TSRT78 Fundamental Signal Processing

Peter Forsberg
Liu ID: *foppa021*
19730720-XXXX

Lionel Messi
Liu ID: *goat987*
19300519-YYYY

***Abstract*—The purpose of the lab was to investigate signal modeling in practice by modeling different signals. This was divided into three tasks were a whistle was modeled, two vowels and a complete sentence. The major findings of the lab was that an AR(2)-model is good for modeling periodic signals, and that when modeling and simulating a signal, better results are achieved if the signal is divided into segments that each are modelled on their own and then composed into one simulation of the signal.**

***Index Terms*—digital signal processing, TSRT78**

I. Introduction

In this report, the results obtained for the lab tasks are to be reviewed and explained. First, the theory behind the methods is presented, and then the solutions to each of the three task are gone through.

II. Theory

A short review of the theory behind the lab methods follows.

A. Signal energy

The signal energy E_x in a discrete signal x composed by N samples is calculated in the time and frequency domain the following way:

$$E_x = \sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N}\sum_{n=0}^{N-1} |X[n]|^2, \tag{1}$$

where $X[n]$ is the DFT of signal x evaluated at step n [1].

B. Power spectrum

The estimated power spectrum $\hat{\hat{\phi}}_N[n]$ of the signal is given by [1]:

$$\hat{\hat{\phi}}_N[n] = \frac{T_s}{N}|X[n]|^2. \tag{2}$$

C. Discrete Fourier Transform (DFT)

Let the signal $x[n] = x(nT_s)$ where T_s is the sampling time of the continuous signal $x(t)$. Then, the discrete-time Fourier transform (DTFT) of the $x[n]$ is

$$X_{T_s}(e^{i2\pi fT_s}) = T_s \sum_{k=-\infty}^{\infty} x[k]e^{-i2\pi fkT_s}, \tag{3}$$

where $X_{T_s}(e^{i2\pi fT_s})$ is a periodic function of f with period $1/T_s = f_s$ (as $X_{T_s}(e^{i2\pi fT_s}) = X_{T_s}(e^{i2\pi(f+f_s)T_s})$.

Notice, though, that it is far beyond ideal that there would be infinitely many samples of signal x available. Therefore, the discrete Fourier transform (DFT) is used instead. It can be seen as a sampled and truncated version of the DTFT, and it is defined as:

$$X[n] = T_s \sum_{k=0}^{N-1} x[k]e^{-i2\pi kn/N}, \tag{4}$$

where $X[n]$ is periodic with period N since $X[n] = X[n+N]$ [1].

D. Auto regressive model (AR-model)

The parametric method used to model signals in this lab is the AR-model. Let the signal $y(t)$ be modeled by an AR-model of order n. Then: $y(t) = T(q)e(t) = e(t) - a_1 y(t-1) - ... - a_n y(t-n)$, where $e(t)$ is white noise input signal and a_i the model parameters. The linear filter $T(q)$'s z-transform can then be described by [1]:

$$T(z) = \frac{z^n}{z^n + a_1 z^{n-1} + a_2 z^{n-2} + ... + a_n}. \tag{5}$$

E. Estimation of parameters in AR-model

Upon deciding what the order n of the model should be, one can estimate the model parameters a_i for $i = 1, 2, ..., n$ the following way:

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{N}\sum_{k=1}^{N}(y[k] - \hat{y}[k;\theta])^2 \tag{6}$$

where $\hat{\theta} = [a_1 a_2 ... a_n]^\top$ is the vector containing the parameters that minimize the total sum of errors between the predictions $\hat{y}[k;\hat{\theta}]$ and the real data $y[k]$ [1].

F. Harmonic distortion

Harmonic distortion is a way to measure the purity of a periodic signal, and it's given by:

$$1 - \frac{E_{dom.freq}}{E_{tot}}, \tag{7}$$

where E_{dom_freq} is the energy concentrated around the signal's dominating frequency (i.e. the one with highest frequency top), and E_{tot} is the frequency of the signal over the whole frequency spectrum [2].

G. Mirroring poles

As described in section 4.2.3 page 134 in [1], if n_i is a zero (pole) to $\tilde{\phi}_{yy}(z)$, then $\frac{1}{n_i}$ is also a zero (pole). This is due to the fact that the covariance function is symmetric, i.e. $R_{yy}(k) = R_{yy}(-k)$, which implies that $\tilde{\phi}_{yy}(z) = \tilde{\phi}_{yy}(1/z)$ [1].

III. Task 1 Whistle

For this assignment, a whistle was recorded with sample frequency $f_s = 8000$ [Hz] and loaded into MATLAB. The tasks were in summary to run frequency calculations using non-parametric and parametric methods. The non-parametric method consisted of transformation to frequency spectrum and filtering, and the parametric method consisted of modeling the signal with an AR-model and controlling the modeled signal's properties.

A. Method

To achieve the objective of the first task, a signal whistle was recorded with a sample frequency of 8000 [Hz] and saved as a wav-file. The recording was read into MATLAB and plotted in the time domain. The plot was analysed and the two best seconds of the whistle where chosen to be the working signal. The energy of this part of the signal was then calculated from the data in the time domain according to equation (1). The signal was also plotted in the frequency domain where the dominating frequency component was determined by looking at the frequency content for different frequencies. This dominating frequency was then used as the central frequency for a bandpass filter which was applied to the signal, with cut-off frequencies close to the central frequency. The energy of the resulting signal was then calculated and recorded in the same way as previously mentioned.

