TSRT78 - Lab 1 Report, group nr 04

Axel Söderlind
Liu ID: *axeso712*
19980612-XXXX

Philip Welin-Berger
Liu ID: *phiwe030*
19300519-XXXX

***Abstract*—The purpose of this lab was to record natural signals, such as voice and whistles, and recreate these sounds in Matlab using signal processing models from the course. In this lab the focus was on the linear AR model, of different orders. Vowel sounds and even a longer string of words were successfully recreated using well-chosen AR models and techniques involving segmenting and dynamically choosing parameters.**

I. Introduction

This is a report for a laboration in the course Digital Signal Processing TSRT78. The laboration consists of three assignments. All of the tasks are related to AR models and signal estimations. The assignments are implemented in Matlab. Input data for the assignments where collected by recording sounds and importing them into Matlab.

II. Task 1 Whistle

The goal of the first assignment was to whistle as purely as possible with regards to two different measures. The first measure was harmonic distortion, which is defined as

$$1 - \frac{E_{dom.freq}}{E_{tot}} \tag{1}$$

Where a distortion of 0 means the whistle is completely pure, and a value closer to 1 means it is less pure. The second measure was fitting a second order AR model to the recorded whistle and measure the distance of the poles of the transfer function to the unit circle in the complex plane. The transfer function of a second order AR model is defined as

$$T(z) = \frac{1}{A(z)} = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{2}$$

Which can be compared to the Z-transform of a pure sinus wave

$$\begin{aligned} Z\{\sin \omega t\} &= Z\{\frac{1}{2j}(e^{j\omega t} + e^{-j\omega t})\} \\ &= \frac{1}{2j}(\frac{z}{z - e^{j\omega t}} - \frac{z}{z - e^{-j\omega t}}) \\ &= \frac{1}{2j}\frac{z(z - e^{-j\omega t}) - z(z - e^{j\omega t})}{(z - e^{j\omega t})(z - e^{-j\omega t})} \end{aligned}$$

In which the two poles lie on the unit circle, since $|e^{jwt}| = 1$. Since this is true for all ω, the distance of the poles in the AR(2) transfer function to the unit circle is a good proxy for measuring closeness to the corresponding pure sine wave. A second order model is a good choice when the aim is to imitate a simple sine wave since it is the lowest model order necessary to generate a periodic signal.

The natural whistle was recorded with a sampling frequency of 8000 Hz and loaded into Matlab using the `audioread` function. The most stable 2 seconds of the signal was chosen and cut out. The dominant frequency was identified to be located around $x = 1090$ Hz by observation of the graph of the Fourier transform of the signal, see figure 1. For the calculation of the energy of the dominant frequency in the time domain, a bandpass filter was used to filter the signal around the dominant frequency. An allowance of 1 percent around the dominant frequency was chosen, meaning the signal was band-limited around 1080 Hz and 1100 Hz, a clearer view of the FFT can be seen in figure 2. The energy in the time domain was subsequently calculated on the full, and bandpassed signal respectively using

$$E = \sum_{k \in samples} |x(k)|^2 \, dt \tag{3}$$

For the frequency domain the whistle was first transferred to the fourier domain using the FFT, then the energy for the full frequency span was calculated for the total energy. For the energy around the dominant frequency the coordinate with the largest amplitude was identified and then similiarly the frequencies within 10 Hz was considered and the energy calculated using

$$E = \frac{1}{N} \sum_{n=0}^{N-1} |X(n)|^2 \, df \tag{4}$$

The results can be seen in table I.

Table I
Energy of the whistle

Whistle Energy Distribution			
Domain	E_{tot}	E_{dom}	*Distortion*
Time	0.4549	0.1628	0.6422
Frequency	0.4549	0.1595	0.6494

The second order AR model is calculated using built-in Matlab functions. The AR-model of the whistle has a transfer function with roots $z = 0.6561 \pm 0.7519i$, with absolute values of 0.9979. The absolute values are very close to 1, which means the poles lie close to the unit circle. This implies the whistle is very pure.

