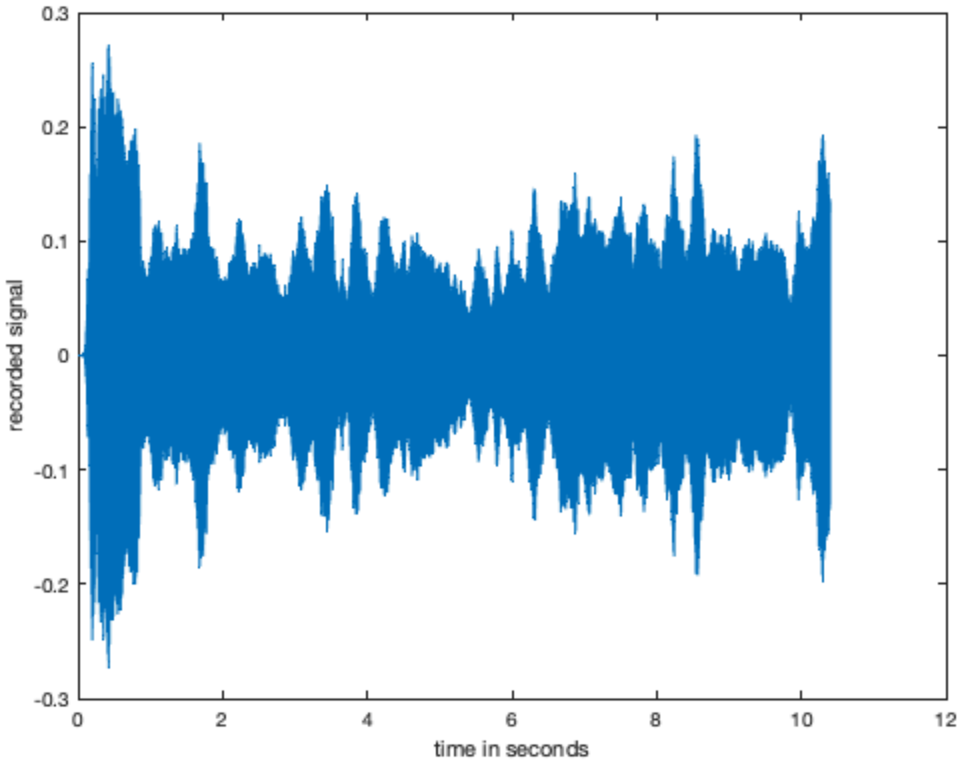
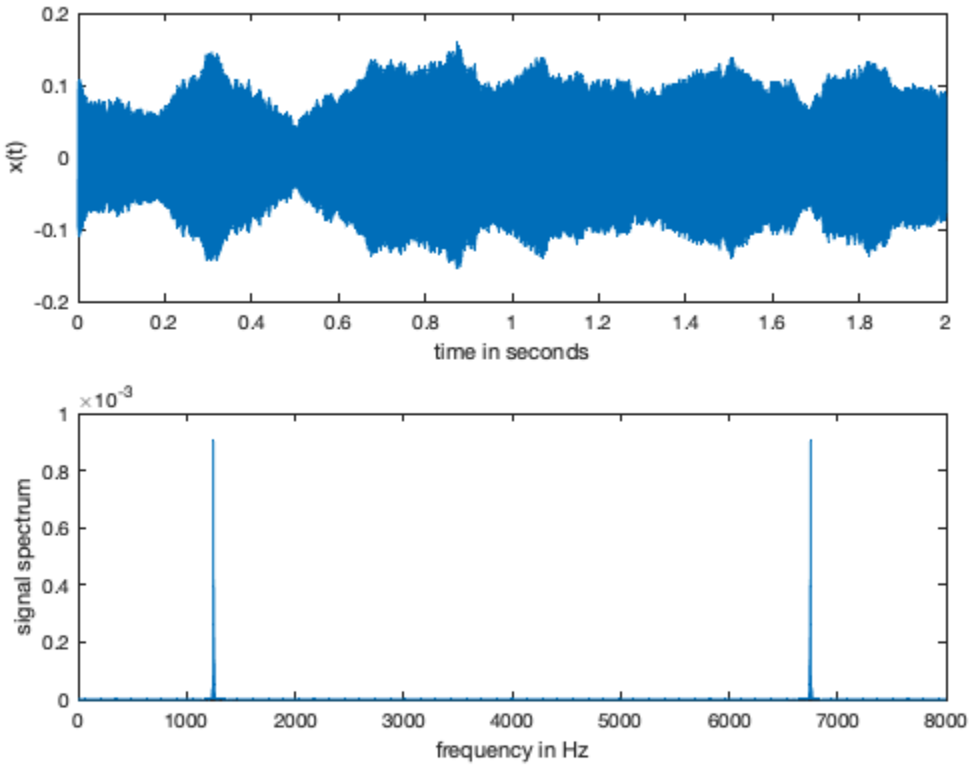
```
% TSRT78, Lab 1: Fundamental Signal Processing
% Matheus Bernat (matvi959) & Caspian Süsskind (cassu286)

% ===================== 4 Assignment: Whistle
 ============================


% Read wav file: extract data and sampling frequency
[y, fSamp] = audioread('whistle.wav');

% Check that 8000Hz
fSamp;
nSamp = size(y,1);


% Hear sound:
sound(y,fSamp);

% ------------------ QUESTION 1 ------------------
% A lot of stuff

% --------------- Plot signal in time axis
ts = 1/fSamp;
timeVector = ts*(0:nSamp-1); % time vector in seconds

figure;
plot(timeVector, y)
xlabel('time in seconds');
ylabel('recorded signal');

% --------------- Calculate energy of signal in time domain

% Get signal from 6 to 8 seconds
idx = (timeVector >= 6) & (timeVector <= 8);
y = y(idx);

nSamp = size(y, 1);
timeVector = ts*(0:nSamp-1);
figure; subplot(2,1,1);
plot(timeVector, y)
xlabel('time in seconds');
ylabel('x(t)');

totalEnergy_t = 0;
for i = 1:length(y)
    totalEnergy_t = totalEnergy_t + abs(y(i))^2;
end

% --------------- Plot spectrum
Yf = fft(y);
frequencyVector = ((0:nSamp-1)/nSamp)*fSamp;
subplot(2,1,2);
```