Next, the energy of the signal was calculated in the frequency domain using equation (1). The energy of the dominating frequency component was also calculated in the frequency domain by choosing a few points close to the dominating frequency (positive and negative) and inserting them into the same equation.

Using the calculated energies of the signal and the dominating frequency component, the harmonic distortion was calculated twice, first using the energies calculated in the time domain and then with the energies calculated in the frequency domain. These calculations where made using equation (7). The two values were analysed and observations regarding the purity of the whistle were made.

The signal was then modelled using an AR(2)-model. The poles of the transfer function for the model were calculated, and their distance to the unit circle was computed and analysed.

Finally, the AR(2)-model's bode plot was found using MATLAB and the dominating frequency was searched for by looking at the frequency content for different frequencies.

B. Results

The energy of the signal calculated in the time domain was found to be 85.1374. The dominating frequency, f_d, was read to be approximately 1250 [Hz] by looking at figure 1.

By filtering using a bandpass filter with cut-off frequencies 1240 [Hz] and 1260 [Hz], the energy of the dominating frequency component was determined to be 84.8809. The energy of the signal calculated in the frequency domain was determined to be 85.1374, and the energy of the dominating frequency component, also calculated in the frequency domain, was 79.3470.

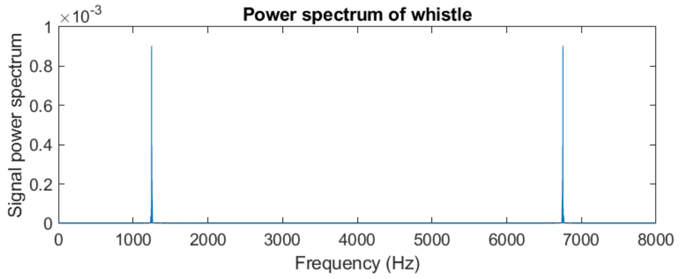


Figure 1. Power spectrum of whistle

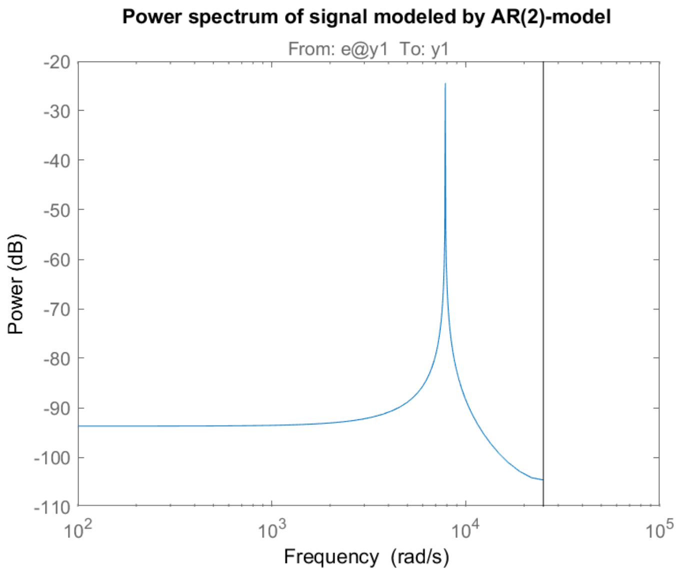


Figure 2. Power spectrum of whistle modeled by AR(2)-model

Using the energy values, the harmonic distortion was calculated using equation (7) to be 0.0030 and 0.0680 from the calculations made in the time and frequency domain respectively. The difference between the two harmonic distortions is 0.065 which is small, and so are the two harmonic distortions. Therefore, the whistle is relatively pure.

The purity of the AR(2)-model was measured as the distance of the transfer function poles to the unit circle. This is because the poles of a pure sine are located on the unit circle. The distance of the poles was calculated to be 0.00020 which is close to the unit circle, meaning that the whistle was pure based on an AR(2)-model. The AR(2)-model was deemed to be suitable for modeling a sine because it means that the transfer function will have two poles, just like the transform of a sine.

The power spectrum of the whistle was calculated using both a non-parametric method and a parametric method. In the non -parametric method, the DFT of the signal was taken and the power spectrum was estimated according to equation (2). The parametric method used was based on the AR(2)-model.

C. Conclusions

The energies for the dominating frequency that were calculated in the time domain and frequency domain were quite close to each other but not exactly the same. This was expected seeing that the techniques used to calculate the two never guaranteed in any way that they would be the exactly the same. Instead they were only used to get an estimate of the energy for the dominating frequency component. The total energies were the same on the other hand, which they were supposed to be according to equation (1).

Judging by the purity measures that were calculated, one can conclude the whistle was pure. It can also be said that the AR(2)-model was good for modeling a sine since it gave poles that were close to the unit circle.

IV. Task 2 Vowels

For the second assigment of the lab two signals of vowels were recorded and investigated. The goal of the task was to build an AR-model that could be used to simulate the original signals.

A. Method

To achieve the goal of the assignment, two signals were recorded with a sample frequency of 8000 [Hz]. The signals that were recorded were of a person saying the vowels "a" and "o" for about 5 seconds, one vowel for each signal. Then the two signals were plotted, and for each of them, two consistent seconds without many disturbances were chosen as the working signals. These working signals were then plotted in the frequency domain using the DFT according to equation (4). The number of peaks for each signal were counted to get an estimate of what an appropriate model order of an AR-model could be. The model orders found were then used to create AR-models for the signals using MATLAB from $\frac{2}{3}$ of the data. The other $\frac{1}{3}$ of the data was then used to validate the models using two different validation methods.