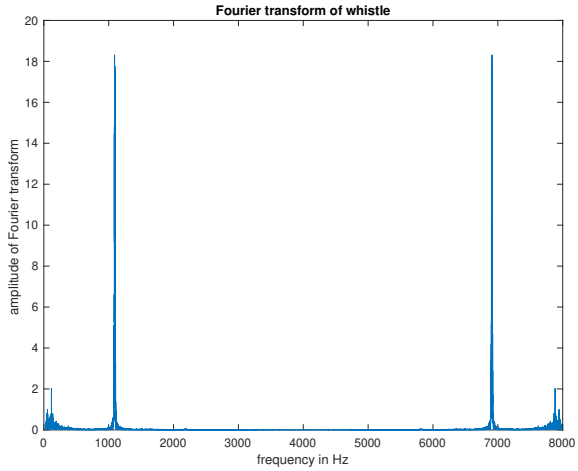

Figure 1. The FFT of the whistle

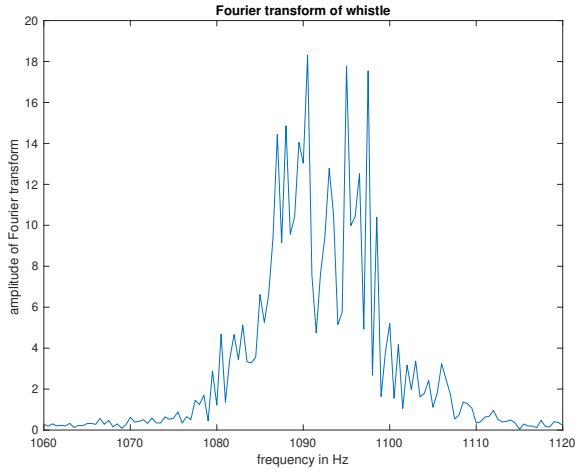

Figure 2. The FFT of the whistle (zoomed in)

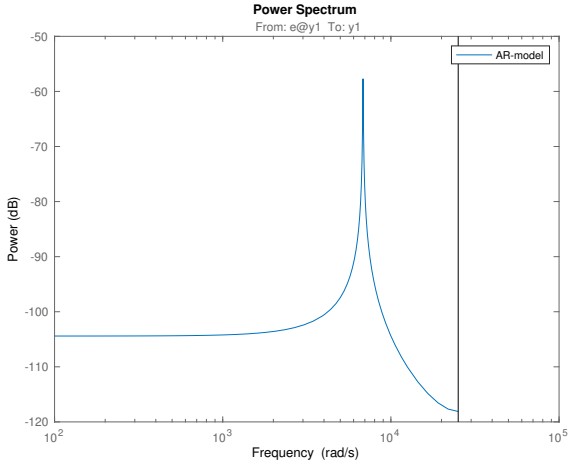

Figure 3. The parametric spectrum of the whistle

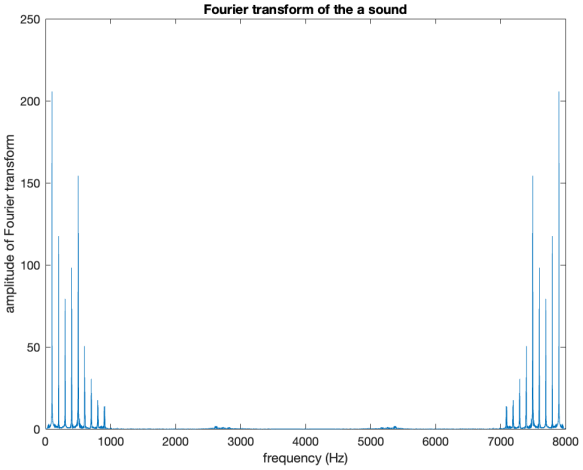

Figure 4. The FFT of the a sound

The non-parametric spectra, see figure 1, can be compared to the parametric spectra, generated using the AR(2) model, see figure 3. In the parametric spectra we identify a clear peak at 6830 rad/s, which can be converted to around 1087 Hz. This is the same frequency (± 3 Hz) that was identified as the max in the Fourier transform of the whistle in the non-parametric case.

III. Task 2 Vowels

The goal of task 2 was to model the sound of a vowel using an AR model of an appropriate order and then simulate the model using a suitable input signal and then listen to the result.

Two vowel sounds, a and o, were recorded at 8000 Hz for 10 seconds each, whereupon they were loaded into Matlab using the `audioread` function. Then the most stable 2 seconds of each signal was chosen and cut out.

A. AR model order estimation

To properly be able to model the sounds with an AR model an appropriate order must be chosen. Several methods was used in order to accomplish this. The first was to estimate the number of peaks in each recordings frequency spectra. Each peak represents one frequency present in the signal and thus should represent one additional order that is needed in the AR model in order to be able to represent the signal accurately. As such the frequency spectra for the a sound, seen in figure 4, and the o sound, seen in figure 5, were examined which gave an estimated model order of 18 for the a sound and 12 for the o sound. The second method was to minimize the loss function

$$W_N(n) = V(\hat{\theta}_N^{(n)}) = \frac{1}{N}\sum_{t=1}^{N} \epsilon(t, \hat{\theta}_N^{(n)})^2$$

