Audio Equalizer

Name	ID	Group
Seif el-Din Mahmoud	6773	3
Yasmine Emad	7107	3

CODE:

```
function varargout = GUI2(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI2_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end


function GUI2_OpeningFcn(hObject, eventdata, handles,
varargin)
handles.output = hObject;
set(handles.FIR,'value',1);
set(handles.edit1,'String',0);
set(handles.edit2,'String',0);
set(handles.edit3,'String',0);
set(handles.edit4,'String',0);
set(handles.edit5,'String',0);
set(handles.edit6,'String',0);
set(handles.edit7,'String',0);
set(handles.edit8,'String',0);
set(handles.edit9,'String',0);
```

```
guidata(hObject, handles);

function varargout = GUI2_OutputFcn(hObject, eventdata,
handles)

varargout{1} = handles.output;


% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
number = get(handles.slider1,'value');
set(handles.edit1,'string',num2str(number));


% --- Executes during object creation, after setting all
properties.
function slider1_CreateFcn(hObject, eventdata, handles)

if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)
number = get(handles.slider2,'value');
set(handles.edit2,'string',num2str(number));

function slider2_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on slider movement.
```

```
function slider3_Callback(hObject, eventdata, handles)
number = get(handles.slider3,'value');
set(handles.edit3,'string',num2str(number));

function slider3_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes on slider movement.
function slider4_Callback(hObject, eventdata, handles)
number = get(handles.slider4,'value');
set(handles.edit4,'string',num2str(number));

function slider4_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function slider5_Callback(hObject, eventdata, handles)
number = get(handles.slider5,'value');
set(handles.edit5,'string',num2str(number));

function slider5_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
```

```
function slider6_Callback(hObject, eventdata, handles)
number = get(handles.slider6,'value');
set(handles.edit6,'string',num2str(number));

function slider6_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function slider7_Callback(hObject, eventdata, handles)
number = get(handles.slider7,'value');
set(handles.edit7,'string',num2str(number));

function slider7_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function slider8_Callback(hObject, eventdata, handles)
number = get(handles.slider8,'value');
set(handles.edit8,'string',num2str(number));

function slider8_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function slider9_Callback(hObject, eventdata, handles)
```

```
number = get(handles.slider9,'value');
set(handles.edit9,'string',num2str(number));

function slider9_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function edit1_Callback(hObject, eventdata, handles)
assignin('base','filter1db',str2double(get(handles.edit1,'String')))
;

function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
assignin('base','filter2db',str2double(get(handles.edit2,'String')))
;

% --- Executes during object creation, after setting all
properties.
function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)
assignin('base','filter3db',str2double(get(handles.edit3,'String')))
;
```

```
% --- Executes during object creation, after setting all
properties.
function edit3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit4_Callback(hObject, eventdata, handles)
assignin('base','filter4db',str2double(get(handles.edit4,'String')))
;

% --- Executes during object creation, after setting all
properties.
function edit4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit5_Callback(hObject, eventdata, handles)
assignin('base','filter5db',str2double(get(handles.edit5,'String')))
;

function edit5_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit6_Callback(hObject, eventdata, handles)
assignin('base','filter6db',str2double(get(handles.edit6,'String')))
;

function edit6_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit7_Callback(hObject, eventdata, handles)
assignin('base','filter7db',str2double(get(handles.edit7,'String')))
;

% --- Executes during object creation, after setting all
properties.
function edit7_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit8_Callback(hObject, eventdata, handles)
assignin('base','filter8db',str2double(get(handles.edit8,'String')))
;

% --- Executes during object creation, after setting all
properties.
function edit8_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end



function edit9_Callback(hObject, eventdata, handles)
assignin('base','filter9db',str2double(get(handles.edit9,'String')))
;

% --- Executes during object creation, after setting all
properties.
function edit9_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in Done.
function Done_Callback(hObject, eventdata, handles)
Filter_function(hObject, eventdata, handles);

function edit10_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all
properties.
function edit10_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in FIR.
function FIR_Callback(hObject, eventdata, handles)
```

```
set(handles.IIR,'value',0);

% --- Executes on button press in IIR.
function IIR_Callback(hObject, eventdata, handles)
set(handles.FIR,'value',0);

function edit11_Callback(hObject, eventdata, handles)
assignin('base','out',str2double(get(handles.edit11,'String')));

% --- Executes during object creation, after setting all
properties.
function edit11_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function Filter_function(hObject, eventdata, handles)
[y,fs] = audioread(get(handles.edit10,'String'));
outFreq = str2double(get(handles.edit11,'String'));
filter1 = 10^(str2double(get(handles.edit1,'String'))/10);
filter2 = 10^(str2double(get(handles.edit2,'String'))/10);
filter3 = 10^(str2double(get(handles.edit3,'String'))/10);
filter4 = 10^(str2double(get(handles.edit4,'String'))/10);
filter5 = 10^(str2double(get(handles.edit5,'String'))/10);
filter6 = 10^(str2double(get(handles.edit6,'String'))/10);
filter7 = 10^(str2double(get(handles.edit7,'String'))/10);
filter8 = 10^(str2double(get(handles.edit8,'String'))/10);
filter9 = 10^(str2double(get(handles.edit9,'String'))/10);

gains =
[filter1,filter2,filter3,filter4,filter5,filter6,filter7,filter8,filter9];
disp('Gains: '); disp(gains);
```

```
sampleRate = 40000;

%plot the original signal in time & frequency domain
figure;
subplot(3,2,[1 2]);
plot(y);
xlabel('Time');
title('Original signal before filtering');
yf=fftshift(fft(y/sampleRate));
f=linspace(-fs/2,fs/2,length(yf));
subplot(3,2,[3 4]);
plot(f,real(yf));
xlim([-1000 1000]);
ylim([-1 1]);
xlabel('Frequency');
subplot(3,2,5);
plot(f,abs(yf));
xlim([-1000 1000]);
xlabel('Magnitude');
subplot(3,2,6);
plot(f,angle(yf));
xlabel('Phase');

%Filtering Prossece
filters = [];
if get(handles.IIR,'value') == 1 %IIR
    %Filter no. 1 ran0ge 0hz to 170khz
    [num1, den1] = butter(3, 2*170/sampleRate,'low');
    filters = [filters filter(num1,den1,y)];
    %Filter no. 2 range 170hz to 310khz
    [num2, den2] = butter(3,
[2*170/sampleRate,2*310/sampleRate],'bandpass');
    filters = [filters filter(num2,den2,y)];
    %Filter no. 3 range 310khz to 600hz
```

```
    [num3, den3] = butter(3,
[2*310/sampleRate,2*600/sampleRate],'bandpass');
    filters = [filters filter(num3,den3,y)];
    %Filter no. 4 range 600hz to 1khz
    [num4, den4] = butter(3,
[2*600/sampleRate,2*1000/sampleRate],'bandpass');
    filters = [filters filter(num4,den4,y)];
    %Filter no. 5 range 1khz to 3khz
    [num5, den5] = butter(3,
[2*1000/sampleRate,2*3000/sampleRate],'bandpass');
    filters = [filters filter(num5,den5,y)];
    %Filter no. 6 range 3khz to 6khz
    [num6, den6] = butter(3,
[2*3000/sampleRate,2*6000/sampleRate],'bandpass');
    filters = [filters filter(num6,den6,y)];
    %Filter no. 7 range 6khz to 12khz
    [num7, den7] = butter(3,
[2*6000/sampleRate,2*12000/sampleRate],'bandpass');
    filters = [filters filter(num7,den7,y)];
    %Filter no. 8 range 12khz to 14khz
    [num8, den8] = butter(3,
[2*12000/sampleRate,2*14000/sampleRate],'bandpass');
    filters = [filters filter(num8,den8,y)];
    %Filter no. 9 range 14khz to 16khz
    [num9, den9] = butter(3,
[2*14000/sampleRate,2*16000/sampleRate],'bandpass');
    filters = [filters filter(num9,den9,y)];
end
if get(handles.FIR,'value') == 1 %FIR type
    %Filter no. 1 range 0hz to 170khz
    [num1, den1] = fir1(50, 2*170/sampleRate,'low');
    filters = [filters filter(num1,den1,y)];
    %Filter no. 2 range 170hz to 310khz
    [num2, den2] = fir1(50,
[2*170/sampleRate,2*310/sampleRate],'bandpass');
```

```
    filters = [filters filter(num2,den2,y)];
    %Filter no. 3 range 310khz to 600hz
    [num3, den3] = fir1(50,
[2*310/sampleRate,2*600/sampleRate],'bandpass');
    filters = [filters filter(num3,den3,y)];
    %Filter no. 4 range 600hz to 1khz
    [num4, den4] = fir1(50,
[2*600/sampleRate,2*1000/sampleRate],'bandpass');
    filters = [filters filter(num4,den4,y)];
    %Filter no. 5 range 1khz to 3khz
    [num5, den5] = fir1(50,
[2*1000/sampleRate,2*3000/sampleRate],'bandpass');
    filters = [filters filter(num5,den5,y)];
    %Filter no. 6 range 3khz to 6khz
    [num6, den6] = fir1(50,
[2*3000/sampleRate,2*6000/sampleRate],'bandpass');
    filters = [filters filter(num6,den6,y)];
    %Filter no. 7 range 6khz to 12khz
    [num7, den7] = fir1(50,
[2*6000/sampleRate,2*12000/sampleRate],'bandpass');
    filters = [filters filter(num7,den7,y)];
    %Filter no. 8 range 12khz to 14khz
    [num8, den8] = fir1(50,
[2*12000/sampleRate,2*14000/sampleRate],'bandpass');
    filters = [filters filter(num8,den8,y)];
    %Filter no. 9 range 14khz to 16khz
    [num9, den9] = fir1(50,
[2*14000/sampleRate,2*16000/sampleRate],'bandpass');
    filters = [filters filter(num9,den9,y)];
end

ranges = [0 170 310 600 1000 3000 6000 12000 14000 16000];

%output signal after filtering
 for i = 1:9
```

```
    figure;
    subplot(3,2,[1 2]);
    plot(filters(:,i));
    xlabel('Time');
    ylabel('Values');
    title(['Signal after filtering ' num2str(ranges(i)) ' - '
num2str(ranges(i+1)) ':']);
    filterfreq=fftshift(fft(filters(:,i)/sampleRate));
    f=linspace(-sampleRate/2,sampleRate/2,length(filterfreq));
    subplot(3,2,[3 4]);
    plot(f,real(filterfreq));
    xlim([-1000 1000]);
    ylim([-1 1]);
    xlabel('Frequency');
    ylabel('Values');
    subplot(3,2,5);
    plot(f,abs(filterfreq));
    xlim([-1000 1000]);
    xlabel('Frequency');
    ylabel('Values');
    subplot(3,2,6);
    plot(f,angle(filterfreq));
    xlabel('Frequency');
    ylabel('Values');
 end


%amplifying the filtered signals
ampFilter = [];
for i =1:9
     ampFilter = [ampFilter gains(i)*filters(:,i)];
end

%Output signals after filtering and amplification
for i=1:9
```

```
    figure;
    subplot(3,2,[1 2]);
    plot(ampFilter(:,i));
    xlabel('Time');
    ylabel('Values');
    title(['Signal after Amplifiered ' num2str(ranges(i)) ' - '
num2str(ranges(i+1)) ':']);
    ampfilterfreq=fftshift(fft(ampFilter(:,i)/sampleRate));
    f=linspace(-
sampleRate/2,sampleRate/2,length(ampfilterfreq));
    subplot(3,2,[3 4]);
    plot(f,real(ampfilterfreq));
    xlim([-1000 1000]);
    ylim([-1 1]);
    xlabel('Frequency');
    ylabel('Values');
    subplot(3,2,5);
    plot(f,abs(ampfilterfreq));
    xlim([-1000 1000]);
    xlabel('Magnituide');
    subplot(3,2,6);
    plot(f,angle(ampfilterfreq));
    xlabel('Phase');
 end

%add the amplified signals in time domain to form composite
signal
compositeSignal =
ampFilter(:,1)+ampFilter(:,2)+ampFilter(:,3)+ampFilter(:,4)+amp
Filter(:,5)+ampFilter(:,6)+ampFilter(:,7)+ampFilter(:,8)+ampFilte
r(:,9);

%plot the original signal in time & frequency domain
figure;
subplot(3,2,[1 2]);
```

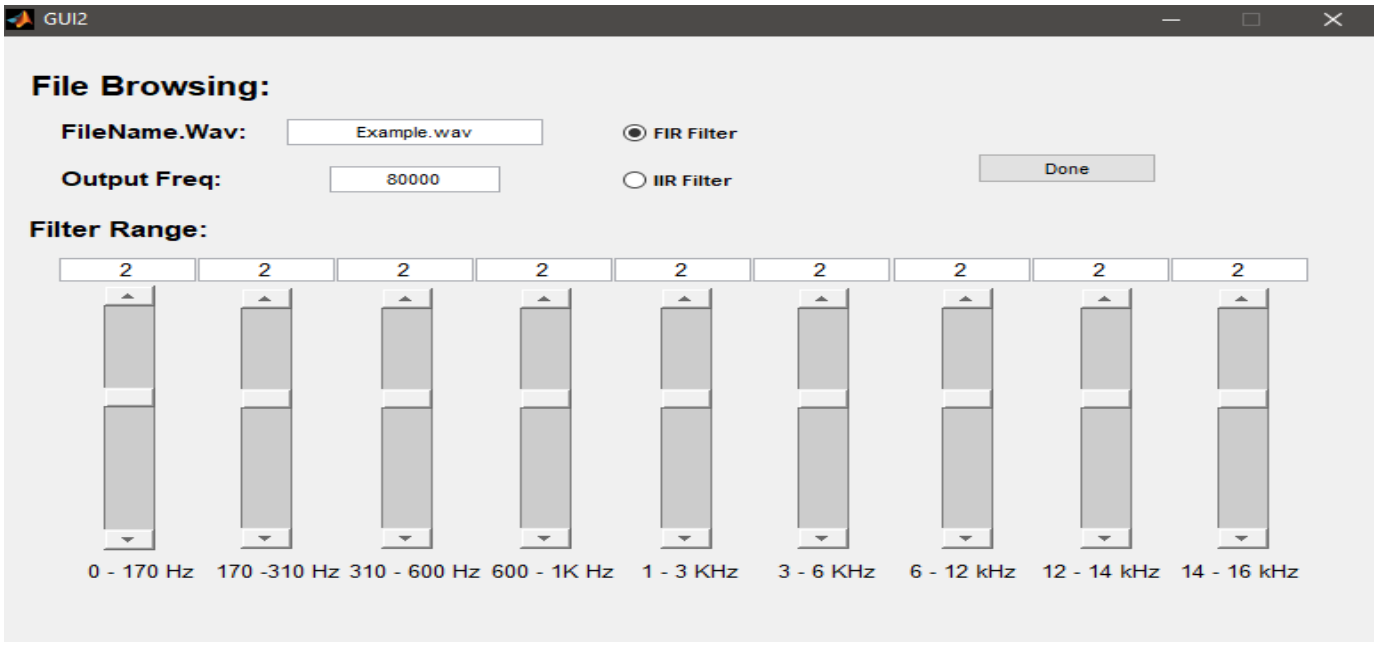
```
plot(y);
xlabel('Time');
ylabel('Values');
title('Original signal before filtering');
yf=fftshift(fft(y/sampleRate));
f=linspace(-fs/2,fs/2,length(yf));
subplot(3,2,[3 4]);
plot(f,real(yf));
xlim([-500 500]);
ylim([-1 1]);
ylabel('Values');
xlabel('Frequency');
subplot(3,2,5);
plot(f,abs(yf));
xlim([-1000 1000]);
xlabel('Magnitude');
subplot(3,2,6);
plot(f,angle(yf));
xlabel('Phase');



%plot the composite signal in time&frequency domain
figure;
subplot(3,2,[1 2]);
plot(compositeSignal);
xlabel('Time');
ylabel('Values');
title('Composite signal');
compositeSignalF=fftshift(fft(compositeSignal/sampleRate));
f=linspace(-outFreq/2,outFreq/2,length(compositeSignalF));
subplot(3,2,[3 4]);
plot(f,real(compositeSignalF));
xlim([-1000 1000]);
xlabel('Frequency');
```

```
ylabel('Values');
subplot(3,2,5);
plot(f,abs(compositeSignalF));
xlim([-1000 1000]);
xlabel('Magnitude');
subplot(3,2,6);
plot(f,angle(compositeSignalF));
xlabel('Phase');

% play the composite wave signal
sound(compositeSignal,outFreq);

%Analysis

choice = menu('Do you want to analyse a filter?', 'Yes','No');
if choice == 1
    while(1)
        choice1=menu('Choose a filter', '0 - 170 Hz','170 - 310
Hz','310 - 600 Hz','600 - 1000 Hz','1 - 3 KHz','3 - 6 KHz','6 - 12
KHz','12 - 14 KHz','14 - 16 KHz','Exit');
        if choice1 ~=10
            switch choice1
                case 1
                    a = num1;
                    b = den1;
                case 2
                    a = num2;
                    b = den2;
                case 3
                    a = num3;
                    b = den3;
                case 4
                    a = num4;
                    b = den4;
                case 5
```

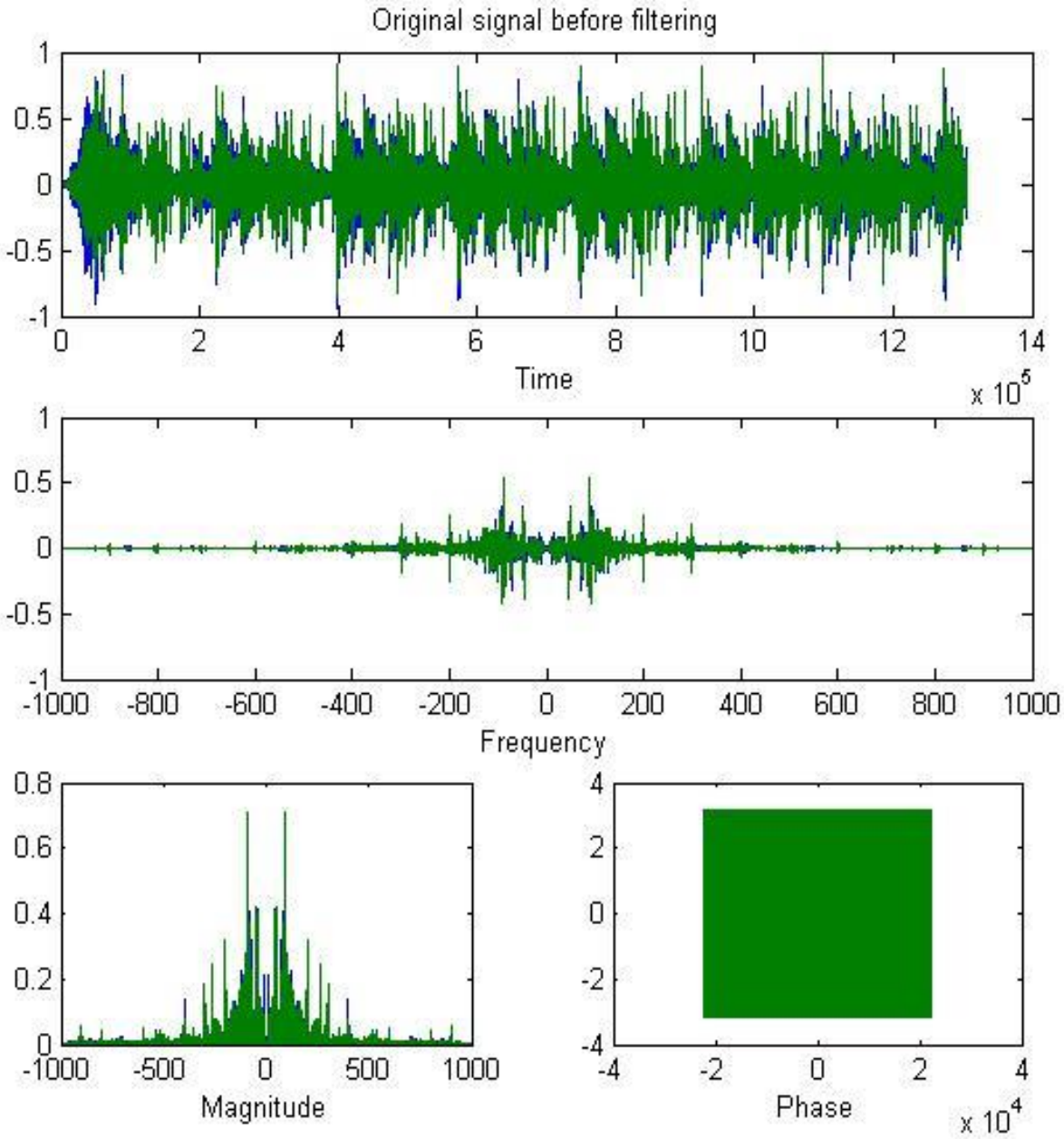
```
                a = num5;
                b = den5;
            case 6
                a= num6;
                b = den6;
            case 7
                a = num7;
                b = den7;
            case 8
                a = num8;
                b = den8;
            case 9
                a= num9;
                b = den9;
        end
    [z,p,k]=tf2zpk(a,b);
    disp('Gain:'); disp(k);
    n=filtord(a,b); %get the order of the filter
    disp('Order:'); disp(n)
    [Yphase, w]=phasez(a,b); %get the phase of the filter
    figure;
    subplot(2,2,1)
    zplane(p,z);
    title(['Filter ' num2str(ranges(choice1)) ' - '
num2str(ranges(choice1+1)) ':']);
    subplot(2,2,2);
    plot(Yphase);
    title('Phase response of filter');
    [h,t] = impz(a,b); %get the impulse response of the filter
    subplot(2,2,3);
    plot(h);
    title('impulse response of filter');
    [s,t] = stepz(a,b); %get the step response of the filter
    subplot(2,2,4);
    plot(s);
```

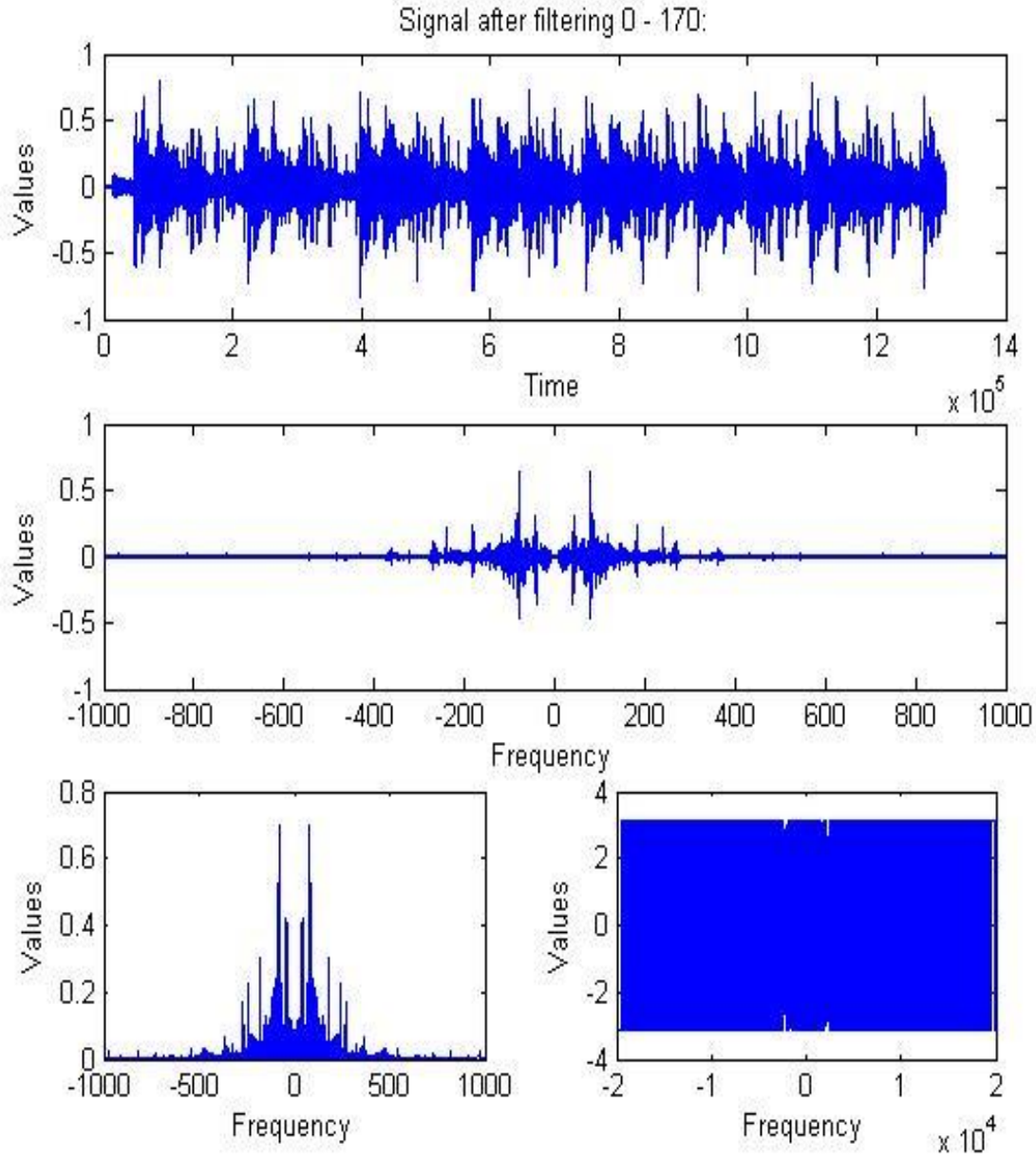
```
        title('step response of filter');
    else
        break;
    end
end
end
```

