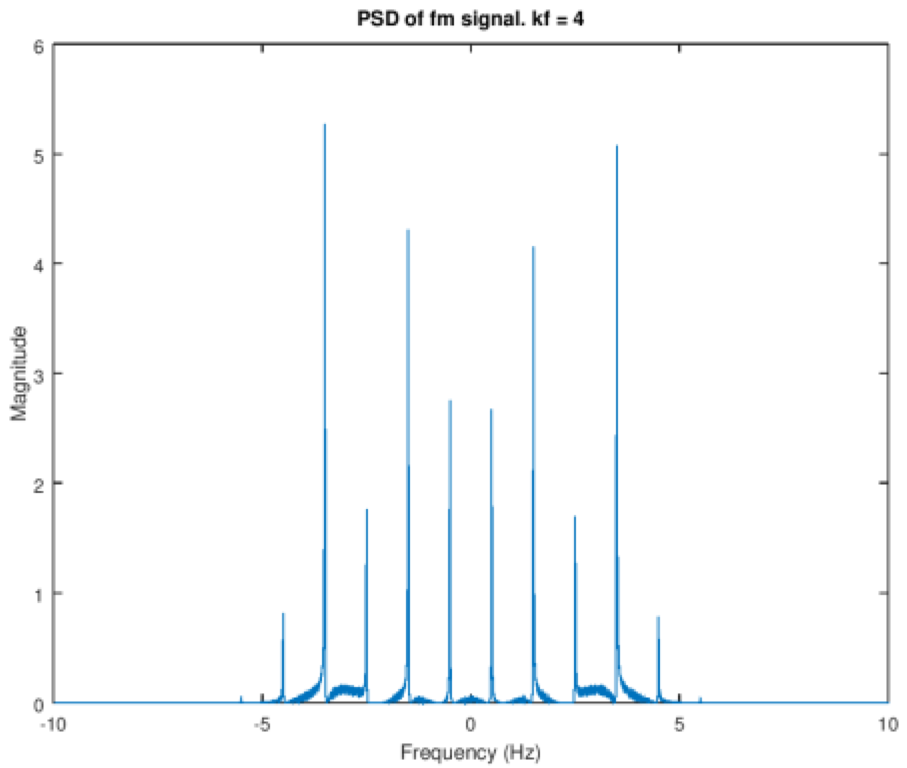

Fig. 9b – PSD of FM for Kf = 4

From these plots,
for k = 0.25, bandwidth is approximately 1 Hz
for k = 0.40, bandwidth is approximately 5 Hz

We know from previous calculations that message signals have bandwidths as follows
From Carson's formula,
for k = 0.25, bandwidth is approximately 2.50 Hz
for k = 0.40, bandwidth is approximately 10.0 Hz

So no, the results are again not in consistence with Carson's formula.

Question 10

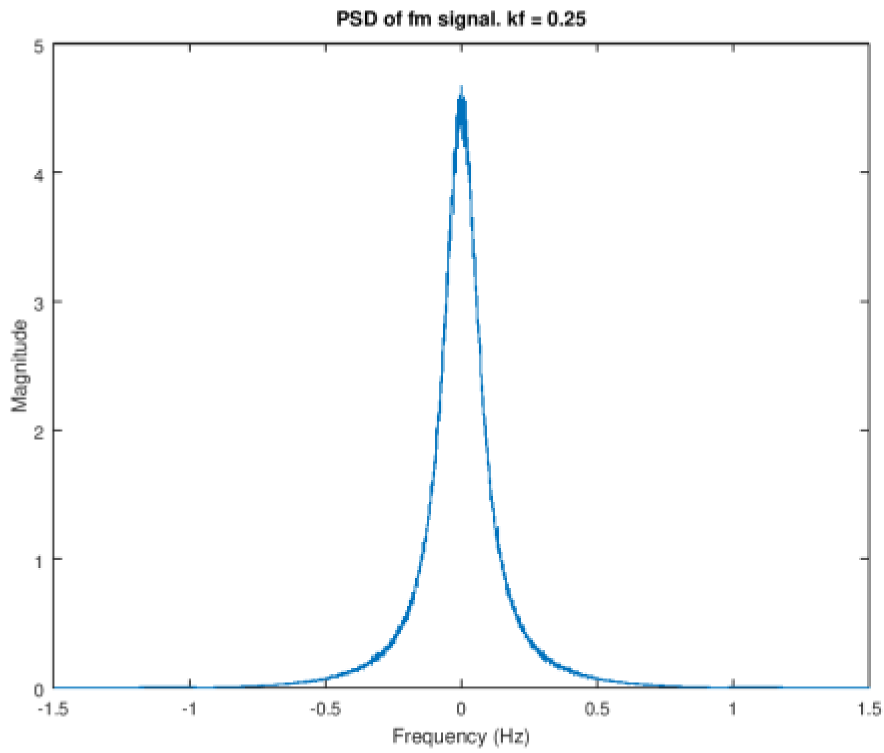

Fig. 10a – PSD of FM with message bits containing numbers drawn from a Gaussian distribution with the same variance, kf = 0.25. Bandwidth = 1 Hz

