Nicholas Marcopoli

Introduction to Microprocessors and Multimedia

April 28, 2017

Parametric Equalizer: Technical Manual

INTRODUCTION

The following document will list and analyze the signal processing techniques used in the Parametric Equalizer GUI.

SIGNAL PROCESSING TECHNIQUE LISTING

Audioread:

Reads in an audio signal within a selected directory and returns the sampled audio along with the sample rate of the data. This data can then be used to manipulate the audio file.

Fast Fourier Transform (FFT):

The FFT of a set of audio samples are a set of complex numbers which describe the frequency content of the signal. The MATLAB function “fft” will generate an array of this set of numbers. Using the MATLAB function “abs” in conjunction with “fft” will compute the magnitude of the FFT of a sample. The first half of this magnitude FFT spectrum is the only part that is needed, as the second half is a mirrored version of the first half.

2nd Order Transfer Function:

The second order transfer function creates a particular analog notch filter, which can either amplify or attenuate a range of frequencies while leaving all other frequencies unaltered. In a parametric equalizer, this is quite useful in order to ensure that only the selected frequencies are amplified or attenuated, as opposed to a FIR filter such as MATLAB’s “firpm” algorithm, which modifies undesired frequencies by creating a ripple in the passband of the sample.

This specific notch filter employs the following parameters: fc (center frequency), zeta (width of frequency), and alpha (attenuation/amplification of altered frequency). The zeta value is known as the “damping” factor. Altering this value will change the range of affected frequencies. In effect, a high zeta value will affect a larger range of frequencies about the center frequency, while a low zeta value will affect a smaller range of frequencies.

The alpha value will attenuate or amplify the selected frequencies by a magnitude value. The formula used to convert decibel values into the magnitude values used by alpha was given by Kyle Hunte and is as follows:

$$\alpha \;=\; 10^{(dB/20)}\text{ , }\textit{Equation 1}$$

The specific continuous time transfer function which was used in this project was given by Vincent Marcopoli and is as follows:

$$H(s) = (s^2 + 2\zeta\alpha\omega_c s + \omega_c{}^2)/(s^2 + 2\zeta\omega_c s + \omega_c{}^2)\text{ , }\textit{Equation 2}$$

$$\text{Where }\ \omega_c \ = 2\pi f_c$$

Bilinear:

MATLAB's "bilinear" function is able to transform an analog filter into a digital IIR filter by converting the transfer function into tap coefficients that can be applied to a digital signal at a particular sampling frequency.

Freqz:

The MATLAB function freqz returns a frequency response for a given digital filter.

USE OF ABOVE TECHNIQUES IN PROJECT:

Audioread:

Simply used to read in a selected audio file chosen by a user manipulating the GUI tool. The sample was then shortened to a duration specified in the GUI and the audio file was reloaded to account for the shortened file. If the "Simulate Microphone Feedback" box was checked, a 400 Hz tone was added to the loaded signal.

Fast Fourier Transform (FFT):

The magnitude of the FFT was computed for the original audio sample and the filtered audio sample, and the magnitudes of the first half were then plotted on a log scaled graph in order to easily view lower frequencies and low magnitude values.

Second Order Transfer Function:

The parameters zeta, fc, and decibels can be specified by the user manipulating the GUI. These values are then inputted into the transfer function described in *Equation 2,* with the decibel values first being converted into alpha values by *Equation 1.* The transfer function was programmed into the

MATLAB code by creating two arrays, num and den, which corresponded to the coefficients on *s* in the numerator and denominator of the transfer function, respectively.

Bilinear:

Bilinear was invaluable in converting the transfer function to a digital filter. To convert the transfer function to tap coefficients, simply run the numerator and denominator of the transfer function, as well as the sampling frequency, through bilinear, and then to pass the audio through the digital filter simply run the new digital numerator, digital denominator, and audio signal through the "filter" function.

Freqz:

The freqz function was used to plot the original audio FFT and frequency response on the same graph. The numerator and denominator of the digital IIR filter created by the bilinear function, as well as the sampling frequency, were inputted into freqz. The output values were then plotted on the same graph as the original audio FFT.

To plot both the audio FFT and frequency response on the same graph, a conversion needed to be made to make sure the two were plotted along the same axes. This was done by dividing the freqz's resulting x value (w) by pi, thus normalizing the frequency by pi rad/sample, then multiplying by the Nyquist frequency to achieve the same scale as the FFT graph.