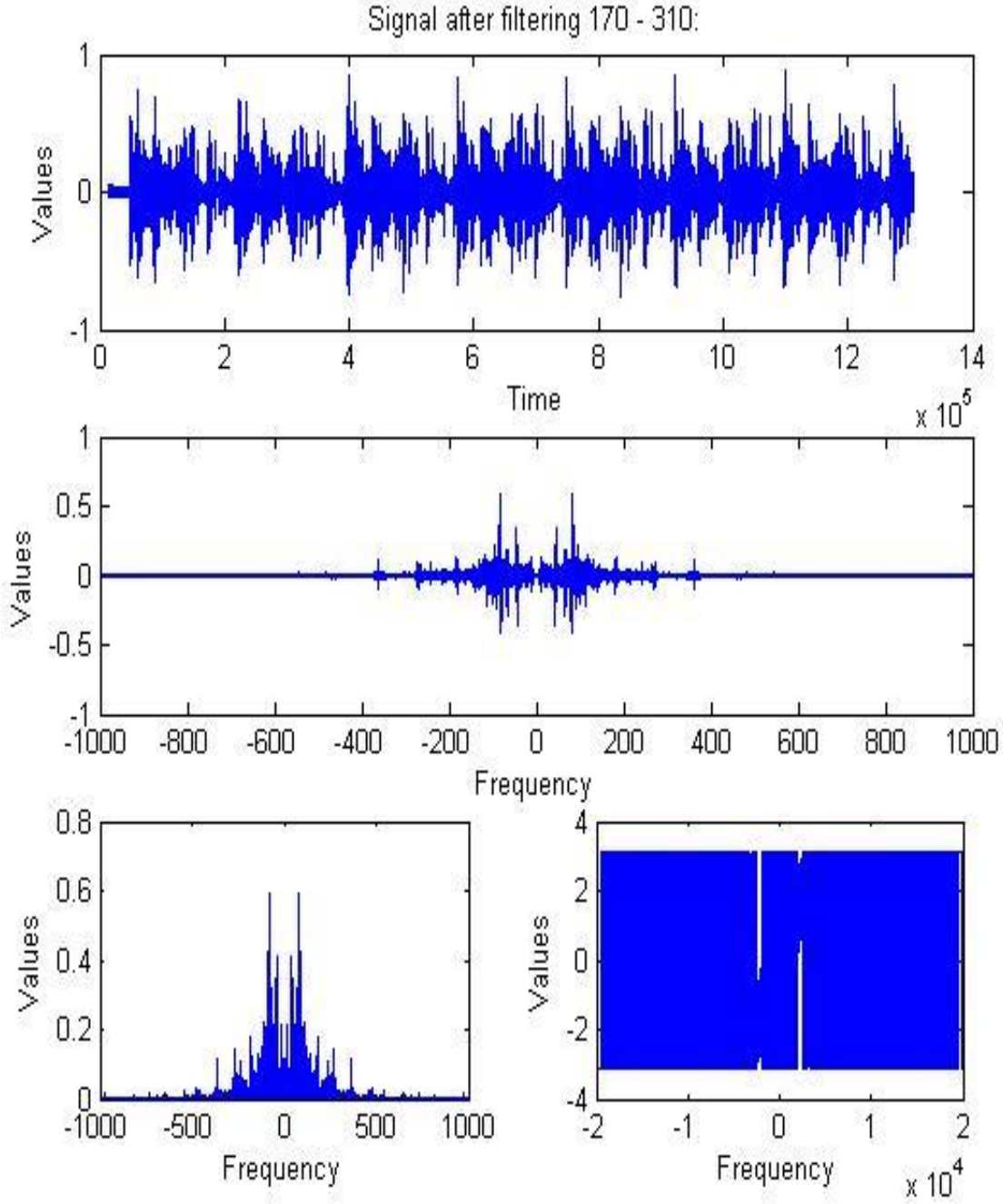
Sample Run

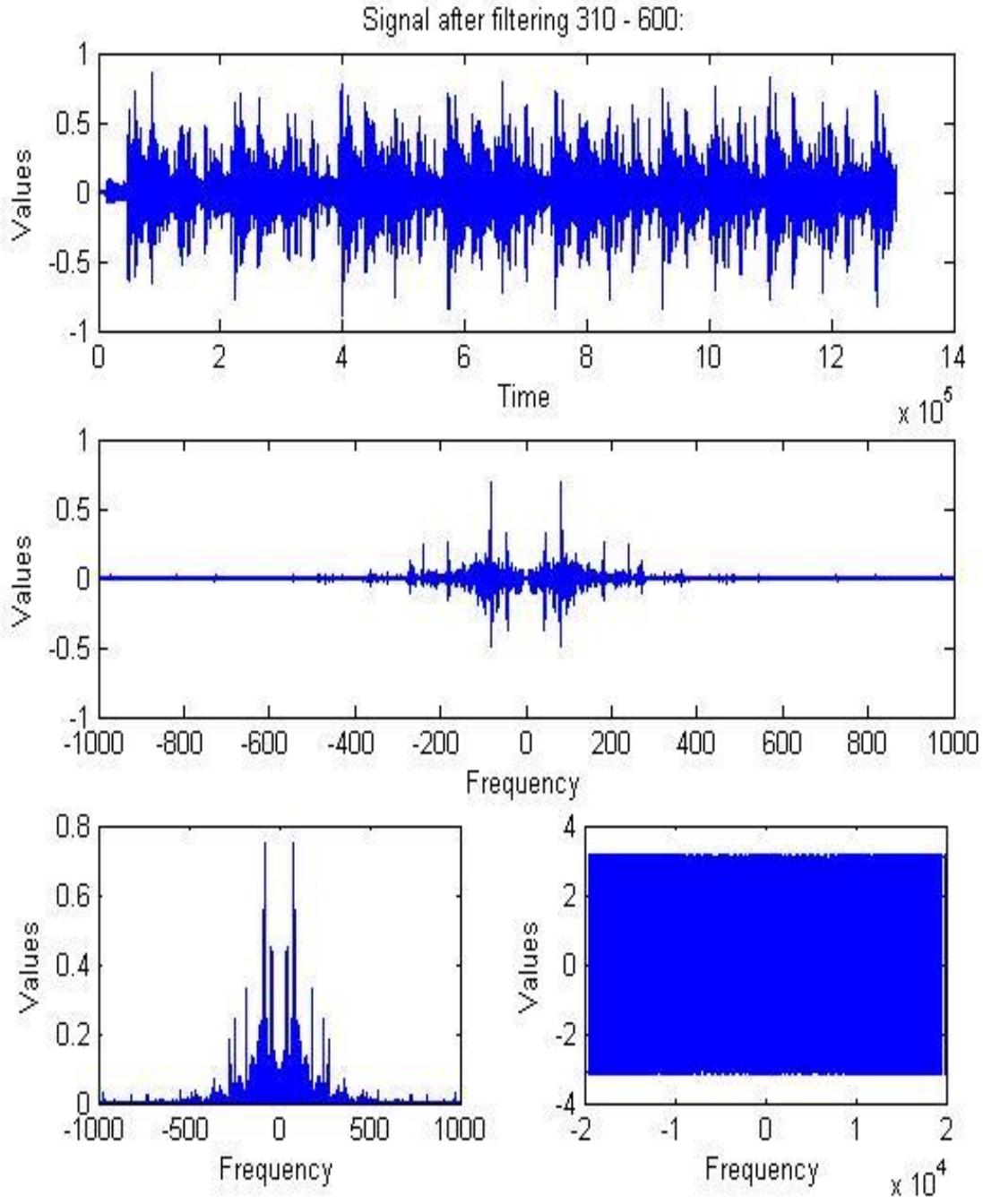


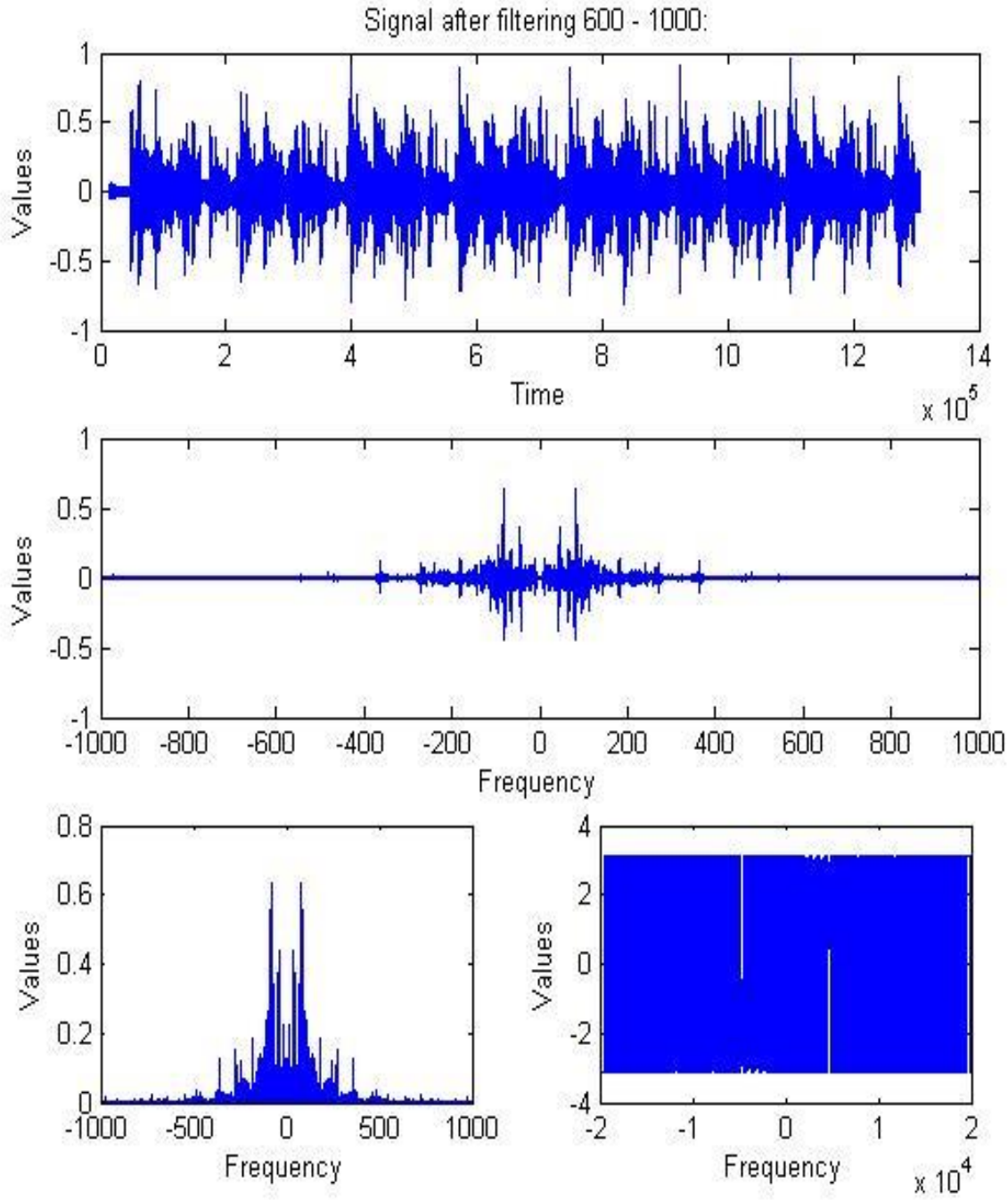
FIR Filter 1st with order of 3 and gain of 2 db.

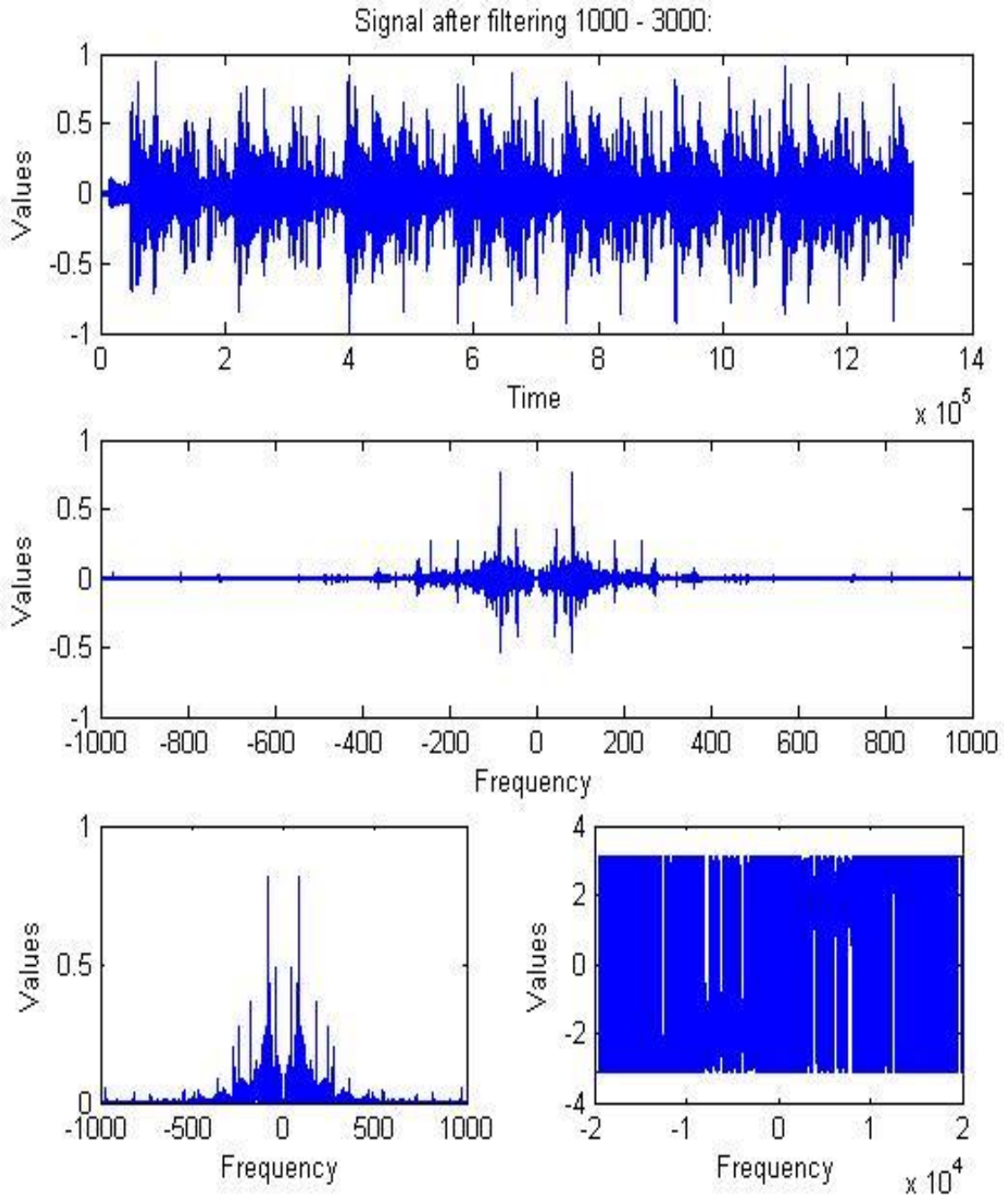


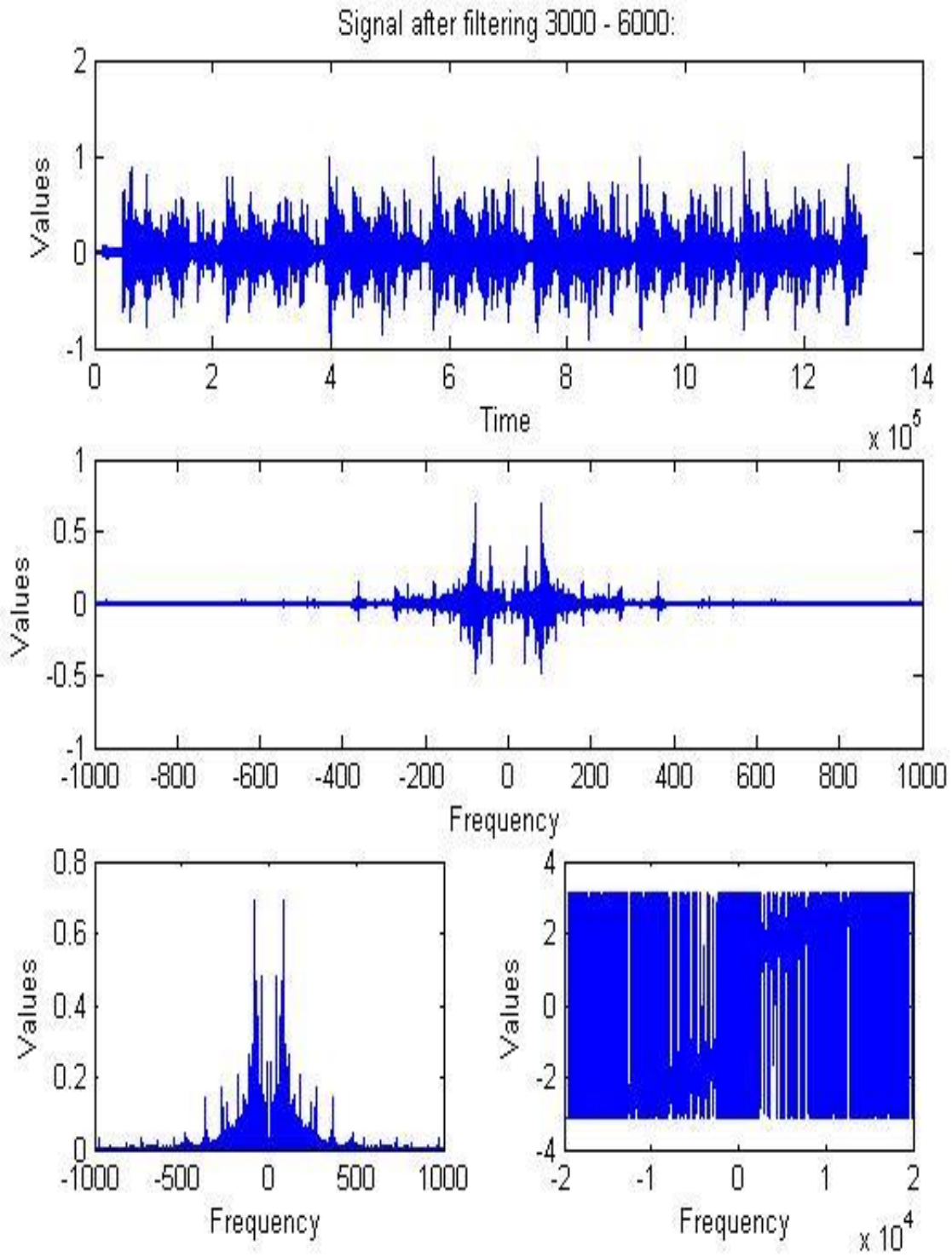

Signal after filtering 0 - 170:
Values
Time
x 10^5
Values
Frequency
Values
Frequency
Values
Frequency
x 10^4

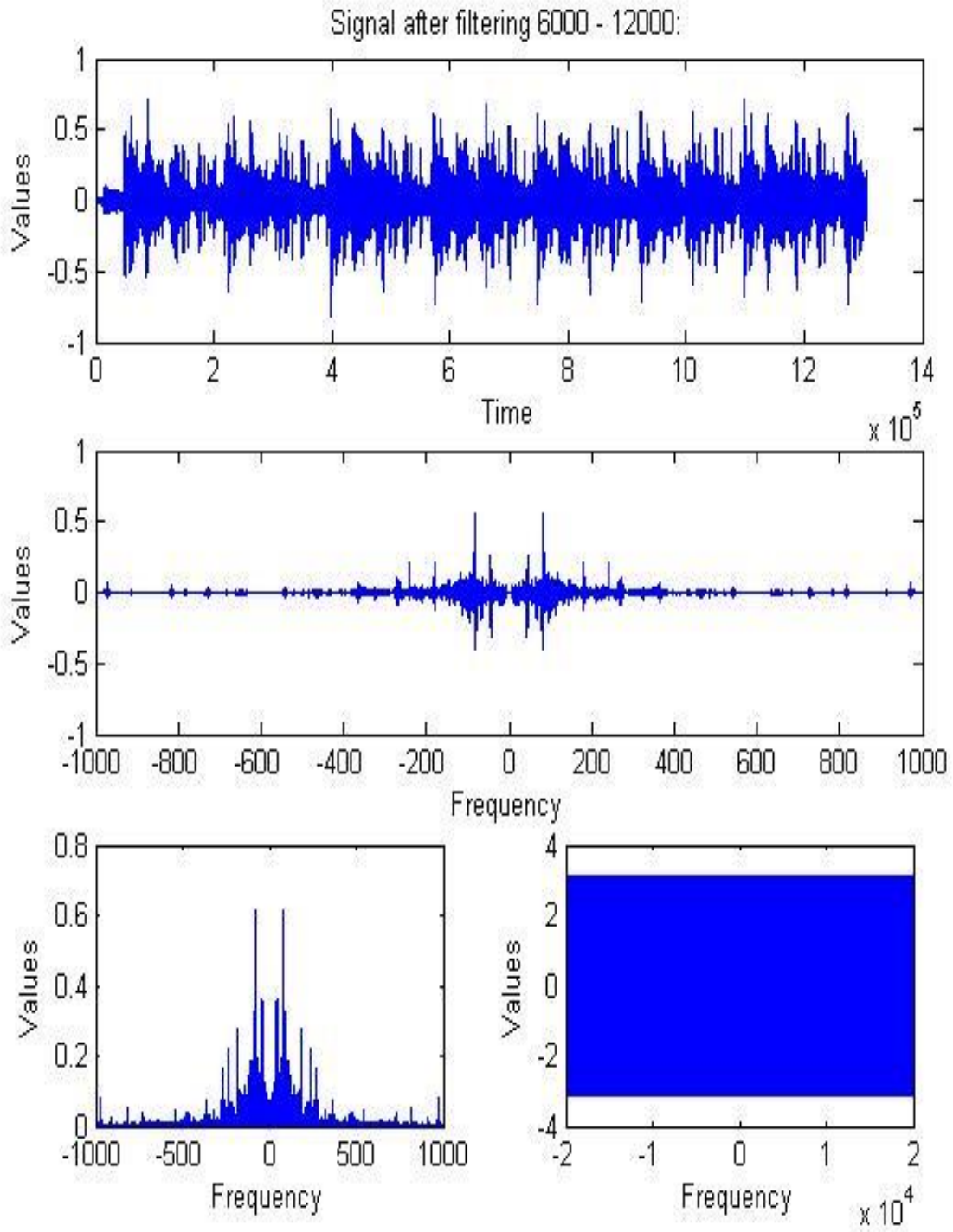

Signal after filtering 170 - 310:
Values
Time
x 10^5
Values
Frequency
Values
Frequency
Values
Frequency
x 10^4

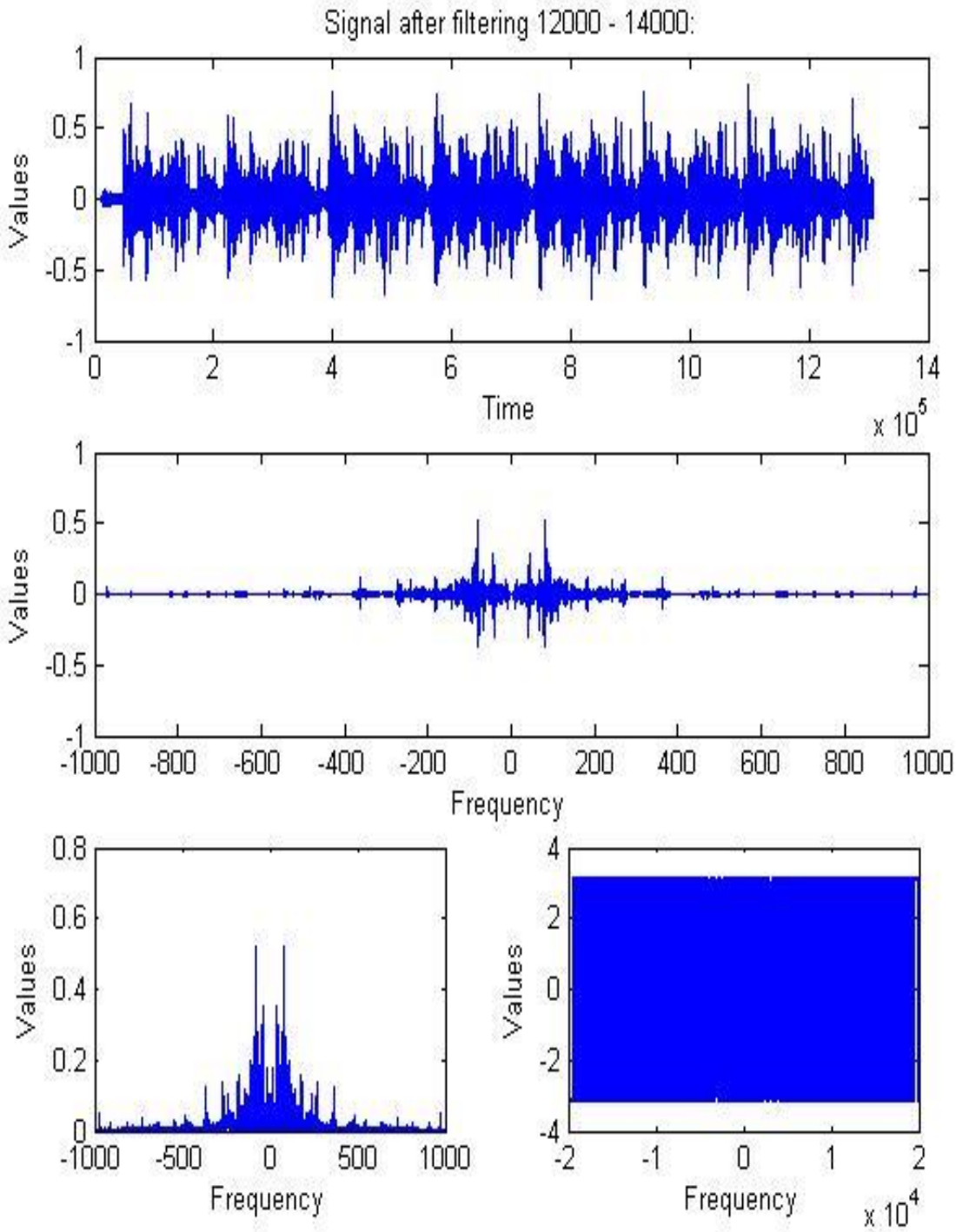

Signal after filtering 310 - 600:
Values
Time
x 10^5
Values
Frequency
Values
Frequency
Values
Frequency
x 10^4

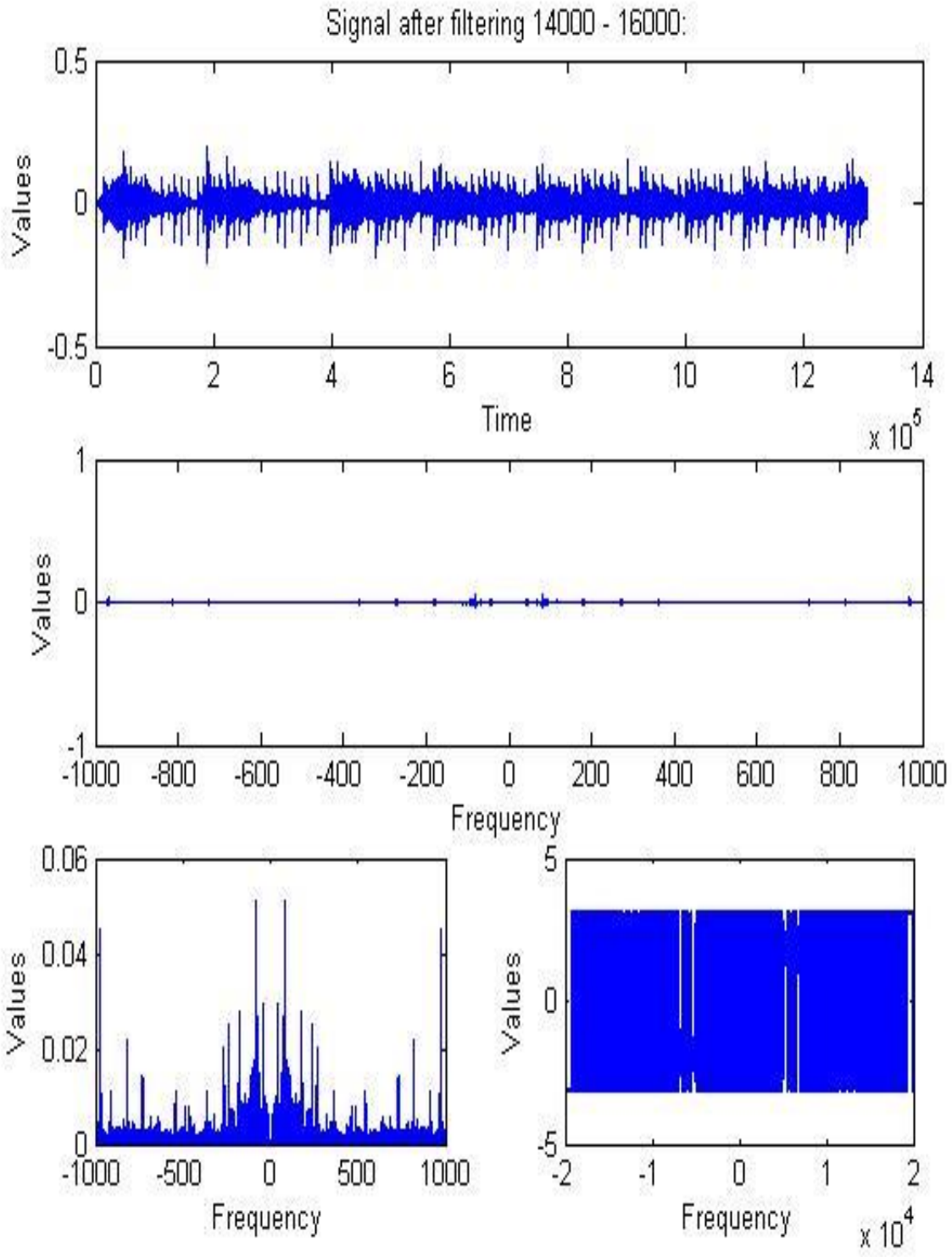

Signal after filtering 600 - 1000:
Values
Time
x 10^5
Values
Frequency
Values
Frequency
Values
Frequency
x 10^4