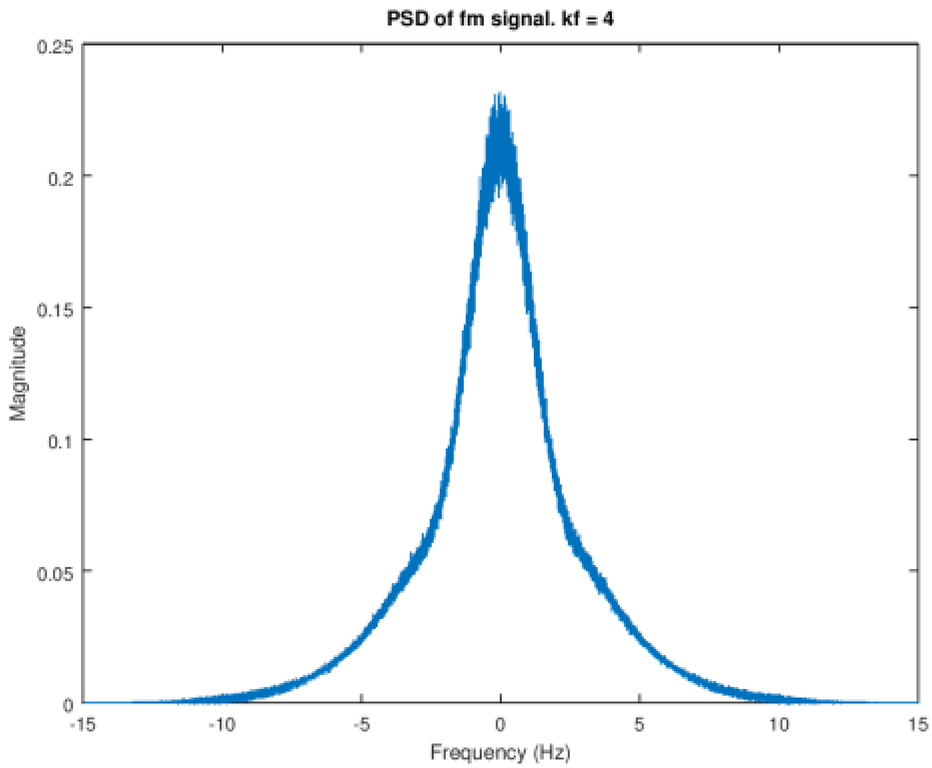

Fig. 10b - PSD of FM with message bits containing numbers drawn from a Gaussian distribution with the same variance, kf = 4. Bandwidth = 10 Hz

Spectral occupancy in kf = 0.25 is almost the same
Spectral occupancy in kf = 4 is strikingly different, however. In the previous case, the distribution was concentrated within 5 Hz, but here it spreads till 10 Hz but with major part within 5 Hz.

<u>Question 11</u>

The bandwidth increases by 1000 because the unit of time in divided by 1000. So the bandwidth is
for k = 0.25, bandwidth is approximately 1 KHz
for k = 0.40, bandwidth is approximately 5 KHz

