BME 306 Lab 5 - Getting your program in order

Table of Contents

Alexander Ross 11/11/19

Question 1

```
% See attached CochlearImplant.m
```

Question 2

```
% See attached CochlearImplant.m
```

Question 3

```
% See attached CochlearImplant.m
```

Question 4

```
% See attached CochlearImplant.m
```

Question 5

```
% See attached CochlearImplant.m
```

Question 6

```
% See attached CochlearImplant.m
```

Published with MATLAB® R2019a

```
% Alexander Ross
% 10/14/19
% Introductory Discription: CochlearImplant is designed to be the software
% component of a standard cochlear implant device. It records an audio
% recording, filters it, generates an envelope of it, thresholds it, and
% adds a stimulus signal to it.  This is done by the Recoder function,
% TheFilter function, EnvelopeDetector function, Threshold function, and
% StimulusSignal function.  See the individual function scripts for more
% detailed information about their functionality.

%% Recorder Function

% Description: The purpose of the Recorder function is to record an audio
% recording and generate a signal vector from that recording.  The ouput of
% the function is the signal vector

fs = 8000; % Sampling Frequence in Hertz

time = input("How long would you like to record for (in seconds)?   "); %
Speaking time in seconds of the Recorder Function

Recorder(time); % Calling the Recorder Function

doubleArray = ans; % Defines doubleArray as the output of the Recorder
Function

%% TheFilter Function

% Description: The purpose of the TheFilter function is to filter the
% output of the Recorder function and return a 4-column filtered matrix
% with each column being the original vector filtered at a different
% frequency band.

TheFilter(doubleArray,fs); % Calling the TheFilter Function

FilteredMatrix = ans; % Defines FilteredMatrix as the output of the TheFilter
Function

%% EnvelopeDetector Function

% Description: The purpose of the EnvelopeDetector function is to detect
% the envelope of the output of the TheFilter function and return a
% matrix with the envelope of each column.

EnvelopeDetector(FilteredMatrix,fs,time); % Calling the EnvelopeDetector
Function

EnvelopeDetectedMatrix = ans; % Defines EnvelopeDetectorMatrix as the output
of the EnvelopeDetector Function

%% Threshold Function

% Description: The purpose of the Threshold function is to threshold the
% output of the TheFilter function.  This makes any noise recorded in the
% signal matrix zero.  The output is a matrix of thresholded signals.
```

```
Threshold(EnvelopeDetectedMatrix); % Calling the Threshold Function

ThresholdedMatrix = ans; % Defines ThresholdedMatrix as the output of the
Threshold Function

%% StimulusSignal Function

% Description: The purpose of the StimulusSignal Function is to create a
% stimulus signal and turn it into the signal matrix by multiplying each
% column by a train of stimulus pulses.  This is known as modulation.  The
% output is a matrix of stimulus pulses

StimulusSignal(ThresholdedMatrix,fs,time); % Calling the StimulusSignal
Function

StimulusSignalMatrix = ans; % Defines StimulusSignalMatrix as the output of
the StimulusSignal Function
```

```
function [output] = Recorder(t)

% RECORDER  Records the an audio signal
%   [output] = Recorder(t) records an audio signal for 't' seconds and
%   returns the audio data array

myVoice = audiorecorder; % Initializes the audio recording

pause(2); % Pause before recording the audio

% Define callbacks to show when
% recording starts and completes.
myVoice.StartFcn = 'disp(''Start speaking.'')'; % Starts the audio recording
recordblocking(myVoice,t); % Recording the audio
myVoice.StopFcn = 'disp(''End of recording.'')'; % Ends the audio recording

output = getaudiodata(myVoice); % Converting audio recording into a signal
array

return

end
```