The first validation method consisted of calculating the spectrum for both the AR-models and the signals and comparing the two spectrums. The calculations were made using equation (2) and built in MATLAB functionality. The comparison was made using generated plots from the MATLAB calculations. The second validation method that was used was a residual whiteness test where the residuals were plotted and analysed to see if it resembled white noise.

Lastly, the model was simulated by using a MATLAB generated pulse train with the same period as the signal that was filtered with the AR-model. This signal period was found by looking at the maximum of the covariance function of the residuals $R_e(t)$ for $t > 19$. The period was set to the index of the maximum of $R_e(t)$ plus 19.

B. Results

The frequency spectrum of the working signals can be seen in figure 3 and the number of peaks for the vowel "a" was found to be 28, while the number of peaks for "o" was found to be 12. These numbers were used as initial estimates of the model orders needed to model the two signals. The estimates were deemed to be reasonably good but slightly higher model orders gave better results during the validation. In figure 4 and 5, the result of the validation can be seen for model orders 28 and 25 for "a" and "o" respectively.

When the AR-models were simulated and played back, one could hear that the two signals were an "a" and an "o" but how the simulations sounded changed compared to the original signals. The result sounded a bit more monotonic, but other than that the result was good.

C. Conclusions

The conclusions that can be drawn from the task is that it is possible to model periodic signals such as vowels using AR-models and simulate them with a good result. For the signals recorded, good model orders seem to be 28 and 25 for "a" and "o" respectively.

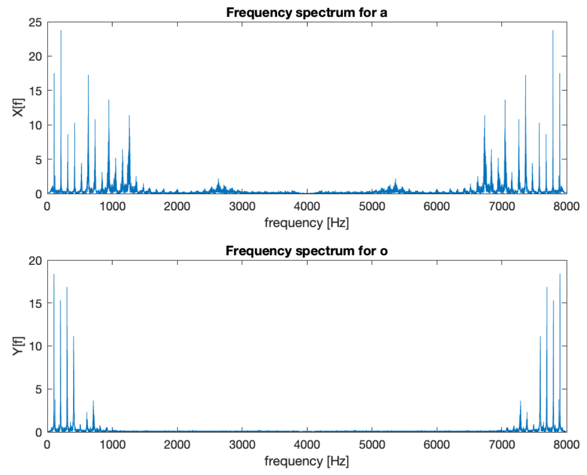


Figure 3. Frequency spectrum of the working signals.

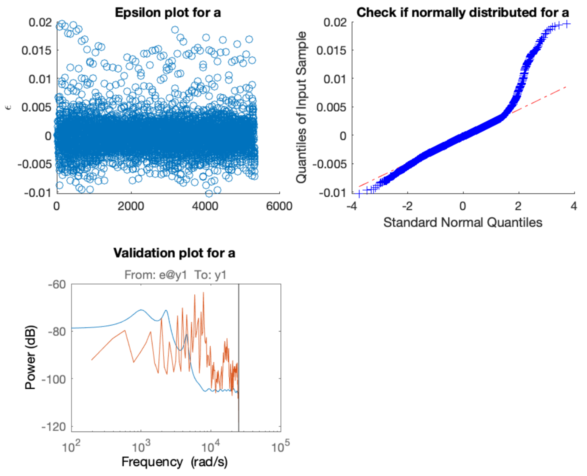


Figure 4. Validation plot for "a".

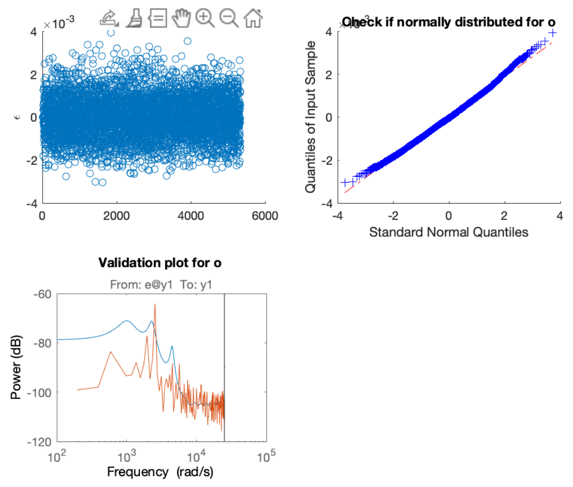


Figure 5. Validation plot for ”o”.

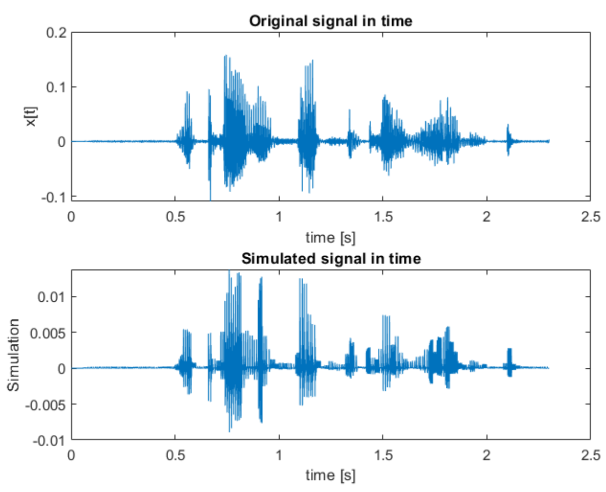


Figure 6. Original and simulated signal plotted in time.