Question 12

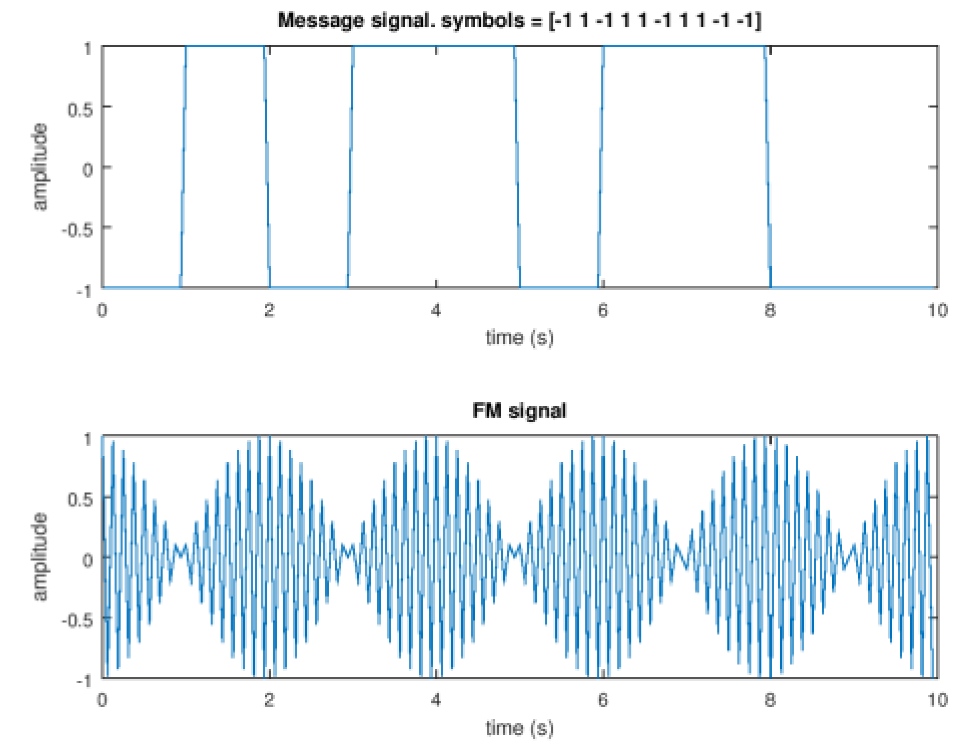

Fig. 12a – FM signal. Kf = 0.25

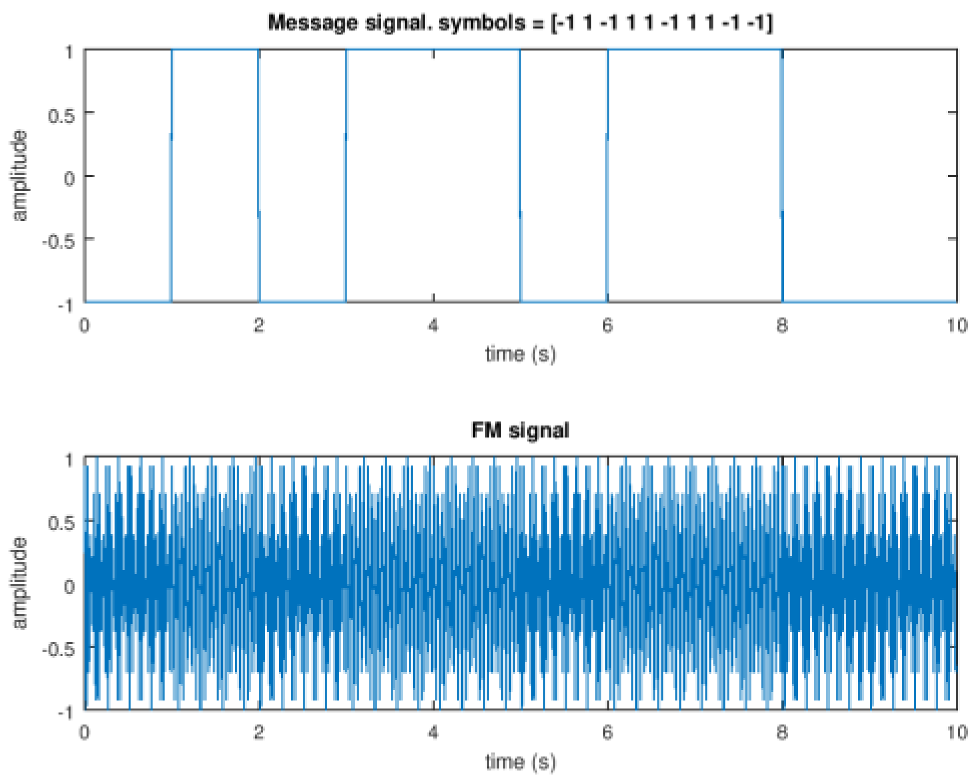

Fig. 12b – FM signal. Kf = 4

Question 13

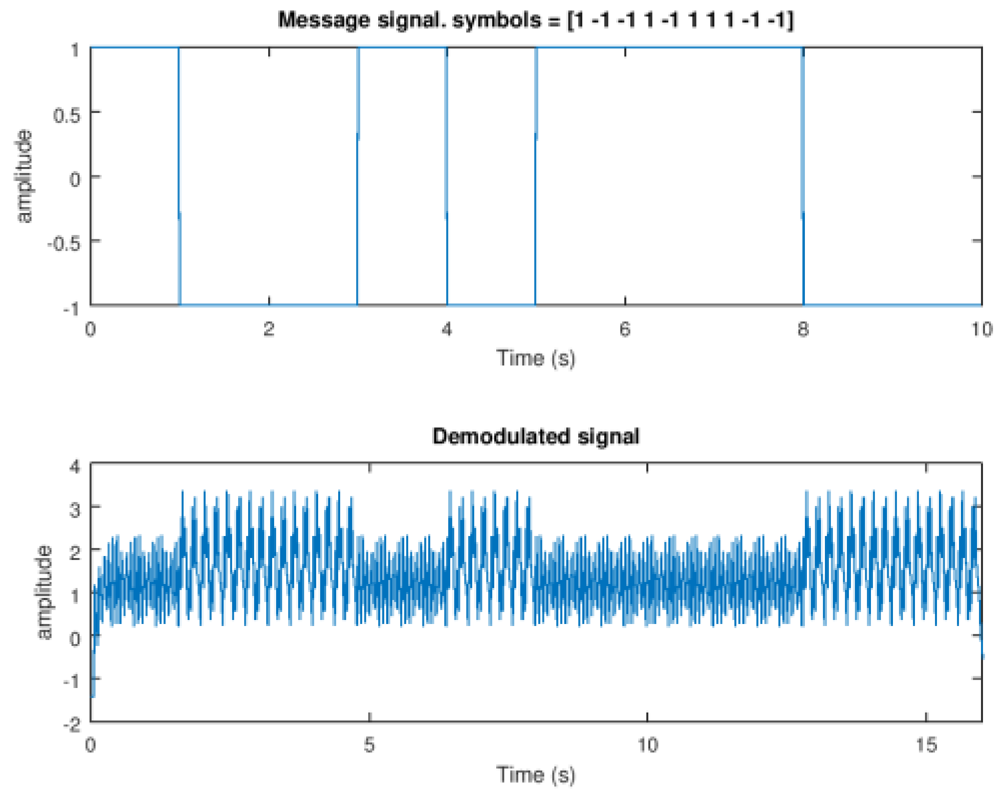

Fig. 13a – Demodulated FM signal

Question 14

Randn returns a matrix with normally distributed random elements having zero mean and variance one.
Randsrc generates a random number between +1 and -1.