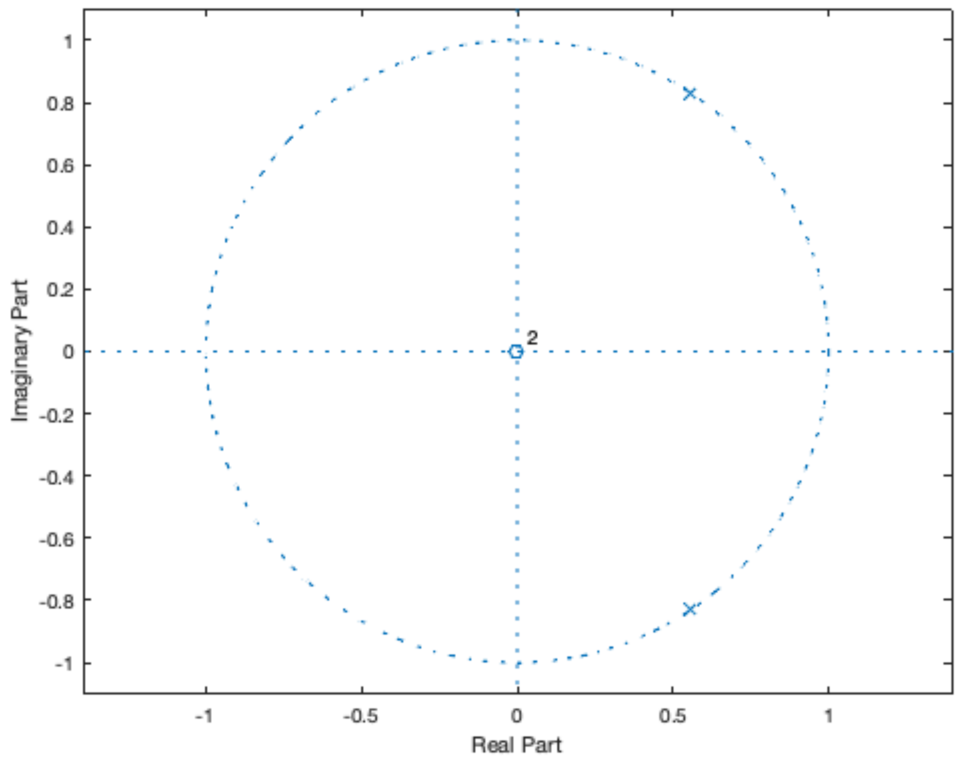
```
plot(frequencyVector , (abs(Yf).^2)*ts/nSamp)
xlabel('frequency in Hz');
ylabel('signal spectrum');

% By looking at spectrum, decide dominant frequency 1250
dominantFreq = 1250;
sth = 10;

yFiltered = bandpass(y, [dominantFreq - sth, dominantFreq + sth],
 fSamp);

% --------------- Calc energy of dominant frequency signal in time
 domain
dominantFreqEnergy_t = 0;
for i = 1:length(yFiltered)
    dominantFreqEnergy_t = dominantFreqEnergy_t+ abs(yFiltered(i))^2;
end

% ANSWERS:
totalEnergy_t;
dominantFreqEnergy_t;
% ------------------ QUESTION 2 ------------------
% Same calculations as in question 1, but in the frequency domain.

% Calculate energy in the frequency domain
totalEnergy_f = 0;
for i = 1:length(Yf)
    totalEnergy_f = totalEnergy_f + abs(Yf(i)^2)*ts/nSamp;
end

% Calculate energy of dominant frequency in frequency domain
idx = (frequencyVector >= dominantFreq - sth) & (frequencyVector <=
 dominantFreq +sth);
dominantFreqSignal = Yf(idx);

dominantFreqEnergy_f = 0;
for i = 1:length(dominantFreqSignal)
    dominantFreqEnergy_f = dominantFreqEnergy_f +
 2*abs(dominantFreqSignal(i))^2/nSamp;
end

% ANSWERS:
totalEnergy_f;
dominantFreqEnergy_f;

% ------------------ QUESTION 3 ------------------
% Calc harm. distortion using energy calculations from time and freq
 domain

% ANSWERS:
hdist_t = 1 - dominantFreqEnergy_t/totalEnergy_t; % 0.0030
hdist_f = 1 - dominantFreqEnergy_f/totalEnergy_f; % 0.0680
```

```
% ------------------- QUESTION 4 -------------------
% Estimate the purity measure based on an AR(2) and motivate why this
 model
% is suitable. How can this measure be compared to the harmonic
 distortion?

modelOrder = 2;
[th,P,lam,epsi] = sig2ar(y,modelOrder);
a1 = th(1,1); a2 = th(2,1);

figure;
zplane([1],[1 a1 a2]);

pole1 = -a1/2 + sqrt(((a1^2)/4)-a2);
pole2 = -a1/2 - sqrt(((a1^2)/4)-a2);

% ANSWERS:
distance = 1 - abs(pole1);

% ------------------- QUESTION 5 -------------------
arMod = ar(y, 2, 'Ts', ts);
figure, bode(arMod)

% Plot for non parametric method in QUESTION 1
```