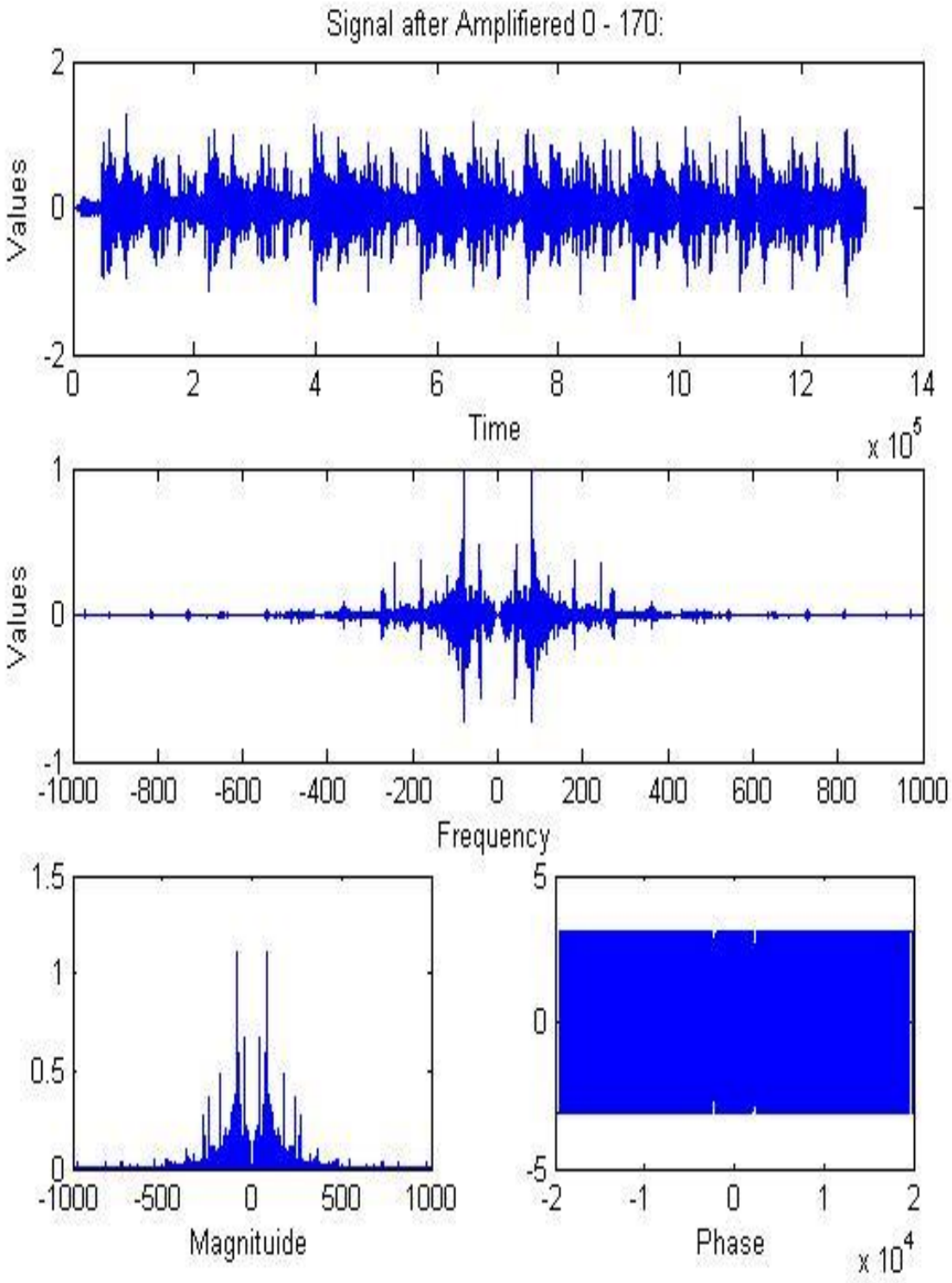

Signal after filtering 1000 - 3000:
Values
Time
x 10^5
Values
Frequency
Values
Frequency
Values
Frequency
x 10^4

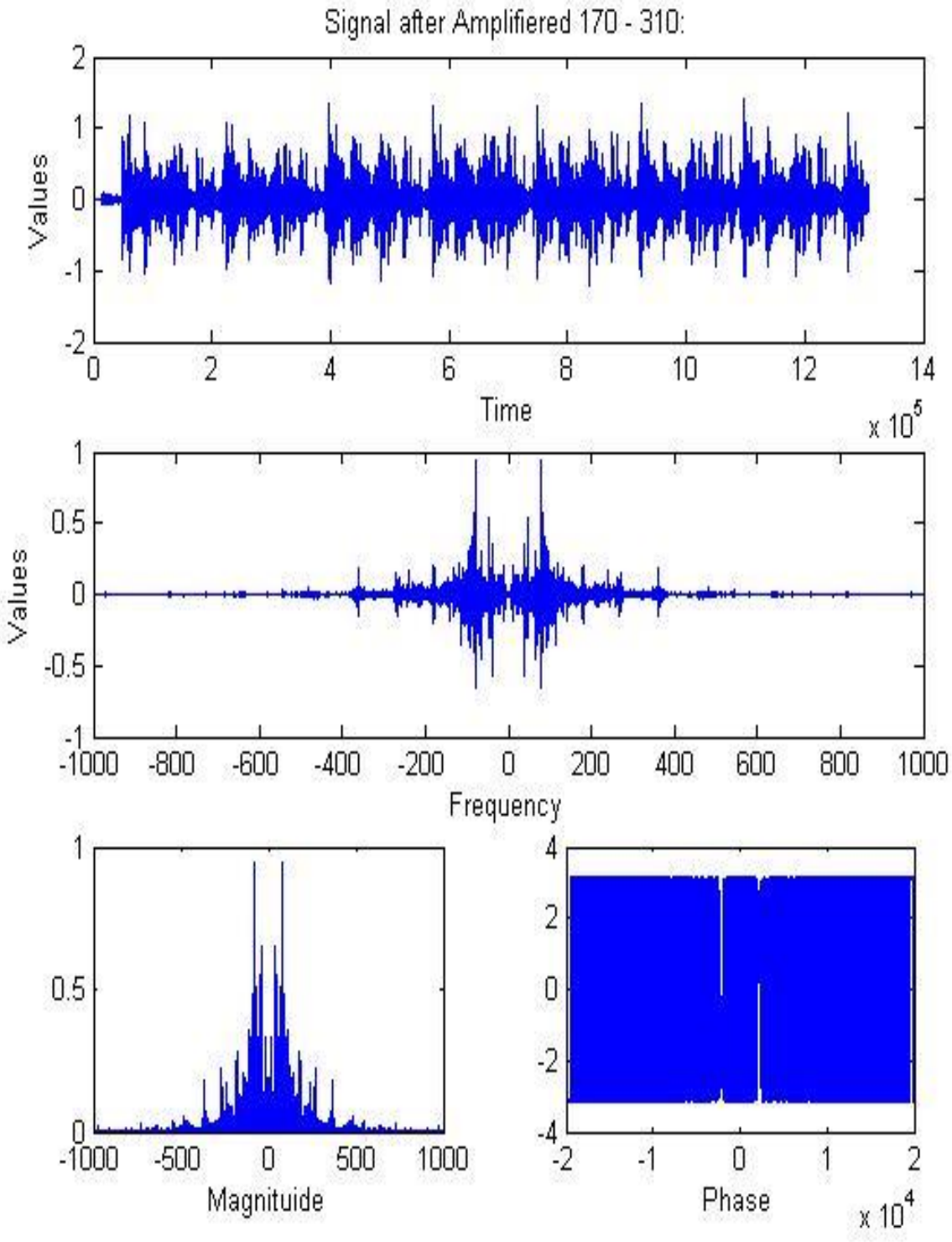

Signal after filtering 3000 - 6000:
Values
Time
x 10^5
Values
Frequency
Values
Frequency
Values
Frequency
x 10^4

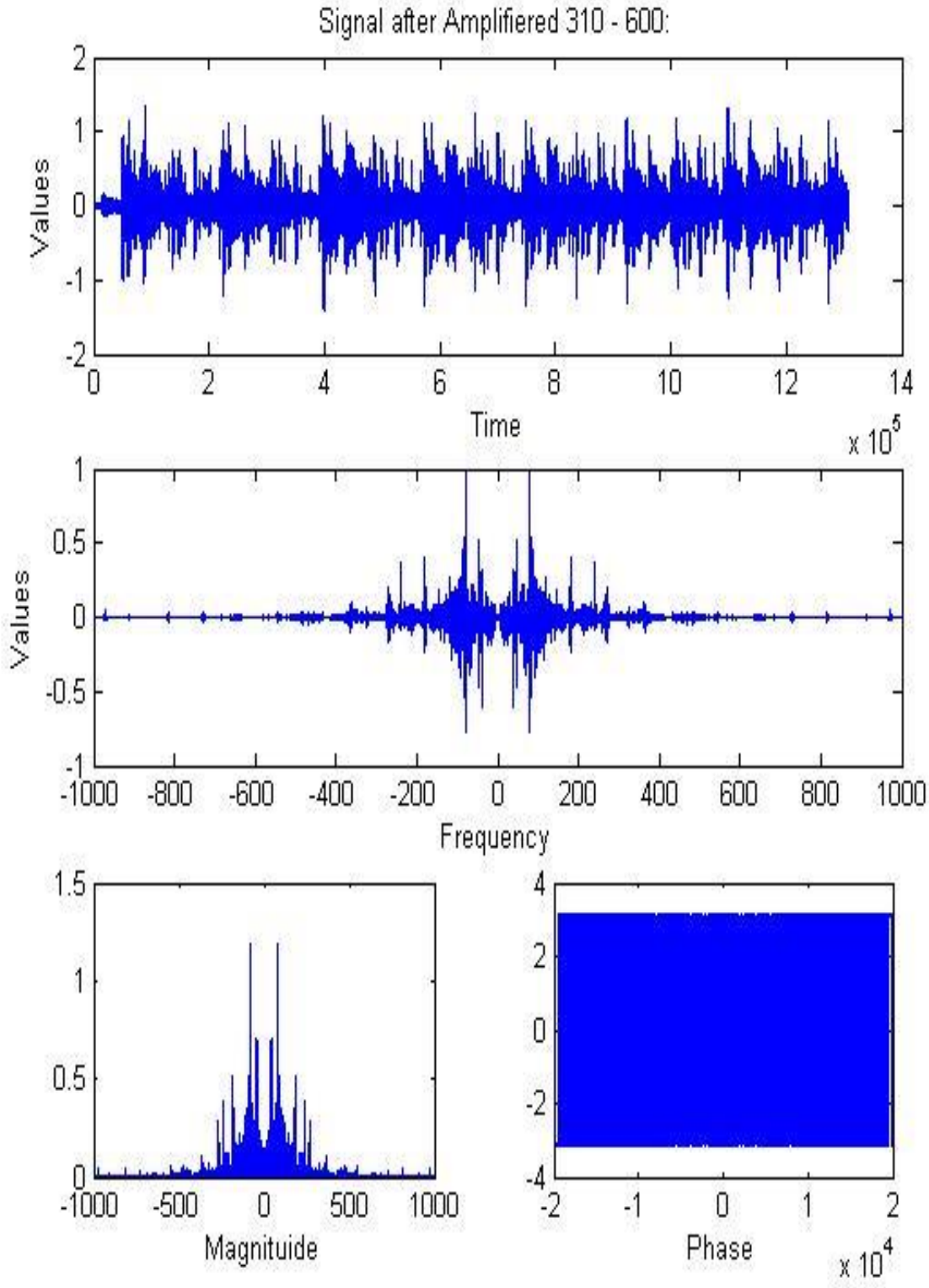

Signal after filtering 6000 - 12000:
Values
Time
x 10^5
Frequency
Frequency
x 10^4

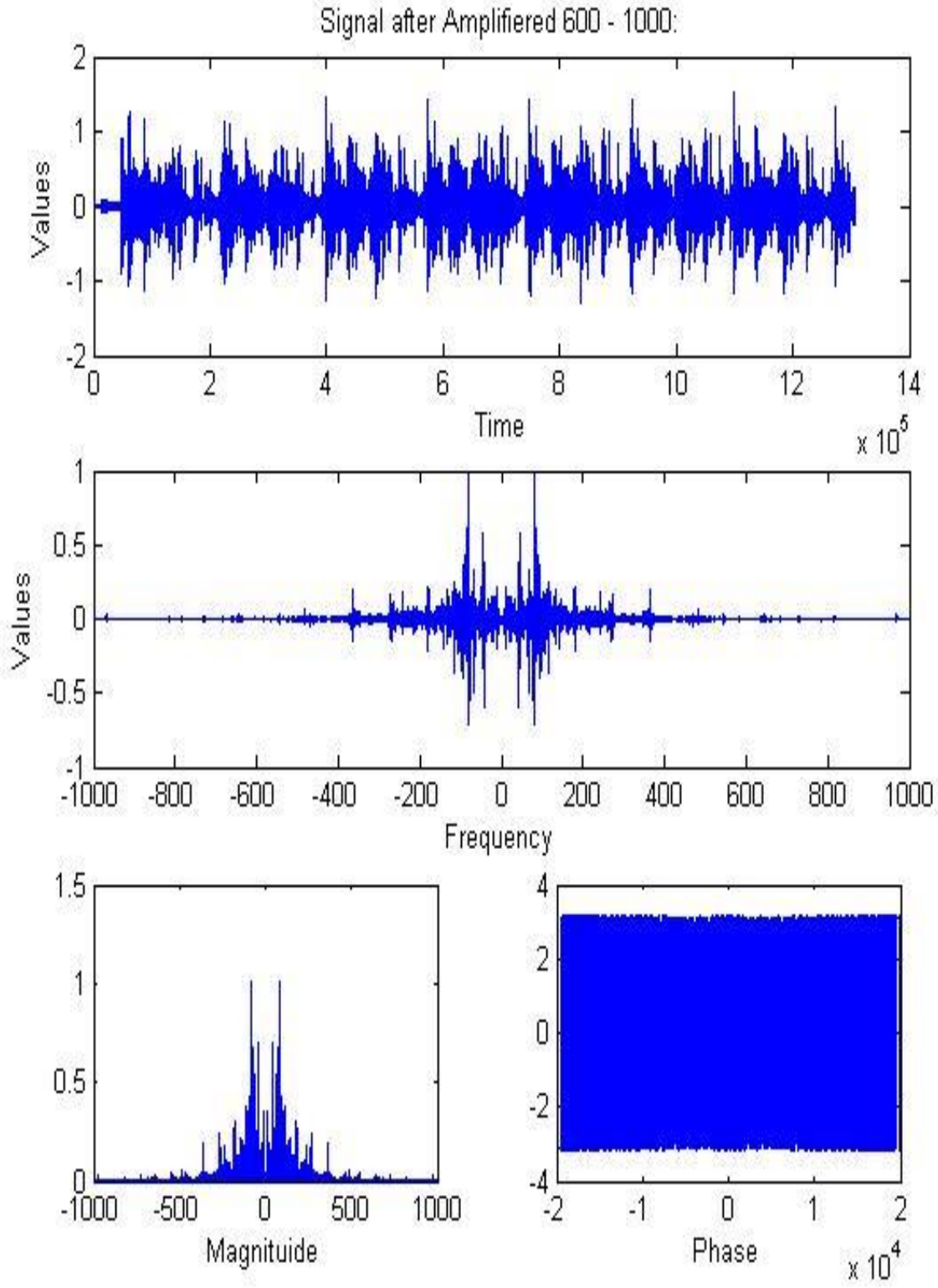

Signal after filtering 12000 - 14000:
Values
Time
x 10^5
Values
Frequency
Values
Frequency
Values
Frequency
x 10^4

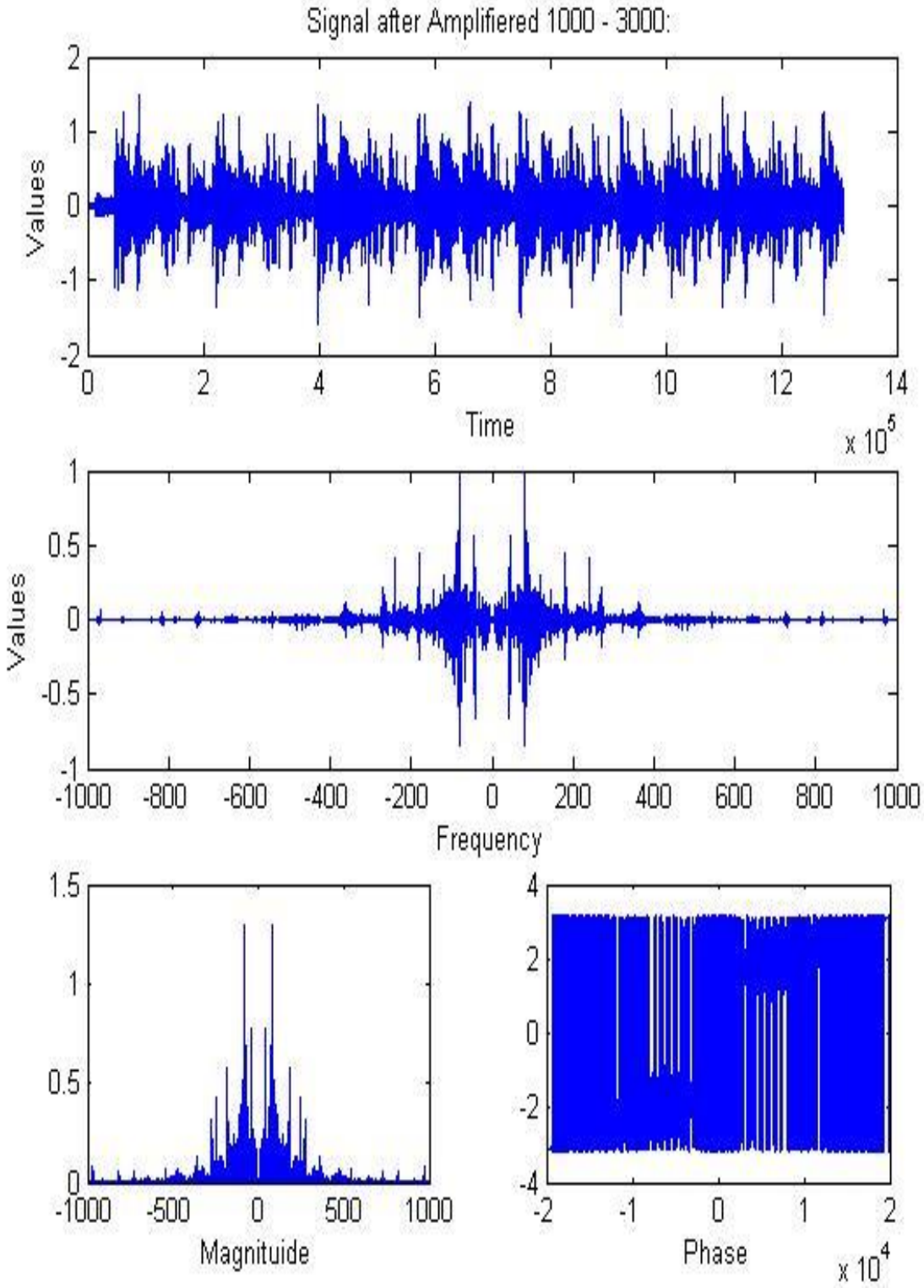

Signal after filtering 14000 - 16000:
Values
Time
x 10^5
Frequency
Values
Frequency
Values
Frequency
x 10^4

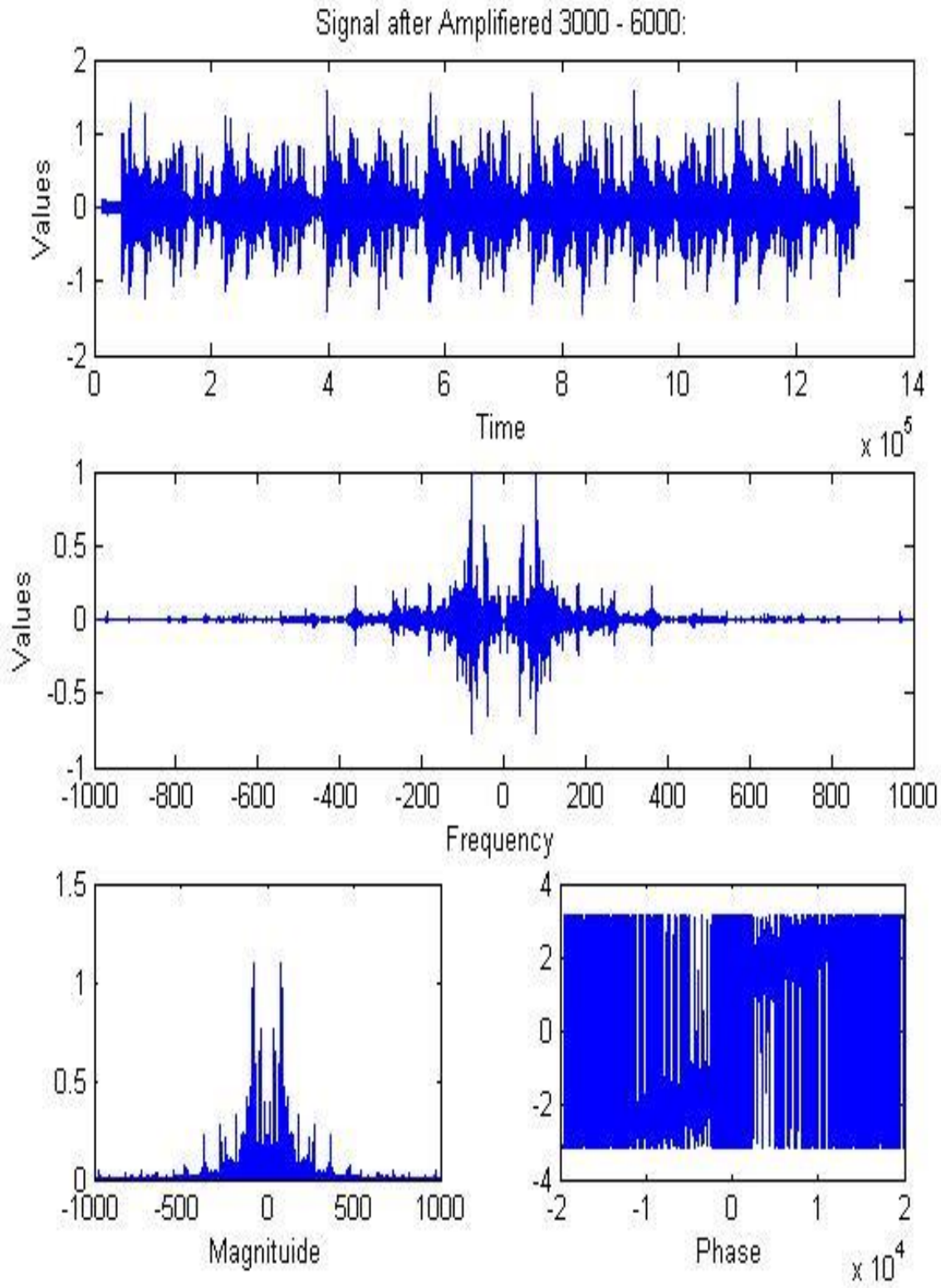

Signal after Amplifiered 0 - 170:
Values
Time
x 10^5
Values
Frequency
Magnituide
Phase
x 10^4

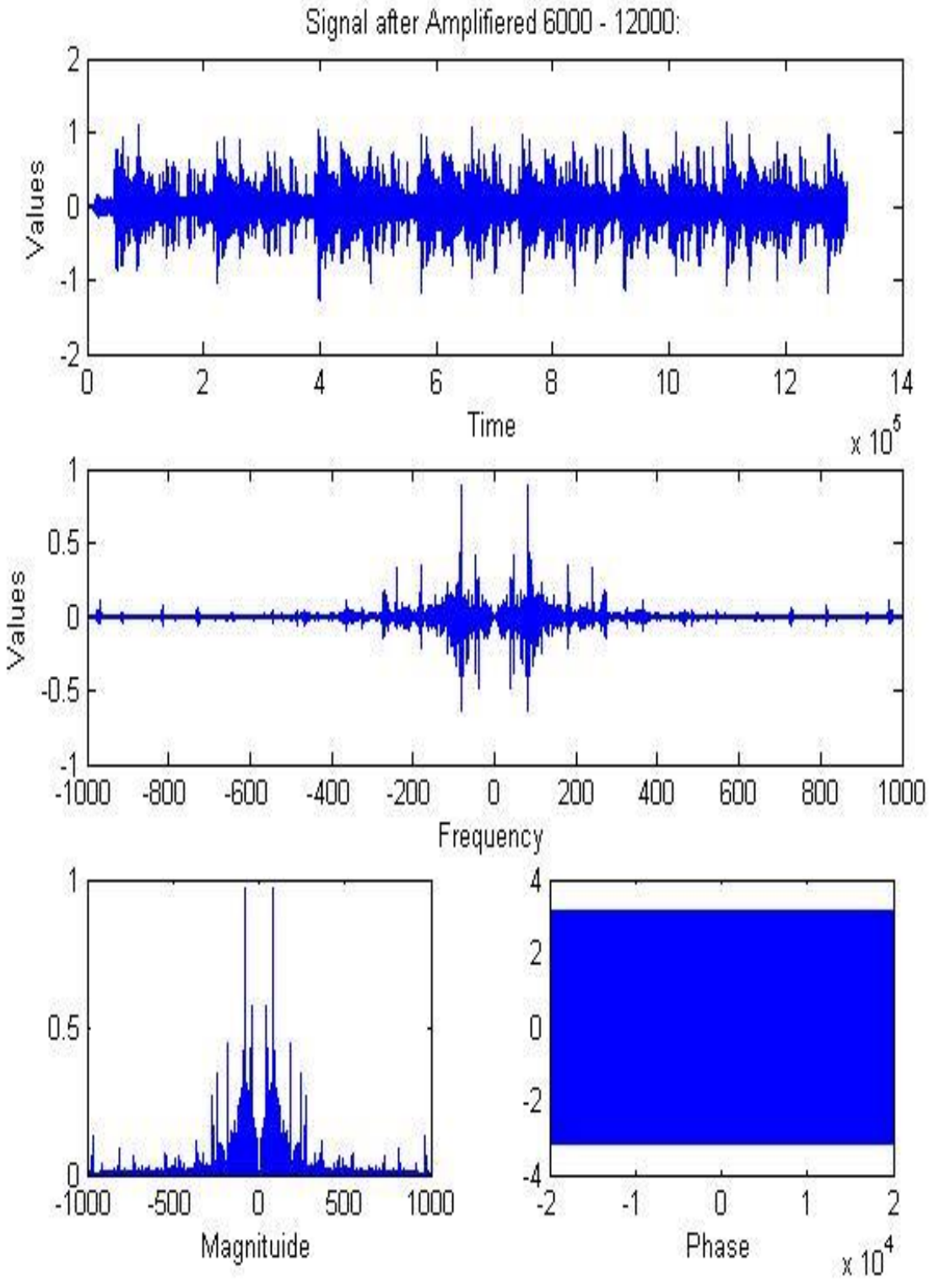

Signal after Amplifiered 170 - 310:
Values
Time
x 10^5
Values
Frequency
Magnituide
Phase
x 10^4