Codes:

Note: You can also download all files from this link: https://goo.gl/mPKhZP

1.

```
oversampling_factor = 16;

%for a pulse with amplitude one, the max frequency deviation is given
by kf
kf=0.25;

%increase the oversampling factor if kf (and hence frequency deviation,
and hence bw of FM s
oversampling_factor = ceil(max(kf,1)*oversampling_factor);

ts=1/oversampling_factor;%sampling time

nsamples = ceil(1/ts);
pulse = ones(nsamples,1); %rectangular pulse

nsymbols =10;
symbols=zeros(nsymbols,1);

%random symbol sequence
symbols = sign(rand(nsymbols,1)-0.5);

symbols

%generate digitally modulated message
nsymbols_upsampled=1+(nsymbols-1)*nsamples;
symbols_upsampled=zeros(nsymbols_upsampled,1);
symbols_upsampled(1:nsamples:nsymbols_upsampled)=symbols;

message = conv(symbols_upsampled,pulse);
%FM signal phase obtained by integrating the message

theta = 2*pi*kf*ts*cumsum(message);
cenvelope=exp(j*theta);

L=length(cenvelope);
time=(0:L-1)*ts;

Icomponent = real(cenvelope);
Qcomponent= imag(cenvelope);

subplot(4, 1, 1);
%plot Message
plot(time, message);
title(["Message signal. symbols = ", mat2str(symbols)]);
xlabel("time (s)");
ylabel("amplitude");

subplot(4, 1, 2);
%plot I component
plot(time,Icomponent);
title("I component");
```

```
xlabel("time (s)");
ylabel("amplitude");

subplot(4, 1, 3);
%plot Q component
plot(time, Qcomponent);
title("Q component");
xlabel("time (s)");
ylabel("amplitude");

subplot(4, 1, 4);
%plot theta
plot(time, theta./pi);
title(["\\theta", "(t)/", "\\pi"]);
xlabel("time (s)");
ylabel("amplitude");

print -dpng 1.png
```

2.

```
oversampling_factor = 16;

%for a pulse with amplitude one, the max frequency deviation is given
by kf
kf=4;

%increase the oversampling factor if kf (and hence frequency deviation,
and hence bw of FM s
oversampling_factor = ceil(max(kf,1)*oversampling_factor);

ts=1/oversampling_factor;%sampling time

nsamples = ceil(1/ts);
pulse = ones(nsamples,1); %rectangular pulse

nsymbols =10;
symbols=zeros(nsymbols,1);

%random symbol sequence
symbols = sign(rand(nsymbols,1)-0.5);

symbols

%generate digitally modulated message
nsymbols_upsampled=1+(nsymbols-1)*nsamples;
symbols_upsampled=zeros(nsymbols_upsampled,1);
symbols_upsampled(1:nsamples:nsymbols_upsampled)=symbols;

message = conv(symbols_upsampled,pulse);
%FM signal phase obtained by integrating the message

theta = 2*pi*kf*ts*cumsum(message);
cenvelope=exp(j*theta);

L=length(cenvelope);
time=(0:L-1)*ts;
```

```
Icomponent = real(cenvelope);
Qcomponent= imag(cenvelope);

subplot(4, 1, 1);
%plot Message
plot(time, message);
title(["Message signal. symbols = ", mat2str(symbols)]);
xlabel("time (s)");
ylabel("amplitude");

subplot(4, 1, 2);
%plot I component
plot(time,Icomponent);
title("I component");
xlabel("time (s)");
ylabel("amplitude");

subplot(4, 1, 3);
%plot Q component
plot(time, Qcomponent);
title("Q component");
xlabel("time (s)");
ylabel("amplitude");

subplot(4, 1, 4);
%plot theta
plot(time, theta./pi);
title(["\\theta", "(t)/", "\\pi"]);
xlabel("time (s)");
ylabel("amplitude");

print -dpng 2.png
```