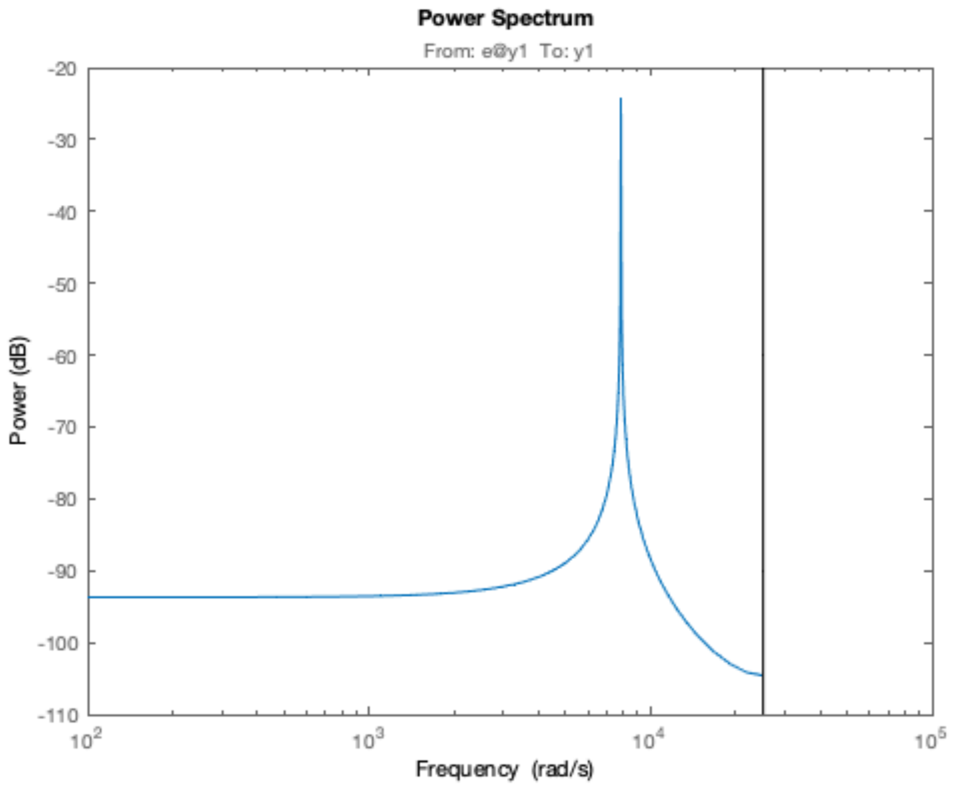
```
Warning: A bode plot is not well defined for a time series model. The
 plot will
show the output spectrum of the model. Consider using the "idlti/
spectrum"
command instead.
```


recorded signal
time in seconds


x(t)
time in seconds
signal spectrum
frequency in Hz



Imaginary Part
Real Part
2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1



Power Spectrum
From: e@y1 To: y1
Power (dB)
Frequency (rad/s)
-20
-30
-40
-50
-60
-70
-80
-90
-100
-110
10^2
10^3
10^4
10^5

