Lab 1 - Simulation of Communication System with ECG Signal Transmission

Object:

1. Enhance the understanding of communication theory, especially the modulation schemes (such as analog modulation – AM, and digital modulation – OOK (On/Off keying));
2. Learn how to use Matlab to build signal modulation systems;
3. Learn the transmission of medical signals (ECG) through AM/OOK.

Note: <u>All lab questions (18 of them) are embedded in the Matlab codes.</u>

Background knowledge: ECG signals:

ECG (electrocardiogram) signals are detected by the electrodes which are attached on human body near heart area and the limbs.

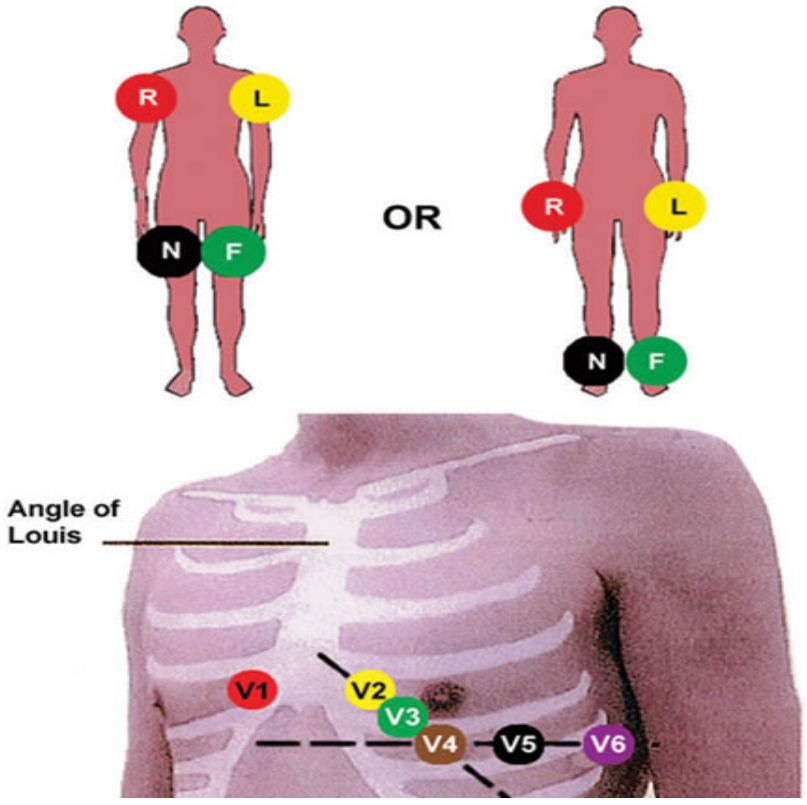


Figure 1. ECG signal measurement locations

There are different leads and positions to settle the electrodes. Different leads generate different ECG signal waveforms. Through the delineation and analysis of the ECG signals the

monitor system and doctor are able to detect the abnormal status of the heart and thus diagnose the disease.

Figure 2 shows the ECG signal waveforms from different leads. The patient has a right bundle branch block at this time.