APPENDIX - MATLAB PROGRAM LISTING

List of Programs included in project:
DSPProject.m
DSPProject.fig
para_eq.m
BadNews.mp3
Beethoven.mp3
Brad.mp3
Gorillaz.mp3
Jackson.mp3
Kanye.mp3
Lorde.mp3

para_eq Code (Most signal processing work done here):

```
function para_eq(handles)
% Works in conjunction with DSPProject.m to produce a notch filter which
% creates a parametric equalizer
% Nicholas Marcopoli
% Microprocessors and Multimedia
% 4/28/17

% read in audio selection
switch handles.audioPop.Value
    case 1
        song = 'Kanye.mp3';
    case 2
        song = 'Jackson.mp3';
    case 3
        song = 'Gorillaz.mp3';
    case 4
        song = 'Lorde.mp3';
    case 5
        song = 'BadNews.mp3';
    case 6
        song = 'Beethoven.mp3';
    case 7
        song = 'Brad.mp3';
end
[audio, fs] = audioread(song);

% shortens audio sample to value specified in GUI
duration = [1,str2double(handles.durationText.String) * fs];
clear audio fs
[audio, fs] = audioread(song,duration);

Ts = 1/fs;                                   % Sampling Period
T = str2double(handles.durationText.String); % Duration of Sample
N = T/Ts;                                    % Number of samples

t = 0: Ts : (N-1)*Ts; % Time vector - sampling times

% Add a tone to simulate mic feedback in the sample
if handles.checkbox.Value == 1
    micfeedback = 400;              % Hz
    tone = cos(2*pi*micfeedback*t); % tone at 400 hz
    tone = tone';                   % transpose tone to be added to audio
    audio = audio + tone;           % add tone to audio
end

fc=handles.freqSlider.Value;      % Center frequency of boost/cut
```

```
w=2*pi*fc;                        % Convert to rad/sec
% "Damping" factor (controls frequency width of cut/boost)
zta=handles.widthSlider.Value;
decib = handles.ampSlider.Value;  % Desired amplification/attenuation in db
% Converts desired amplification/attenuation using db = 20log(alpha)
alph=10^(decib/20);

% Allows operations later in code to be executed upon button press
dofvtool = handles.viewFvButton.Value;
doaudio = handles.audioButton.Value;
dofaudio = handles.audioFButton.Value;

% Create filter from 2nd order transfer function - from Vincent Marcopoli
num=[1 2*zta*alph*w w^2];         % Continuous transfer function numerator
den=[1 2*zta*w w^2];              % Denominator
[numd,dend]=bilinear(num,den,fs); % Convert to digital IIR filter

% pass audio through filter
faudio = filter(numd,dend,audio);

% plot original audio fft while overlaying filter
spec = abs(fft(audio)); % Take magnitude of the fft
f = 0: 1/T: (N/2 - 1)/T; % plot frequency for audio duration

[h,w] = freqz(numd,dend,fs); % gets filter frequency response
% (needed to plot filter on GUI axis)
h = abs(h);
% plots original signal overlayed with frequency response of the filter
% frequency response is normalized (when w is divided by pi) then is converted
% into the same axis as the FFT graph (up to Nyquist)
plot(handles.origAxes,f,spec(1:N/2),w/pi*fs/2,h);
set(handles.origAxes,'XScale','log');
set(handles.origAxes,'YScale','log');
title(handles.origAxes,'fft of Original Signal');
xlabel(handles.origAxes,'f (Hz)');
legend(handles.origAxes,'Original Signal','Filter to be Applied','Location','Southwest');

% plot filtered audio fft on a log scale
spec = abs(fft(faudio)); % Take magnitude of the fft
f = 0: 1/T: (N/2 - 1)/T; % Plot frequency for audio duration
plot(handles.filtAxes,f,spec(1:N/2));
set(handles.filtAxes,'XScale','log');
set(handles.filtAxes,'YScale','log');
title(handles.filtAxes,'fft of Filtered Signal');
xlabel(handles.filtAxes,'f (Hz)');

% plot fvtool when button pressed
if dofvtool == 1
    fvtool(numd,dend);
end

% play original audio when button pressed
if doaudio == 1
    sound(audio,fs);
end

% play filtered audio when button pressed
if dofaudio == 1
    sound(faudio,fs);
end
```