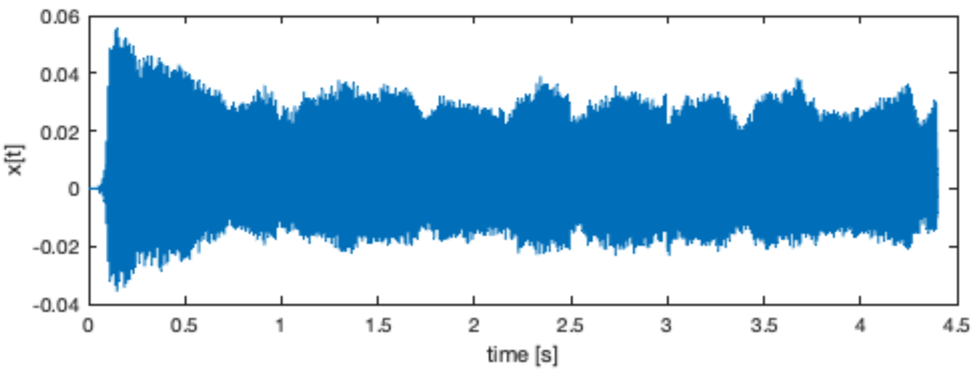
Published with MATLAB® R2020a

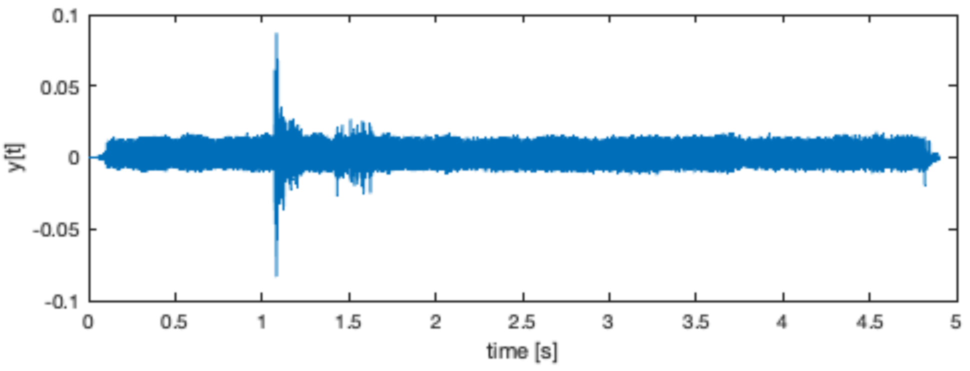
```
% TSRT78, Lab 1: Fundamental Signal Processing
% Matheus Bernat (matvi959) & Caspian Süsskind (cassu286)

% ===================== 5 Assignment: Vowel
 ============================

% ------------------ INTRO ------------------
clear;
% --------------- Read wav file: extract data and sampling frequency
x = audioread('aaaaa.wav');
y = audioread('ooooo.wav');
fs = 8000;
Nx = size(x,1);
Ny = size(y,1);
x = detrend(x);
y = detrend(y);
% --------------- Plot signal in time axis
Ts = 1/fs;
tx = Ts*(0:Nx-1); % time vector in seconds
ty = Ts*(0:Ny-1); % time vector in seconds

figure(1);clf();
subplot(2,1,1); plot(tx, x); xlabel('time [s]'); ylabel('x[t]');
subplot(2,1,2); plot(ty, y); xlabel('time [s]'); ylabel('y[t]');

% --------------- Get the 2 most consistent seconds of both signals
idx_x = (tx >= 1) & (tx <= 3); x = x(idx_x);
idx_y = (ty >= 2) & (ty <= 4); y = y(idx_y);

Nx = size(x, 1); tx = Ts*(0:Nx-1);
Ny = size(y, 1); ty = Ts*(0:Ny-1);

figure(2);clf();
subplot(2,1,1); plot(tx, x); xlabel('time [s]'); ylabel('x[t]');
subplot(2,1,2); plot(ty, y); xlabel('time [s]'); ylabel('y[t]');

% --------------- Plot frequency content, count peaks to get model
 order
X = fft(x);
Y = fft(y);
fx = (0:Nx-1)*fs/Nx;
fy = (0:Ny-1)*fs/Ny;
figure(3); clf();
subplot(2,1,1); plot(fx, abs(X)); % Order of AR model should be 14*2 =
 28
xlabel('frequency [Hz]'); ylabel('X[f]');
title('Frequency spectrum for a')
subplot(2,1,2); plot(fy, abs(Y)); % Order of AR model should be 6*2 =
 12
xlabel('frequency [Hz]'); ylabel('Y[f]');
title('Frequency spectrum for o')
```

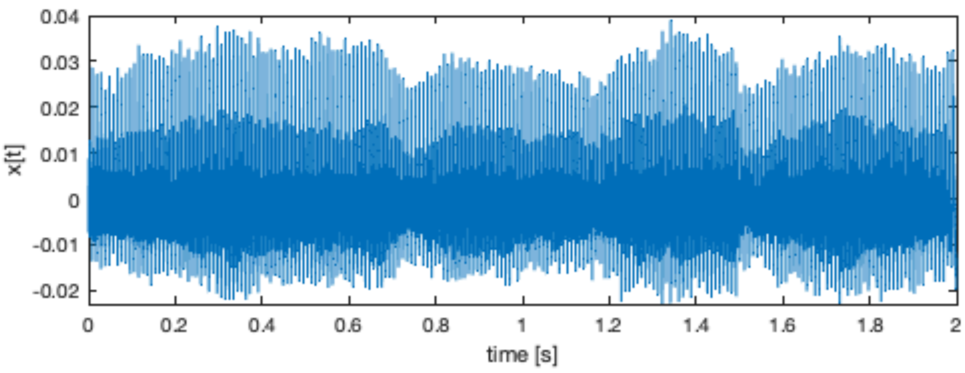
```
% --------------- Model order estimation with loss/AIC/BIC
figure;
subplot(2,1,1);
maxOrder = 50;
arorder(x, maxOrder);
legend('Minimal loss function',...
       'Akaikes information criterion (AIC)',...
       'Akikes information criterion B (BIC)');
title('Order selection analysis of vowel a');
xlabel('Model order');
subplot(2,1,2);
arorder(y, maxOrder);
legend('Minimal loss function',...
       'Akaikes information criterion A (AIC)',...
       'Akikes information criterion B (BIC)');
title('Order selection analysis of vowel o');
xlabel('Model order');

% --------------- Create AR models

% AR models for vowel 'a' of orders 2, 9 and 18
estIdx_x = floor(2*Nx/3);
modelOrder_x = 2;
arMod_x_2 = ar(x(1:estIdx_x), modelOrder_x, 'Ts', Ts);
modelOrder_x = 9;
arMod_x_9 = ar(x(1:estIdx_x), modelOrder_x, 'Ts', Ts);
modelOrder_x = 18;
arMod_x_18 = ar(x(1:estIdx_x), modelOrder_x, 'Ts', Ts);

% AR models for vowel 'o' of orders 3, 5 and 10
estIdx_y = floor(2*Ny/3);
modelOrder_y = 3;
arMod_y_3 = ar(y(1:estIdx_y), modelOrder_y, 'Ts', Ts);
modelOrder_y = 5;
arMod_y_5 = ar(y(1:estIdx_y), modelOrder_y, 'Ts', Ts);
modelOrder_y = 10;
arMod_y_10 = ar(y(1:estIdx_y), modelOrder_y, 'Ts', Ts);

% ------------------ Validation of model ------------------

% Validation method 1: compare power spectra
f = 0:0.05:1;
Phi1 = arMod_x_18.NoiseVariance*Ts*abs(freqz(1, arMod_x_18.a,
 pi*f)).^2;
Phi2 = arMod_x_9.NoiseVariance*Ts*abs(freqz(1, arMod_x_9.a, pi*f)).^2;
Phi3 = arMod_x_2.NoiseVariance*Ts*abs(freqz(1, arMod_x_2.a, pi*f)).^2;
[Phi4, f4] = sig2blackmantukey(x(estIdx_x+1:end), 30, Ts);
figure;
subplot(2,1,1);
semilogy(fs*f/2,Phi1, fs*f/2,Phi2,fs*f/2, Phi3, f4/2, Phi4);
legend('AR(18)', 'AR(9)', 'AR(2)', 'Blackman-Tukey Estimate');
xlabel('Frequency (Hz)'); title('Spectrum Estimates for a');
```