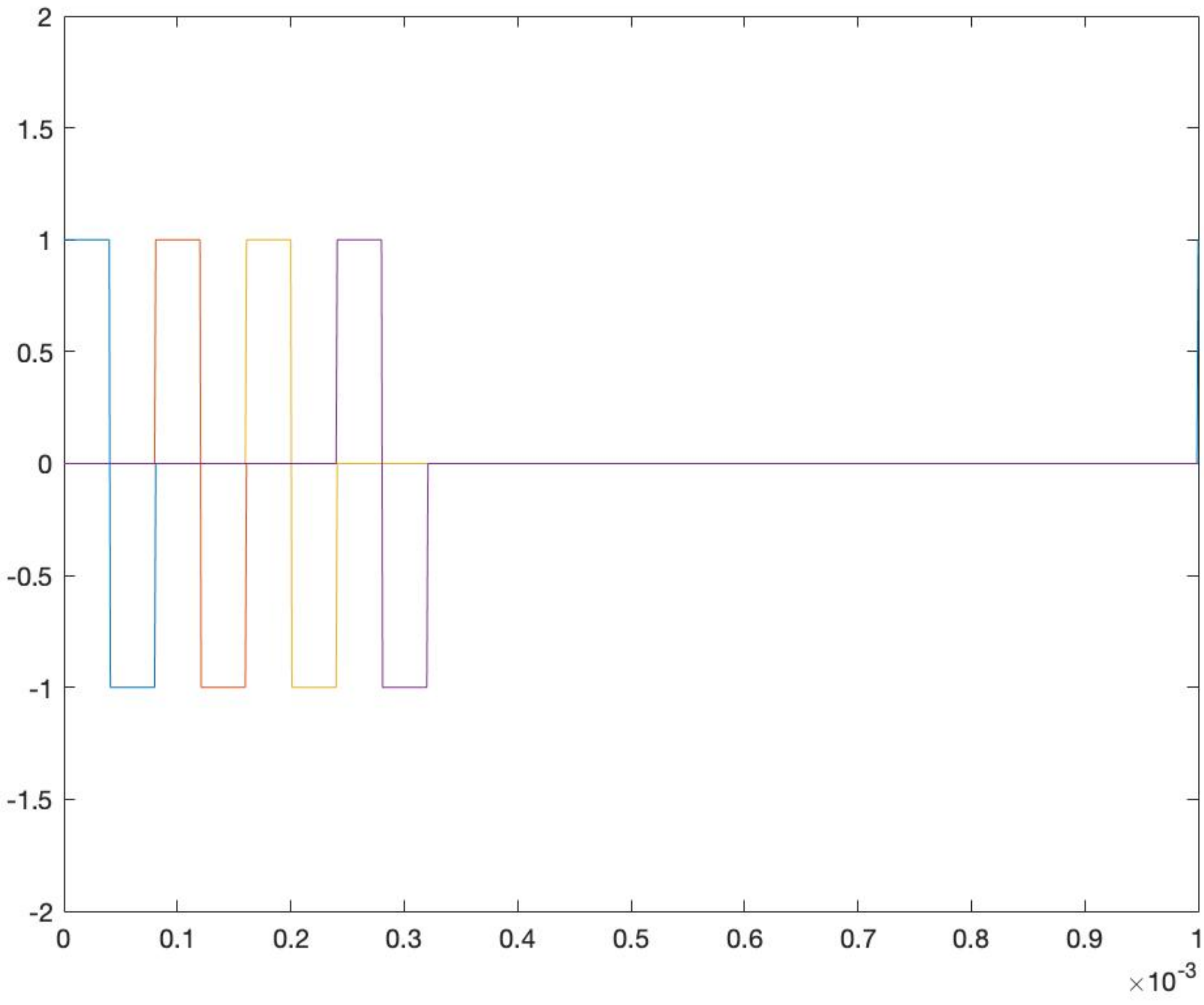
```
function [output] = TheFilter(AudioSignal,SamplingFrequency)

% THEFILTER  Filters an audio signal vector at 4 different frequency bands
%   [output] = TheFilter(doubleArray,fs) filters the audio signal
%   doubleArray at the given sampling frequency fs and returns the filtered
%   signal as an 4-column matrix, with each column being the filtered
%   signal filtered at a different fequency band.

Audiosignal = AudioSignal'; % Transposes the audio signal for the matrix

output = [Audiosignal;Audiosignal;Audiosignal;Audiosignal]'; % Returns a
matrix of the same audio signal in 4 different columns

end
```

```
function [output] = EnvelopeDetector(filteredsignal,sampf,time)

% ENVELOPEDETECTOR  Detects the envelope of a given signal
%   [output] = EnvelopeDetector(filteredsignal,samplingfrequency) detects
%   the envelope of filteredsignal at the given sampling frequency,
%   samplingfrequency, and returns the envolope as an array or matrix.

inputsignal = abs(filteredsignal(:,1))'; % Initializes column 1 as the signal
1
inputsignal2 = abs(filteredsignal(:,2))'; % Initializes column 2 as the
signal 2
inputsignal3 = abs(filteredsignal(:,3))'; % Initializes column 3 as the
signal 3
inputsignal4 = abs(filteredsignal(:,4))'; % Initializes column 4 as the
signal 4

t2 = 0:(1/2*(sampf-(1/time))):(time); % Time vector for the impulse reponse
function
tc = 0.0079577; % Time constant for the impulse response function
impulseresponse = 1/tc.*exp(-t2/tc).*heaviside(t2); % Defines the impulse
response function

convolution = (1/(sampf-(1/time))).*conv(inputsignal,impulseresponse); %
Envelope-Detection of signal 1
convolution2 = ((1/(sampf-(1/time))).*conv(inputsignal2,impulseresponse)); %
Envelope-Detection of signal 2
convolution3 = ((1/(sampf-(1/time))).*conv(inputsignal3,impulseresponse)); %
Envelope-Detection of signal 3
convolution4 = ((1/(sampf-(1/time))).*conv(inputsignal4,impulseresponse)); %
Envelope-Detection of signal 4

output(:,1) = convolution';
output(:,2) = convolution2';
output(:,3) = convolution3';
output(:,4) = convolution4';

%output = [convolution;convolution2;convolution3;convolution4]'; % Returns
ouput as envelope-detected matrix

end
```

```
function [output] = Threshold(signalmatrix)

% THRESHOLD  Thresholds the input signal matrix to prevent unwanted
stimulation
% of the cochlear electrodes
%   [output] = Threshold(signalmatrix) thresholds the input matrix and
%   returns the thresholded signal matrix as an output

inputsignal = signalmatrix(:,1)'; % Initializes column 1 as the signal 1
inputsignal2 = signalmatrix(:,2)'; % Initializes column 2 as the signal 2