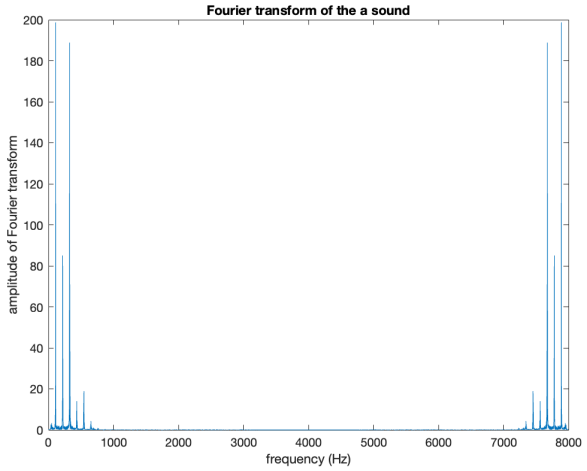

Figure 5. The FFT of the o sound

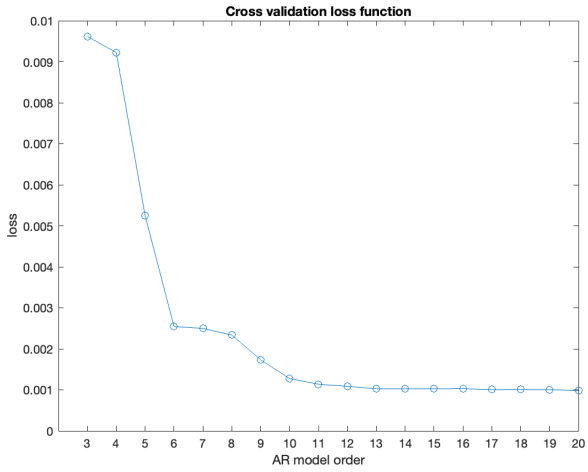

Figure 6. The cross validation loss function for the o sound for different model orders.

on validation data were the signals of the respective vowels got split up into $\frac{2}{3}$ estimation data and $\frac{1}{3}$ validation data. This is because the loss function is strictly decreasing on the entire signal and this cannot be minimized directly. This method was implemeted using the `arordercv` function of the signal processing library and the loss for various orders was plotted in figure 6 and figure 7 respectively. For the a sound a model order of 13 seemed appropriate and an order of 3 for the o sound. Thirdly, the order was estimated using Akaike's Information Criterion (AIC) and the Bayesian Information Criterion (BIC). They compute the loss function, but add a penalty term which becomes

$$U_N(n) = W_N(n)\left(1 + \frac{2n}{N}\right)$$

for AIC and

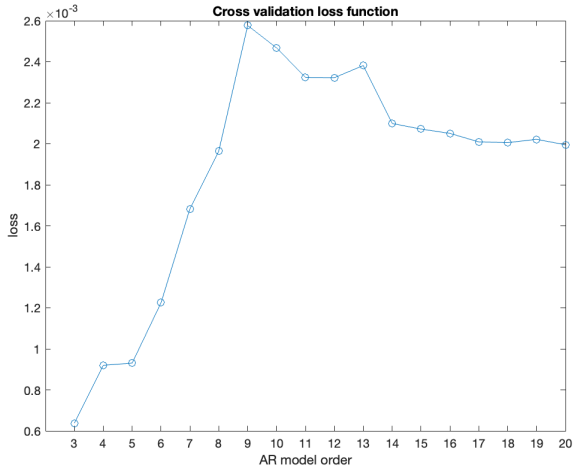

Figure 7. The cross validation loss function for the o sound for different model orders.

$$U_N(n) = W_N(n)\left(1 + \frac{n\log(N)}{N}\right)$$

for BIC. These were implemented using the `arorder` function from the signal processing library and plotted for the a and o sound respectively seen in figure 8 and figure 9. The `arorder` function also returned a regular loss function that also was plotted for reference in the same figures. This method suggested a order of 13 would be appropriate for the a sound and an order of 16 for the o sound.

Ultimately, the orders were chosen for testing for each sound. 3, 18, and 30 for the a sound and 3, 16, and 30 for the o sound. These were deemed to be interesting based on the result from the different estimation methods and also to see how upper and lower bounds of those estimations would preform.

B. Model validation

The chosen orders were then validated by using two methods. For both methods the signals were once split up into $\frac{2}{3}$ estimation data and $\frac{1}{3}$ validation data.

Firstly, the power spectra was calculated using and plotted in a bode plot together with the chosen ar models for the respective sounds as seen in figure 10 and figure 11. Analysis of the graph suggested that neither of the chosen model orders for any of the seem to model the signal accurately. Although the higher orders were able to capture more of the signals peaks.