V. Task 3 GSM

A. Method

In the last assignment of this lab, the objective was to implement a version of the speech encoding used in GSM (**G**lobal **S**ystem for **M**obile Communications). In order to achieve that, first a two-second long sentence was recorded, *The clever fox surprised the rabbit*, using the sampling frequency of 8000 [Hz]. The number of samples, N, became 18400, which was divided equally into 115 segments of 160 samples each. Then, each of these 160 segments were detrended and modeled using an AR-model of order 8, as required in the assignment. When deciding the coefficients of the AR-model, it was made sure that unstable poles (outside the unit circle) were stabilized by mirroring them in the unit circle as described in II-G.

After the linear filter for each segment was calculated, an input signal in the form of a pulse train was formed. The amplitude of the individual pulses were set to be equal to $\sqrt{A}$, where A was the maximum of the covariance function of the residuals, while the pulse period D was set to be equal to the index that gave the maximum of the covariance function. Finally, the input was passed through the linear filters of each segment so that the output sound was generated.

Then the amplitude of the pulse train was set to 1 instead of $\sqrt{A}$, which was simply done by changing the amplitude variable in MATLAB, and the result was analysed. For the third question, the model orders were varied by changing a variable in MATLAB. Finally, an analysis of what data a receiver would need to simulate a signal was done.

B. Results

The sentence was reconstructed using the generated AR-model. See in figure 6 how alike the real and the simulated were in time. The simulated sentence sounded just like the original; one was even able to identify the person that was speaking. When the pulse amplitude was set to one, the simulated signal became substantially more noisy.

Regarding the chosen model order, it was decided that it would be 16 since it gave an almost non-noisy result with a relatively low amount of model parameters. With the recommended model order 8, the original sentence was recognizable, but not as clearly as with model order 16.

Finally, the data that needed to be sent to the receiver (instead of the value of each sample) so that the original sound could be simulated was: the 16 model parameters, the pulse amplitude A and the pulse period D. This means that the data sent is $\frac{18}{160} = 11.25\%$ of what would have to be sent if the original signal was sent.

C. Conclusions

By simulating the AR-model one can clearly hear what the person said without having access to the entire signal. Therefore it is clear that the compression is substantial when using GSM modeling, while still keeping good quality of the sound.

VI. Conclusions

A short conclusion that can be drawn by comparing the results achieved in IV and V is that dividing the signal in smaller segments and modeling each one of the segments with its AR-model achieved substantially better simulations.

References

[1] Gustafsson, F. and Ljung, L. and Millner, M., “Signal Processing,” Studentlitteratur: Lund Sweden, 2011.
[2] https://www.control.isy.liu.se/student/tsrt78/lab1FundSigProc.pdf

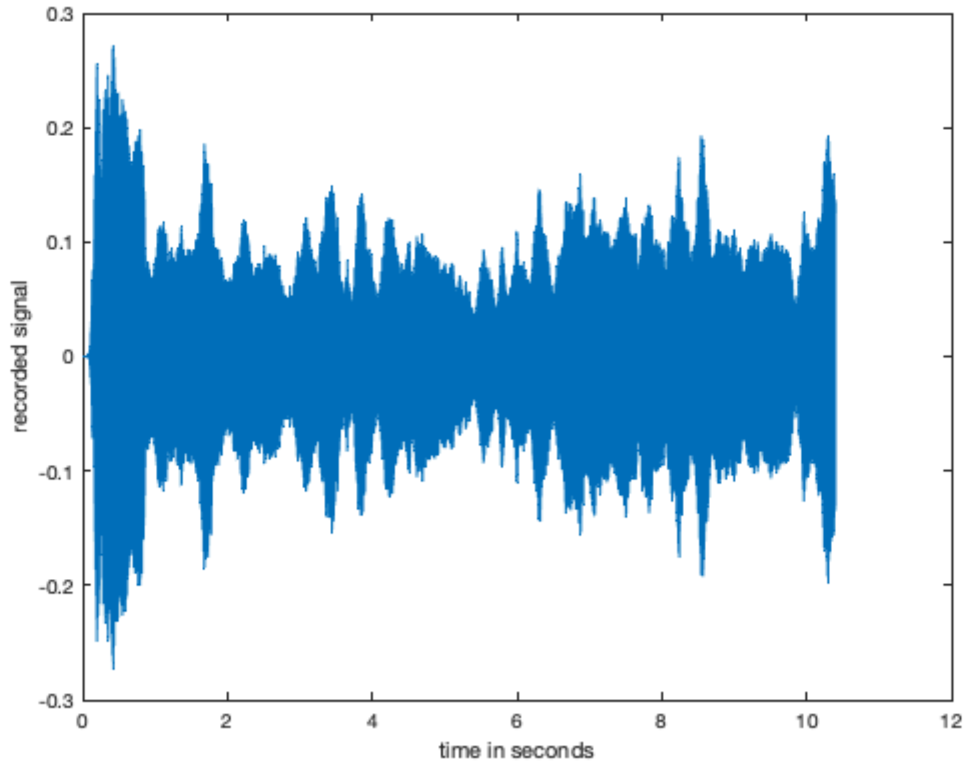
VII. Appendix

```
% 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(1);

figure;clf();
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;clf();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;
```