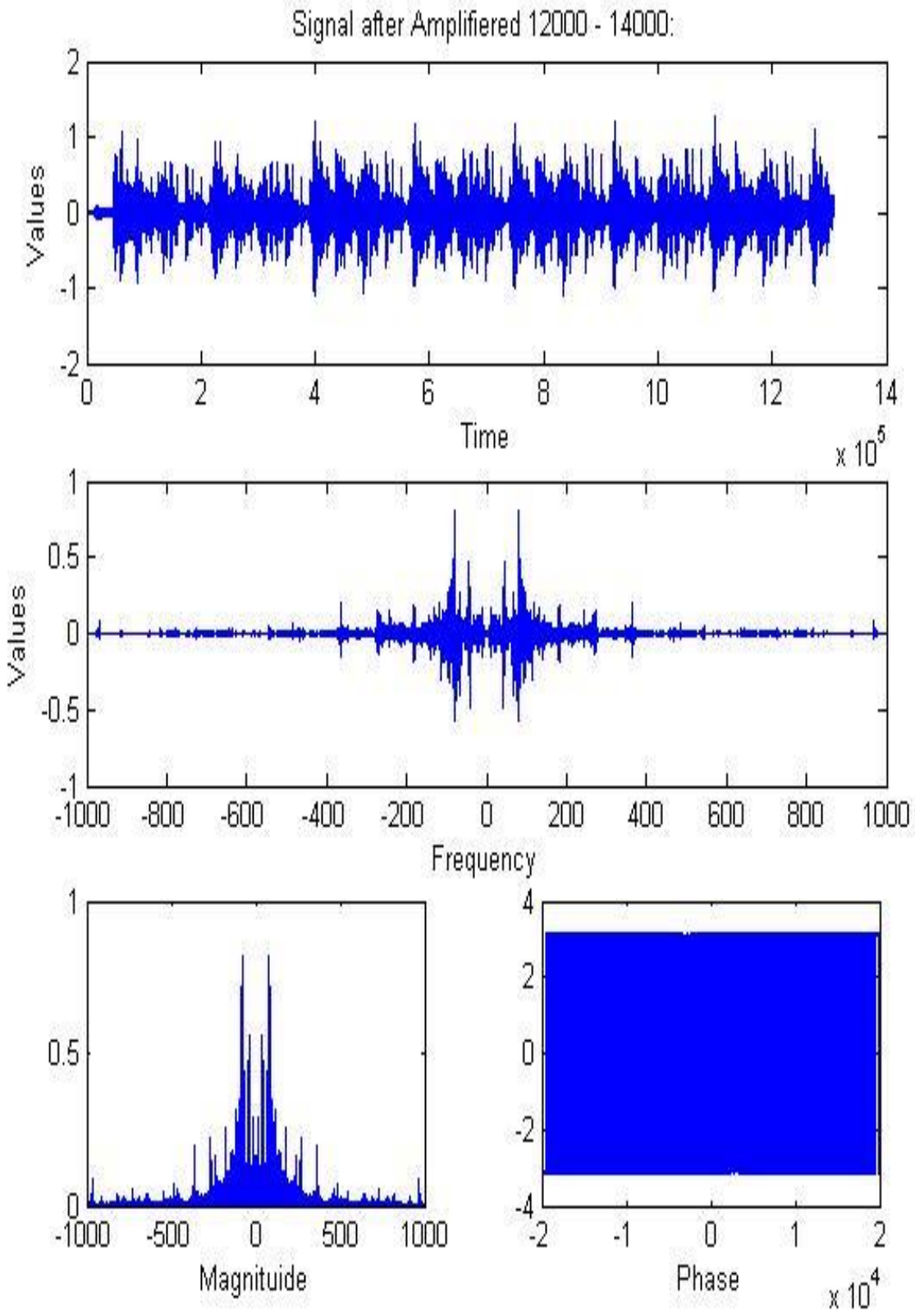

Signal after Amplifiered 310 - 600:
Values
Time
x 10^5
Values
Frequency
Magnituide
Phase
x 10^4

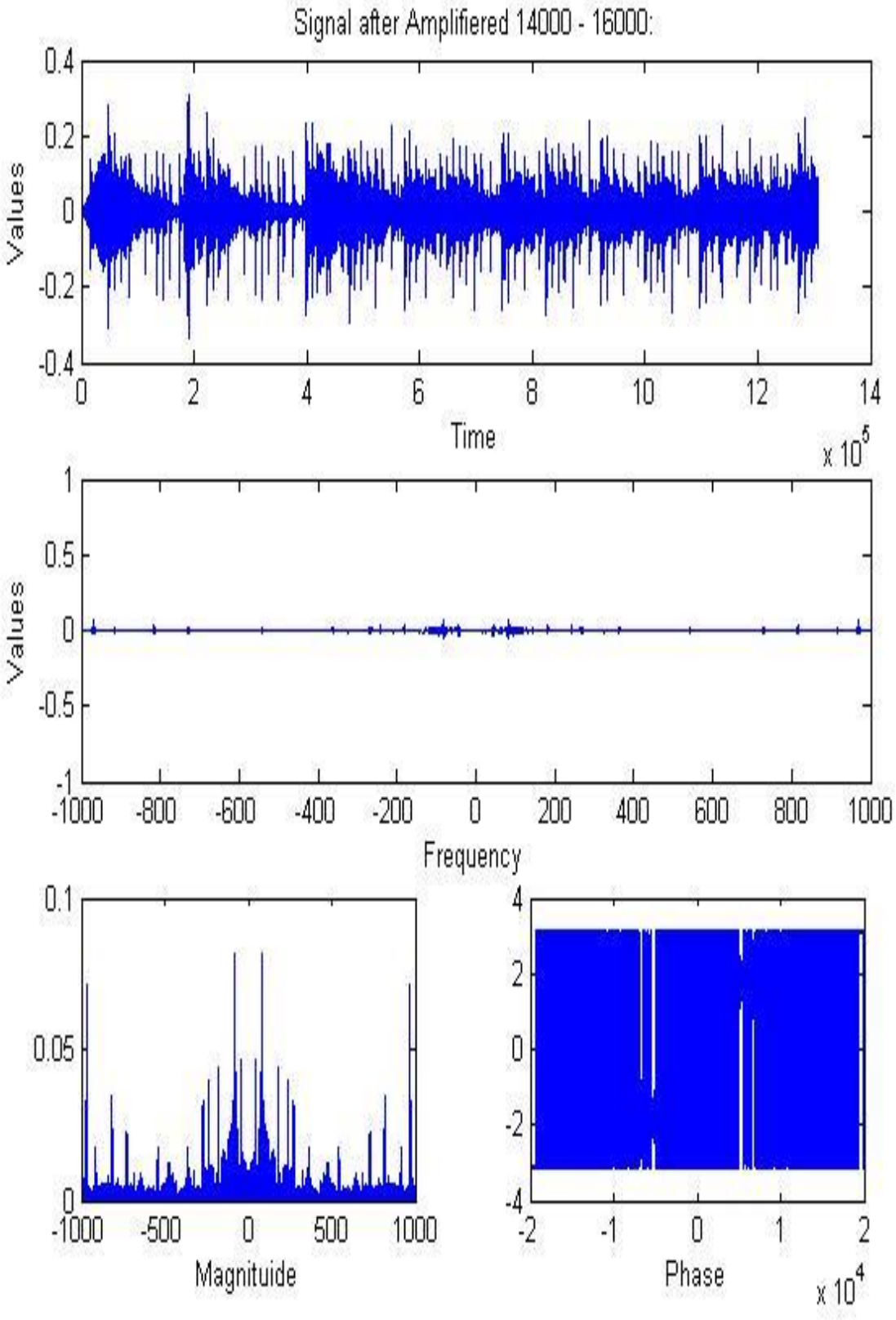

Signal after Amplifiered 600 - 1000:
Values
Time
x 10^5
Values
Frequency
Magnituide
Phase
x 10^4

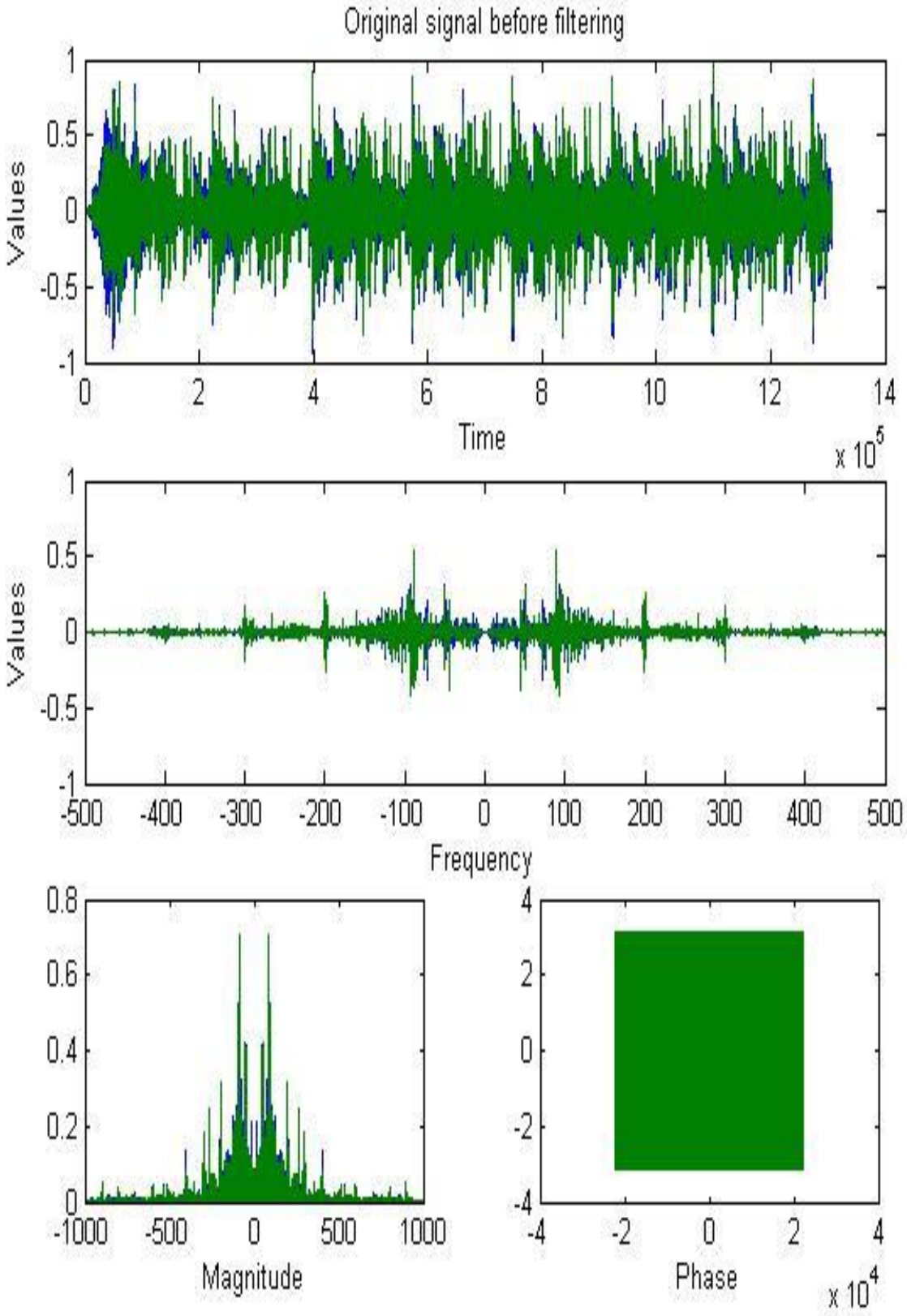

Signal after Amplifiered 1000 - 3000:
Values
Time
x 10^5
Values
Frequency
Magnituide
Phase
x 10^4

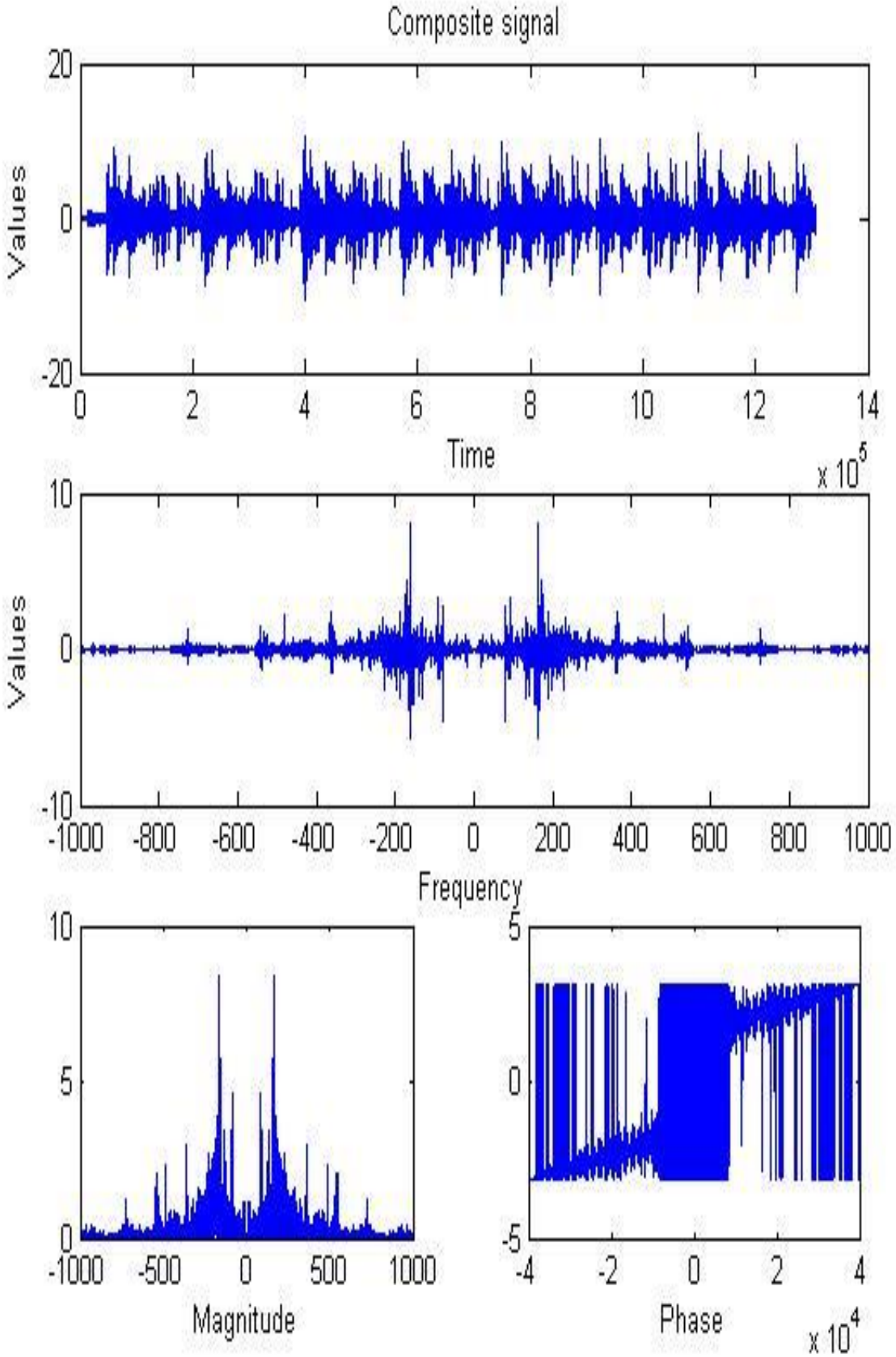

Signal after Amplifiered 3000 - 6000:
Values
Time
x 10^5
Values
Frequency
Magnituide
Phase
x 10^4


Signal after Amplifiered 6000 - 12000:
Values
Time
x 10^5
Values
Frequency
Magnituide
Phase
x 10^4


Signal after Amplifiered 12000 - 14000:
Values
Time
x 10^5
Values
Frequency
Magnituide
Phase
x 10^4

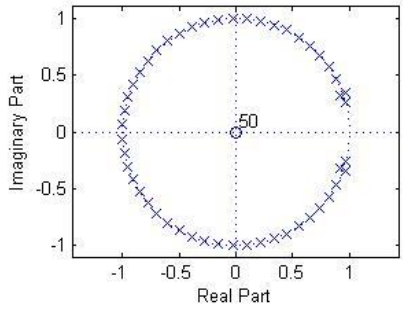

Signal after Amplifiered 14000 - 16000:
Values
Time
x 10^5
Values
Frequency
Magnituide
Phase
x 10^4

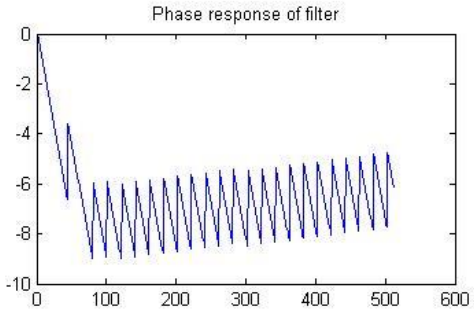

Original signal before filtering
Values
Time
x 10^5
Values
Frequency
Magnitude
Phase
x 10^4

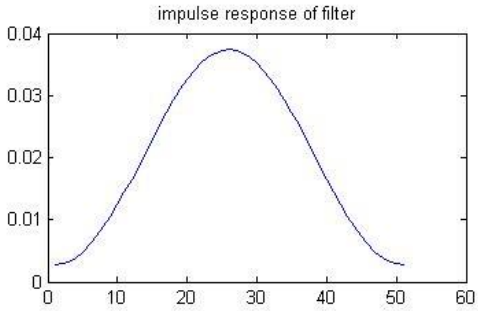

Composite signal
Values
Time
x 10^5
Values
Frequency
Magnitude
Phase
x 10^4

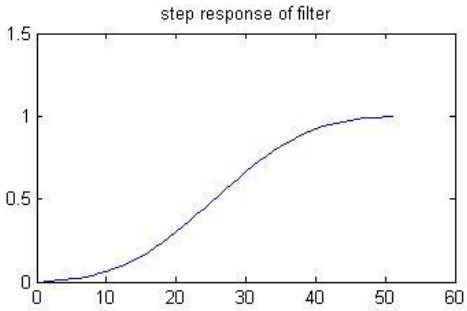
Filter analysis of each:

Filter of range 0 -> 170 Hz

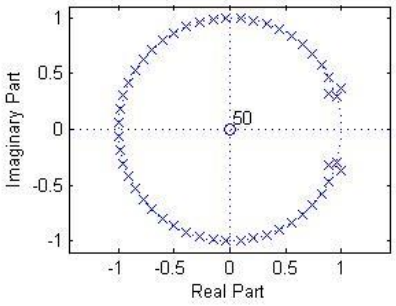


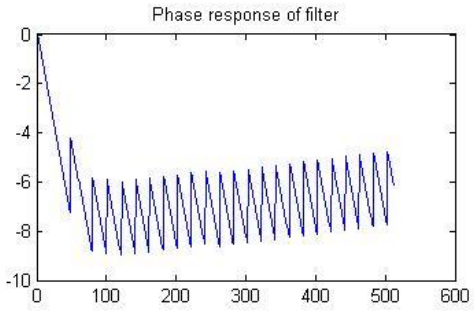


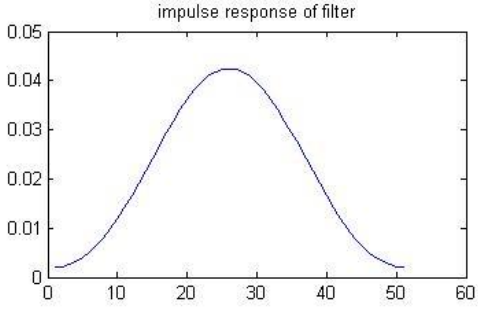


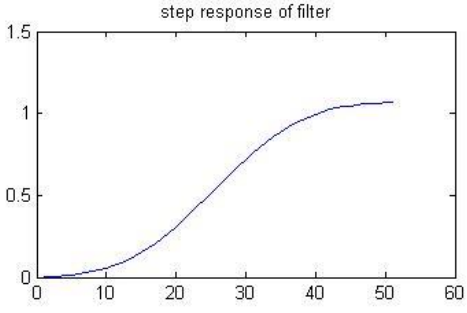


Filter of range 170 -> 310 Hz

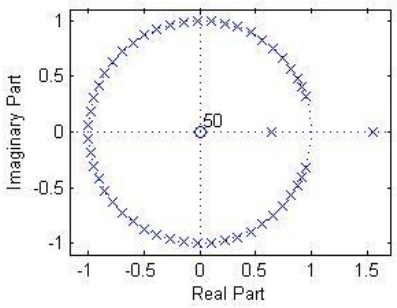


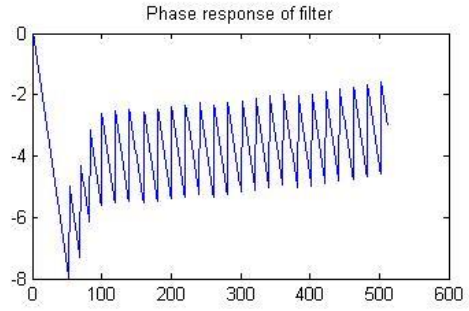


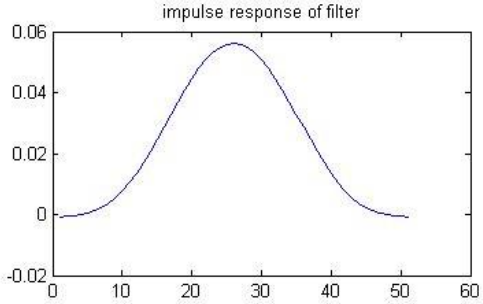


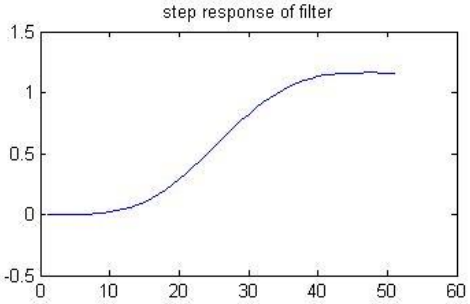


Filter of range 310 -> 600 Hz

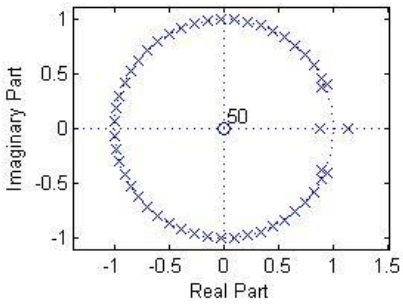


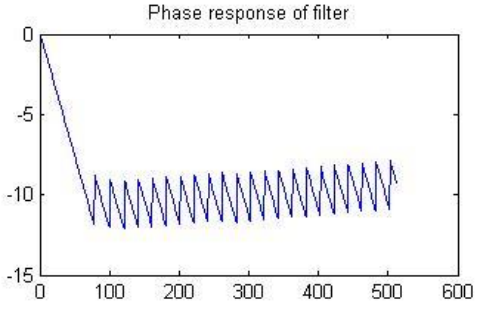


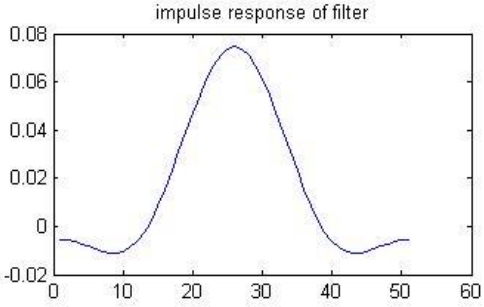


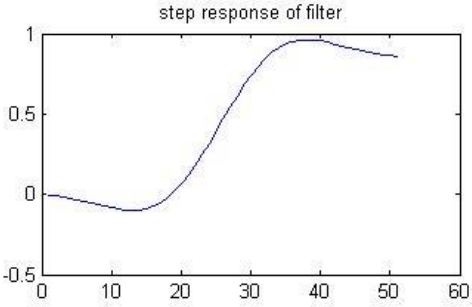


Filter of range 600 -> 1000 Hz

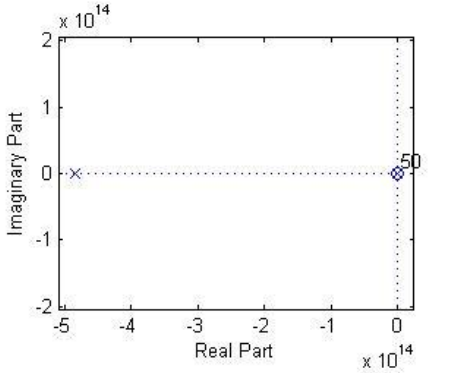


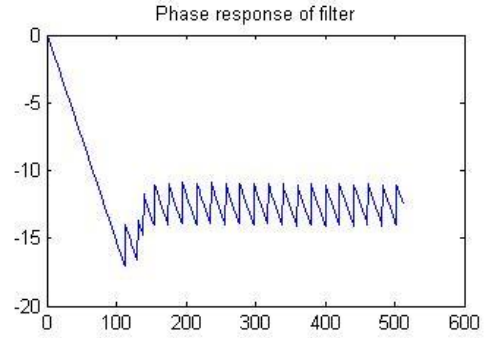


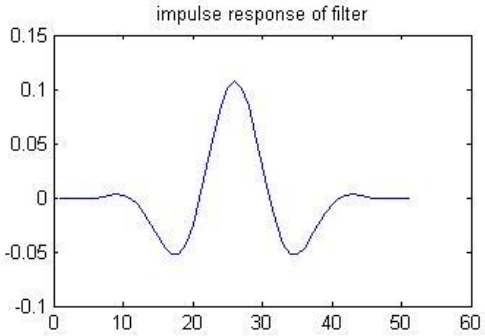


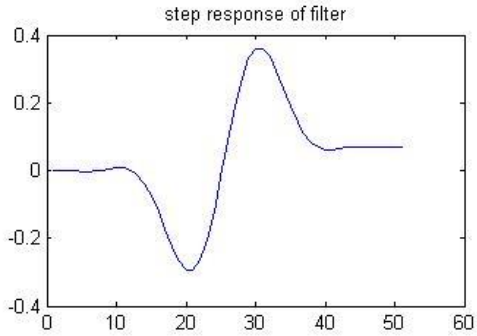


Filter of range 1 -> 3 kHz

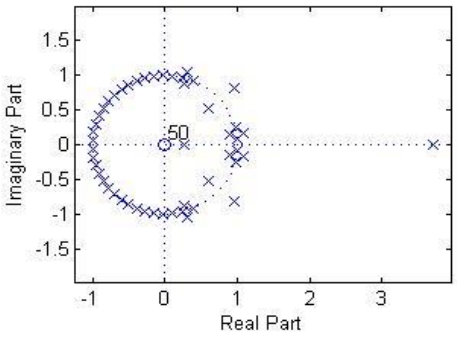


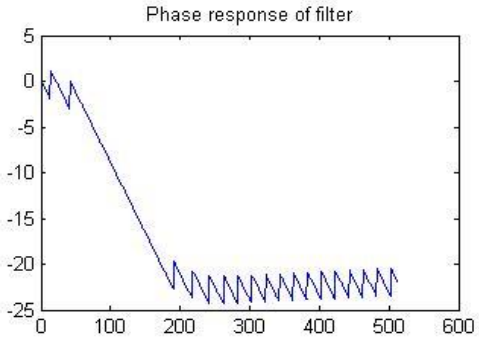


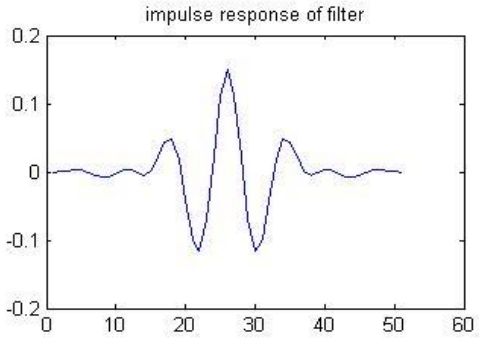


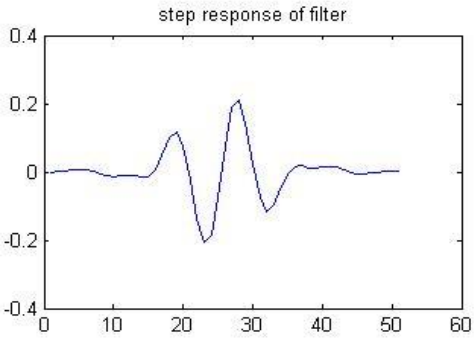


Filter of range 3 -> 6 kHz

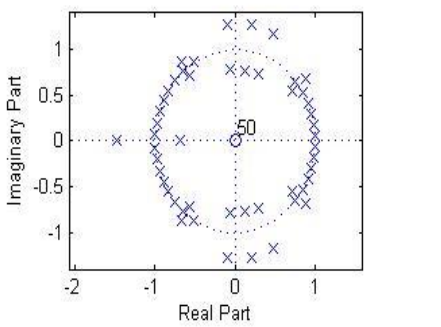


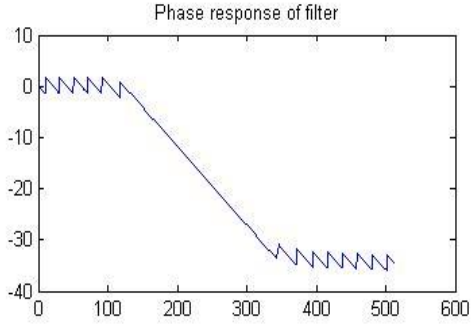


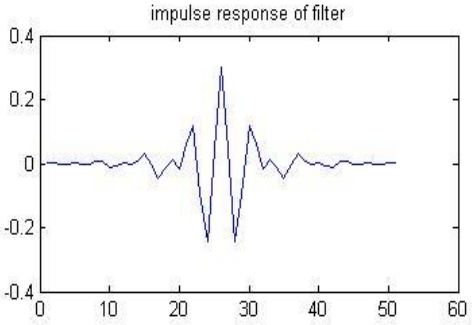


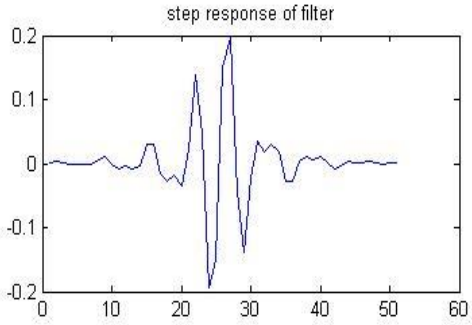


Filter of range 6 -> 12 kHz

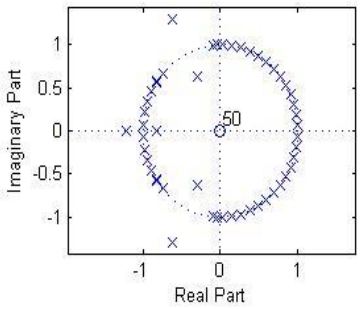


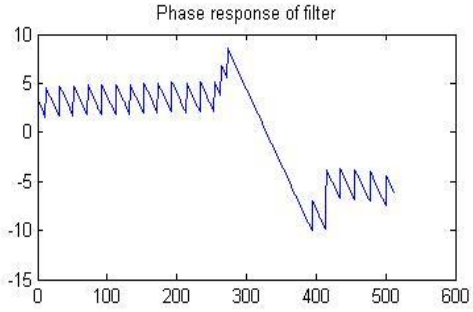


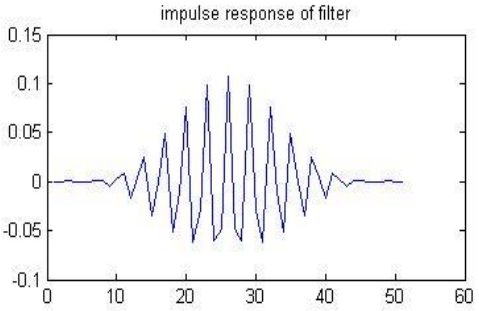


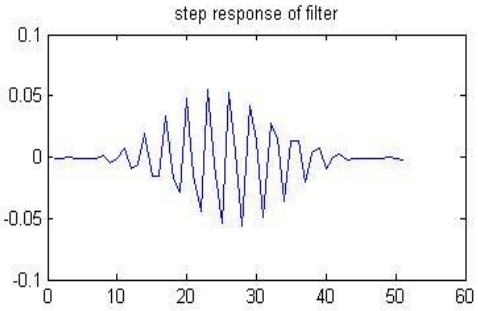


Filter of range 12 -> 14 kHz

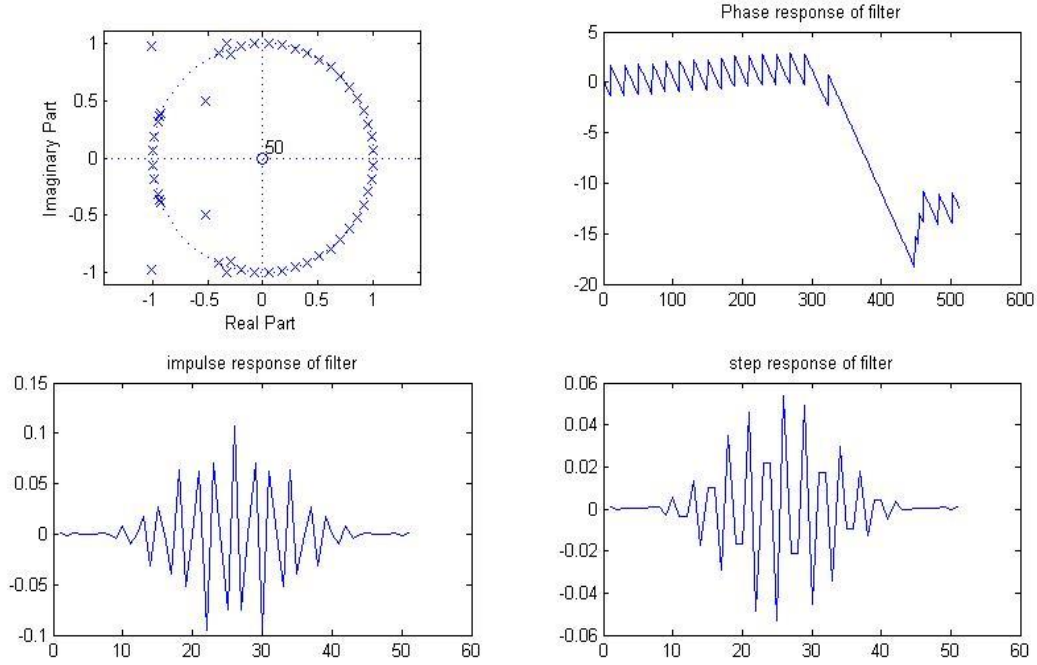




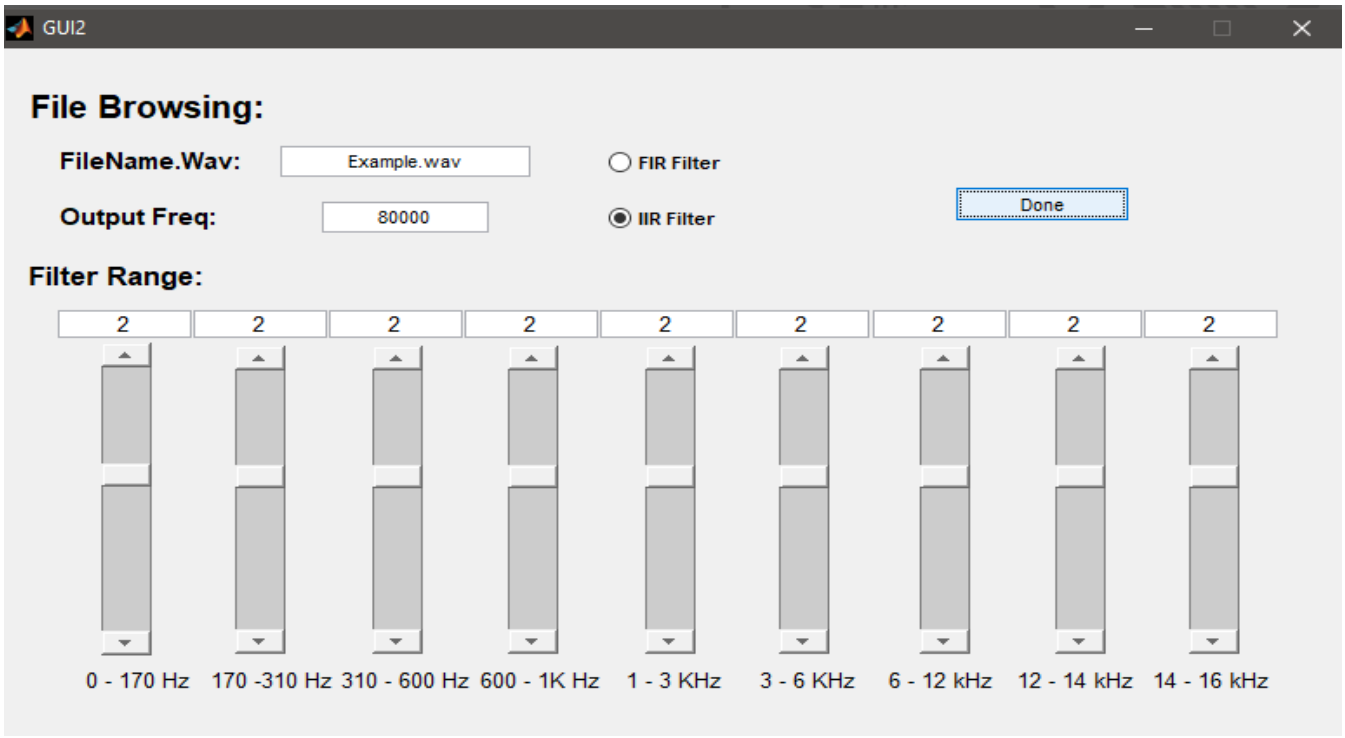


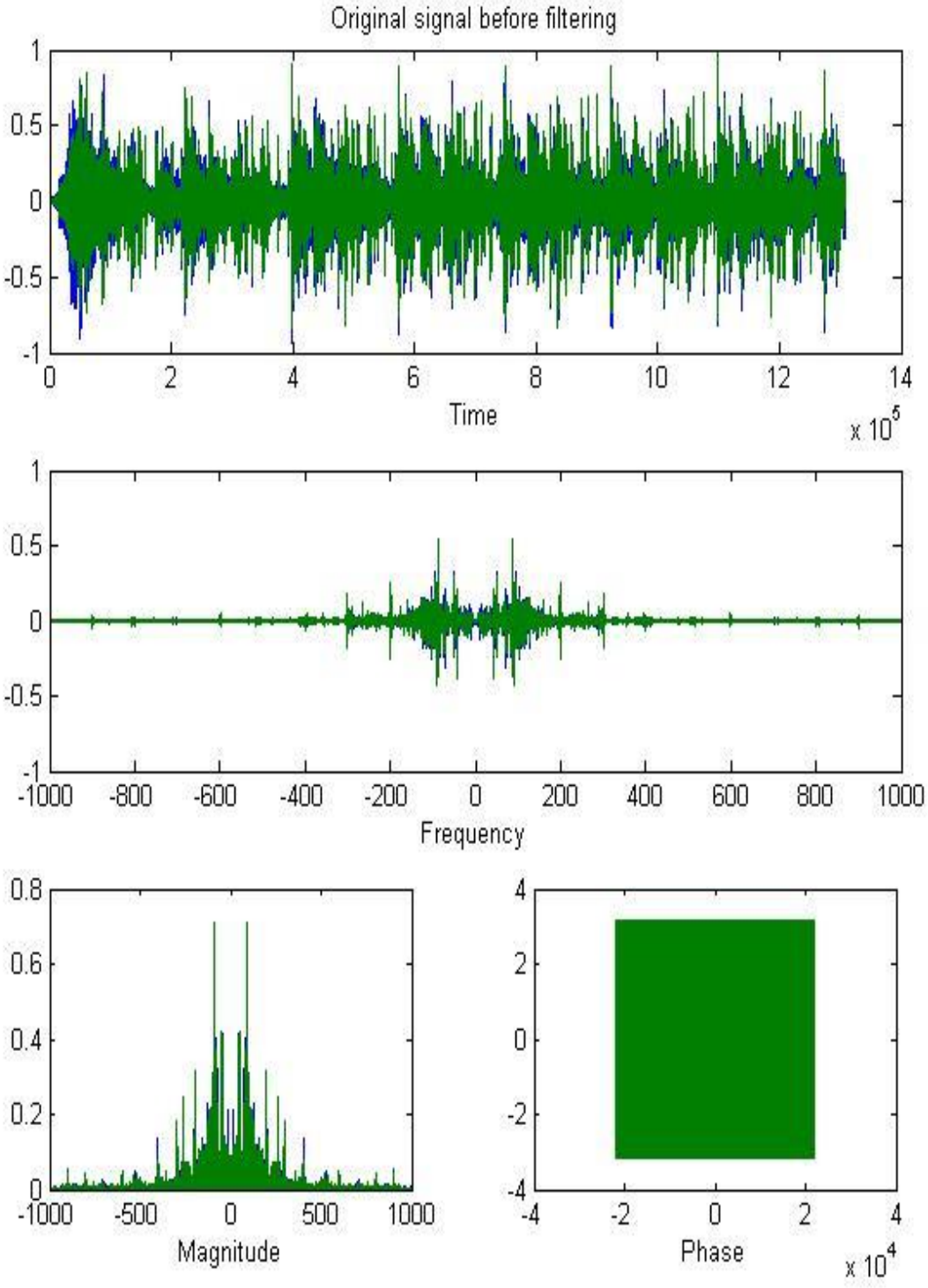


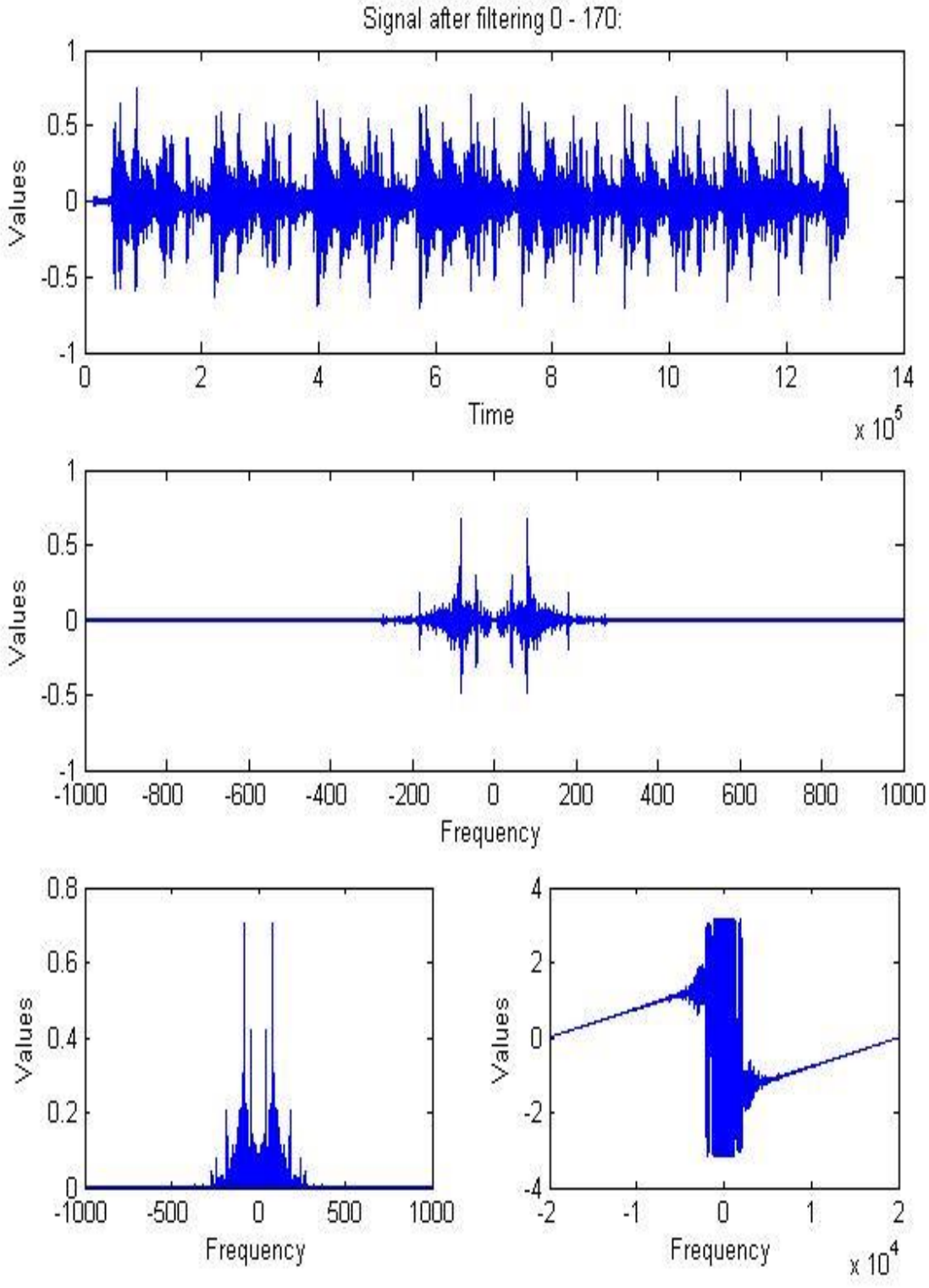
Filter of range 14 -> 16kHz

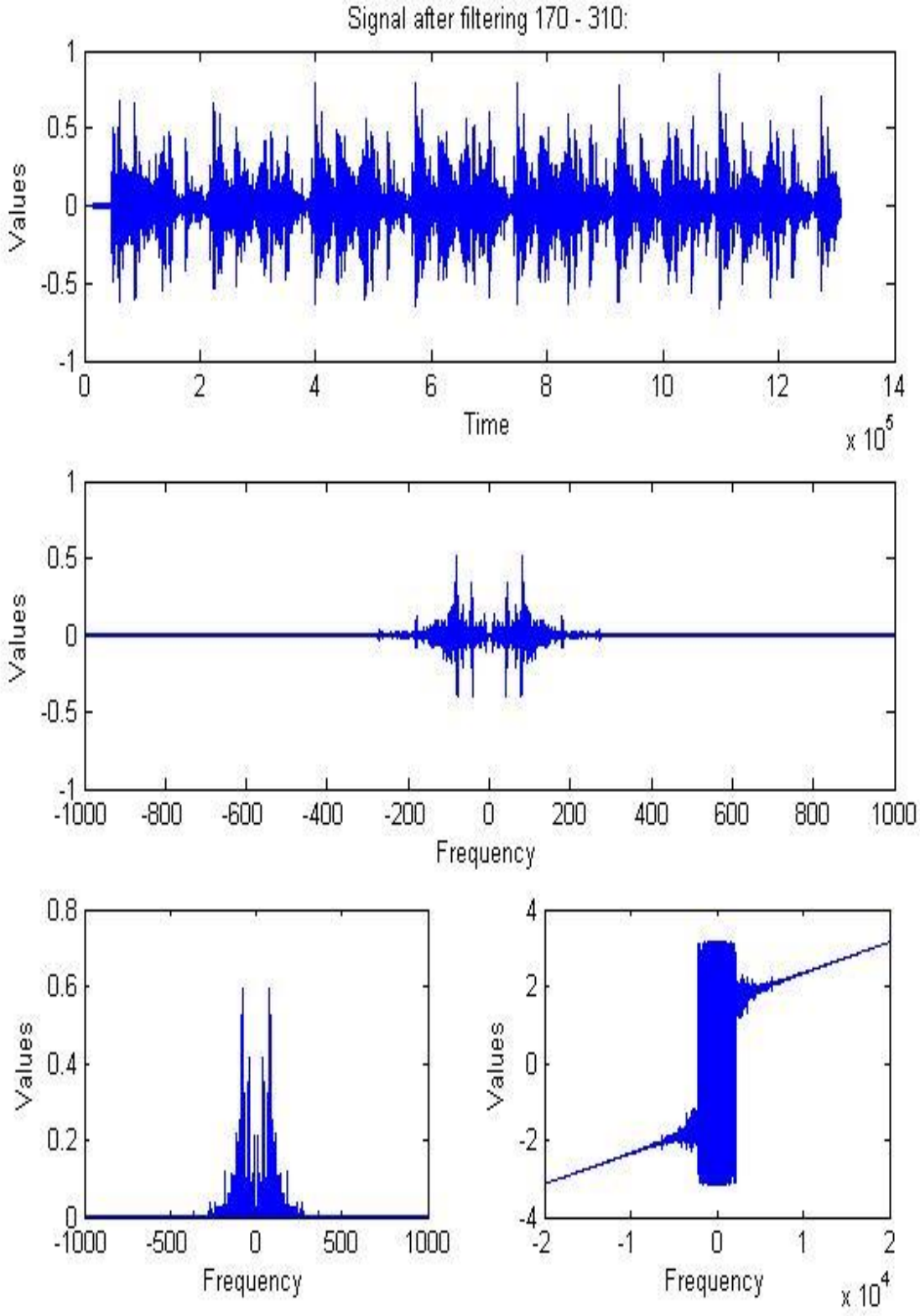


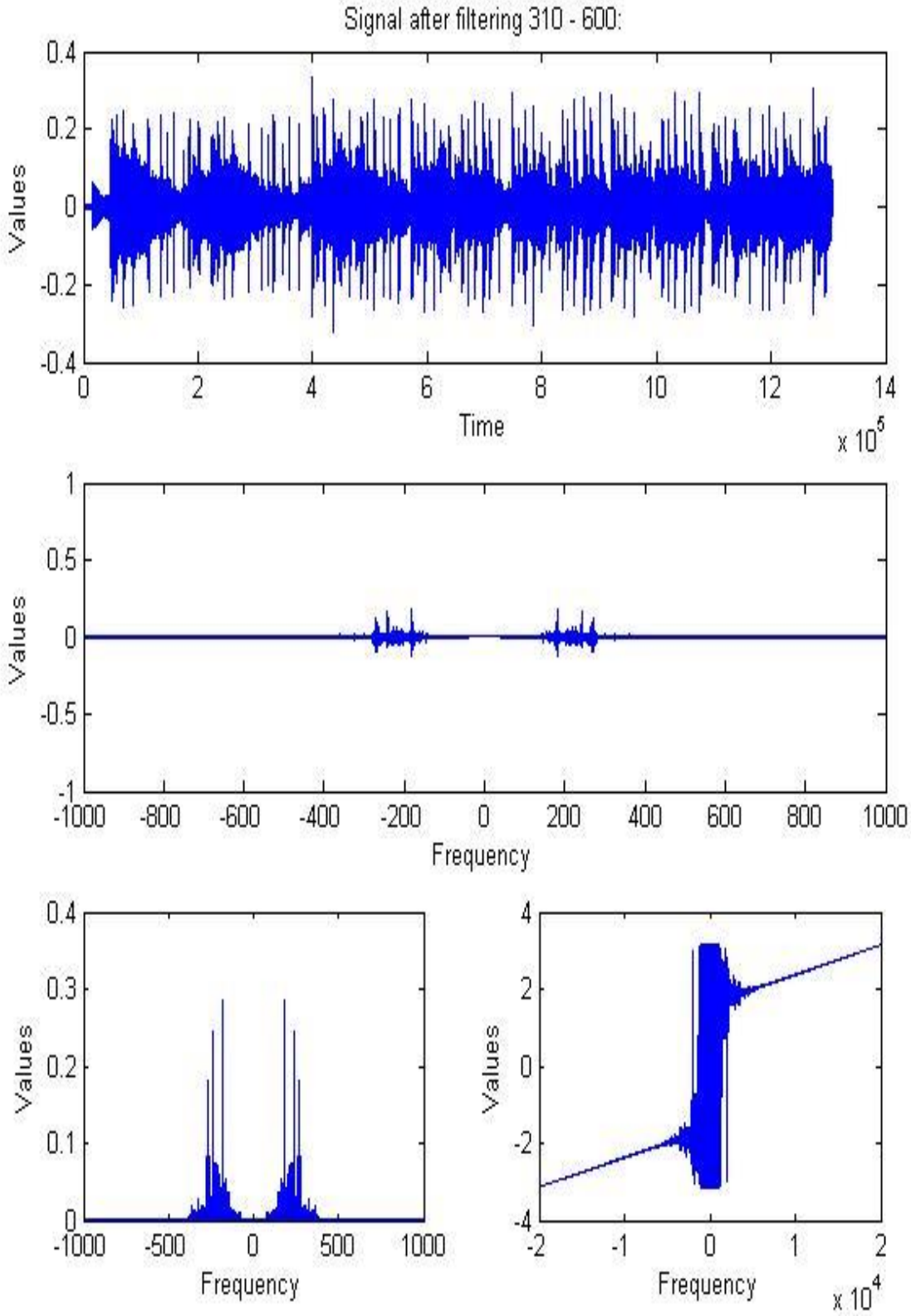
IIR Filter 2st with order of 50 and gain of 2 db