Secondly, 1-step prediction were carried out using Matlab's `predict` and `compare` functions to check how accurately the models could predict the signals. Note that the models were created using the estimation part of the signal and then

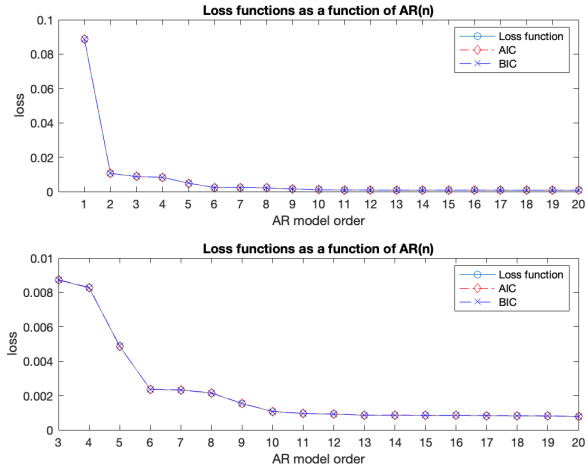

Figure 8. AIC, BIC, and loss function for the a sound for orders between 1 and 20.

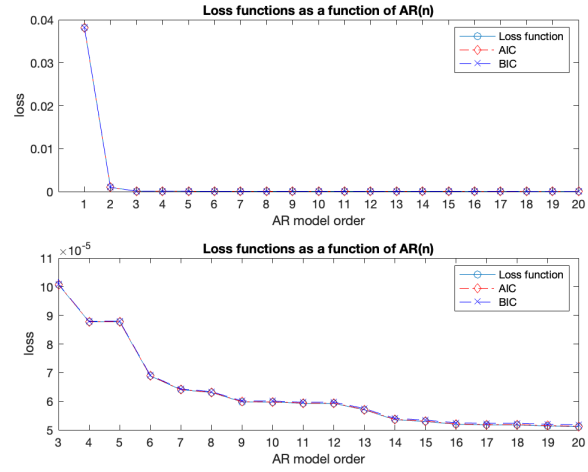

Figure 9. AIC, BIC, and loss function for the o sound for orders between 1 and 20.

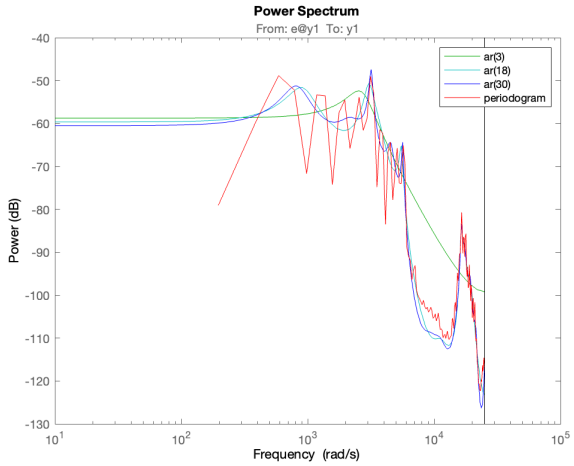

Figure 10. Bode plot for the spectra of the validation data of the a sound together with its chosen ar model orders.

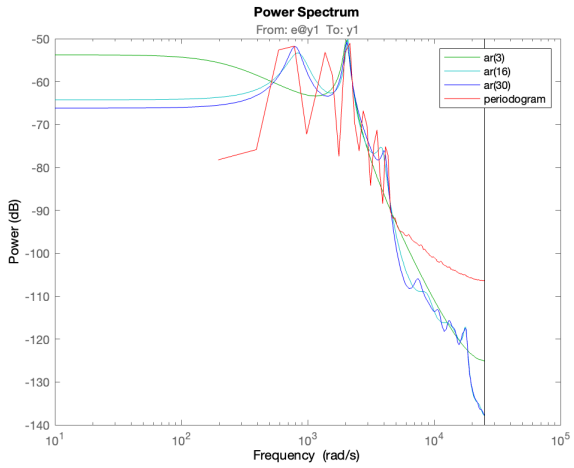

Figure 11. Bode plot for the spectra of the validation data of the o sound together with its chosen ar model orders.

tested on the validation part, the result of which can be seen in figure 12 and figure 13. For the a sound model order 18 had good coverage with model order 30 being marginally better. The o sound had virtually the same high prediction accuracy for all three model orders. For both sounds, contrary to the spectral analysis all model orders seem to have the capability to model the signal accurately to a certain degree.

C. Signal reconstruction

Finally the signals were reconstructed using the different model orders with a pulse train as the input signal. For the a sound order 18 were closest to the original signal with order 30 sounding the same. For the o sound order 3 sounded the best. For both vowels the the reconstructed signal was recognisable, but sounded more robotic and flat compared to their original signal.

The model order of the a signal was within expectations based on analysis of the frequency spectra and the two loss function methods. However, the o sound sounding the best for model order 3 was unexpected. It shows that the order estimation methods provide a suggestion for model order, but are not necessarily accurate.

In both cases however, the conclusion can be drawn that it is possible to model the vowel sounds with a relatively small number of parameters and still retain a recognisable signal.