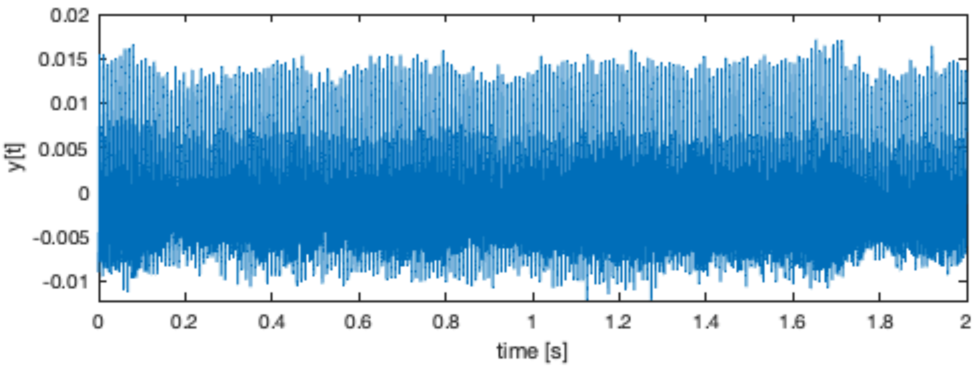
```
Phi5 = arMod_y_10.NoiseVariance*Ts*abs(freqz(1, arMod_y_10.a,
 pi*f)).^2;
Phi6 = arMod_y_5.NoiseVariance*Ts*abs(freqz(1, arMod_y_5.a, pi*f)).^2;
Phi7 = arMod_y_3.NoiseVariance*Ts*abs(freqz(1, arMod_y_3.a, pi*f)).^2;
[Phi8, f8] = sig2blackmantukey(y(estIdx_x+1:end), 30, Ts);


subplot(2,1,2);
semilogy(fs*f/2,Phi5, fs*f/2,Phi6,fs*f/2, Phi7, f8/2, Phi8);
legend('AR(10)', 'AR(5)', 'AR(3)', 'Blackman-Tukey Estimate');
xlabel('Frequency (Hz)'); title('Spectrum Estimates for o');



% Validation method 2: residual whiteness test
eps_x_18 = pe(arMod_x_18, x(estIdx_x+1:end));
[Rex_18, k] = sig2crosscovfun(eps_x_18, x(estIdx_x+1:end));
eps_x_9 = pe(arMod_x_9, x(estIdx_x+1:end));
Rex_9 = sig2crosscovfun(eps_x_9, x(estIdx_x+1:end));
eps_x_2 = pe(arMod_x_2, x(estIdx_x+1:end));
Rex_2 = sig2crosscovfun(eps_x_2, x(estIdx_x+1:end));

eps_y_10 = pe(arMod_y_10, y(estIdx_y+1:end));
Rey_10 = sig2crosscovfun(eps_y_10, y(estIdx_y+1:end));
eps_y_5 = pe(arMod_y_5, y(estIdx_y+1:end));
Rey_5 = sig2crosscovfun(eps_y_5, y(estIdx_y+1:end));
eps_y_3 = pe(arMod_y_3, y(estIdx_y+1:end));
Rey_3 = sig2crosscovfun(eps_y_3, y(estIdx_y+1:end));

figure;
subplot(2,1,1);
plot(k, Rex_18, 'xm', k, Rex_9, '--b', k, Rex_2, '.-r');
legend('Rea(k) for AR(18)', 'Rea(k) for AR(9)', 'Rea(k) for AR(2)');
title('Cross correlation of residuals for a');
xlabel('k'); ylabel('Rea(k)');
subplot(2,1,2);
plot(k, Rey_10, 'xm', k, Rey_5, '--b', k, Rey_3, '.-r');
legend('Reo(k) for AR(10)', 'Reo(k) for AR(5)', 'Reo(k) for AR(3)');
title('Cross correlation of residuals for o');
xlabel('k'); ylabel('Reo(k)');

% ------------------ Simulation of model ------------------
b = 1;
ax = arMod_x_9.A; % coefficients of the AR-model
ay = arMod_y_5.A;

pulseTrain = ones(1, Nx);

e = filter(ax, 1, x);
r = covf(e, 100);
[A, D] = max(r(20:end));
D = D + 19; % Look at max from t>19, so add 19 to time lag
ehat = (mod(1:Nx, D) == 0);
sim_x = filter(1,ax,ehat);
```

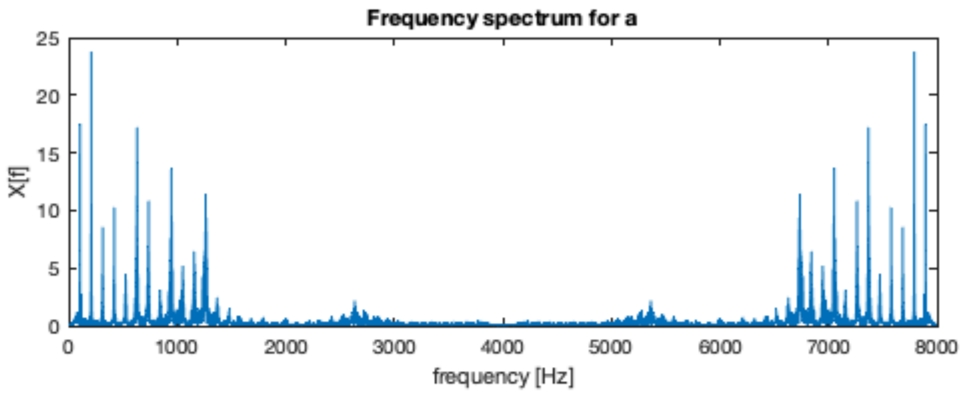
```
e = filter(ay, 1, y);
r = covf(e, 100);
[A, D] = max(r(20:end));
D = D + 19; % Look at max from t>19, so add 19 to time lag
ehat = (mod(1:Ny, D) == 0);
sim_y = filter(1,ay,ehat);

sim_X = fft(sim_x);
sim_Y = fft(sim_y);

sound([sim_x, sim_y]);
```