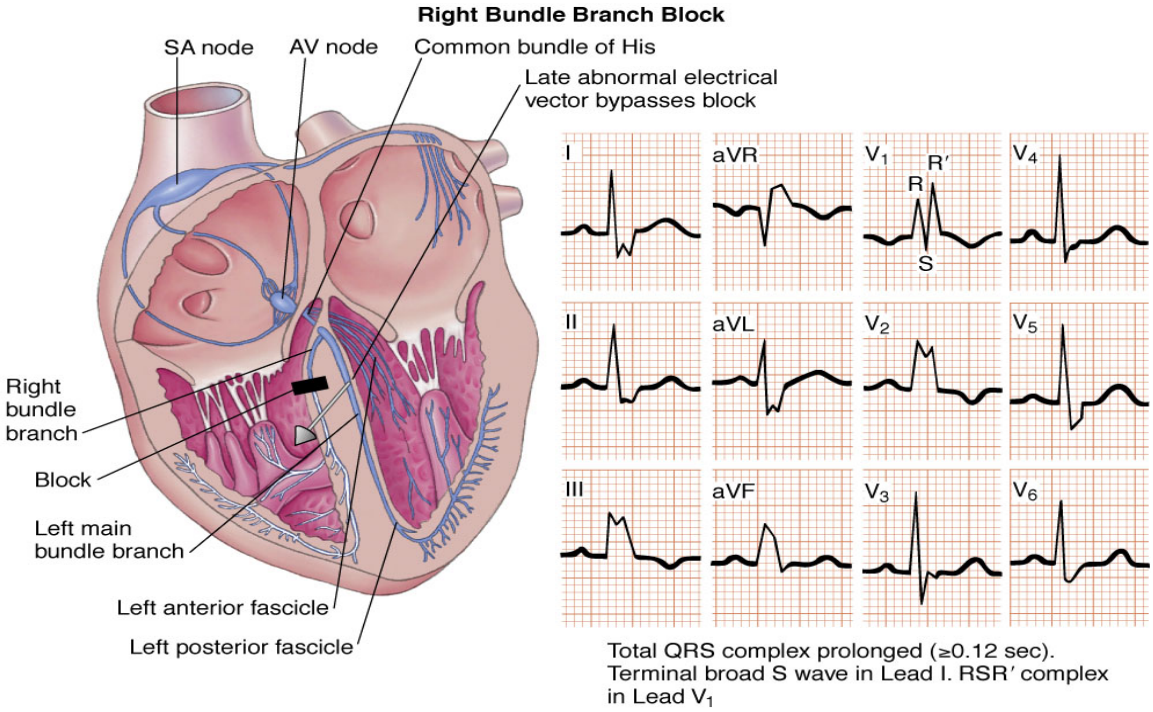


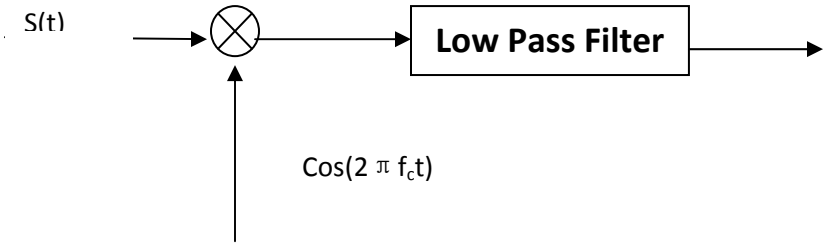
Figure 2. ECG signal waveforms

1. Analog communication

At first we build an analog communication system to transmit the ECG signal.

For AM modulation system, if the signal is m(t), the modulation signal is :

$$s(t)=[A+m(t)]\cos(2\pi f_c t)$$



Matlab simulation is as follows:

a. Load ECG signal

```
% save the data (signals.mat) in your working directory.
% sig1 contains the sample times in seconds.
% sig2, sig3, sig4, sig5 and sig6 contain the ECG data (leads i, ii, iii, iv and vi).

load signals.mat;
```

Question 1: Print out one-page of ECG data from signals.mat.

Question 2: Use EXCEL (copy data from signals.mat to EXCEL data sheet) to display two leads of ECG signal curves.

b. Setup basic parameters for system

```
num_points = 2000; % the number of symbols (the maximum is 38399)

t=sig1(1:num_points); % time in second

dt=t(2)-t(1); % symbol period

mt = sig2(1:num_points)'; % take sig2 for example

T=t(end); % signal duration

t=0:dt:T;

fm=60; % the highest frequency

fc=2*fm; % the carrier frequency
```

c. Modulation

```
%%%%%%%%%%%%%%%%%%%%%%
%%% AM modulation %%%
%%%%%%%%%%%%%%%%%%%%%%
A=2; % the carrier amplitute

s_am = …… ; % modulating
```

Question 3: Please finish the above Matlab code.

Hint: refer to AM principle: $s(t)=[A+m(t)]\cos(2\pi f_c t)$

d. Demodulation

```
%%%%%%%%%%%%%%%%%%%%%%
%%% AM demodulation %%%
%%%%%%%%%%%%%%%%%%%%%%
```

```
Rt = … … ; % demodulating
```

Question 4: Please finish the above Matlab code.