4.

```
oversampling_factor = 16;

%for a pulse with amplitude one, the max frequency deviation is given
by kf
kf=0.25;

%increase the oversampling factor if kf (and hence frequency deviation,
and hence bw of FM s
oversampling_factor = ceil(max(kf,1)*oversampling_factor);

ts=1/oversampling_factor;%sampling time

nsamples = ceil(1/ts);
pulse = ones(nsamples,1); %rectangular pulse

nsymbols =10;
symbols=zeros(nsymbols,1);

%random symbol sequence
symbols = sign(rand(nsymbols,1)-0.5);
```

```
symbols

%generate digitally modulated message
nsymbols_upsampled=1+(nsymbols-1)*nsamples;
symbols_upsampled=zeros(nsymbols_upsampled,1);
symbols_upsampled(1:nsamples:nsymbols_upsampled)=symbols;

message = conv(symbols_upsampled,pulse);
%FM signal phase obtained by integrating the message

theta = 2*pi*kf*ts*cumsum(message);
cenvelope=exp(j*theta);

L=length(cenvelope);
time=(0:L-1)*ts;

Icomponent = real(cenvelope);
Qcomponent= imag(cenvelope);

% subplot(4, 1, 1);
% %plot Message
% plot(time, message);
% title(["Message signal. symbols = ", mat2str(symbols)]);
% xlabel("time (s)");
% ylabel("amplitude");

% subplot(4, 1, 2);
% %plot I component
% plot(time,Icomponent);
% title("I component");
% xlabel("time (s)");
% ylabel("amplitude");

% subplot(4, 1, 3);
% %plot Q component
% plot(time, Qcomponent);
% title("Q component");
% xlabel("time (s)");
% ylabel("amplitude");

% subplot(4, 1, 4);
% %plot theta
% plot(time, theta./pi);
% title(["\\theta", "(t)/", "\\pi"]);
% xlabel("time (s)");
% ylabel("amplitude");

% print -dpng 1.png


%baseband discriminator
%differencing operation approximates derivative
Iderivative = [0;diff(Icomponent)]/ts;
Qderivative = [0;diff(Qcomponent)]/ts;
message_estimate = (1/(2*pi*kf))*(Icomponent.*Qderivative -
Qcomponent.*Iderivative)./(Icomponent.^2 .+ Qcomponent.^2);

subplot(2, 1, 1);
```

```
plot(time, message);
title(["Message signal. symbols = ", mat2str(symbols)]);
xlabel("time (s)");
ylabel("amplitude");

subplot(2, 1, 2);
plot(time, message);
title("Estimated message. kf = 0.25");
xlabel("time (s)");
ylabel("amplitude");

print -dpng 4a.png
```

5.

```
oversampling_factor = 16;

%for a pulse with amplitude one, the max frequency deviation is given
by kf
kf=0.25;

%increase the oversampling factor if kf (and hence frequency deviation,
and hence bw of FM s
oversampling_factor = ceil(max(kf,1)*oversampling_factor);

ts=1/oversampling_factor;%sampling time

nsamples = ceil(1/ts);
pulse = ones(nsamples,1); %rectangular pulse

nsymbols =10;
symbols=zeros(nsymbols,1);

%random symbol sequence
symbols = sign(rand(nsymbols,1)-0.5);

symbols

%generate digitally modulated message
nsymbols_upsampled=1+(nsymbols-1)*nsamples;
symbols_upsampled=zeros(nsymbols_upsampled,1);
symbols_upsampled(1:nsamples:nsymbols_upsampled)=symbols;

message = conv(symbols_upsampled,pulse);
%FM signal phase obtained by integrating the message

theta = 2*pi*kf*ts*cumsum(message);
cenvelope=exp(j*theta);

phi = 2*pi*rand; %phase uniform over [0,2pi]
phi = 0
cenvelope = cenvelope.*exp(j*phi); % adding random phase to theta
% e^(x)*e^(y) = e^(x+y)
%now apply baseband discriminator

L=length(cenvelope);
time=(0:L-1)*ts;
```

```
Icomponent = real(cenvelope);
Qcomponent= imag(cenvelope);


%baseband discriminator
%differencing operation approximates derivative
Iderivative = [0;diff(Icomponent)]/ts;
Qderivative = [0;diff(Qcomponent)]/ts;
message_estimate = (1/(2*pi*kf))*(Icomponent.*Qderivative -
Qcomponent.*Iderivative)./(Icomponent.^2 .+ Qcomponent.^2);

subplot(2, 1, 1);
plot(time, message);
title(["Message signal. symbols = ", mat2str(symbols)]);
xlabel("time (s)");
ylabel("amplitude");

subplot(2, 1, 2);
plot(time, message);
title("Estimated message. kf = 0.25");
xlabel("time (s)");
ylabel("amplitude");

print -dpng 5a.png
```