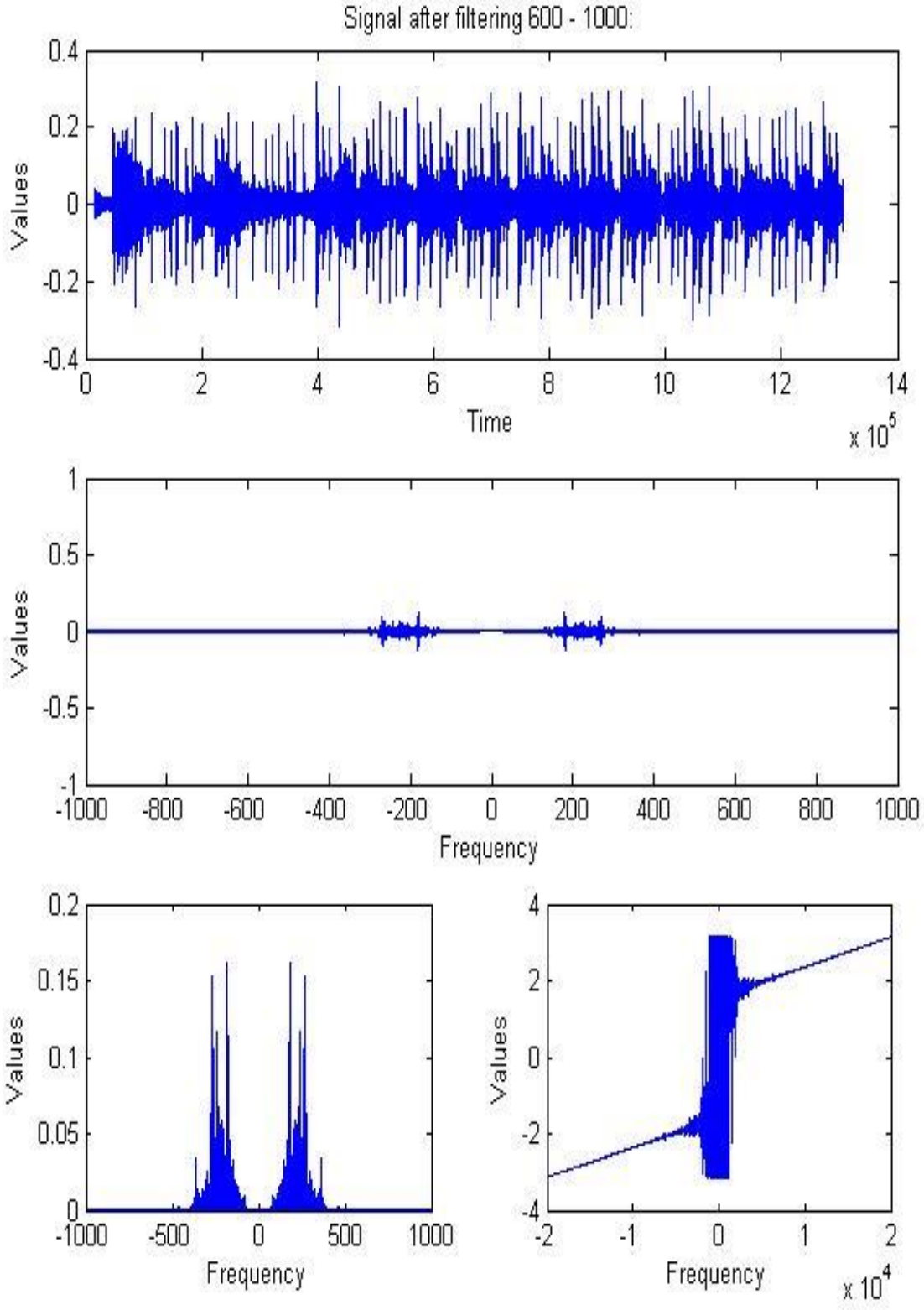


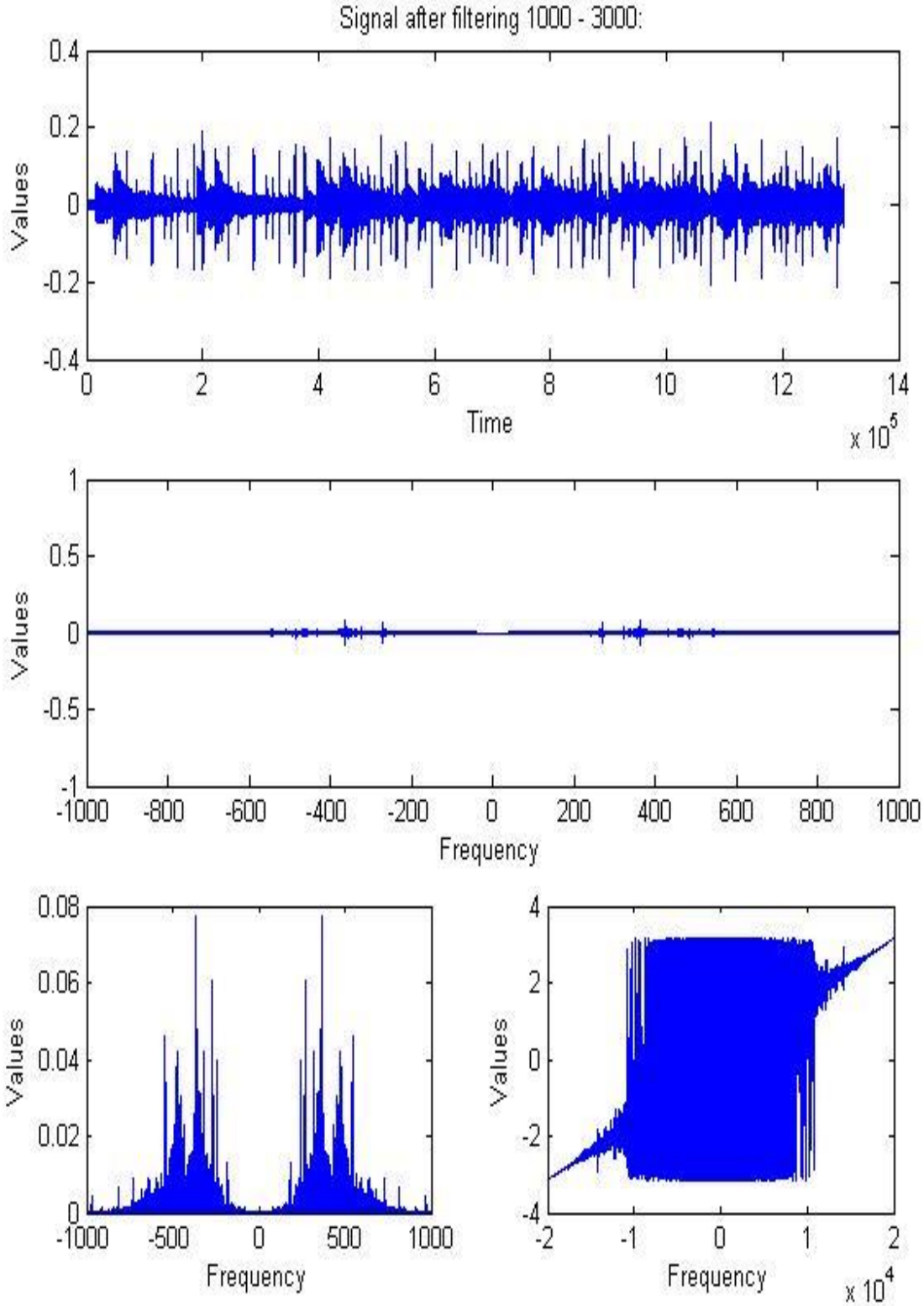

Original signal before filtering
Time
x 10^5
Frequency
Magnitude
Phase
x 10^4

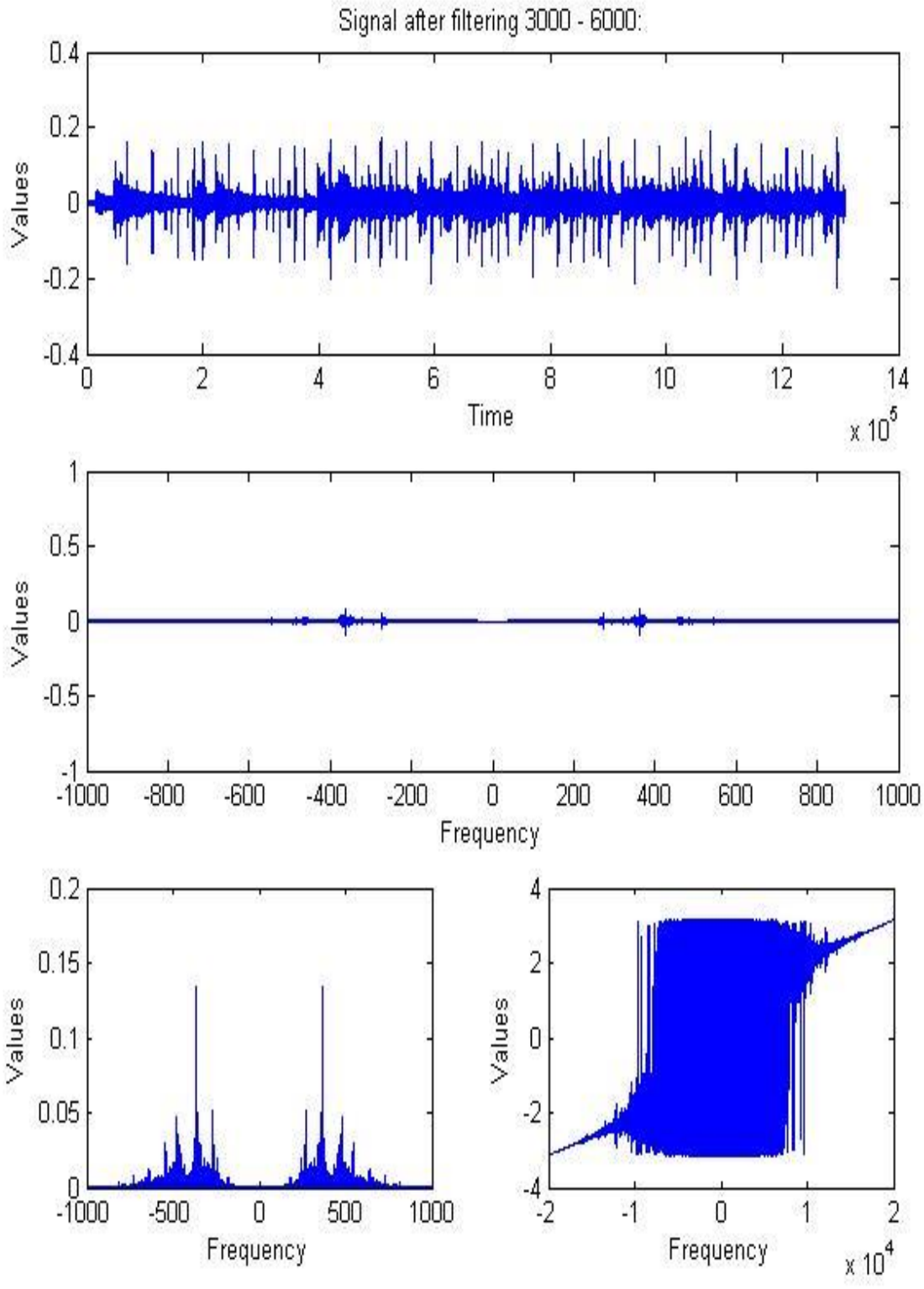

Signal after filtering 0 - 170:
Values
Time
x 10^5
Values
Frequency
Values
Frequency
Values
Frequency
x 10^4

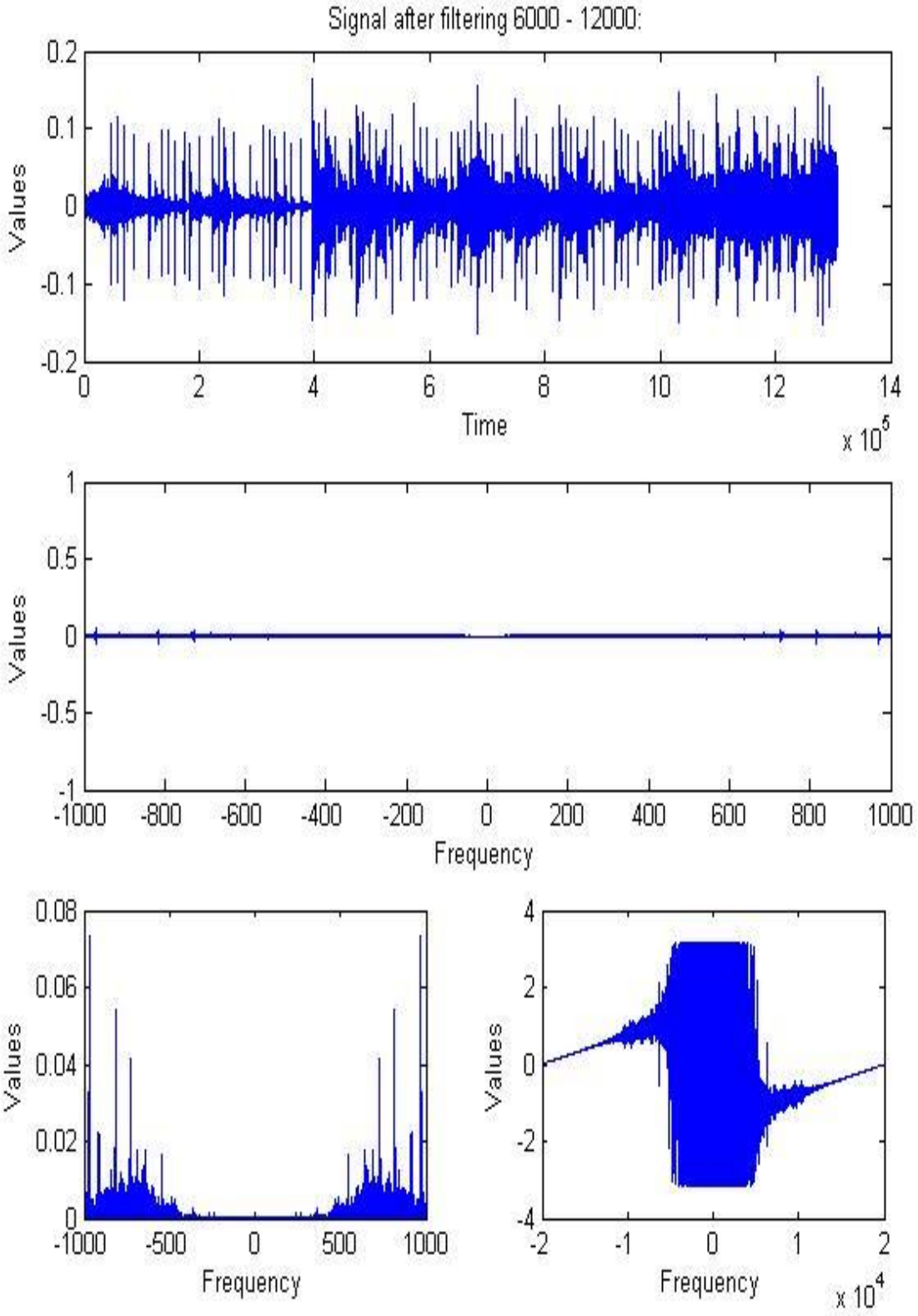

Signal after filtering 170 - 310:
Values
Time
x 10^5
Values
Frequency
Values
Frequency
Values
Frequency
x 10^4

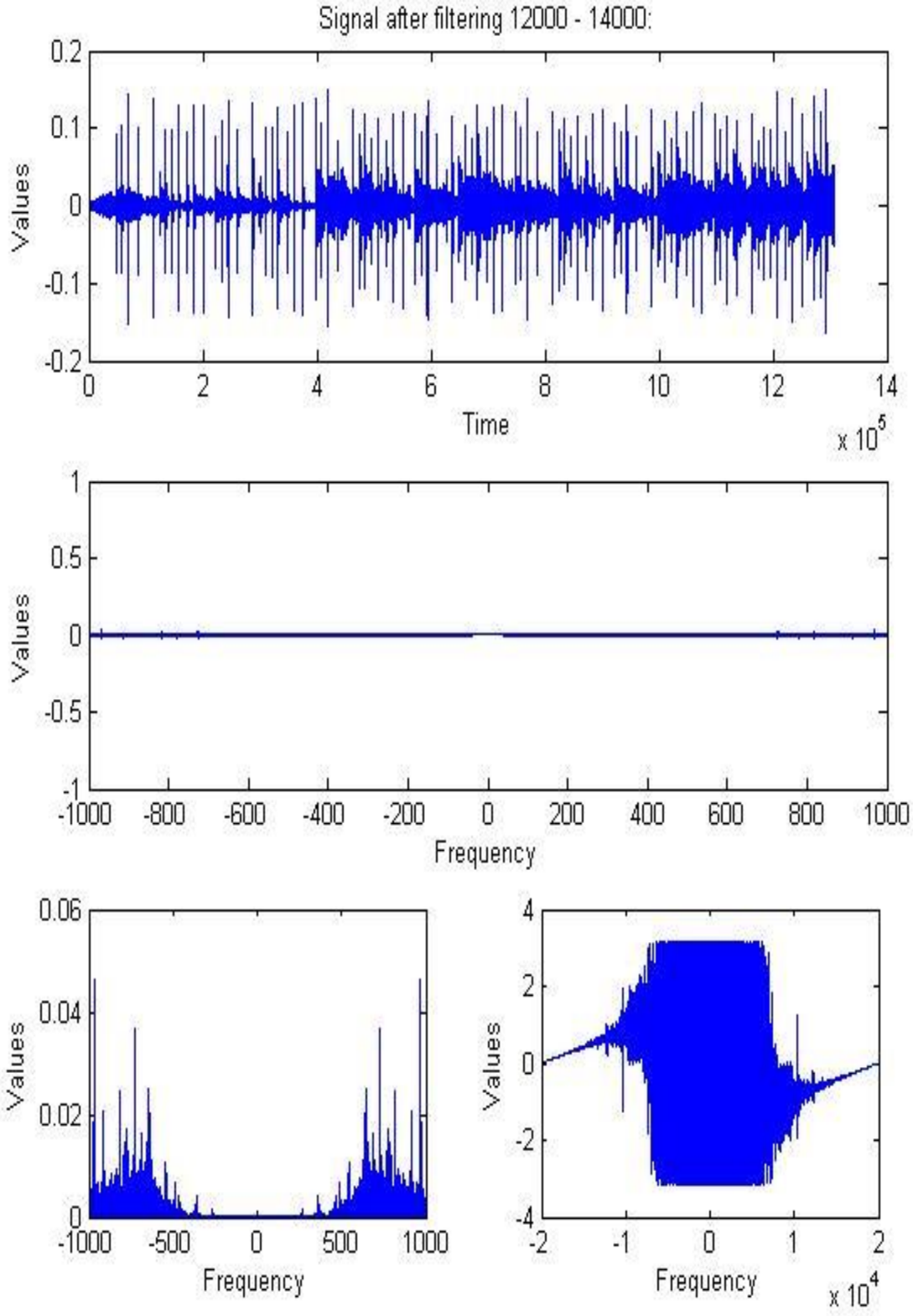

Signal after filtering 310 - 600:
Values
Time
x 10^5
Values
Frequency
Values
Frequency
Values
Frequency
x 10^4

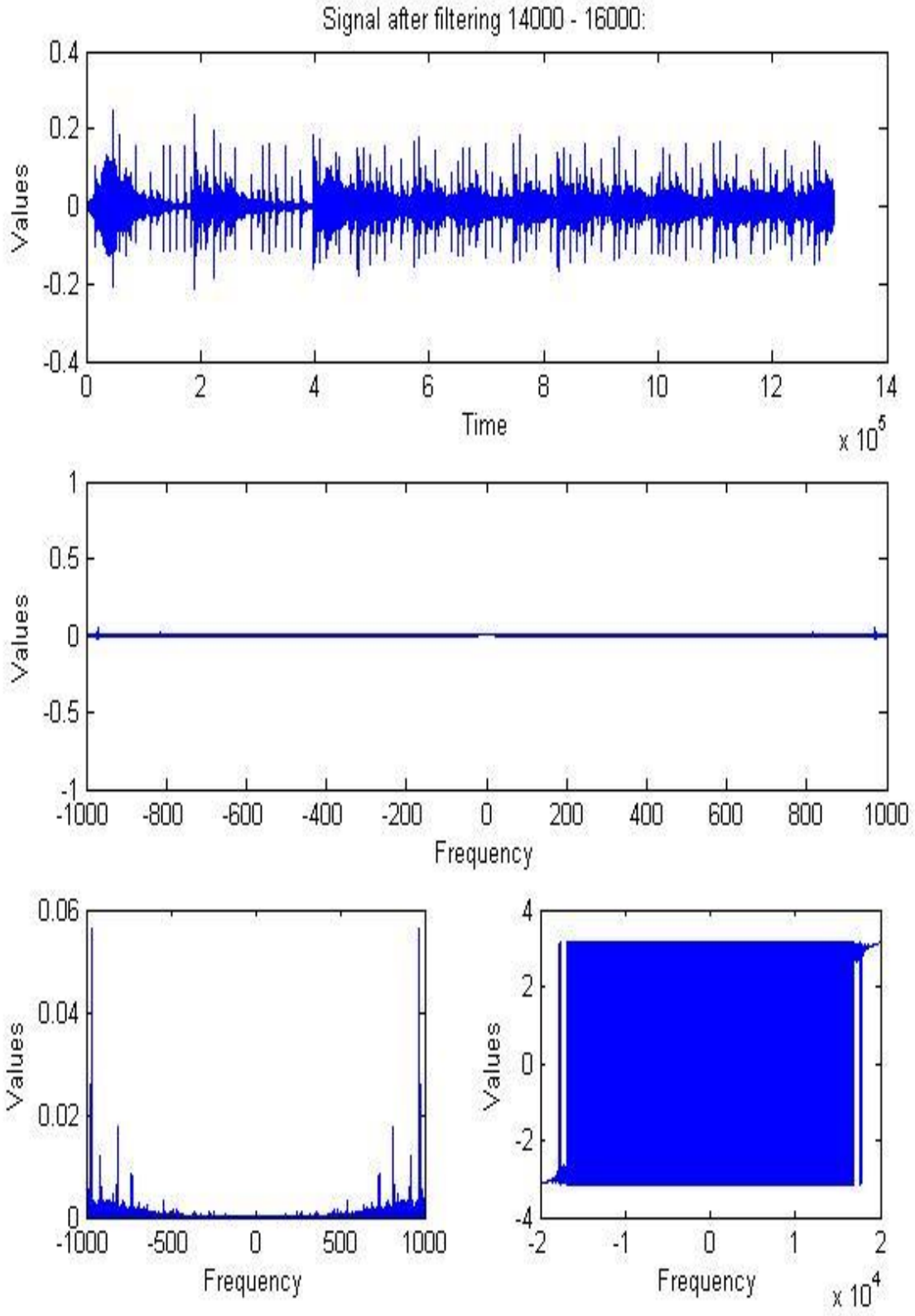

Signal after filtering 600 - 1000:
Values
Time
x 10^5
Values
Frequency
Values
Frequency
Values
Frequency
x 10^4

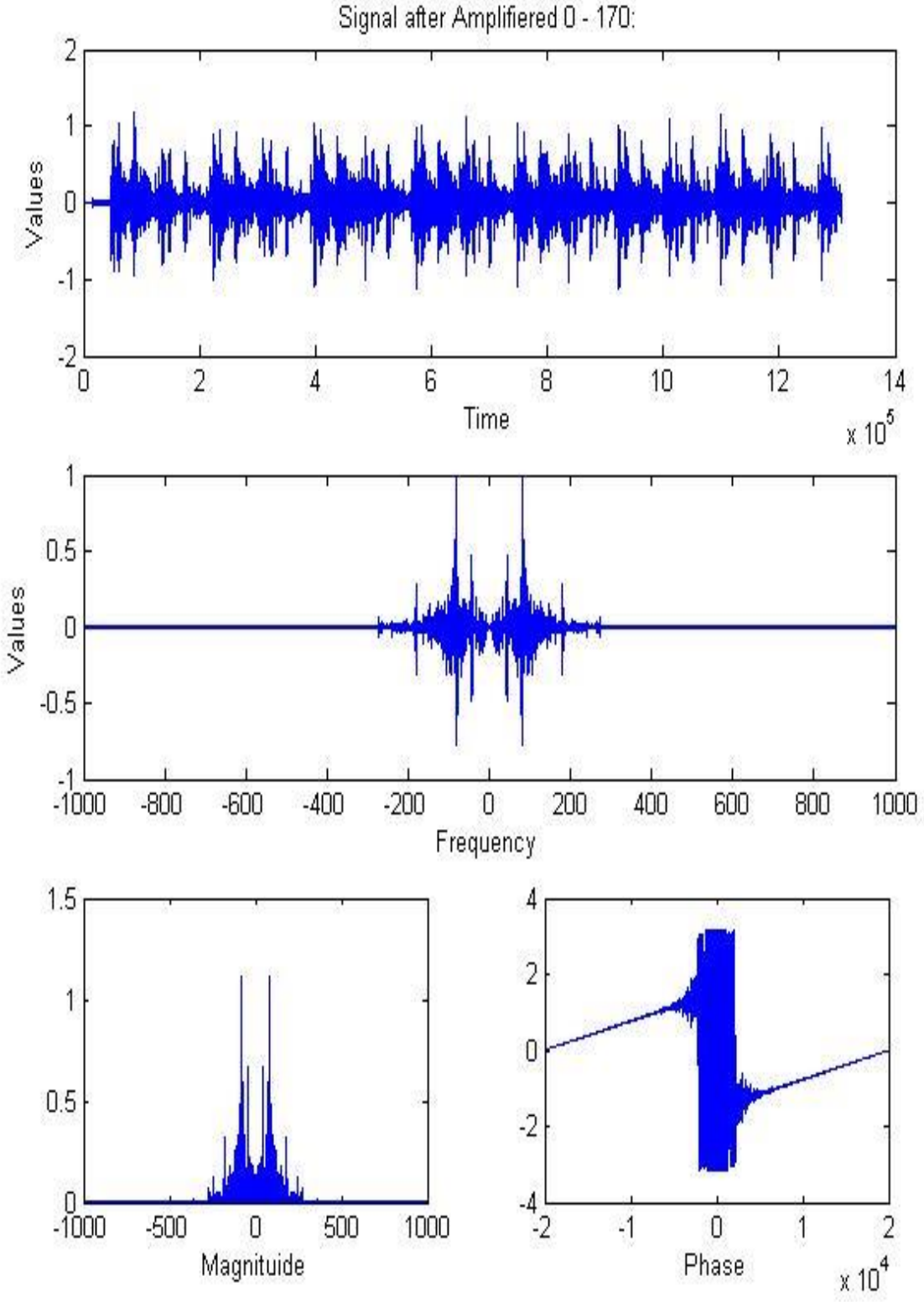

Signal after filtering 1000 - 3000:
Values
Time
x 10^5
Values
Frequency
Values
Frequency
Values
Frequency
x 10^4

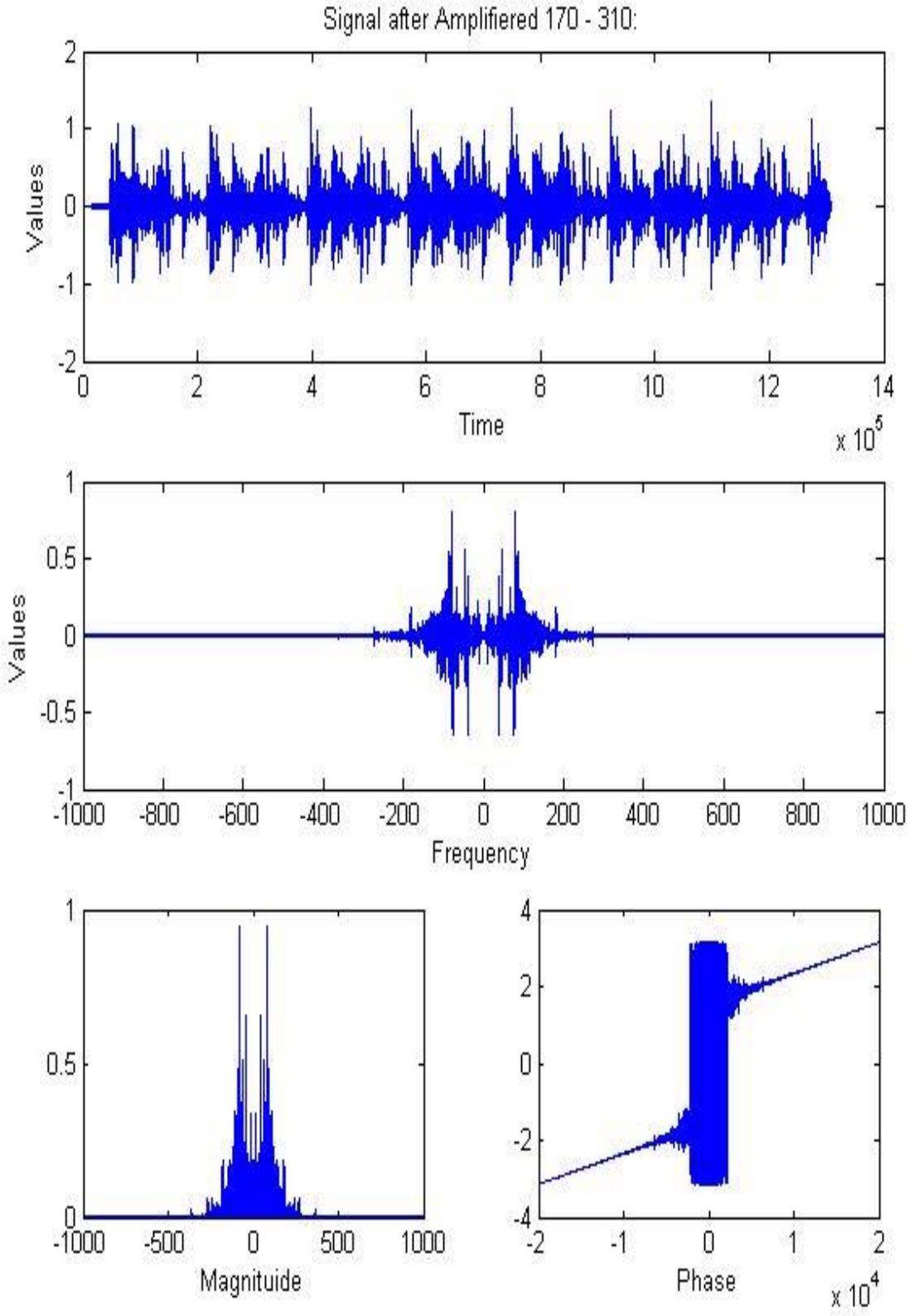

Signal after filtering 3000 - 6000:
Values
Time
x 10^5
Values
Frequency
Values
Frequency
Values
Frequency
x 10^4

