# EC519 BinSrc Code

Jiangyu Wang

May 4, 2017

```
% Author: Jiangyu Wang
% Date: Apr 26 2017
% Email: jiangyu@bu.edu

clear;clc

%% Parameters Setting
bitrate = 2400;
[x,fs] = audioread('s5.wav');
lporder = 10;
winlen = 240; % 30ms
winshift = 240; % Non-overlapping
frame_num = round((length(x)-3*winlen-150)/winlen);
load('pp5.mat');

if bitrate == 2400
    bit_alc = [7 7 7 6 6 5 5 4 4 3];
elseif bitrate == 3600
    bit_alc = [12 10 10 9 9 8 8 8 7 7];
elseif bitrate == 4800
    bit_alc = [16 16 15 14 14 14 13 13 11 9];
end

%% Spectrum
coeff = []; Gain = [];
 for k=1:frame_num
```

```
    frame_procs=x(1+(k-1)*winlen:((k-1)*winlen+winlen));
    coeff_frame3=lpc(frame_procs,lporder);
    r = zeros(lporder+1,1);
    for i=0:lporder
       r(i+1) = frame_procs(1:winlen-i)' * frame_procs(1+i:winlen);
    end
    Gain_frame=sqrt(coeff_frame3*r);
    coeff_frame3=-coeff_frame3(2:lporder+1);
    coeff_frame3(1:lporder,1)=0;
    for i=1:lporder
        coeff_frame3(i)=coeff_frame3(i);
    end
    coeff=[coeff coeff_frame3'];
    Gain = [Gain Gain_frame];
end
[h1,w1]=freqz(1,[1 -coeff(:,31)']); ww=w1*fs/pi/2; H1=10*log10(abs(h1)
plot(ww,H1);title('Power Spectrum'); hold on; plot(ww,H1,'-o'); legend

%% Residual Error & Synthesized File from Err
residual_err = []; sync_re_err = [];
for k=1:frame_num
        coeff_frame2=coeff(:,k);
        frame_procs=x(1+(k-1)*winlen:((k-1)*winlen+winlen));
        error_frame=frame_procs-filter([0 coeff_frame2'],1,frame_procs)
        syn_err_frame=filter(1,[1 -coeff_frame2'],error_frame);
        residual_err=[residual_err error_frame'];
        sync_re_err=[sync_re_err syn_err_frame'];
end
figure(2)
subplot(2,1,1);plot(1:winlen,residual_err(1+(21-1)*winlen:21*winlen),1
+(21-1)*winlen:21*winlen),'-o');
legend('Unquantized','Quantized');title('Residual Error');
subplot(2,1,2);plot(1:winlen,sync_re_err(1+(21-1)*winlen:21*winlen),1:
+(21-1)*winlen:21*winlen),'-o');
legend('Unquantized','Quantized');title('Synthesizing from Residual');

%% Binary Source & Synthesized File from Binary Src
pitch_bit = 7;
```

```
 pitch_quantized(1,1:length(pp5))=0;
for i=1:length(pp5)
    if pp5(i)==0
        pitch_quantized(1,i)=0;
    else
        pitch_quantized(1,i)=quantiz(pp5(i,1)-54,1:1:2^pitch_bit);
    end
end

index=0;
bin_src=[];
for i=1:96
    pitch_per=pitch_quantized(i);
    if(pitch_per == 0)
        bin_src=[bin_src randn(1,winlen)*0.05];
    else
        pulse=zeros(1,winlen);
        if(index==0)
            pulse(1)=1;
            pos=1;
            while (pitch_per+pos < winlen)
                pulse(pos+pitch_per)=1;
                pos=pos+pitch_per;
            end
        else
           pos=pos-winlen;
           if (pos+pitch_per <1)
               pos=1-pitch_per;
           end
           while(pos+pitch_per < winlen)
               pulse(pos+pitch_per)=1;
               pos=pos+pitch_per;
           end
       end
       bin_src=[bin_src pulse];
   end
   index=pitch_per;
end
```

```
 sync_binary=[];
for k=1:96
    coeff_frame3=coeff(:,k);
    bin_src_frame=bin_src(1+(k-1)*winlen:k*winlen);
    sync_binary_frame=filter(1,[1 -coeff_frame3'],bin_src_frame);
    sync_binary=[sync_binary sync_binary_frame];
end
figure(3)
subplot(2,1,1);plot(bin_src(1+(1-1)*winlen:1*winlen),'-o');
legend('Unquantized','Quantized');title('Binary Source');
subplot(2,1,2);plot(1:winlen, sync_binary(1+(1-1)*winlen:1*winlen),'-o'
legend('Unquantized','Quantized');title('Synthesizing from Binary Source
```