Hint: refer to AM demodulation principle: $r(t)=s(t)\cos(2\pi f_c t)$

```
[f,rf]=T2F(t,rt); % the Fourier Transform

[t,rt] = lpf(f,rf,fm);
```

Question 5: Why do we use the above function?

```
rt = 2*rt - A; % amplitute adjustment
```

e. Results:

```
figure(1)
subplot(211);
plot(t,mt);
title('ECG signal');
xlabel('t');

subplot(212);

…  …. ; % the Fourier Transform
```

Question 6: Please add one code here to call the Fourier Transform function in order to obtain the frequency domain waveform of ECG signal **mt** (one of the leads).

```
pds = 10*log10(abs(sf).^2/T); % the power density spectrum
plot(f, pds);
title('ECG signal PDS');
xlabel('f');


figure(2)
subplot(311)
```

```
plot(t,s_am);hold on;
plot(t, A+mt,'r--');
title('AM modulation signal');
xlabel('t');

subplot(312)
plot(t,rt);hold on;
plot (t,mt,'r--');
title('Demodulated signal')
xlabel('t');

subplot(313)

… … ; % the Fourier Transform
```

Question 7: Please add one code here to call the Fourier Transform function in order to obtain the frequency domain waveform of modulated AM signal s_am.

```
pds=(abs(sf).^2)/T; % the power density spectrum
plot(f,pds);
axis([-2*fc 2*fc 0 max(pds)]);
title('AM PDS');
xlabel('t');
```

Question 8: Please show the figures of ECG signal and its PDS here.

Question 9: Please show the figure of the modulated signal here (S_am). Explain how the AM works based on such a figure.

Question 10: Please show the figure of the demodulated signal here (rt). Explain how the demodulation recovers original signals.

2. Digital Communication

After we finish the above analog communication case, let's simulate the OOK (On/Off Keying) modulation case in digital communication system. As we know, digital system needs to do more things than analog one, such as signal sampling, quantization, encoding, digital waveform generating, digital modulation (here we use OOK), etc.

Matlab simulation:

a) Sampling:
Nyquist theory (fs>=2fh)

```
%%%%%%%%%%%%%%%%%
%%% sampling %%%
%%%%%%%%%%%%%%%%%

fs= 100; % sampling rate

dts=1/fs; % sampling period

ns=dts/dt; % sampled every ns symbols, where dt is the symbol period

ts=1:ns:n; % time index

sigs=sig(ts); % sample signal
```

b) PCM encoding: we use an uniform PCM quantization encoder function here. (Please google PCM details).

Question 11: Explain PCM quantization principle. (use one example to illustrate it).

```
%%%%%%%%%%%%%%%%%%%%
%%% pcm encoding %%%
%%%%%%%%%%%%%%%%%%%%

num_bits=8; % number of bits per sample

bits = PCM_encoder(sigs,num_bits);
```

c) NRZ waveform (again, please google *non-return-to-zero* (*NRZ*) line code).

```
%%%%%%%%%%%%%%%%%%%%
%%% NRZ waveform %%%
%%%%%%%%%%%%%%%%%%%%
```

Question 12: Explain NRZ principle. (use one example to illustrate it).

```
dd=sigexpand(bits,fc*N_sample);
```

Question 13: What does this code do? Check the appended sigexpand function.

```
gt=ones(1,fc*N_sample); % rectangular window

d_NRZ = conv(dd, gt); % generating NRZ waveform
```

d) Modulation

```
%%%%%%%%%%%
%%% OOK %%%
%%%%%%%%%%%

ht = A*cos(1*pi*fc*tb); % A is the amplitude of the carrier


s_2ask = d_NRZ(1:Lt).*ht;
```

Question 14: What does the above code do?

e) Results:

```
figure(1)
subplot(221);
plot(tb,d_NRZ(1:Lt));
axis([0 100 -0.2 1.2]);ylabel('binary data');

subplot(222);
[f, d_NRZf] = T2F(tb, d_NRZ(1:length(tb))); % Fourier Transform
plot(f, 10*log10(abs(d_NRZf).^2/Ts));
axis([-2 2 -30 50]);ylabel('dB/Hz');

subplot(223);
plot(tb,s_2ask);
axis([0 13 -1.2 1.2]);ylabel('OOK');

[f, s_2ask] = T2F(tb, s_2ask); % Fourier Transform
subplot(224);
plot(f, 10*log10(abs(s_2ask).^2/Ts));
axis([-fc-4 fc+4 -50 50]);ylabel('dB/Hz');
```

Question 15: Please provide your results here.