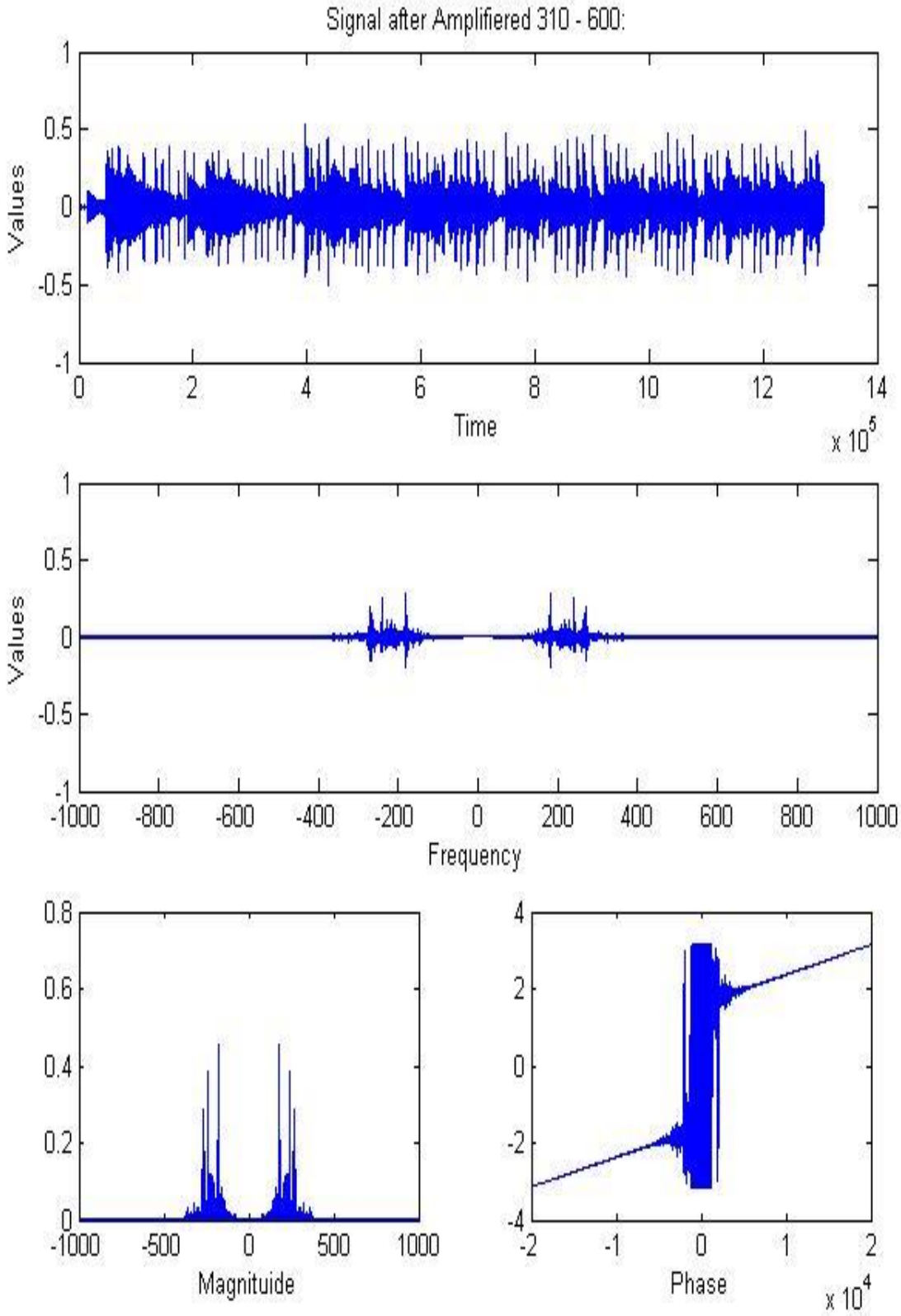

Signal after filtering 6000 - 12000:
Values
Time
x 10^5
Values
Frequency
Values
Frequency
Values
Frequency
x 10^4

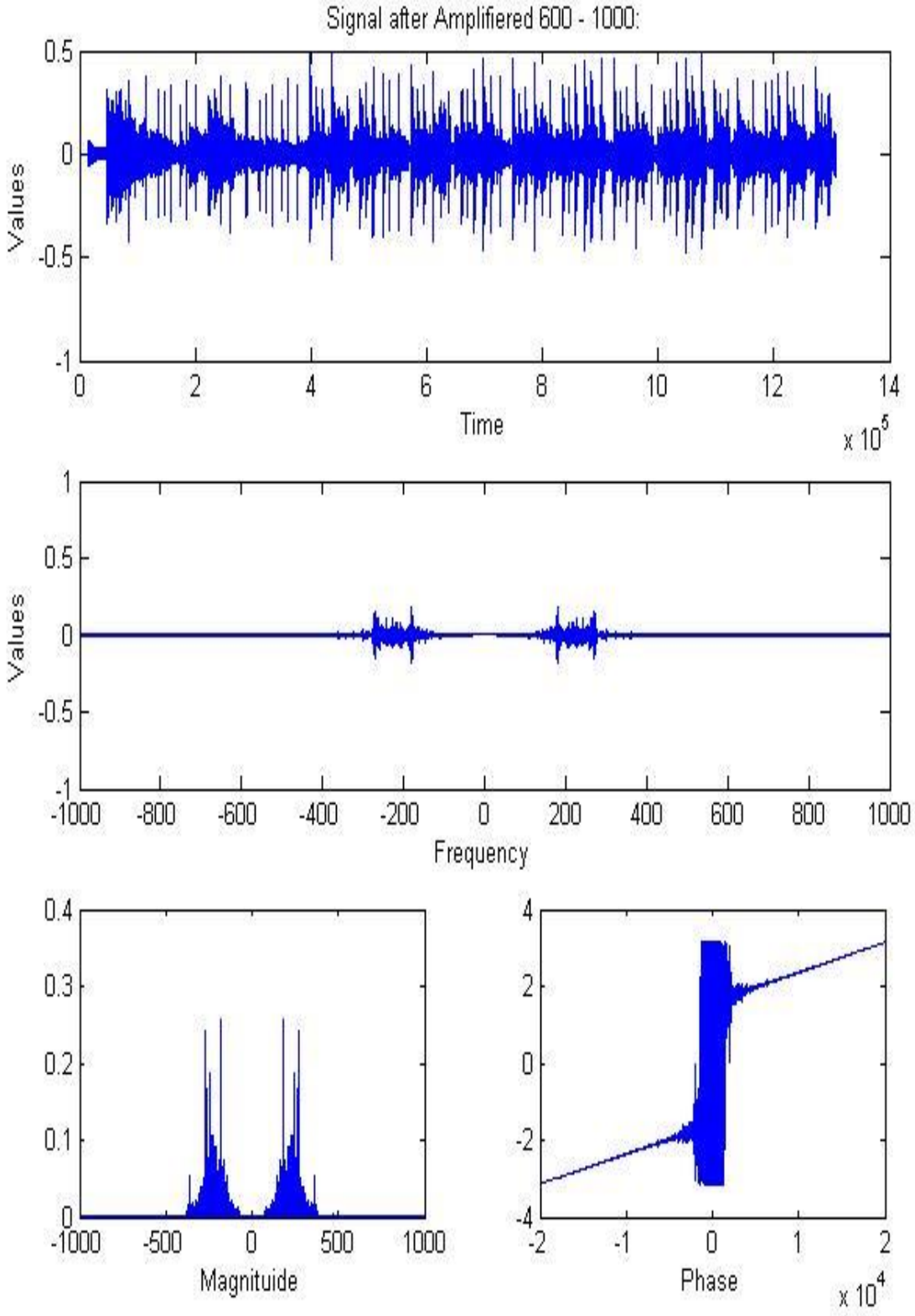

Signal after filtering 12000 - 14000:
Values
Time
x 10^5
Values
Frequency
Values
Frequency
Values
Frequency
x 10^4

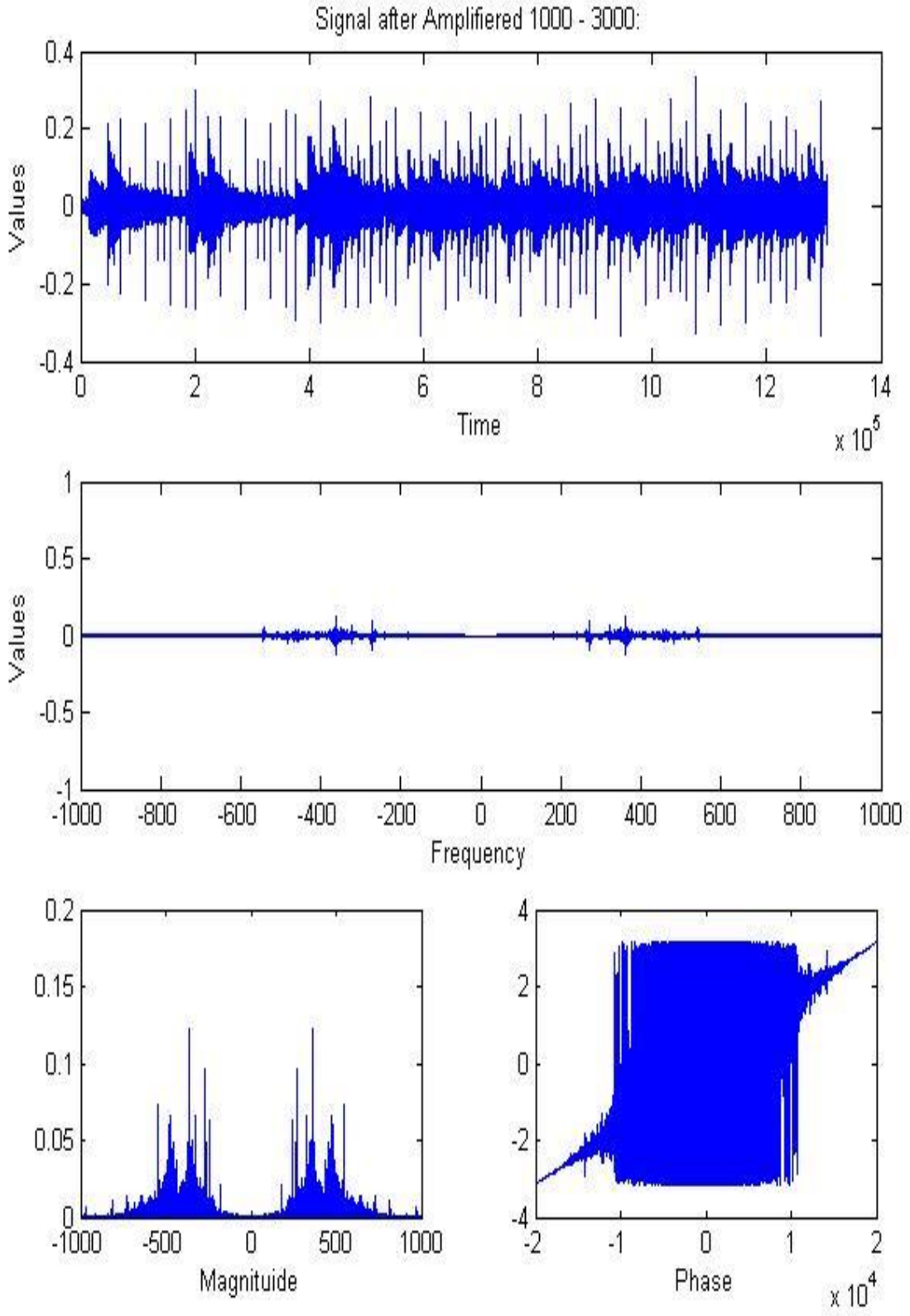

Signal after filtering 14000 - 16000:
Values
Time
x 10^5
Frequency
Frequency
Values
Frequency
x 10^4

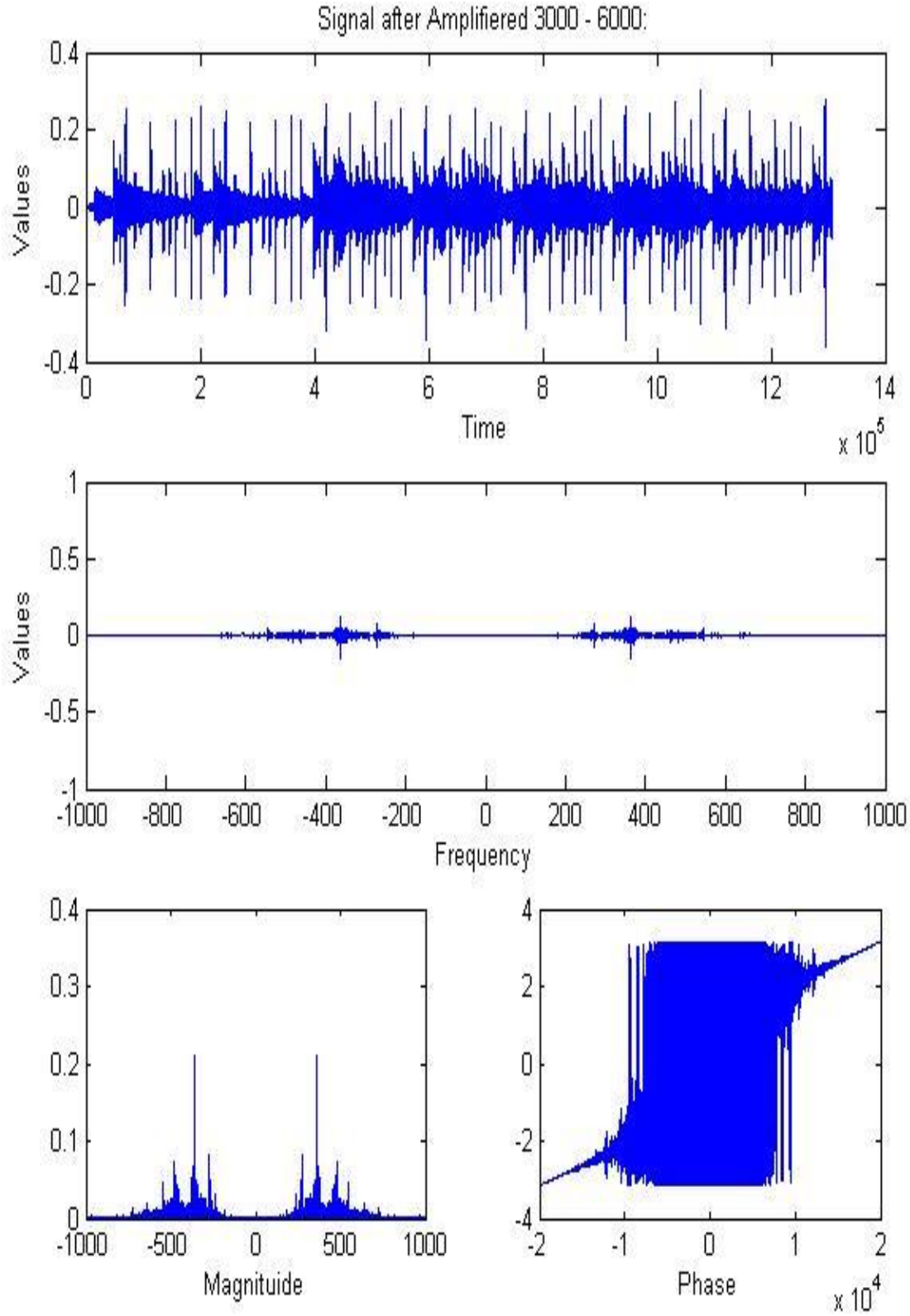

Signal after Amplifiered 0 - 170:
Values
Time
x 10^5
Values
Frequency
Magnituide
Phase
x 10^4

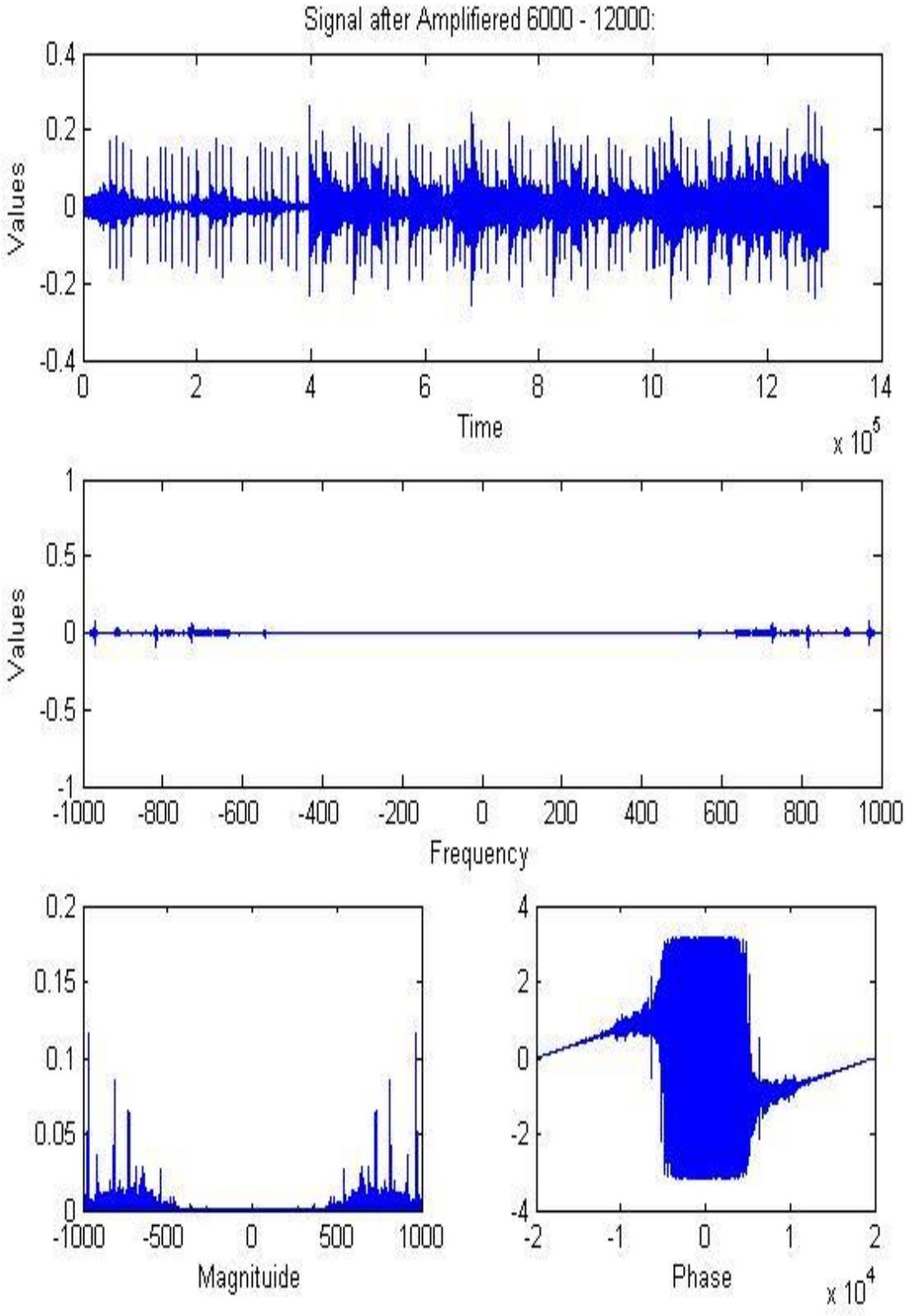

Signal after Amplifiered 170 - 310:
Values
Time
x 10^5
Values
Frequency
Magnituide
Phase
x 10^4

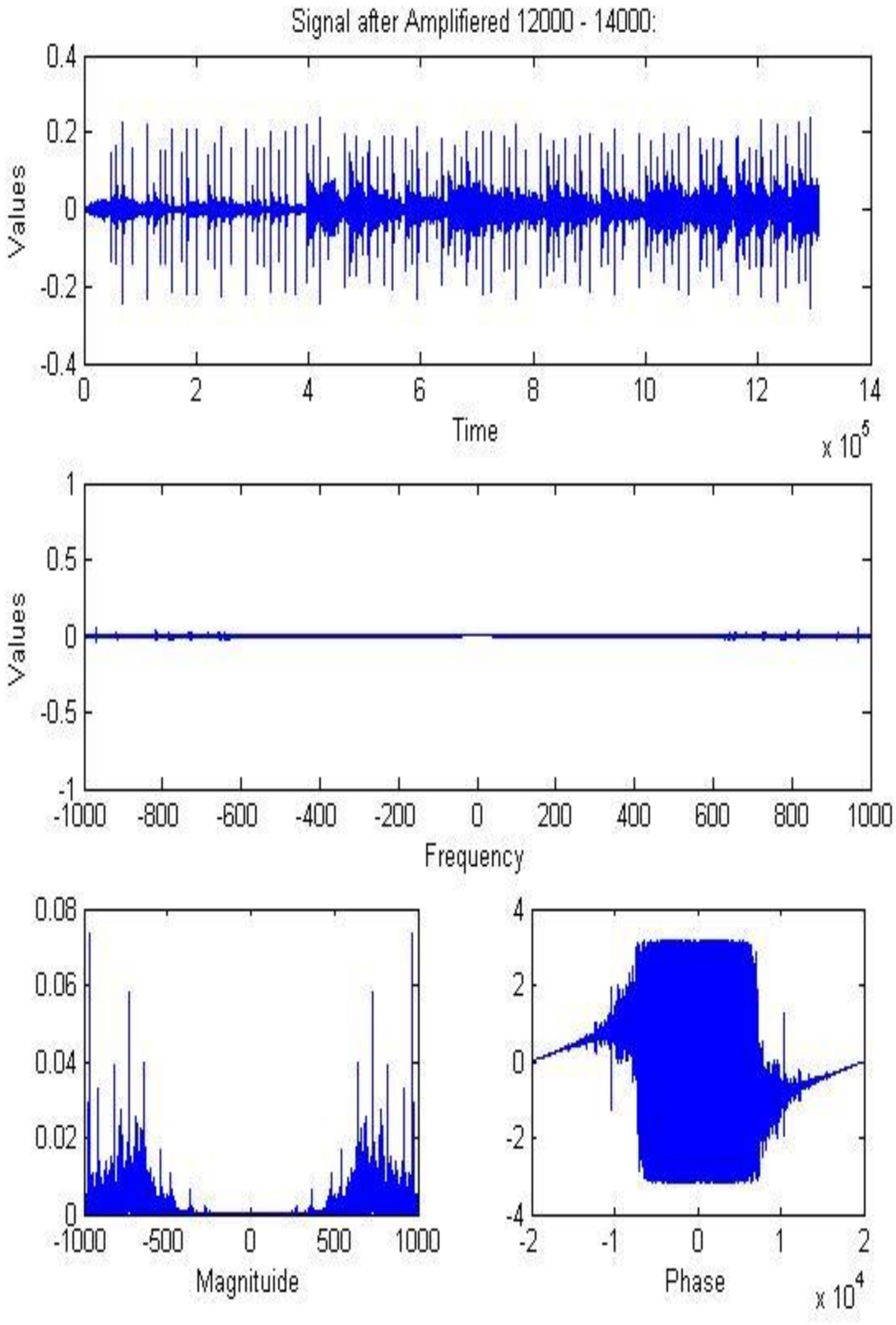

Signal after Amplifiered 310 - 600:
Values
Time
x 10^5
Values
Frequency
Magnituide
Phase
x 10^4

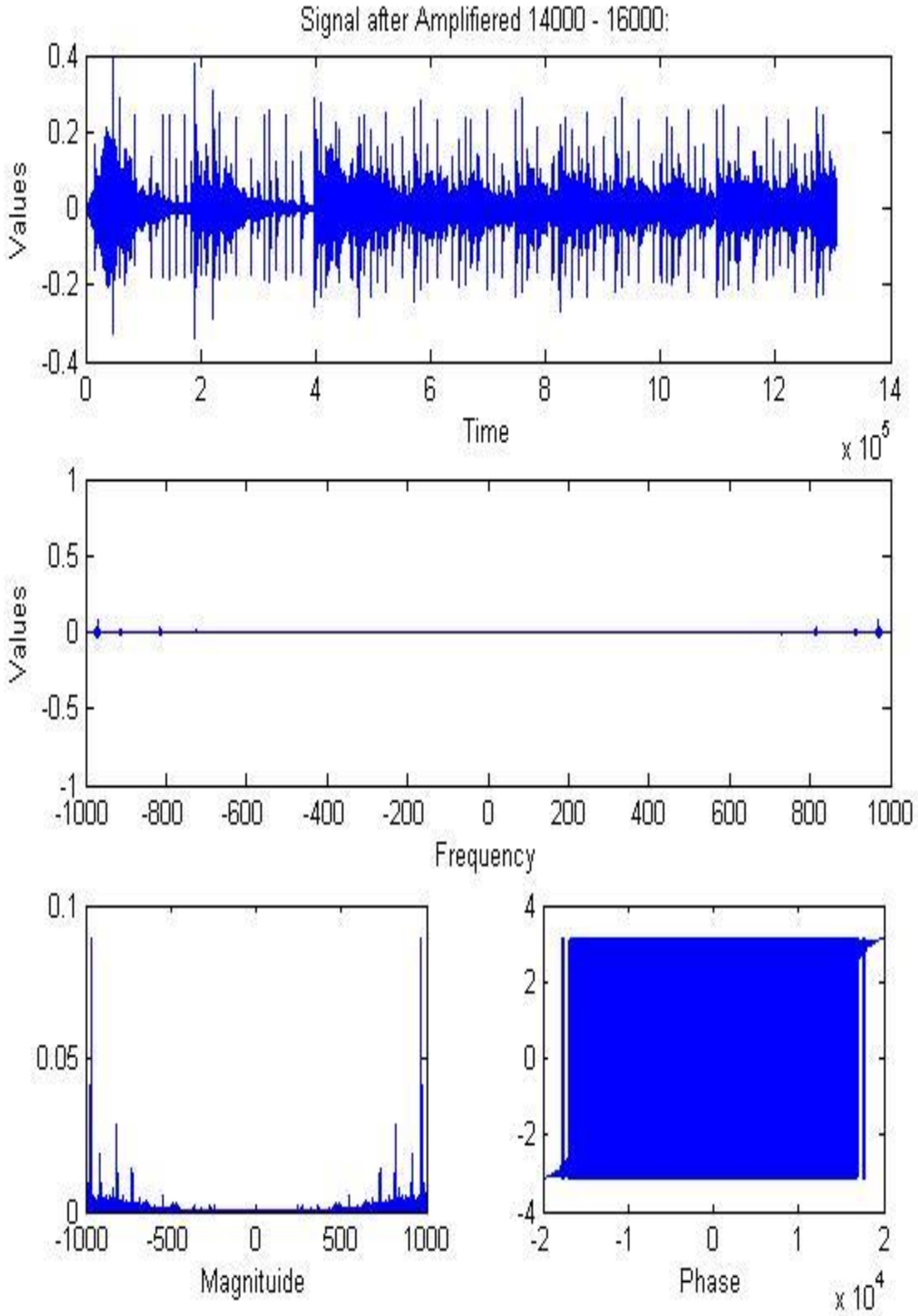

Signal after Amplifiered 600 - 1000:
Values
Time
x 10^5
Values
Frequency
Magnituide
Phase
x 10^4