inputsignal3 = signalmatrix(:,3)'; % Initializes column 3 as the signal 3
inputsignal4 = signalmatrix(:,4)'; % Initializes column 4 as the signal 4

THR = 0.005; % Defines the Threshold

for ii = 1:1:length(inputsignal) % Loops through all values of inputsignal
    if inputsignal(ii) >= THR % Selects all signal values above the threshold
        continue % Keeps all values above the threshold in the signal
    else % Selects all signal values below the threshold
        inputsignal(ii) = 0; % Makes any value below the threshold zero
    end
end

for ii = 1:1:length(inputsignal2) % Loops through all values of inputsignal2
    if inputsignal2(ii) >= THR % Selects all signal values above the
threshold
        continue % Keeps all values above the threshold in the signal
    else % Selects all signal values below the threshold
        inputsignal2(ii) = 0; % Makes any value below the threshold zero
    end
end

for ii = 1:1:length(inputsignal3) % Loops through all values of inputsignal3
    if inputsignal3(ii) >= THR % Selects all signal values above the
threshold
        continue % Keeps all values above the threshold in the signal
    else % Selects all signal values below the threshold
        inputsignal3(ii) = 0; % Makes any value below the threshold zero
    end
end

for ii = 1:1:length(inputsignal4) % Loops through all values of inputsignal4
    if inputsignal4(ii) >= THR % Selects all signal values above the
threshold
        continue % Keeps all values above the threshold in the signal
    else % Selects all signal values below the threshold
        inputsignal4(ii) = 0; % Makes any value below the threshold zero
    end
end

output = [inputsignal;inputsignal2;inputsignal3;inputsignal4]'; % Returns
ouput as envelope-detected matrix

end
```

```
function [output] = StimulusSignal(signalmatrix,fs,time)

% STIMULUSSIGNAL  Multiplies each each column of the audio matrix by
% a train of stimulus pulses, modulating the original matrix
%   [output] = StimulusSignal(signalmatrix) modulates the input matrix
%   with a stimulus signal and outputs the stimulus pulses in a matrix

inputsignal = signalmatrix(:,1)'; % Initializes column 1 as the signal 1
inputsignal2 = signalmatrix(:,2)'; % Initializes column 2 as the signal 2
inputsignal3 = signalmatrix(:,3)'; % Initializes column 3 as the signal 3
inputsignal4 = signalmatrix(:,4)'; % Initializes column 4 as the signal 4

t = 0:0.000001:time; % Time vector
pps = 1000; % Number of pulses per second
widthA = 0.00004; % Width of the postive pulse in seconds
widthB = 0.00004; % Width of the negative pulse in seconds
dutyA = widthA*pps*100; % Duty of the positive pulse in seconds