3. Function used

Fourier Transform and Inverse Fourier Transform

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
%% Complete the Fourier Transform by FFT
%  Input: t - time index
%         st - signals in the time domain
%  Output: f - frequency index
%         sf - signals in the frequency domain
%             i.e., the signal spectrum
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%

function[f,sf]=T2F(t,st)

dt=t(2)-t(1); % symbol period
T=t(end); % sigmal duration
df=1/T; % frequency resolution
N=length(st); % number of symbols

f=-N/2*df:df:N/2*df-df; % frequency index
sf=fft(st);
sf=T/N*fftshift(sf); % normalize and shift zero-frequency component to center of
spectrum
```

Question 16: Please provide rectangle function's Fourier Transform result here.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
%% Complete the Inverse Fourier Transform by IFFT
%  Input: f - frequency index
%         sf - signals in the frequency domain
%             i.e., the signal spectrum
%  Output: t - time index
%         st - signals in the time domain
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%

function [t,st]=F2T(f,sf)

df=f(2)-f(1); % frequency resolution
Fmx=(f(end)-f(1)+df); % frequency upper bound
dt=1/Fmx; % time resolution
N=length(sf); % number of symbols (points)
T=dt*N; % time duration

t=0:dt:(T-dt);  % time index
```

```
sff=fftshift(sf); % shift back to the original spectrum (corresponding to fftshift in T2F)
st=Fmx*ifft(sff);
```

> **Question 17:** Please provide **Sinc** function's Inverse Fourier Transform result here.

Low pass filter function:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%% Pass the signal through a Low Pass Filter
% Input: f - frequency index
%        sf - signals in the frequency domain
%            i.e., the signal spectrum
%        B - the pass bandwidth of the filter
% Output: t - time index
%        st - signals in the time domain
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%

function [t st]=lpf(f,sf,B)

df=f(2)-f(1); % frequency resolution
T=1/df; % signal duration
hf=zeros(1,length(f));
bf=[(-floor(B/df)):floor(B/df)]+floor(length(f)/2); % frequency index of the filter
hf(bf)=1; % rectangular pass band
yf=hf.*sf; % filtering

[t,st]=F2T(f,yf); % Inverse Fourier Transform
st=real(st); % taking the real part
```

> **Question 18:** Use a flow chat to show LPF's procedure based on the above codes.

Insert zeros in sequence

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
%% Insert zeros
% Input: d - the signal to be expanded
%        M - the number of samples per symbol after expanding
%            i.e., insert M-1 zeros
% Output: out - the signal after expanding
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
```

```
function out = sigexpand(d, M)

N = length(d);
out = zeros(M, N);
out(1,:) = d;
out = reshape(out, 1, M*N);
```

PCM encoder function:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%% Complete PCM coding
%  Input: signal - the signal to be encoded
%         num_bits - number of bits per sample
%  Output: bits - the bit stream after encoding
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

function bits = PCM_encoder(signal,num_bits)

n = length(signal);

% determine the range of the input signal
min_abs = min(abs(signal));
max_abs = max(abs(signal));

num = 2^(num_bits-1); % the first bit is the sign bit
step = (max_abs - min_abs)/num; % the length of intervals

partition = [min_abs:step:max_abs]; % uniform intervals

bits = zeros(n,num_bits);

for ii = 1:n % one-by-one processing
   % determine the sign bit
   if signal(ii)>0
      bits(ii,1)=1;
   else
      bits(ii,1)=0;
   end

   tmp = abs(signal(ii)); % focus on the absolute value

   % tranverse all the intervals
   for jj = 1:num
      if tmp >= partition(jj) & tmp < partition(jj+1)
         bits(ii,2:end) = dec2bin(jj-1,num_bits-1)-48; % converting to bits, 48 is the numeric
value of '0'
```

```
            break;
        end
    end
end

bits = reshape(bits',1,n*num_bits); % generating the bit stream
```