IV. Task 3 GSM

In the last assignment an AR(8) model is used to estimate a recorded sentence. Here the same technique as in the previous tasks was used to encode sounds, the difference being that the sentence was split up into segments of 160 samples each.

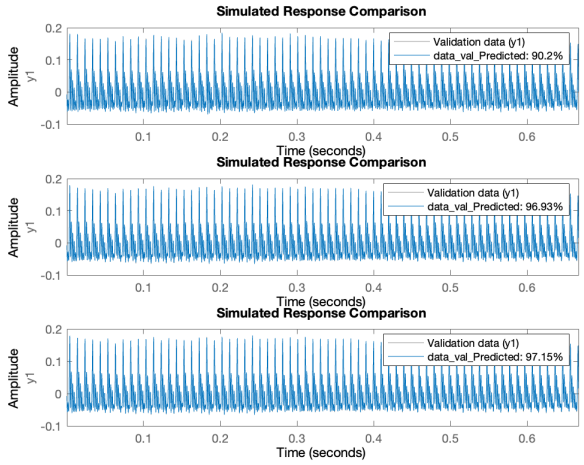

Figure 12. 1-step prediction for 3 different model orders of the a sound.

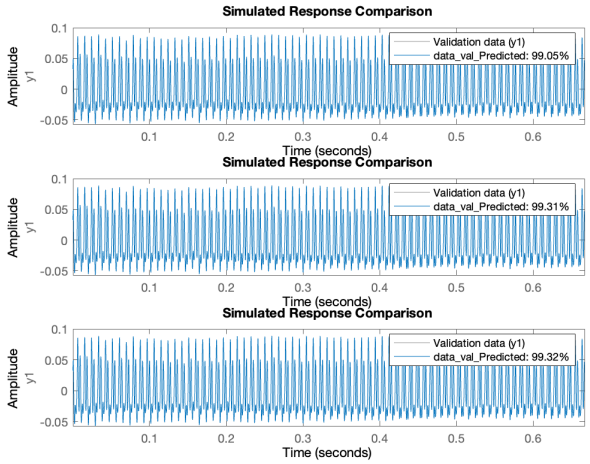

Figure 13. 1-step prediction for 3 different model orders of the o sound.

Each segment was modeled using an AR(8) model and later reassembled to create the final result. For the pulse train amplitude the square root of the max of the auto-covariance, $\sqrt{A}$ was used. To make sure the all the poles were stable, the inverse were taken in case they were outside of the unit circle. Also for every segment we remove the mean by way of the function `detrend`. The resulting voice synthesis sounds legible but very compressed. If the amplitude for the pulse-trains is set to a constant 1 instead of $\sqrt{A}$ the result sounds very bad and intelligible, also the volume becomes very unstable. Varying the order of the AR model used yields varying results. Lower than 4 gives a very compressed noisy sound, and higher than 20 gives spikes in volume at certain parts. In the case of voice transmission the data that would have to be sent for a specific segment are the coefficients $a_1, a_2, ...a_8$ for the AR(8) model, as well as the amplitude A and pulse length D. In the case of an AR(8) model, there would be $8+1+1 = 10$ data-points instead of 160, which leads to a compression ratio of about 94%, since $10/160 = 0.9375$.

V. Conclusions

In conclusion it was first shown that it's possible to recreate a sine wave fairly well by whistling, however the purity measure is heavily dependent on the allowed frequency range around the dominant frequency. In task 1 the frequency range was chosen as 1% around the dominant frequency, which might be a bit too restrictive, but the measure is ultimately down to subjective judgement. In task 2 AR models were used to recreate vowel sounds with some success, and finally the same techniques were used to encode a full sentence by segmenting the samples and creating a model for each, along with a dynamic way of choosing a suitable pulse train width and amplitude. The final compression rate was around 94%, which is very good considering the sound was legible.

