# “main.m” file

close all;  
clear;  
  
[file, path] = uigetfile('\*.wav');  
fullFileName = fullfile(path, file);  
  
[data, original\_fs] = audioread(fullFileName);  
  
disp('File information:');  
disp(['Path: ' fullFileName]);  
disp(['Data dimensions: ' mat2str(size(data))]);  
disp(['Frequency: ' num2str(original\_fs)]);  
  
bands = get\_bands();  
  
fs = original\_fs;  
if bands(end, end) >= fs / 2  
 margin = 11;  
 fs = 2 \* (bands(end, end) + margin);  
 data = resample(data, fs, original\_fs);  
end  
  
gains = zeros(1, length(bands));  
for i = 1:length(bands)  
 gains(i) = get\_number(['Enter gain for ' mat2str(bands(i,:)) 'Hz (between -20 dB and 20 dB): '], @(x) x >= -20 && x <= 20);  
end  
  
filter\_type = get\_string('Enter filter type ("fir" or "iir"):', @(x) strcmp('fir', x) || strcmp('iir', x));  
  
output\_fs = get\_number('Enter a valid output sample rate: ', @(x) x > 340);  
  
if strcmp('fir', filter\_type)  
 fir\_order = 150;  
 filters = fir\_filters(fir\_order, fs, bands);  
else  
 iir\_order = 4;  
 filters = iir\_filters(iir\_order, fs, bands);  
end  
  
acc\_filtered = data .\* 0;  
  
freq\_range\_plt = [' 0-170 Hz'; '170-310 Hz'; '310-600 Hz'; '0.6-1 kHz'; '1-3 kHz'; '3-6 kHz'; '6-12 kHz'; '12-14 kHz'; '14-16 KHZ'];  
for i = 1:length(filters)  
 plot\_requirements(filters(i), fs, [mat2str(bands(i,:)) 'Hz']);  
 filtered = filter(filters(i).Numerator, filters(i).Denominator, data);  
 plot\_time\_frequency\_domain(filtered, fs, ['Output in time-domain for filter ' freq\_range\_plt(i,:)], ['Output in frequency-domain for filter ' freq\_range\_plt(i,:)]);  
 acc\_filtered = acc\_filtered + filtered \* db2mag(gains(i));  
 [z, p, k] = tf2zpk(filters(i).Numerator, filters(i).Denominator);  
 order = filtord(filters(i).Numerator, filters(i).Denominator);  
 fprintf('The gain of %s filter : %s is %f , Order is %d \n',filter\_type, freq\_range\_plt(i,:), k, order);  
end  
  
acc\_filtered = resample(acc\_filtered, output\_fs, fs); % resample to the output fs  
plot\_time\_frequency\_domain(data, original\_fs, 'Input in time ', 'Input in freq. ', acc\_filtered, output\_fs, 'Output in freq.', 'Output in time.');  
  
[file, path] = uiputfile('\*.wav');  
fullFileName = fullfile(path, file);  
  
audiowrite(fullFileName, acc\_filtered, output\_fs);

# “plot\_requirements” helper function

Plots the frequency response (magnitude and phase), impulse response, step response, and the pole-zero plot of a given filter.

function plot\_requirements(filter, fs, name)  
 x = fvtool(filter.Numerator, filter.Denominator);  
 set\_common\_properties(x, fs);  
 x.Analysis = 'freq';  
 x.Name = ['Magnitude response (dB) and phase response of filter: ' name];  
  
 x = fvtool(filter.Numerator, filter.Denominator);  
 set\_common\_properties(x, fs);  
 x.Analysis = 'impulse';  
 x.Name = ['Impulse response of filter: ' name];  
  
 x = fvtool(filter.Numerator, filter.Denominator);  
 set\_common\_properties(x, fs);  
 x.Analysis = 'step';  
 x.Name = ['Step response of filter: ' name];  
  
 x = fvtool(filter.Numerator, filter.Denominator);  
 set\_common\_properties(x, fs);  
 x.Analysis = 'polezero';  
 x.Name = ['Pole-zero plot of filter: ' name];  
end  
  
function set\_common\_properties(x, fs)  
 x.NormalizedFrequency = 'off';  
 x.fs = fs;  
 x.WindowStyle = 'normal';  
 x.NumberTitle = 'off';  
end

# “plot\_time\_frequency\_domain” helper function

Plots one or two signals in both time and frequency domains.