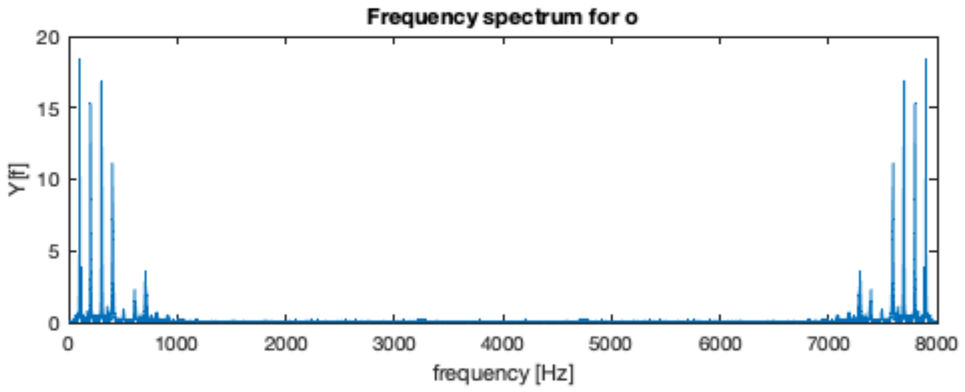


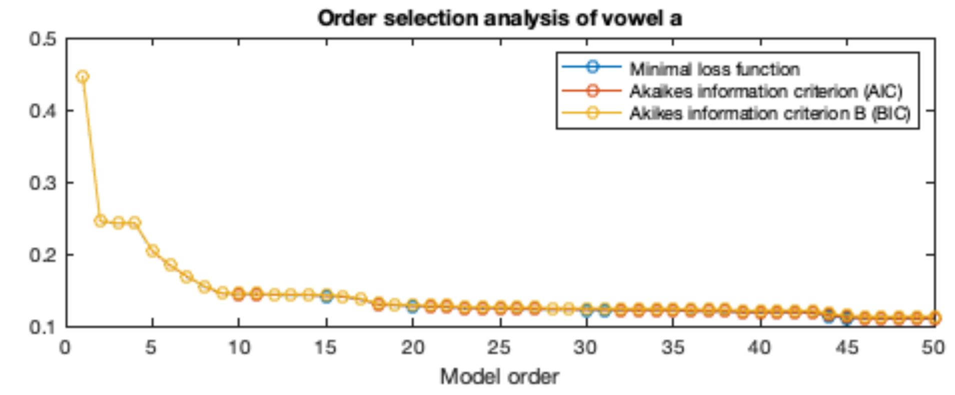


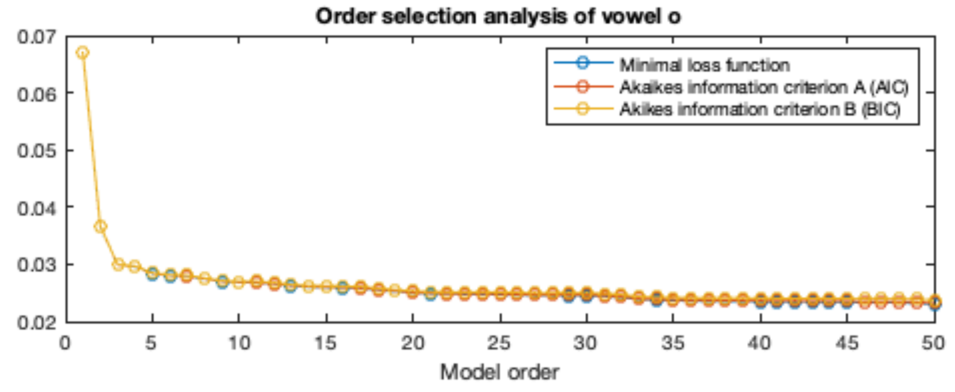

x[t]
time [s]
0.04
0.03
0.02
0.01
0
-0.01
-0.02
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

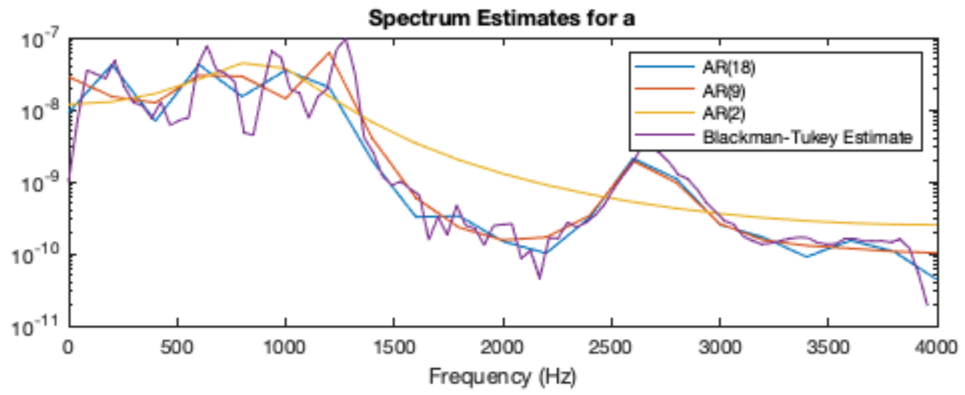

y[t]
time [s]
0.02
0.015
0.01
0.005
0
-0.005
-0.01
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