VI. Author rebuttal

VII. MATLAB CODE

A. *lab1_whistle.m*

```
%% Get signal from file and plot data
fs = 8000;
fn = 4000;

[whistle, fSamp] = audioread('data/whistle.wav');
y = whistle(fSamp*4:fSamp*6-1);             % the best 2 seconds
nSamp = length(y);                          % number of samples
t = (0:nSamp - 1) / fs;                     % time vector in seconds

figure(1);clf;
plot(t, y);
xlabel('time in seconds')
ylabel('recorded signal')

%% Fourier transform of the signal
N = 16000;
Y = fft(y);                                 % FFT of the signal
Ysamp = length(Y);                          % Length of FFT vector
ff = (0:N-1)*(fSamp/N);                     % frequency vector

figure;
plot(ff, abs(Y));

% Filter around the dominant frequency
[BBand, ABand] = butter(7, [1100*2/fSamp 1200*2/fSamp], "bandpass");

y_BP = filtfilt(BBand, ABand, y);           % apply BP filter to signal
Y_BP = fft(y_BP,N);                         % FFT of the filtered signal

% Compensate for loss of amplitude for butter
[~, idx] = max(Y);
Y_max = Y(idx);
[~, idxBP] = max(Y_BP);
Y_BP_max = Y_BP(idxBP);
ratio = Y_max / Y_BP_max;
Y_BP = Y_BP * ratio;

% Energy in the time domain
E_tot_t = sum(abs(y).^2)
E_dom_t = sum(abs(ifft(Y_BP)).^2)
purity_t = (1 - E_dom_t/E_tot_t);           % Harmonic distortion

% Energy in the frequency domain
[~, idy_lowmax] = max(abs(Y(1:Ysamp/2)));
[~, idy_highmax] = max(abs(Y(Ysamp/2+1:Ysamp)));

% Number of frequencies around peak to add
b = 15;

band_pass = zeros(1, N);
band_pass(idy_lowmax-b:idy_lowmax+b) = 1;
band_pass(idy_highmax-b:idy_highmax+b) = 1;

Y_BP= band_pass'.*Y;

E_tot_f = 1/N*sum(abs(Y).^2)
E_dom_f = 1/N*sum(abs(Y_BP).^2)
purity_f = (1 - E_dom_f/E_tot_f);           % Harmonic distortion

%% AR (parametric)

% AR model of order 2, least squares,
ar_model = ar(y, 2, 'Ts', 1/fs);
A = ar_model.A;
figure;
zplane(1,A)
present(ar_model)

r = roots(A);
```

```
distance_1 = 1- norm(r(1))
distance_2 = 1- norm(r(2))

% Alrnative method
[R,P,K] = residue(1,A);
dis1 = 1-norm(P(1))
dis2 = 1-norm(P(2))

figure; bode(ar_model); % see highest frequency response in bode plot
legend('AR-model');
```

B. lab1_vowel_a.m

```
close all; clc; clear;
[vowel_a_raw, fs] = audioread('data/aaa.wav');

% ---------- PRE-PROCESSING -------------------------
vowel_a_raw = detrend(vowel_a_raw);
N = size(vowel_a_raw,1); % number of samples
t = (0:N-1)/fs; % time vector in seconds
figure;clf;
plot(t, vowel_a_raw)
title('aaa raw')
xlabel('time in seconds')
ylabel('recorded signal') % axis description is important!

vowel_a = vowel_a_raw(floor(8000*1.7)+1:floor(8000*3.7)); % cut out best 2 seconds

N = size(vowel_a,1); % number of samples
t = (0:N-1)/fs; % time vector in seconds

figure;clf;
plot(t, vowel_a)
axis([])
xlabel('time in seconds')
ylabel('recorded signal') % axis description is important!

% ---------- EST MODEL ORDER FROM GRAPH -------------
% 9*2 peaks = order 18 appropriate for AR
VOWEL_A = fft(vowel_a,N);
ff = (0:(N-1))*(fs/N);
figure; clf;
plot(ff,abs(VOWEL_A));
title("Fourier transform of the a sound")
xlabel("frequency (Hz)")
ylabel("amplitude of Fourier transform")

% ---------- PARTITION DATA -------------------------
% 2/3 for estimation 1/3 for validation
vowel_a_est = vowel_a(1:2*floor(N/3));      % Estimated data
vowel_a_val = vowel_a(2*floor(N/3)+1:N);    % Validation data

% ---------- PRE-defined LOSS function plot ---------
nmax = 20;
[n, W_cv] = arordercv(vowel_a_est, vowel_a_val, nmax);
disp('best order')
disp(n)

%% ---------- HOME-MADE SOLUTION ---------------------
[W, Uaic, Ubic] = arorder(vowel_a, nmax);

figure;
plot(3:nmax, W_cv(3:end), "-o");
title("Cross validation loss function")
xlabel("AR model order")
ylabel("loss")
xticks(3:nmax)

figure;
subplot(2,1,1);
plot(1:nmax,W,'-o', 1:nmax,Uaic, '-.dr', 1:nmax, Ubic, '--xb')
title('Loss functions as a function of AR(n)')
legend('Loss function', 'AIC', 'BIC')
```

```
xlabel('AR model order')
ylabel("loss")
xticks(1:nmax)
subplot(2,1,2);
plot(1:nmax,W,'-o', 1:nmax,Uaic, '-.dr', 1:nmax, Ubic, '--xb')
xlim([3 20])
title('Loss functions as a function of AR(n)')
legend('Loss function', 'AIC', 'BIC')
xlabel('AR model order')
ylabel("loss")
xticks(1:nmax)


%% ---------- SPECTRUM COMPARISON ------------------
data_est = iddata(vowel_a_est, [], 1/fs);
data_val = iddata(vowel_a_val, [], 1/fs);
vowel_a_spect = etfe(data_val, 200);
ar_3 = ar(data_est,3);
ar_18 = ar(data_est,18);
ar_30 = ar(data_est,30);

figure;clf;
bode(ar_3, 'g-', ar_18, 'c-', ar_30, 'b-', vowel_a_spect, 'r');
legend('ar(3)', 'ar(18)', 'ar(30)', 'periodogram')

%% ---------- COMPARE and PREDICT ------------------
figure;
subplot(3,1,1)
YP = predict(ar_3, data_val);
compare(data_val, YP);
subplot(3,1,2)
YP = predict(ar_18, data_val);
compare(data_val, YP);
subplot(3,1,3)
YP = predict(ar_30, data_val);
compare(data_val, YP);

%% ------------------ Simulation of model ------------------
mo18 = ar(vowel_a,18); % 18 because we have 9 clear peaks
e_vec = filter(mo18.a,1,vowel_a); % E = (a_1 + ...) * Y

f_min = 20;
f_max = 100;

r = covf(e_vec,f_max);
[r_max,r_i] = max(r(f_min:end));
f_period = f_min + r_i;

pulsetrain = zeros(1, N);
pulsetrain(1:f_period:end)=sqrt(r_max); % speech synthesis hack

vowel_a_out = filter(1, mo18.a, pulsetrain); % Y = 1/(a_1 + ...) * E
sound(50*vowel_a_out, fs);

figure; clf;
plot(1:length(vowel_a),vowel_a);
hold on;
plot(1:length(vowel_a_out),vowel_a_out);
hold off
```