6.

```
oversampling_factor = 16;

%for a pulse with amplitude one, the max frequency deviation is given
by kf
kf=0.25;

%increase the oversampling factor if kf (and hence frequency deviation,
and hence bw of FM s
oversampling_factor = ceil(max(kf,1)*oversampling_factor);

ts=1/oversampling_factor;%sampling time

nsamples = ceil(1/ts);
pulse = ones(nsamples,1); %rectangular pulse

nsymbols =10;
symbols=zeros(nsymbols,1);

%random symbol sequence
symbols = sign(rand(nsymbols,1)-0.5);

symbols

%generate digitally modulated message
nsymbols_upsampled=1+(nsymbols-1)*nsamples;
symbols_upsampled=zeros(nsymbols_upsampled,1);
symbols_upsampled(1:nsamples:nsymbols_upsampled)=symbols;

message = conv(symbols_upsampled,pulse);
%FM signal phase obtained by integrating the message
```

```
theta = 2*pi*kf*ts*cumsum(message);
cenvelope=exp(j*theta);

L=length(cenvelope);
time=(0:L-1)*ts;

phi = 2*pi*rand; %phase uniform over [0,2 pi]
df = 0.3;
cenvelope = cenvelope.*exp(j*(2*pi*df*time'+phi));
% adding random phase and frequency offset to theta
% e^(x)*e^(y) = e^(x+y)
%now apply baseband discriminator



Icomponent = real(cenvelope);
Qcomponent= imag(cenvelope);


%baseband discriminator
%differencing operation approximates derivative
Iderivative = [0;diff(Icomponent)]/ts;
Qderivative = [0;diff(Qcomponent)]/ts;
message_estimate = (1/(2*pi*kf))*(Icomponent.*Qderivative -
Qcomponent.*Iderivative)./(Icomponent.^2 .+ Qcomponent.^2);

subplot(2, 1, 1);
plot(time, message);
title(["Message signal. symbols = ", mat2str(symbols)]);
xlabel("time (s)");
ylabel("amplitude");

subplot(2, 1, 2);
plot(time, message_estimate);
title("Estimated message. kf = 0.25");
xlabel("time (s)");
ylabel("amplitude");

print -dpng 6a.png
```

7.

```
oversampling_factor = 16;

%for a pulse with amplitude one, the max frequency deviation is given
by kf
kf=0.25;

%increase the oversampling factor if kf (and hence frequency deviation,
and hence bw of FM s
oversampling_factor = ceil(max(kf,1)*oversampling_factor);

ts=1/oversampling_factor; %sampling time

nsamples = ceil(1/ts);
pulse = ones(nsamples,1); %rectangular pulse
% pulse_time = 0:ts:1;
```

```
% pulse = sin(2*pi*pulse_time);

% calculating PSD

nsymbols =1000;
symbols=zeros(nsymbols,1);
nruns=1000;
fs_desired=0.1;
Nmin = ceil(1/(fs_desired*ts)); %minimum length DFT for desired
frequency granularity
message_length=1+(nsymbols-1)*nsamples+length(pulse)-1;
Nmin = max(message_length,Nmin);
% %for efficient computation, choose FFT size to be power of 2
Nfft = 2^(nextpow2(Nmin)) %FFT size = the next power of 2 at least as
big as Nmin
psd=zeros(Nfft,1);

for runs=1:nruns,
      %random symbol sequence
      symbols = sign(rand(nsymbols,1)-0.5);
      nsymbols_upsampled = 1+(nsymbols-1)*nsamples;
      symbols_upsampled = zeros(nsymbols_upsampled,1);
      symbols_upsampled(1:nsamples:nsymbols_upsampled) = symbols;
      message = conv(symbols_upsampled,pulse);
      %FM signal phase
      theta = 2*pi*kf*ts*cumsum(message);
      cenvelope = exp(j*theta);
      time = (0:length(cenvelope)-1)*ts;
      % %freq domain signal computed using DFT
      cenvelope_freq = ts*fft(cenvelope,Nfft); %FFT of size Nfft,
automatically zeropads as needed
      cenvelope_freq_centered = fftshift(cenvelope_freq); %shifts DC to
center of spectrum
      psd=psd+abs(cenvelope_freq_centered).^2;
end

psd=psd/(nruns*nsymbols);
fs=1/(Nfft*ts) %actual frequency resolution attained
% %set of frequencies for which Fourier transform has been computed
using DFT
freqs = ((1:Nfft)-1-Nfft/2)*fs;
%plot the PSD
plot(freqs,psd);
title(["PSD of fm signal. kf = ", num2str(kf)]);
ylabel("Magnitude");
xlabel("Frequency (Hz)");
xlim([-1.5, 1.5]);
print -dpng 7a.png
```