dutyB = widthB*pps*100; % Duty of the negative pulse in seconds

positive1= (0.5).*square(pps.*2.*pi.*t,dutyA)+0.5; % Positive pulse
negative1= (-0.5).*square(pps.*2.*pi.*(t-widthA),dutyB)-0.5; % Negative pulse
y1= positive1 + negative1; % Summation of positve and negative pulses

positive2= (0.5).*square(pps.*2.*pi.*(t-0.00008),dutyA)+0.5; % Positive pulse
negative2= (-0.5).*square(pps.*2.*pi.*(t-0.00008-widthA),dutyB)-0.5; %
Negative pulse
y2= positive2 + negative2; % Summation of positve and negative pulses

positive3= (0.5).*square(pps.*2.*pi.*(t-0.00016),dutyA)+0.5; % Positive pulse
negative3= (-0.5).*square(pps.*2.*pi.*((t-0.00016)-widthA),dutyB)-0.5; %
Negative pulse
y3= positive3 + negative3; % Summation of positve and negative pulses

positive4= (0.5).*square(pps.*2.*pi.*(t-0.00024),dutyA)+0.5; % Positive pulse
negative4= (-0.5).*square(pps.*2.*pi.*((t-0.00024)-widthA),dutyB)-0.5; %
Negative pulse
y4= positive4 + negative4; % Summation of positve and negative pulses

plot(t,y1,t,y2,t,y3,t,y4); % Plot to demonstrate pulse sequencing
axis([0 0.001 -2 2]);  % Axis of plot to demonstrate pulse sequencing

t3 = 0:(1)/(fs-(1/time)):time;  % Time vector
inputsignala1 = resample(inputsignal,t3,100000); % Resampling to the desired
frequency
inputsignalb2 = resample(inputsignal2,t3,100000); % Resampling to the desired
frequency
inputsignalc3 = resample(inputsignal3,t3,100000); % Resampling to the desired
frequency
inputsignald4 = resample(inputsignal4,t3,100000); % Resampling to the desired
frequency

y1(:,((length(inputsignala1))+1):1:end)=[]; % Reshaping pulse train vectors
y2(:,((length(inputsignalb2))+1):1:end)=[]; % Reshaping pulse train vectors
y3(:,((length(inputsignalc3))+1):1:end)=[]; % Reshaping pulse train vectors
y4(:,((length(inputsignald4))+1):1:end)=[]; % Reshaping pulse train vectors
```

```
outputsignal = y1.*inputsignala1; % Mulitplying the first input signal by its
respective stimulus pulse train
outputsignal2 = y2.*inputsignalb2; % Mulitplying the second input signal by
its respective stimulus pulse train
outputsignal3 = y3.*inputsignalc3; % Mulitplying the third input signal by
its respective stimulus pulse train
outputsignal4 = y4.*inputsignald4; % Mulitplying the fourth input signal by
its respective stimulus pulse train

%t2 = 0:(1)/(100000-(1/time)):time; % New time vector
%t4 = conv((t2),heaviside(t3)); % Time vector for plotting
%plot(t4,outputsignal,t4,outputsignal2,t4,outputsignal3,t4,outputsignal4);%
Plotting to confirm the output

output = [outputsignal;outputsignal2;outputsignal3;outputsignal4]'; % Returns
ouput as envelope-detected matrix

end
```


2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
$\times 10^{-3}$