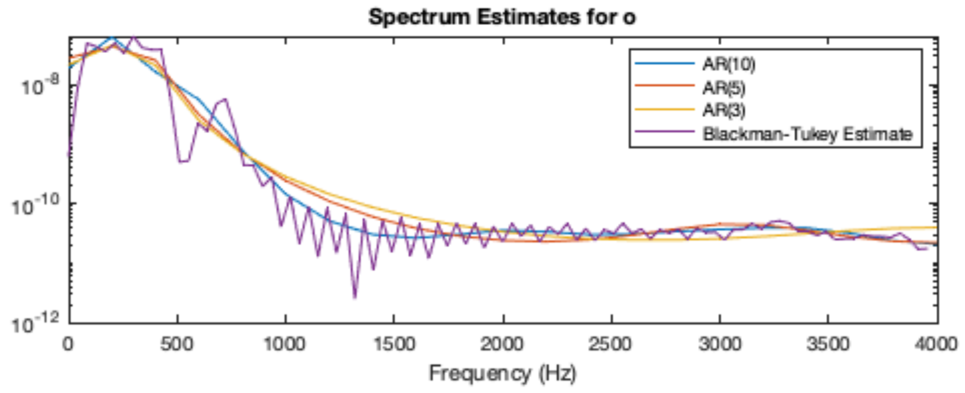

Frequency spectrum for a
X[f]
frequency [Hz]
25
20
15
10
5
0
0
1000
2000
3000
4000
5000
6000
7000
8000

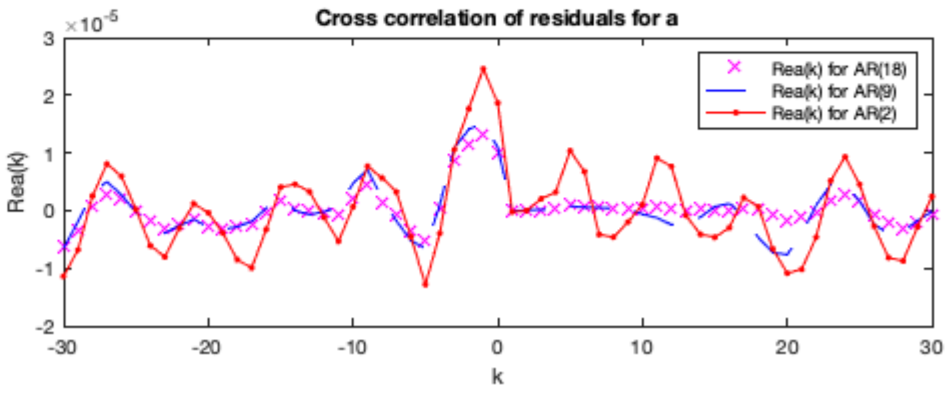

Frequency spectrum for o
Y[f]
frequency [Hz]
20
15
10
5
0
0
1000
2000
3000
4000
5000
6000
7000
8000

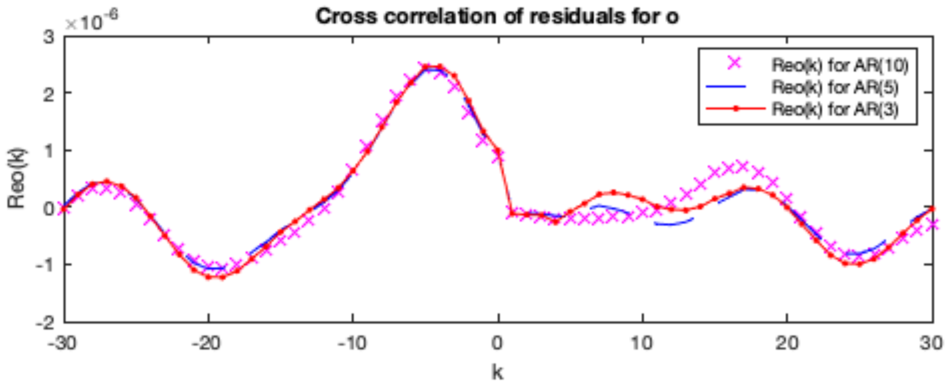

Order selection analysis of vowel a
Minimal loss function
Akaikes information criterion (AIC)
Akikes information criterion B (BIC)
Model order


Order selection analysis of vowel o
Minimal loss function
Akaikes information criterion A (AIC)
Akikes information criterion B (BIC)
Model order


Spectrum Estimates for a
AR(18)
AR(9)
AR(2)
Blackman-Tukey Estimate
Frequency (Hz)