8.

```
oversampling_factor = 16;

%for a pulse with amplitude one, the max frequency deviation is given
by kf
kf=0.25;
```

```
%increase the oversampling factor if kf (and hence frequency deviation,
and hence bw of FM s
oversampling_factor = ceil(max(kf,1)*oversampling_factor);

ts=1/oversampling_factor; %sampling time

nsamples = ceil(1/ts);
pulse = ones(nsamples,1); %rectangular pulse

% calculating PSD

nsymbols =1000;
symbols=zeros(nsymbols,1);
nruns=1000;
fs_desired=0.1;
Nmin = ceil(1/(fs_desired*ts)); %minimum length DFT for desired
frequency granularity
message_length=1+(nsymbols-1)*nsamples+length(pulse)-1;
Nmin = max(message_length,Nmin);
% %for efficient computation, choose FFT size to be power of 2
Nfft = 2^(nextpow2(Nmin)) %FFT size = the next power of 2 at least as
big as Nmin
psd=zeros(Nfft,1);

for runs=1:nruns,
      %random symbol sequence
      symbols = sign(rand(nsymbols,1)-0.5);
      nsymbols_upsampled = 1+(nsymbols-1)*nsamples;
      symbols_upsampled = zeros(nsymbols_upsampled,1);
      symbols_upsampled(1:nsamples:nsymbols_upsampled) = symbols;
      message = conv(symbols_upsampled,pulse);
      %FM signal phase
      % theta = 2*pi*kf*ts*cumsum(message);
      % cenvelope = exp(j*theta);
      % time = (0:length(cenvelope)-1)*ts;
      % %freq domain signal computed using DFT
      message_freq = ts*fft(message,Nfft); %FFT of size Nfft,
automatically zeropads as needed
      message_freq_centered = fftshift(message_freq); %shifts DC to
center of spectrum
      psd=psd+abs(message_freq_centered).^2;
end

psd=psd/(nruns*nsymbols);
fs=1/(Nfft*ts) %actual frequency resolution attained
% %set of frequencies for which Fourier transform has been computed
using DFT
freqs = ((1:Nfft)-1-Nfft/2)*fs;
%plot the PSD
plot(freqs,psd);
title(["PSD of message signal. kf = ", num2str(kf)]);
ylabel("Magnitude");
xlabel("Frequency (Hz)");
% xlim([-1.5, 1.5]);
print -dpng 8a.png
```

9.

```
oversampling_factor = 16;

%for a pulse with amplitude one, the max frequency deviation is given
by kf
kf=0.25;

%increase the oversampling factor if kf (and hence frequency deviation,
and hence bw of FM s
oversampling_factor = ceil(max(kf,1)*oversampling_factor);

ts=1/oversampling_factor; %sampling time

nsamples = ceil(1/ts);
% pulse = ones(nsamples,1); %rectangular pulse
pulse_time = 0:ts:1;
pulse = sin(pi*pulse_time);

% calculating PSD

nsymbols =1000;
symbols=zeros(nsymbols,1);
nruns=1000;
fs_desired=0.1;
Nmin = ceil(1/(fs_desired*ts)); %minimum length DFT for desired
frequency granularity
message_length=1+(nsymbols-1)*nsamples+length(pulse)-1;
Nmin = max(message_length,Nmin);
% %for efficient computation, choose FFT size to be power of 2
Nfft = 2^(nextpow2(Nmin)) %FFT size = the next power of 2 at least as
big as Nmin
psd=zeros(Nfft,1);

for runs=1:nruns,
      %random symbol sequence
      symbols = sign(rand(nsymbols,1)-0.5);
      nsymbols_upsampled = 1+(nsymbols-1)*nsamples;
      symbols_upsampled = zeros(nsymbols_upsampled,1);
      symbols_upsampled(1:nsamples:nsymbols_upsampled) = symbols;
      message = conv(symbols_upsampled,pulse);
      %FM signal phase
      theta = 2*pi*kf*ts*cumsum(message);
      cenvelope = exp(j*theta);
      time = (0:length(cenvelope)-1)*ts;
      % %freq domain signal computed using DFT
      cenvelope_freq = ts*fft(cenvelope,Nfft); %FFT of size Nfft,
automatically zeropads as needed
      cenvelope_freq_centered = fftshift(cenvelope_freq); %shifts DC to
center of spectrum
      psd=psd+abs(cenvelope_freq_centered).^2;
end

psd=psd/(nruns*nsymbols);
fs=1/(Nfft*ts) %actual frequency resolution attained
% %set of frequencies for which Fourier transform has been computed
using DFT
freqs = ((1:Nfft)-1-Nfft/2)*fs;
```

```
%plot the PSD
plot(freqs,psd);
title(["PSD of fm signal. kf = ", num2str(kf)]);
ylabel("Magnitude");
xlabel("Frequency (Hz)");
xlim([-1.5, 1.5]);
print -dpng 9a.png
```

10.

```
oversampling_factor = 16;

%for a pulse with amplitude one, the max frequency deviation is given
by kf
kf=0.25;

%increase the oversampling factor if kf (and hence frequency deviation,
and hence bw of FM s
oversampling_factor = ceil(max(kf,1)*oversampling_factor);

ts=1/oversampling_factor; %sampling time

nsamples = ceil(1/ts);
% pulse = ones(nsamples,1); %rectangular pulse
pulse_time = 0:ts:1;
pulse = sin(pi*pulse_time);

% calculating PSD

nsymbols =1000;
symbols=zeros(nsymbols,1);
nruns=1000;
fs_desired=0.1;
Nmin = ceil(1/(fs_desired*ts)); %minimum length DFT for desired
frequency granularity
message_length=1+(nsymbols-1)*nsamples+length(pulse)-1;
Nmin = max(message_length,Nmin);
% %for efficient computation, choose FFT size to be power of 2
Nfft = 2^(nextpow2(Nmin)) %FFT size = the next power of 2 at least as
big as Nmin
psd=zeros(Nfft,1);

for runs=1:nruns,
      %random symbol sequence
      % symbols = sign(rand(nsymbols,1)-0.5);
      symbols = randn(nsymbols,1);
      nsymbols_upsampled = 1+(nsymbols-1)*nsamples;
      symbols_upsampled = zeros(nsymbols_upsampled,1);
      symbols_upsampled(1:nsamples:nsymbols_upsampled) = symbols;
      message = conv(symbols_upsampled,pulse);
      %FM signal phase
      theta = 2*pi*kf*ts*cumsum(message);
      cenvelope = exp(j*theta);
      time = (0:length(cenvelope)-1)*ts;
      % %freq domain signal computed using DFT
```

```
      cenvelope_freq = ts*fft(cenvelope,Nfft); %FFT of size Nfft,
automatically zeropads as needed
      cenvelope_freq_centered = fftshift(cenvelope_freq); %shifts DC to
center of spectrum
      psd=psd+abs(cenvelope_freq_centered).^2;
end

psd=psd/(nruns*nsymbols);
fs=1/(Nfft*ts) %actual frequency resolution attained
% %set of frequencies for which Fourier transform has been computed
using DFT
freqs = ((1:Nfft)-1-Nfft/2)*fs;
%plot the PSD
plot(freqs,psd);
title(["PSD of fm signal. kf = ", num2str(kf)]);
ylabel("Magnitude");
xlabel("Frequency (Hz)");
xlim([-1.5, 1.5]);
print -dpng 7a.png
```