C. *lab1_vowel_o.m*

```
close all; clc; clear;
[vowel_o_raw, fs] = audioread('data/ooo.wav');

% ---------- PRE-PROCESSING -------------------------
vowel_o_raw = detrend(vowel_o_raw);
N = size(vowel_o_raw,1); % number of samples
t = (0:N-1)/fs; % time vector in seconds
figure;clf;
plot(t, vowel_o_raw)
title('ooo raw')
xlabel('time in seconds')
```

```
ylabel('recorded signal') % axis description is important!

vowel_o = vowel_o_raw(floor(8000*6.5)+1:floor(8000*8.5)); % cut out best 2 seconds

N = size(vowel_o,1); % number of samples
t = (0:N-1)/fs; % time vector in seconds

figure;clf;
plot(t, vowel_o)
axis([])
xlabel('time in seconds')
ylabel('recorded signal') % axis description is important!

% ---------- EST MODEL ORDER FROM GRAPH -------------
% 6*2 peaks = order 12 appropriate for AR
VOWEL_O = fft(vowel_o,N);
ff = (0:(N-1))*(fs/N);
figure; clf;
plot(ff,abs(VOWEL_O));
title("Fourier transform of the a sound")
xlabel("frequency (Hz)")
ylabel("amplitude of Fourier transform")

% ---------- PARTITION DATA -------------------------
% 2/3 for estimation 1/3 for validation
vowel_o_est = vowel_o(1:2*floor(N/3));      % Estimated data
vowel_o_val = vowel_o(2*floor(N/3)+1:N);    % Validation data

% ---------- PRE-defined LOSS function plot ---------
nmax = 20;
[n, W_cv] = arordercv(vowel_o_est, vowel_o_val, nmax);
disp('best order')
disp(n)

%% ---------- HOME-MADE SOLUTION ---------------------
[W, Uaic, Ubic] = arorder(vowel_o, nmax);

figure;
plot(3:nmax, W_cv(3:end), "-o");
title("Cross validation loss function")
xlabel("AR model order")
ylabel("loss")
xticks(3:nmax)

figure;
subplot(2,1,1);
plot(1:nmax,W,'-o', 1:nmax,Uaic, '-.dr', 1:nmax, Ubic, '--xb')
title('Loss functions as a function of AR(n)')
legend('Loss function', 'AIC', 'BIC')
xlabel('AR model order')
ylabel("loss")
xticks(1:nmax)
subplot(2,1,2);
plot(1:nmax,W,'-o', 1:nmax,Uaic, '-.dr', 1:nmax, Ubic, '--xb')
xlim([3 20])
title('Loss functions as a function of AR(n)')
legend('Loss function', 'AIC', 'BIC')
xlabel('AR model order')
ylabel("loss")
xticks(1:nmax)


%% ---------- SPECTRUM COMPARISON ------------------
data_est = iddata(vowel_o_est, [], 1/fs);
data_val = iddata(vowel_o_val, [], 1/fs);
vowel_o_spect = etfe(data_val, 200);
ar_3 = ar(data_est,3);
ar_16 = ar(data_est,16);
ar_30 = ar(data_est,30);

figure;clf;
bode(ar_3, 'g-', ar_16, 'c-', ar_30, 'b-', vowel_o_spect, 'r');
legend('ar(3)', 'ar(16)', 'ar(30)', 'periodogram')
```

```
%% ---------- COMPARE and PREDICT ------------------
figure;
subplot(3,1,1)
YP = predict(ar_3, data_val);
compare(data_val, YP);
subplot(3,1,2)
YP = predict(ar_16, data_val);
compare(data_val, YP);
subplot(3,1,3)
YP = predict(ar_30, data_val);
compare(data_val, YP);

%% ------------------ Simulation of model ------------------
mo18 = ar(vowel_o,3);
e_vec = filter(mo18.a,1,vowel_o); % E = (a_1 + ...) * Y

f_min = 20;
f_max = 100;

r = covf(e_vec,f_max);
[r_max,r_i] = max(r(f_min:end));
f_period = f_min + r_i;

pulsetrain = zeros(1, N);
pulsetrain(1:f_period:end)=sqrt(r_max); % speech synthesis hack

vowel_a_out = filter(1, mo18.a, pulsetrain); % Y = 1/(a_1 + ...) * E
sound(50*vowel_a_out, fs);

figure; clf;
plot(1:length(vowel_o),vowel_o);
hold on;
plot(1:length(vowel_a_out),vowel_a_out);
hold off
```