function plot\_time\_frequency\_domain(data, fs, title\_time, title\_freq, varargin)  
 sub\_plots = 1;  
 figure;  
 if nargin > 5  
 sub\_plots = 2;  
 length\_output = length(varargin{1});  
 df = varargin{2} / length\_output;  
 frequency\_audio = -varargin{2}/2 : df : varargin{2}/2-df;  
  
 not\_shifted\_output\_fft = fft(varargin{1});  
 fft\_data = fftshift(not\_shifted\_output\_fft) / length(not\_shifted\_output\_fft);  
  
 subplot(2, sub\_plots , 3);  
 plot(frequency\_audio, abs(fft\_data));  
 title(varargin{3});  
 xlabel('Frequency (Hz)');  
 ylabel('Amplitude');  
  
 subplot(2, sub\_plots, 4);  
 t = (0:length\_output-1)/ varargin{2};  
 plot(t, varargin{1})  
 title(varargin{4});  
 xlabel('Time (s)');  
 ylabel('Amplitude');  
 end  
  
 length\_data = length(data);  
 df = fs / length\_data;  
 frequency\_audio = -fs/2 : df : fs/2-df;  
 not\_shifted\_fft = fft(data);  
 fft\_data = fftshift(not\_shifted\_fft) / length(not\_shifted\_fft);  
  
 subplot(2, sub\_plots, 1);  
 plot(frequency\_audio, abs(fft\_data));  
 title(title\_freq);  
 xlabel('Frequency (Hz)');  
 ylabel('Amplitude');  
  
 subplot(2, sub\_plots, 2);  
 t = (0:length\_data-1)/ fs;  
 plot(t, data)  
 title(title\_time);  
 xlabel('Time (s)');  
 ylabel('Amplitude');  
end

# “fir\_filters” helper function

Returns a list of fir filters of a given order, fs, and bands.

function filters = fir\_filters(order, fs, bands)  
 filters = [];  
 for i = 1:length(bands)  
 band = bands(i,:) / (fs / 2);  
 if band(1) == 0  
 b = fir1(order, band(2));  
 else  
 b = fir1(order, band);  
 end  
 filter = Filter(b, 1);  
 filters = [filters filter];  
 end  
end

# “iir\_filters” helper function

Returns a list of iir (butter) filters of a given order, fs, and bands.

function filters = iir\_filters(order,fs,bands)  
 filters = [];  
 for i = 1:length(bands)  
 band = bands(i,:) / (fs / 2);  
 if band(1) == 0  
 [b, a] = butter(order,band(2));  
  
 else  
 [b, a] = butter(order,band,'bandpass');  
 end  
 filters = [filters Filter(b, a)];  
 end  
end

# “Filter” class

Represents a filter by its numerator and denominator. It makes it easier for us to have a vector of filters.

classdef Filter  
  
 properties  
 Numerator  
 Denominator  
 end  
  
 methods  
 function obj = Filter(numerator, denominator)  
 obj.Numerator = numerator;  
 obj.Denominator = denominator;  
 end  
 end  
end

# “get\_bands” helper function

Returns a list of filter bands as required.

function bands = get\_bands()  
 bands = [0 170;  
 170 310;  
 310 600;  
 600 1000;  
 1000 3000;  
 3000 6000;  
 6000 12000;  
 12000 14000;  
 14000 16000];  
end

# “get\_number” helper function

Takes a message to display to the user, and keep asking the user to enter a valid real number based on the validation predicate.

function f = get\_number(prompt, predicate)  
 while 1  
 x = inputdlg(prompt);  
 n = str2double(x{1});  
 if length(n) == 1 && isreal(n) && predicate(n)  
 f = n;  
 break;  
 end  
 end  
end

# “get\_string” helper function

Takes a message to display to the user, and keep asking the user to enter a valid string based on the validation predicate.

function f = get\_string(prompt, predicate)  
 while 1  
 s = inputdlg(prompt);  
 if predicate(s{1})  
 f = s{1};  
 break;  
 end  
 end  
end

**Sample run 1:**

Type of filter: fir

Specified gains for each of the nine filters: 0, 0, 0, 0, 0, 0, 0, 0, 0 dB

Specified output frequency: 8000 Hz

**Output:**

File information:  
Path: C:\Users\PC\Desktop\FinalProject\_DSP\voice.wav  
Data dimensions: [268237 2]  
Frequency: 8000  
The gain of fir filter : 0-170 Hz is 0.000277 , Order is 150   
The gain of fir filter : 170-310 Hz is -0.001567 , Order is 150   
The gain of fir filter : 310-600 Hz is 0.000804 , Order is 150   
The gain of fir filter : 0.6-1 kHz is 0.000115 , Order is 150   
The gain of fir filter : 1-3 kHz is -0.000228 , Order is 150   
The gain of fir filter : 3-6 kHz is 0.000054 , Order is 150   
The gain of fir filter : 6-12 kHz is 0.000099 , Order is 150   
The gain of fir filter : 12-14 kHz is -0.000537 , Order is 150   
The gain of fir filter : 14-16 KHZ is 0.000384 , Order is 150





