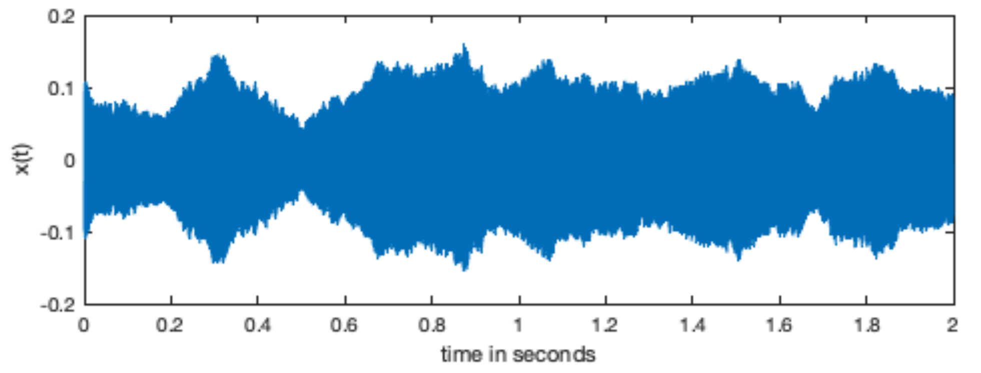
```
subplot(2,1,2);
plot(frequencyVector , (abs(Yf).^2)/nSamp)
xlabel('frequency in Hz');
ylabel('signal spectrum');

% 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)/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
```

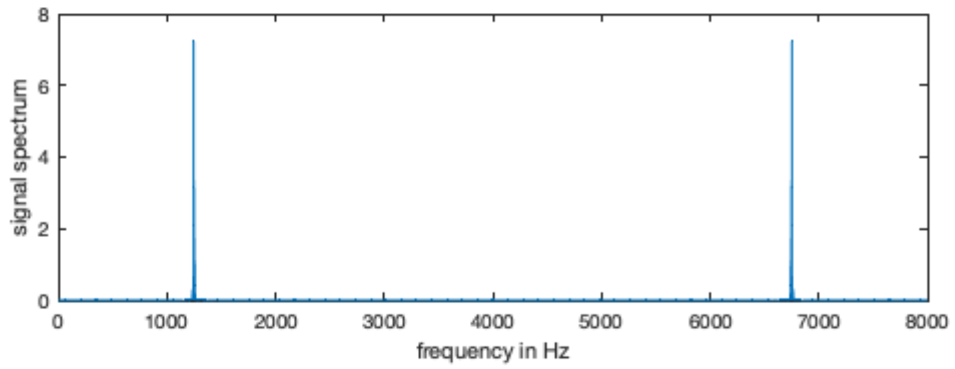
```
% ------------------- QUESTION 4 -------------------
% Estimate the purity measure based on an AR(2) and motivate why this
 model
% is suitable. How can this measure be compared to the harmonic
 distortion?

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

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
```

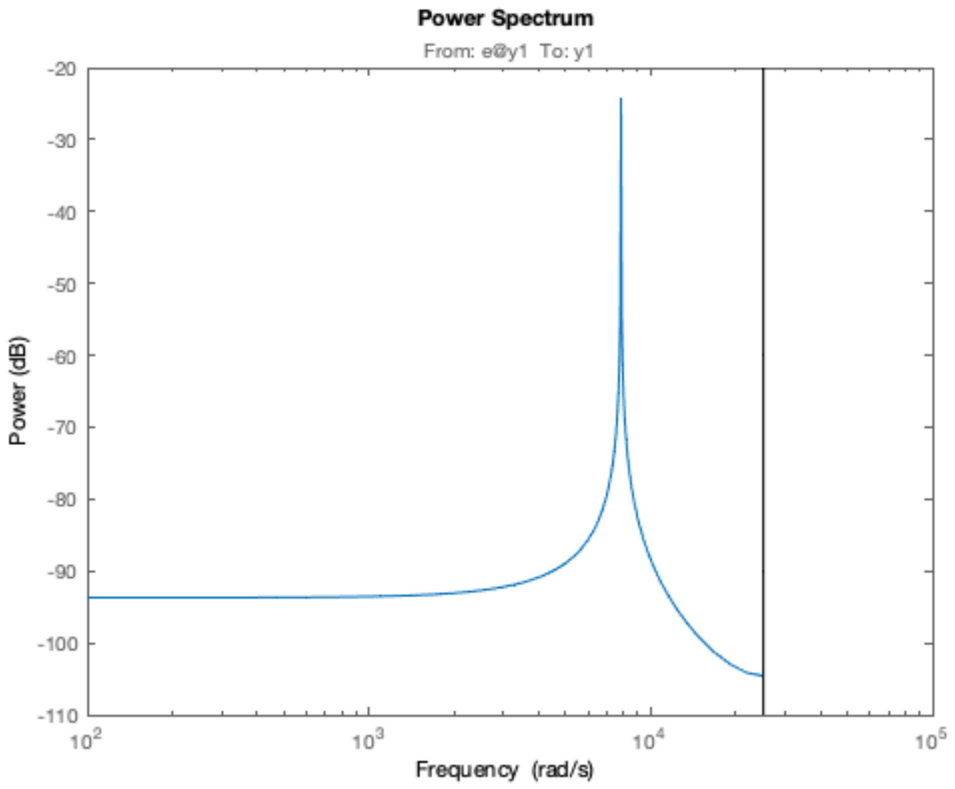
```
Warning: A bode plot is not well defined for a time series model. The
 plot will
show the output spectrum of the model. Consider using the "idlti/
spectrum"
command instead.
```



0.3
0.2
0.1
0
-0.1
-0.2
-0.3
recorded signal
0
2
4
6
8
10
12
time in seconds


x(t)
time in seconds


signal spectrum
frequency in Hz


Power Spectrum
From: e@y1 To: y1
Power (dB)
Frequency (rad/s)