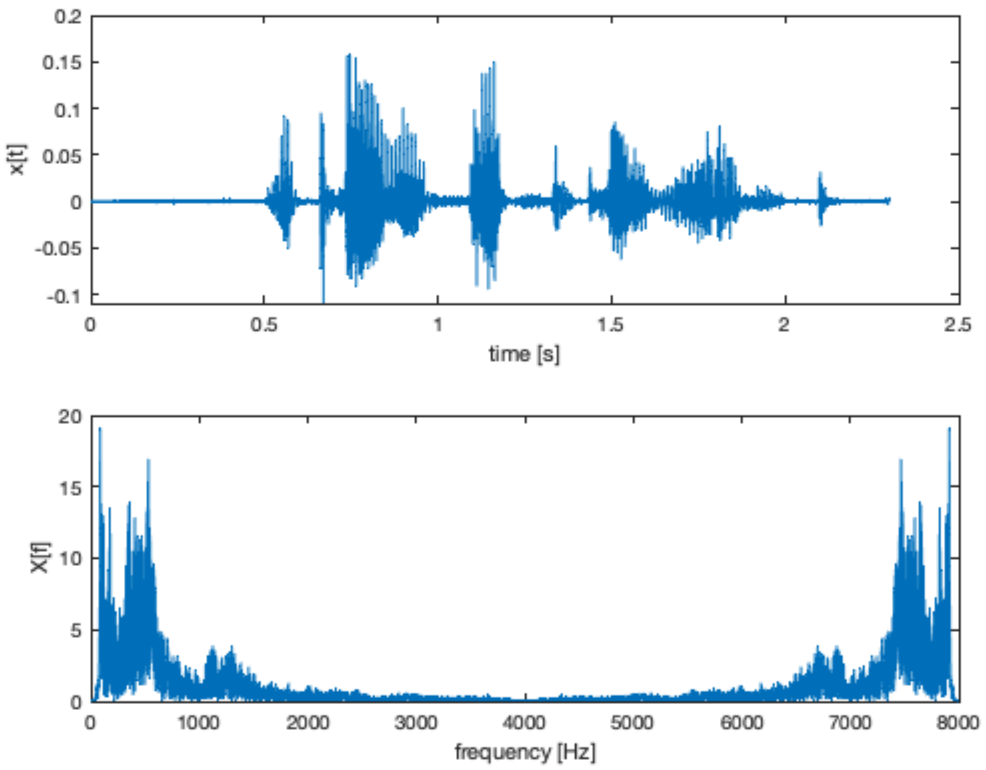

Spectrum Estimates for o
AR(10)
AR(5)
AR(3)
Blackman-Tukey Estimate
Frequency (Hz)

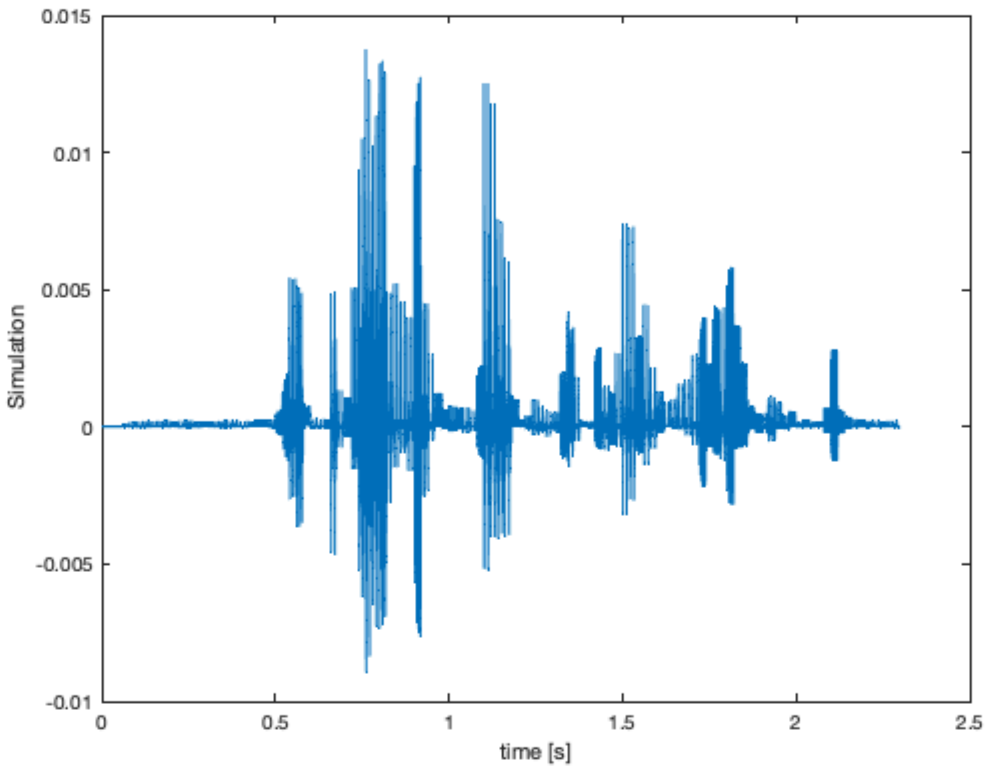




Published with MATLAB® R2020a

```
% TSRT78, Lab 1: Fundamental Signal Processing
% Matheus Bernat (matvi959) & Caspian Süsskind (cassu286)

% ================= 6 Assignment: Speech encoding as in GSM
 ===============


% ------------------ INTRO ------------------
clear;
% --------------- Read wav file: extract data and sampling frequency
[x,fs] = audioread('cleverFox.wav');
N = size(x,1);
segmentLength = 160;
% --------------- Plot signal in time and frequency axes
Ts = 1/fs;
t = Ts*(0:N-1); % time vector in seconds
numSegments = floor(N/segmentLength);
figure(1);clf();
subplot(2,1,1); plot(t, x), xlabel('time [s]'); ylabel('x[t]');
X = fft(x);
f = (0:N-1)*fs/N;

subplot(2,1,2); plot(f, abs(X))
xlabel('frequency [Hz]'); ylabel('X[f]');



% --------------- Create 115 AR models, one for each 160 point segment
modelOrder = 16;
sounds = zeros(size(x));

for row = 1:numSegments
    segment = detrend(x(1+(row-1)*segmentLength:row*segmentLength)); %
 Fetch 160 points from recording
    m = ar(segment, modelOrder, 'Ts', Ts);

    % Check for unstable poles and mirror
    poles = roots(m.a);
    if max(abs(poles)) > 1
        disp('Pole outside unit circle');
        for idx = 1:size(poles,1)
            if abs(poles(idx)) > 1
                poles(idx) = 1/poles(idx);
            end
        end
        m.a = poly(poles);
    end

    e = filter(m.a, 1, x(1+(row-1)*segmentLength:row*segmentLength));
    r = covf(e, 100);
    [A, D] = max(r(20:end));
    D = D + 19; % Look at max from t>19, so add 19 to time lag
    amp = sqrt(A);
    ehat = amp*(mod(1:160, D) == 1);
    yhat = filter(1, m.a, ehat);
    sounds(1+(row-1)*segmentLength:row*segmentLength) = yhat;
end

% --------------- Play up sound by the reconstructed sound
sounds = reshape(sounds, N, 1);
sound(10*sounds, fs);
figure;clf();
plot(t, sounds), xlabel('time [s]'); ylabel('Simulation');
```





Published with MATLAB® R2020a