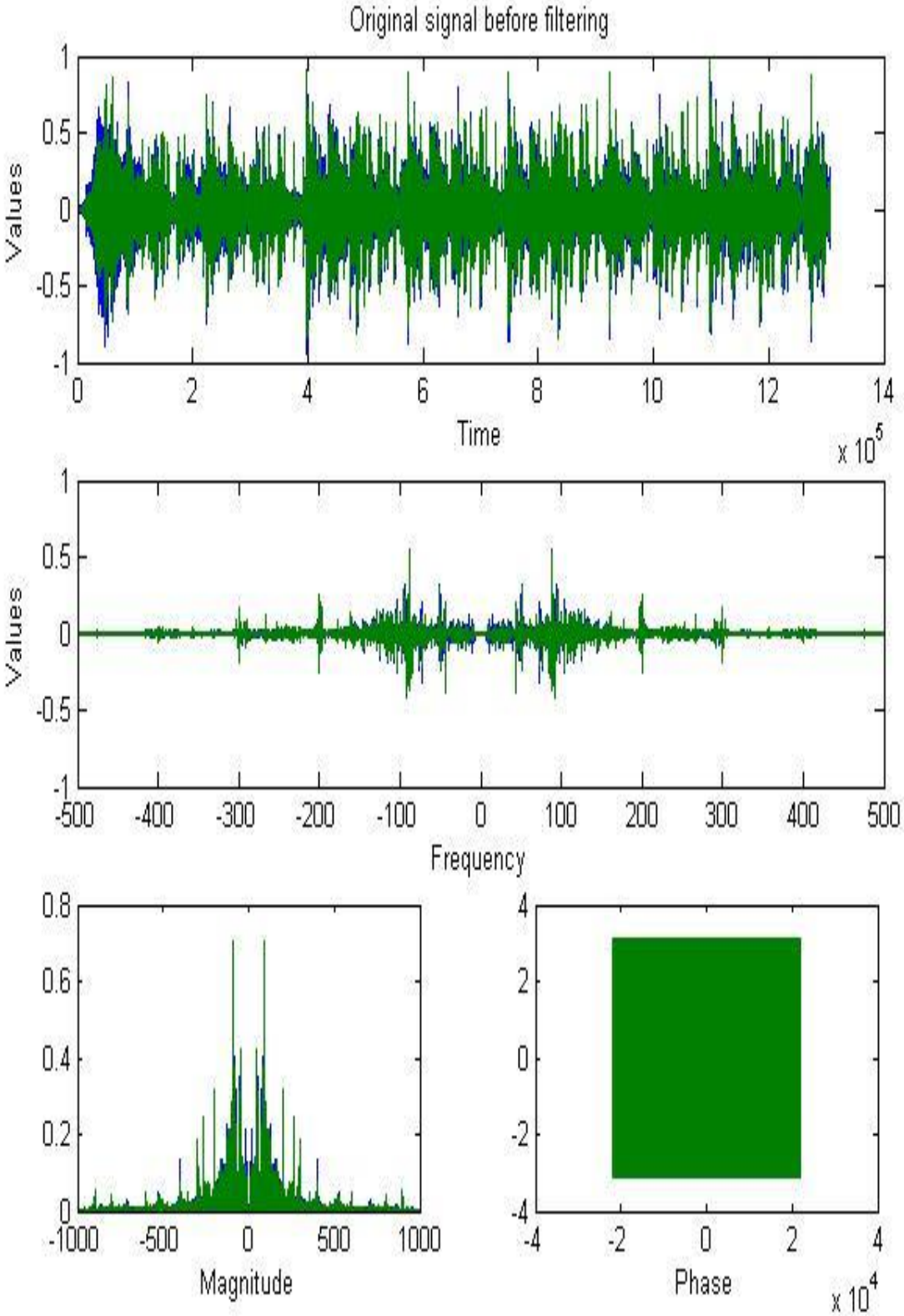

Signal after Amplifiered 1000 - 3000:
Values
Time
x 10^5
Values
Frequency
Magnituide
Phase
x 10^4

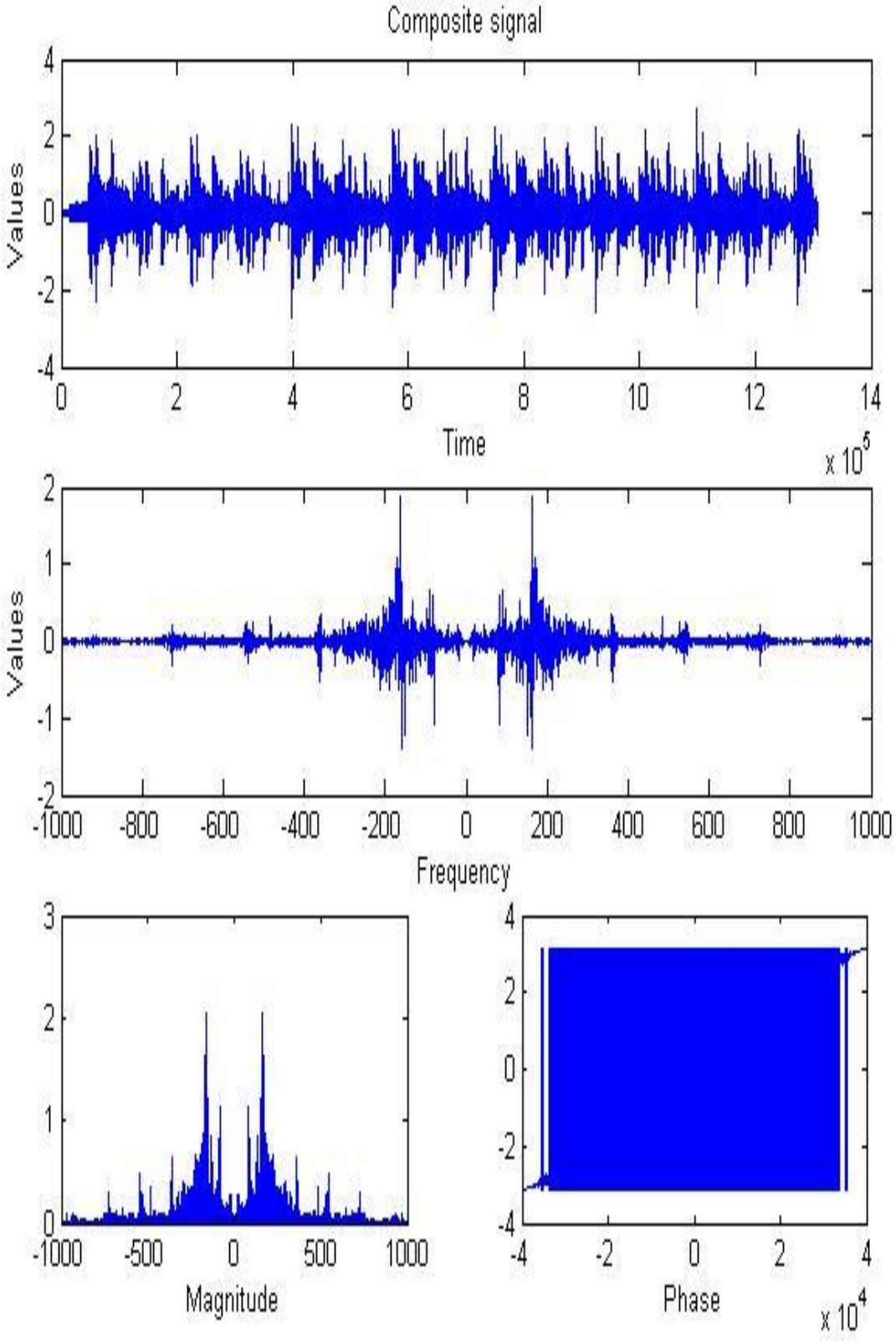

Signal after Amplifiered 3000 - 6000:
Values
Time
x 10^5
Values
Frequency
Magnituide
Phase
x 10^4


Signal after Amplifiered 6000 - 12000:
Values
Time
x 10^5
Values
Frequency
Magnituide
Phase
x 10^4


Signal after Amplifiered 12000 - 14000:
Values
Time
x 10^5
Values
Frequency
Magnituide
Phase
x 10^4

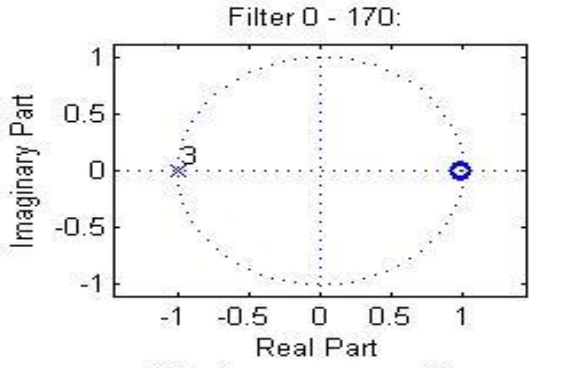

Signal after Amplifiered 14000 - 16000:
Values
Time
x 10^5
Values
Frequency
Magnituide
Phase
x 10^4

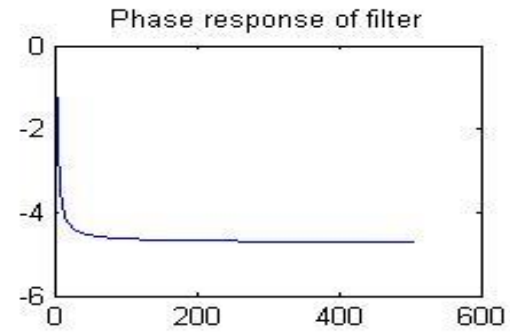

Original signal before filtering
Values
Time
x 10^5
Values
Frequency
Magnitude
Phase
x 10^4

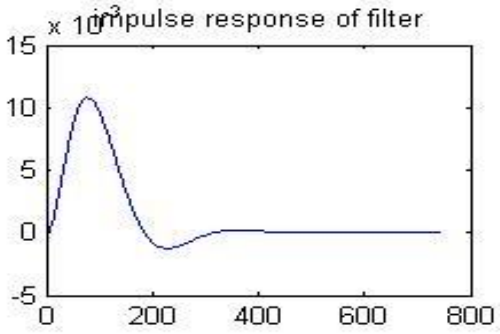

Composite signal
Values
Time
x 10^5
Values
Frequency
Magnitude
Phase
x 10^4

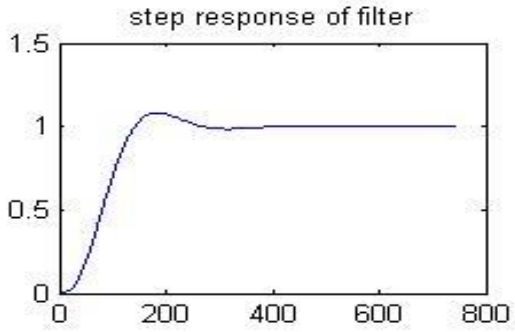
Filter analysis of each:

Filter of range 0 -> 170 Hz

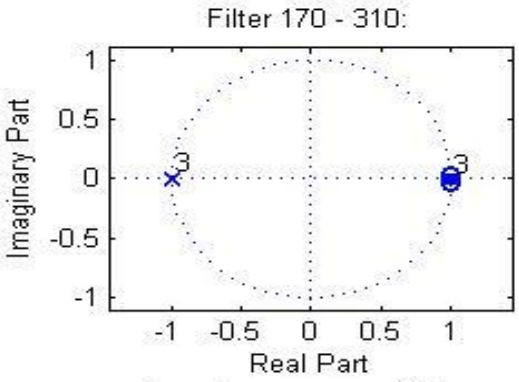


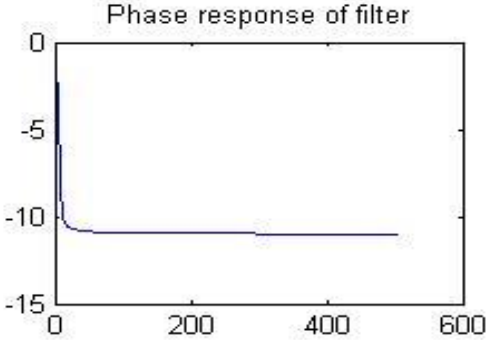


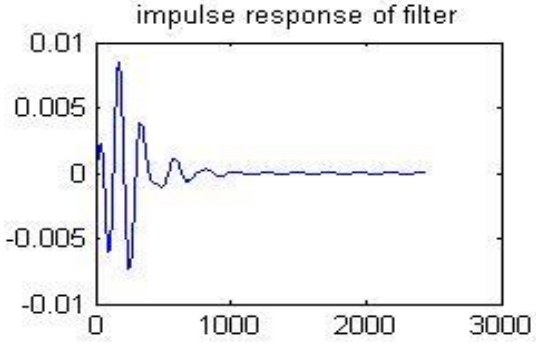


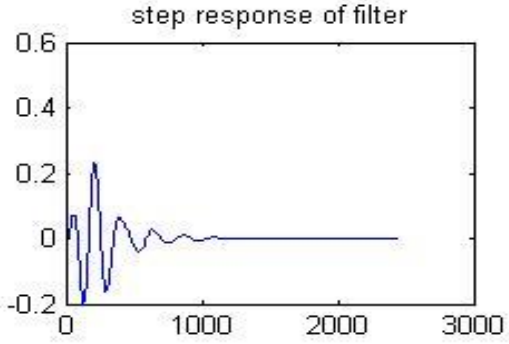


Filter of range 170 -> 310 Hz

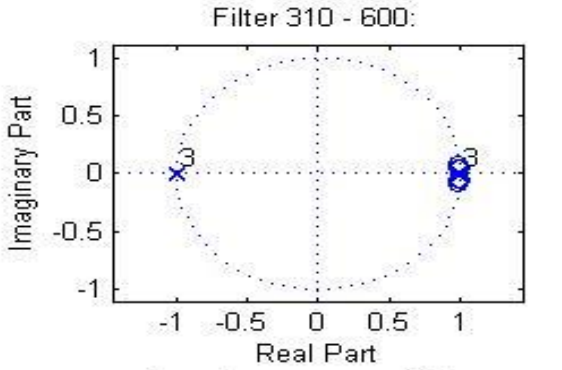


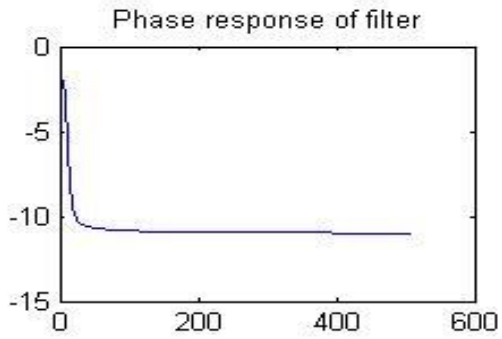


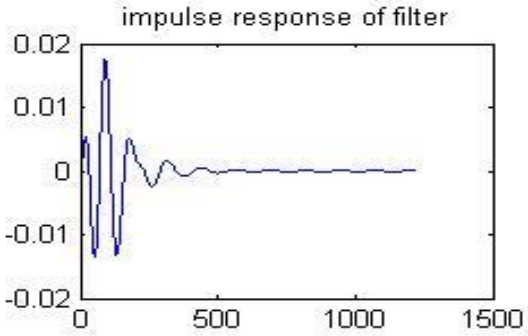


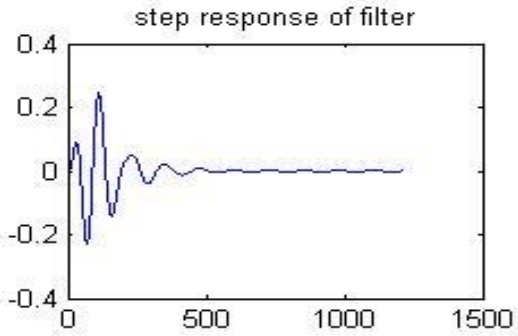


Filter of range 310 -> 600 Hz

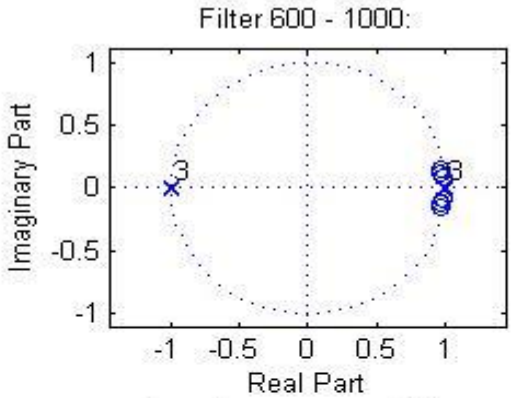


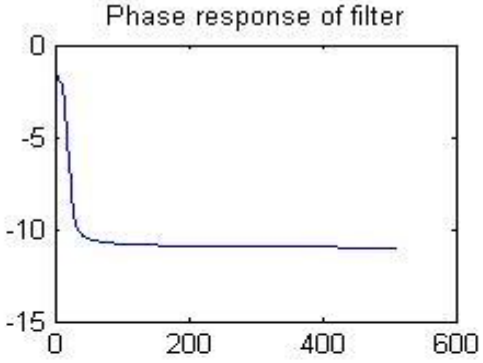


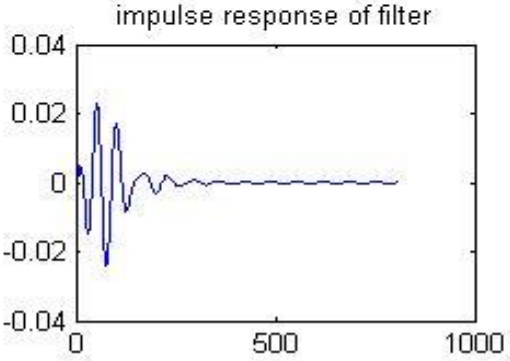


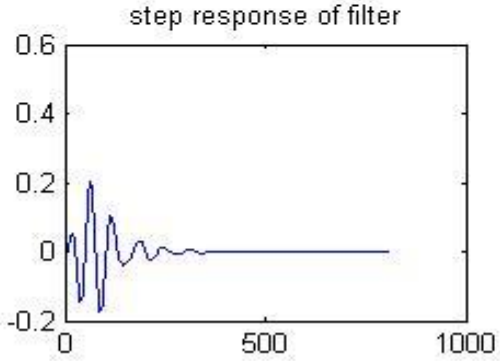


Filter of range 600 -> 1000 Hz

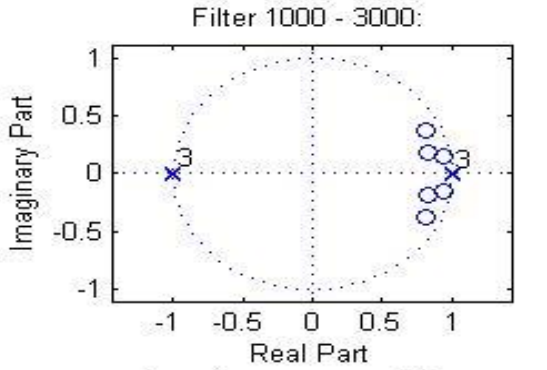


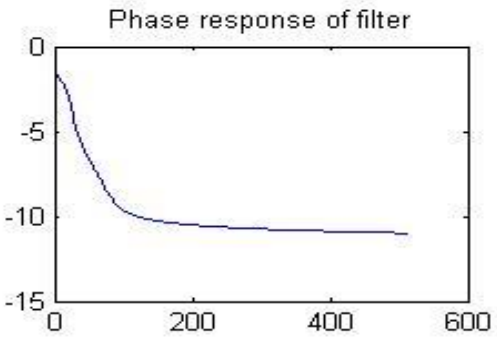


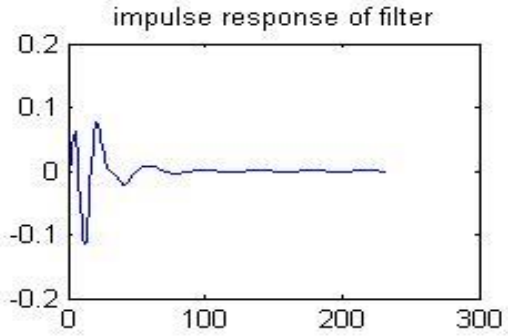


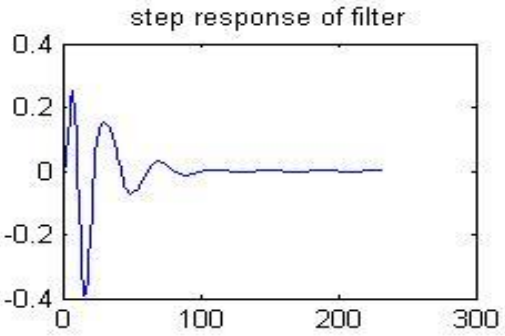


Filter of range 1 -> 3 kHz

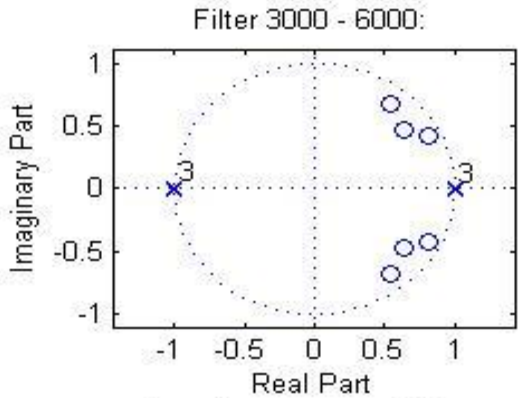


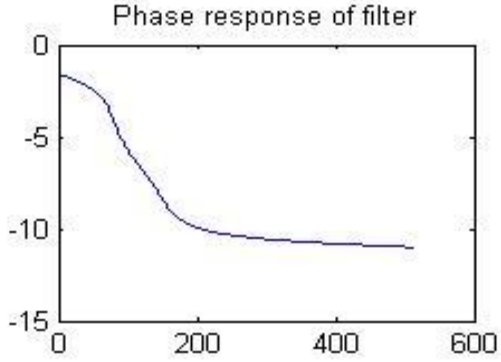


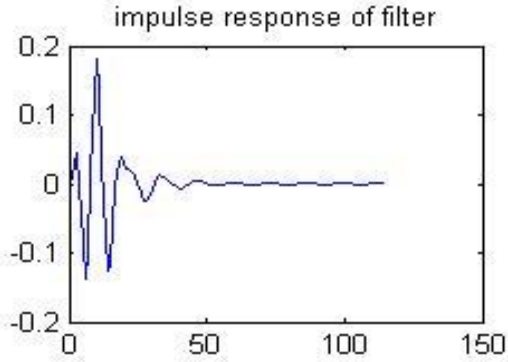


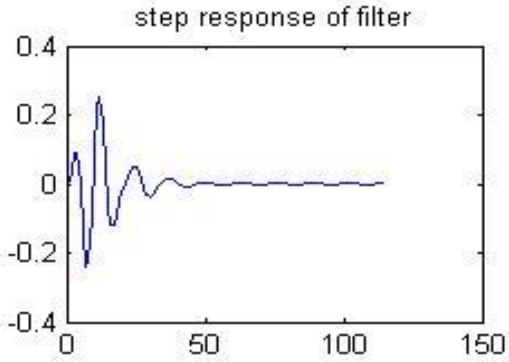


Filter of range 3 -> 6 kHz

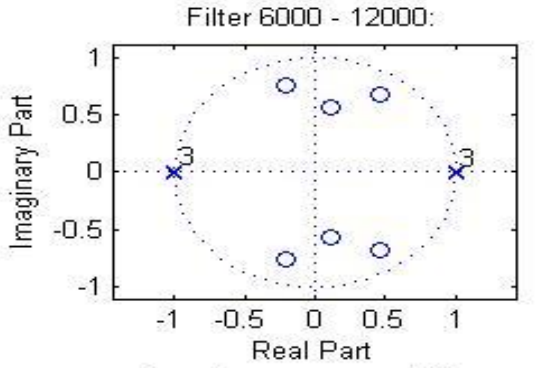


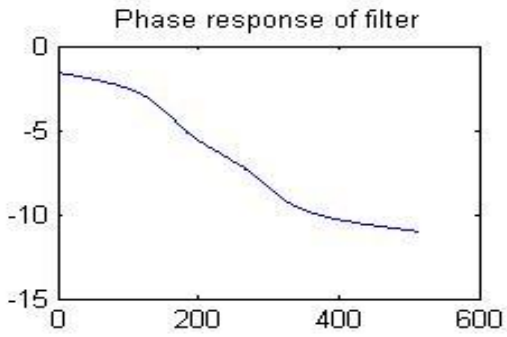


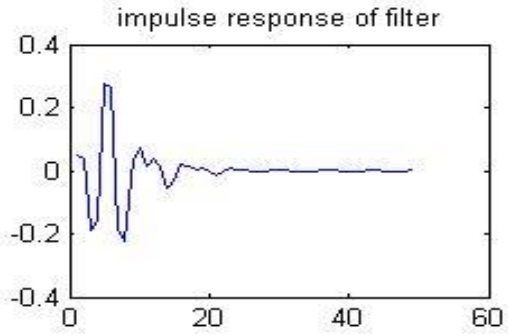


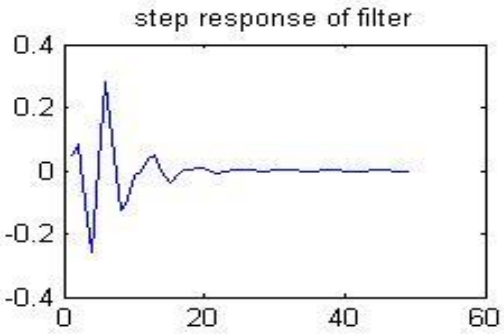


Filter of range 6 -> 12 kHz

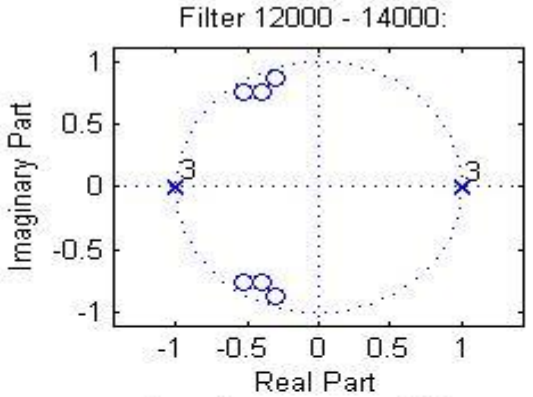


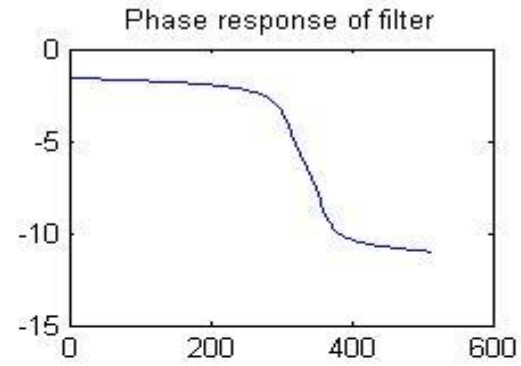


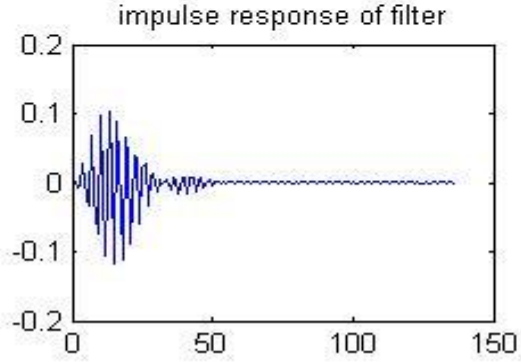


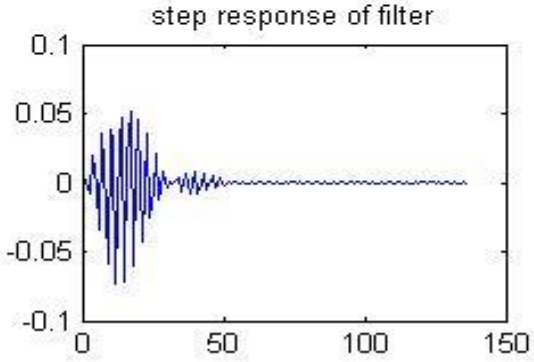


Filter of range 12 -> 14 kHz









Filter of range 14 -> 16kHz