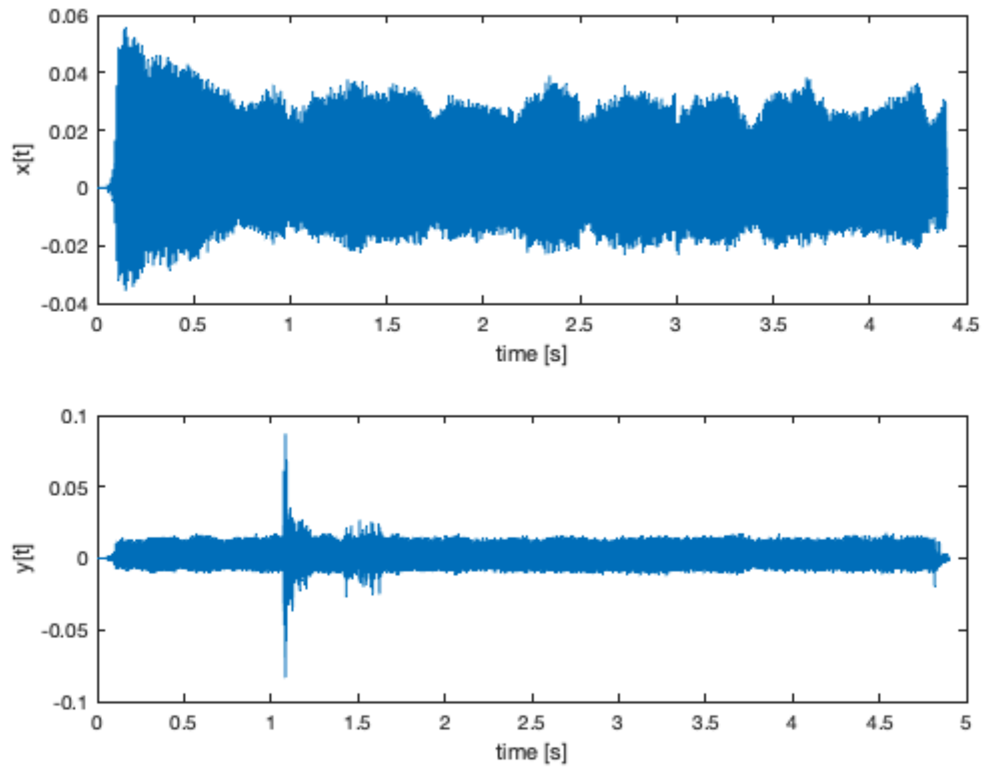
Published with MATLAB® R2020a

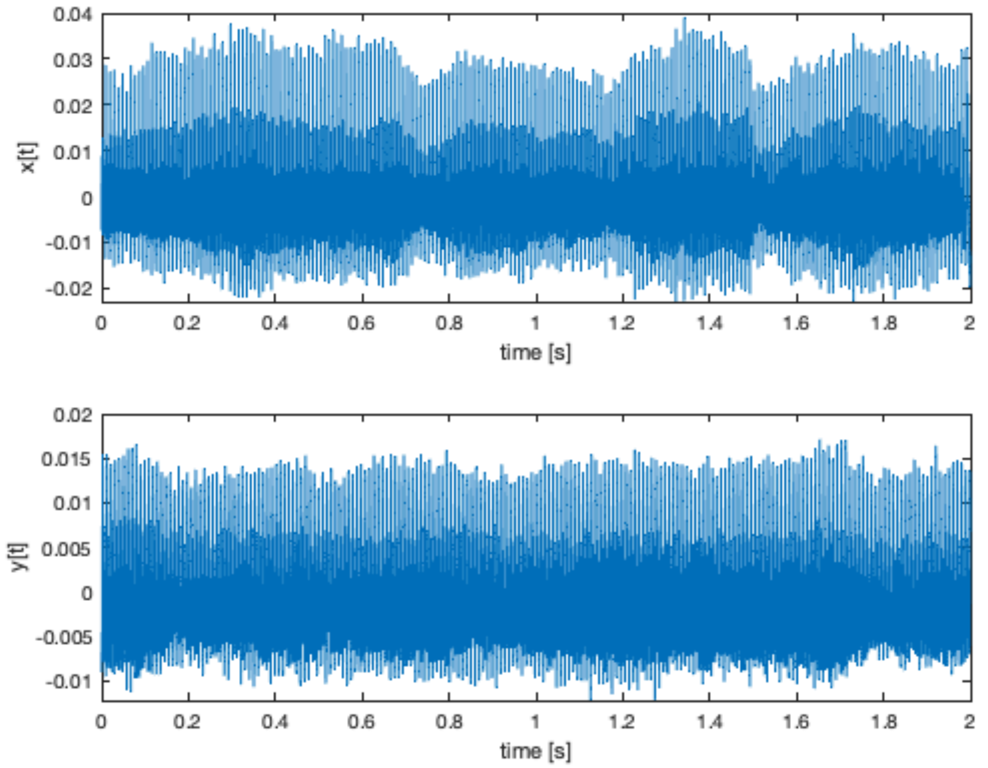
```
% TSRT78, Lab 1: Fundamental Signal Processing
% Matheus Bernat (matvi959) & Caspian Süsskind (cassu286)

% ===================== 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')
```