12.

```
oversampling_factor = 16;

%for a pulse with amplitude one, the max frequency deviation is given
by kf
kf=0.25;

%increase the oversampling factor if kf (and hence frequency deviation,
and hence bw of FM s
oversampling_factor = ceil(max(kf,1)*oversampling_factor);

ts=1/oversampling_factor;%sampling time

nsamples = ceil(1/ts);
pulse = ones(nsamples,1); %rectangular pulse

nsymbols =10;
symbols=zeros(nsymbols,1);

%random symbol sequence
% symbols = sign(rand(nsymbols,1)-0.5);

% symbols
symbols = [-1, 1, -1, 1, 1, -1, 1, 1, -1, -1]
%generate digitally modulated message
nsymbols_upsampled=1+(nsymbols-1)*nsamples;
symbols_upsampled=zeros(nsymbols_upsampled,1);
symbols_upsampled(1:nsamples:nsymbols_upsampled)=symbols;

message = conv(symbols_upsampled,pulse);
%FM signal phase obtained by integrating the message

theta = 2*pi*kf*ts*cumsum(message);
cenvelope=exp(j*theta);
```

```
L=length(cenvelope);
time=(0:L-1)*ts;

Fc = 1000;
% cos(2pi*Fc*t + theta(t))
FM = cos(2*pi*Fc*time.+theta');

subplot(2, 1, 1);
plot(time, message);
title(["Message signal. symbols = ", mat2str(symbols)]);
xlabel("time (s)");
ylabel("amplitude");

subplot(2, 1, 2);
plot(time, FM);
title("FM signal");
xlabel("time (s)");
ylabel("amplitude");

print -dpng 12a.png
```

13.

```
oversampling_factor = 16;

%for a pulse with amplitude one, the max frequency deviation is given
by kf
kf=4;

%increase the oversampling factor if kf (and hence frequency deviation,
and hence bw of FM s
oversampling_factor = ceil(max(kf,1)*oversampling_factor);

ts=1/oversampling_factor;%sampling time

nsamples = ceil(1/ts);
pulse = ones(nsamples,1); %rectangular pulse

nsymbols =10;
symbols=zeros(nsymbols,1);

%random symbol sequence
% symbols = sign(rand(nsymbols,1)-0.5);

% symbols
symbols = [1, -1, -1, 1, -1, 1, 1, 1, -1, -1]
%generate digitally modulated message
nsymbols_upsampled=1+(nsymbols-1)*nsamples;
symbols_upsampled=zeros(nsymbols_upsampled,1);
symbols_upsampled(1:nsamples:nsymbols_upsampled)=symbols;

message = conv(symbols_upsampled,pulse);
%FM signal phase obtained by integrating the message

theta = 2*pi*kf*ts*cumsum(message);
cenvelope=exp(j*theta);
```

```
L=length(cenvelope);
time=(0:L-1)*ts;

Fc = 1000;
% cos(2pi*Fc*t + theta(t))
FM = cos(2*pi*Fc*time.+theta');

% -------- transmission ----------- %
% passing FM through differentiator
FM_diff = [0;diff(FM')]/ts;

% diode filter - retaining only positive signal
FM_diff_diode = diodeFilter(FM_diff');

% Envelope detector - RC filter

[time_env, fm_env] = RCfilter(time, FM_diff_diode, 0.04);

[time_dcblock, fm_dcblock] = DCblock(time_env, fm_env);

% fm_dcblock = fm_env;
% time_dcblock = time_env;

subplot(2, 1, 1);
plot(time, message);
title(["Message signal. symbols = ", mat2str(symbols)]);
xlabel("Time (s)");
ylabel("amplitude");


subplot(2, 1, 2);
plot(time_dcblock, fm_dcblock);
title("Demodulated signal");
xlim([0, 16]);
xlabel("Time (s)");
ylabel("amplitude");

print -dpng 13a.png
```

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
```