D. lab1_sentence.m

```
close all; clc; clear;
fn = 4000;
fs = 8000;
% ---- Speech encoding as in GSM ---------------------

[sen, fSamp] = audioread('data/sen2.wav');  % extract data
sen = [sen; zeros(1,16000-length(sen))'];   % zero-pad end with 0's
%sound(sen,fSamp);
N = size(sen,1);                        % number of samples
t = (0:N-1)/fSamp;                      % time vector in seconds

%% Split up sentence with corresponding AR models in 100 chunks
order = 15;
sen_div=zeros(100,160);
for i=1:100
   sen_div(i,:)=sen(160*(i-1)+1:160*(i));
end

sen_ar = zeros(100, order+1);
for i=1:100
   % Get AR model
   temp = ar(sen_div(i,:), order);

   % Check for pole stability (above order 33 is unstable sometimes)
   [R,P,K] = residue(1,temp.a);
   for j=1:length(P)
       dist = norm(P(j));
       if dist > 1
           P(j) = 1/P(j);
       end
   end
   [~, new_a] = residue(R,P,K);

   % Set the ar model in list
   sen_ar(i,:) = new_a;
end
```

```
%% Get the covariance function for the middle chunk i=50
chunk_n = 50;
e_vec = filter(sen_ar(chunk_n,:),1,sen_div(chunk_n,:)); %
r = covf(e_vec',100);
figure(7)
plot(r);title('Covariance Function');xlabel('t');
print(7,'convf.eps','-depsc','-loose');

%%
sen_div_hat = zeros(100,160);
f_min = 20;
f_max = 100;

for i=1:100
    e_vec = filter(sen_ar(i,:),1,sen_div(i,:));
    r = covf(e_vec', f_max);              % Covariance matrix
    [r_max,r_index] = max(r(f_min:end));  % skip first f_min
    r_index = r_index + f_min;            % adjust for offset

    pulsetrain = zeros(1,160);               % declare vector
    pulsetrain(1:r_index:end)=sqrt(r_max);   % speech synthesis hack
    sen_div_hat(i,:) = filter(1,sen_ar(i,:),pulsetrain);
end

% Detrend the estimated signals
sen_div_hat_detrend = zeros(1, N);
for i=1:100
    sen_div_hat_detrend(160*(i-1)+1:160*(i)) = detrend(sen_div_hat(i,:));
end

%% Figure zone

tt = (0:(N-1))/fs;

SEN = fft(sen);
SEN_DIV_HAT_DETREND = fft(sen_div_hat_detrend);

figure(4);
subplot(2,1,1)
plot(tt, sen_div_hat_detrend);
title('Parametric Sentence');
xlabel('time (s)');
ylabel('Amplitude')

subplot(2,1,2)
plot(tt, sen);
title('Non-Parametric Sentence');
xlabel('time (s)');
ylabel('Amplitude')
print(4,'sent_t.eps','-depsc','-loose');

ff = (0:N-1)*(fs/N);

figure(5);
subplot(2,1,1);
plot(ff, abs(SEN_DIV_HAT_DETREND));
title('Parametric Sentence');
xlabel('frequency (Hz)');
ylabel('Amplitude')
subplot(2,1,2)
plot(ff, abs(SEN));
title('Non Parametric Sentence');
xlabel('frequency (Hz)');
ylabel('Amplitude')
print(5,'sen_f.eps','-depsc','-loose');
%sound(y,8000)
%% Play the parametric recording

sound(sen_div_hat_detrend,fs);

%%

function [ new_a ] = pole_stable(a)
```

```
% Make sure are the poles are within the
% unit circle

end
```