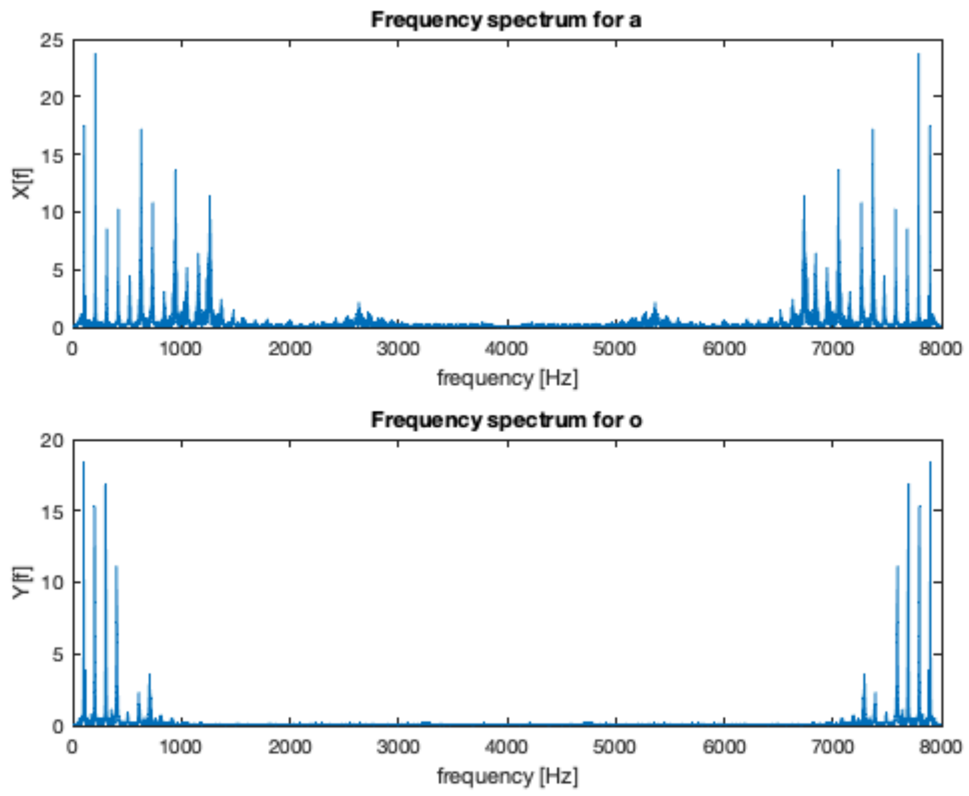
```
% ---------------- Create AR models
modelOrder_x = 28;
modelOrder_y = 25;
estIdx_x = floor(2*Nx/3);
estIdx_y = floor(2*Ny/3);
arMod_x = ar(x(1:estIdx_x), modelOrder_x, 'Ts', Ts);
arMod_y = ar(y(1:estIdx_y), modelOrder_y, 'Ts', Ts);

% ------------------- Simulation of model -------------------
b = 1;
ax = arMod_x.A; % coefficients of the AR-model
ay = arMod_y.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]);


% ------------------- Validation of model -------------------
eps = pe(arMod_x, x(estIdx_x+1:end));
figure;
subplot(2,2,1);
scatter(1:length(eps), eps); x('\epsilon index'); ylabel('\epsilon');
 title('Epsilon plot for a');
subplot(2,2,2);
qqplot(eps) % See if the error is normally distributed
title('Check if normally distributed for a')
subplot(2,2,3);
bode(arMod_y); hold on; bode(etfe(iddata(x(estIdx_x+1:end),[],Ts)));
title('Validation plot for a');

% Get residuals. pe gives us the prediction errors
eps = pe(arMod_y, y(estIdx_y+1:end));
figure;
subplot(2,2,1);
```

```
scatter(1:length(eps), eps); x('\epsilon index'); ylabel('\epsilon');
 title('Epsilon plot for o');
subplot(2,2,2);
qqplot(eps) % See if the error is normally distributed
title('Check if normally distributed for o')
subplot(2,2,3);
bode(arMod_y); hold on; bode(etfe(iddata(y(estIdx_y+1:end),[],Ts)));
title('Validation plot for o');
```