DSPProject Code (Only the GUI code, mostly unaltered):

```
function varargout = DSPProject(varargin)
% DSPPROJECT MATLAB code for DSPProject.fig
%      DSPPROJECT, by itself, creates a new DSPPROJECT or raises the existing
%      singleton*.
%
%      H = DSPPROJECT returns the handle to a new DSPPROJECT or the handle to
%      the existing singleton*.
%
%      DSPPROJECT('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in DSPPROJECT.M with the given input arguments.
%
%      DSPPROJECT('Property','Value',...) creates a new DSPPROJECT or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before DSPProject_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to DSPProject_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help DSPProject

% Last Modified by GUIDE v2.5 26-Apr-2017 18:32:56

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @DSPProject_OpeningFcn, ...
                   'gui_OutputFcn',  @DSPProject_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
% End initialization code - DO NOT EDIT


% --- Executes just before DSPProject is made visible.
function DSPProject_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to DSPProject (see VARARGIN)

% Choose default command line output for DSPProject
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes DSPProject wait for user response (see UIRESUME)
% uiwait(handles.figure1);
para_eq(handles);

% --- Outputs from this function are returned to the command line.
function varargout = DSPProject_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on slider movement.
function freqSlider_Callback(hObject, eventdata, handles)
% hObject    handle to freqSlider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
handles.freqText.String = num2str(handles.freqSlider.Value);
para_eq(handles);


% --- Executes during object creation, after setting all properties.
function freqSlider_CreateFcn(hObject, eventdata, handles)
% hObject    handle to freqSlider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function widthSlider_Callback(hObject, eventdata, handles)
% hObject    handle to widthSlider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
handles.widthText.String = num2str(handles.widthSlider.Value);
para_eq(handles);


% --- Executes during object creation, after setting all properties.
function widthSlider_CreateFcn(hObject, eventdata, handles)
% hObject    handle to widthSlider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function ampSlider_Callback(hObject, eventdata, handles)
% hObject    handle to ampSlider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
handles.ampText.String = num2str(handles.ampSlider.Value);
para_eq(handles);
```

```
% --- Executes during object creation, after setting all properties.
function ampSlider_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ampSlider (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end



function freqText_Callback(hObject, eventdata, handles)
% hObject    handle to freqText (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of freqText as text
%        str2double(get(hObject,'String')) returns contents of freqText as a double
handles.freqSlider.Value = str2double(handles.freqText.String);
para_eq(handles);


% --- Executes during object creation, after setting all properties.
function freqText_CreateFcn(hObject, eventdata, handles)
% hObject    handle to freqText (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function widthText_Callback(hObject, eventdata, handles)
% hObject    handle to widthText (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of widthText as text
%        str2double(get(hObject,'String')) returns contents of widthText as a double
handles.widthSlider.Value = str2double(handles.widthText.String);
para_eq(handles);


% --- Executes during object creation, after setting all properties.
function widthText_CreateFcn(hObject, eventdata, handles)
% hObject    handle to widthText (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ampText_Callback(hObject, eventdata, handles)
% hObject    handle to ampText (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ampText as text
%        str2double(get(hObject,'String')) returns contents of ampText as a double
handles.ampSlider.Value = str2double(handles.ampText.String);
para_eq(handles);


% --- Executes during object creation, after setting all properties.
function ampText_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ampText (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in audioBox.
function audioBox_Callback(hObject, eventdata, handles)
% hObject    handle to audioBox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns audioBox contents as cell array
%        contents{get(hObject,'Value')} returns selected item from audioBox
para_eq(handles);


% --- Executes during object creation, after setting all properties.
function audioBox_CreateFcn(hObject, eventdata, handles)
% hObject    handle to audioBox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in audioButton.
function audioButton_Callback(hObject, eventdata, handles)
% hObject    handle to audioButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
para_eq(handles);

% --- Executes on button press in filterButton.
function filterButton_Callback(hObject, eventdata, handles)
% hObject    handle to filterButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



% --- Executes on button press in audioFButton.
function audioFButton_Callback(hObject, eventdata, handles)
% hObject    handle to audioFButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
para_eq(handles);
```

```
function durationText_Callback(hObject, eventdata, handles)
% hObject    handle to durationText (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of durationText as text
%        str2double(get(hObject,'String')) returns contents of durationText as a double
para_eq(handles);


% --- Executes during object creation, after setting all properties.
function durationText_CreateFcn(hObject, eventdata, handles)
% hObject    handle to durationText (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in audioPop.
function audioPop_Callback(hObject, eventdata, handles)
% hObject    handle to audioPop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns audioPop contents as cell array
%        contents{get(hObject,'Value')} returns selected item from audioPop
para_eq(handles);


% --- Executes during object creation, after setting all properties.
function audioPop_CreateFcn(hObject, eventdata, handles)
% hObject    handle to audioPop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in viewFvButton.
function viewFvButton_Callback(hObject, eventdata, handles)
% hObject    handle to viewFvButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
para_eq(handles);


% --- Executes on button press in selectFile.
function selectFile_Callback(hObject, eventdata, handles)
% hObject    handle to selectFile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
para_eq(handles);


% --- Executes on button press in checkbox.
```

```
function checkbox_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox
para_eq(handles);
```