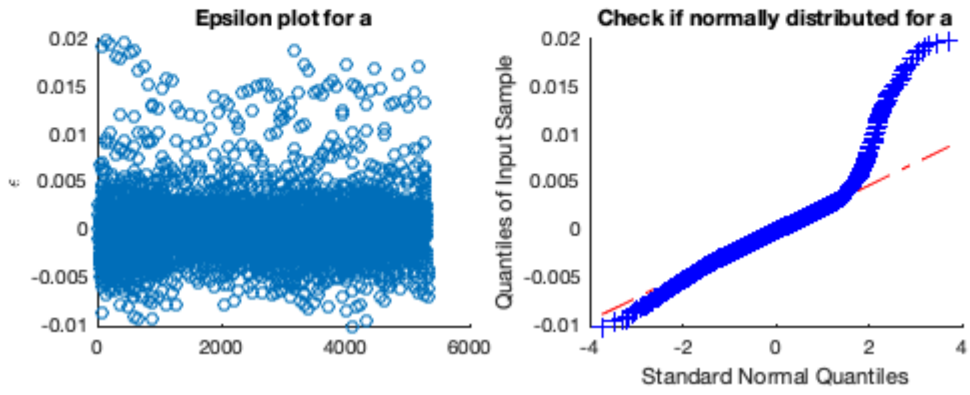
```
Warning: A bode plot is not well defined for a time series model. The
 plot will
show the output spectrum of the model. Consider using the "idlti/
spectrum"
command instead.
Warning: A bode plot is not well defined for a time series model. The
 plot will
show the output spectrum of the model. Consider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


x[t]
time [s]
y[t]
time [s]

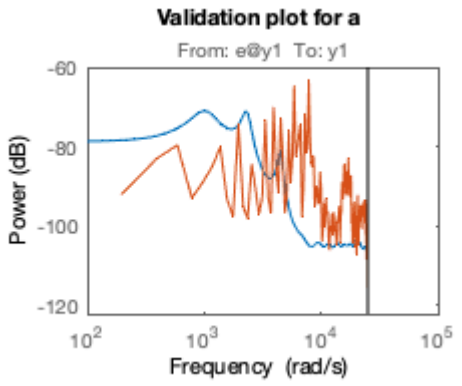

x[t]
time [s]
y[t]
time [s]



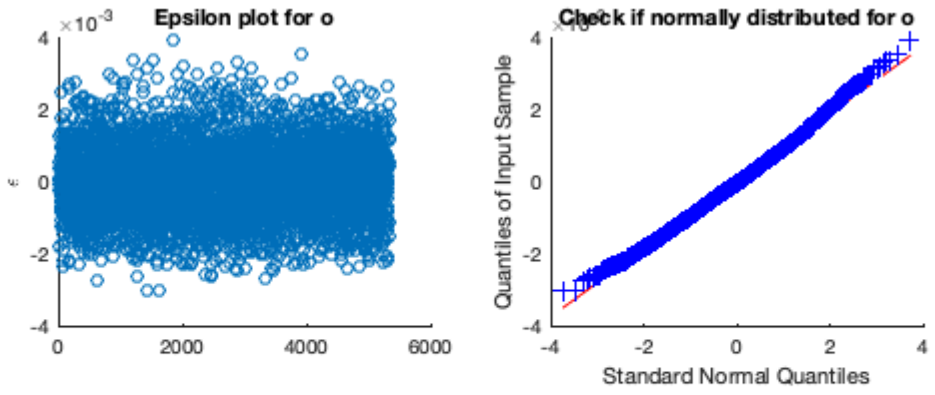
Frequency spectrum for a
X[f]
frequency [Hz]
Frequency spectrum for o
Y[f]
frequency [Hz]

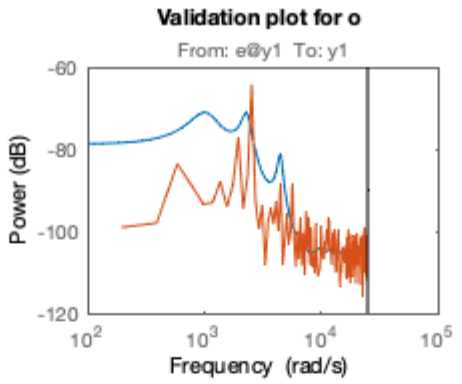


Epsilon plot for a
ε
Check if normally distributed for a
Quantiles of Input Sample
Standard Normal Quantiles



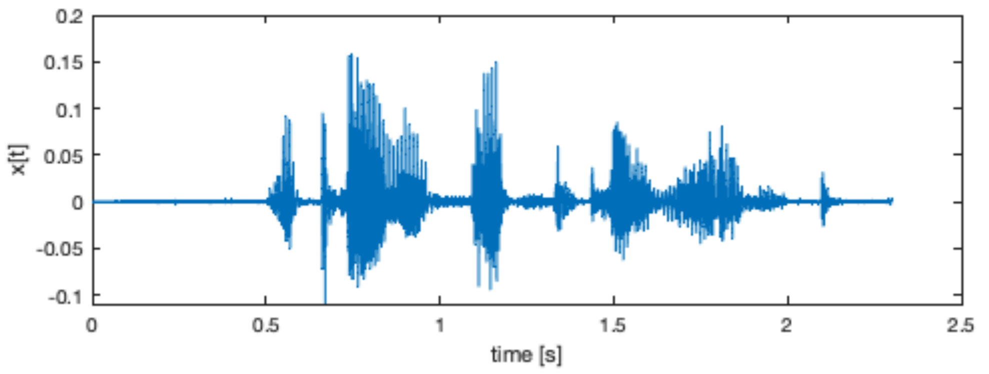
Validation plot for a
From: e@y1 To: y1
Power (dB)
Frequency (rad/s)

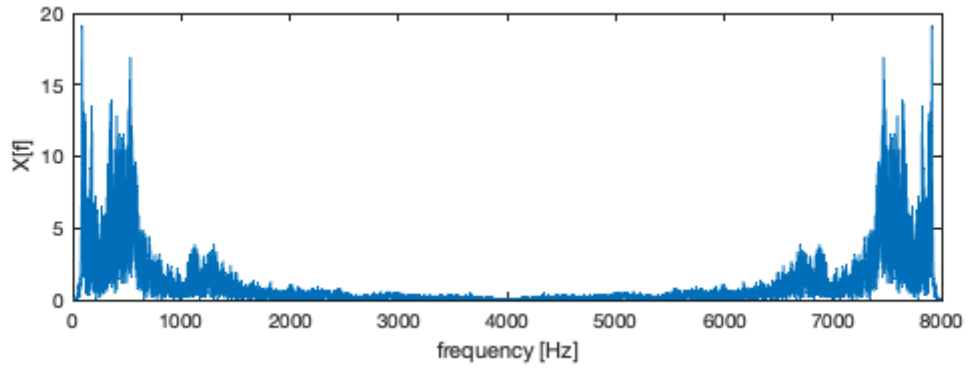


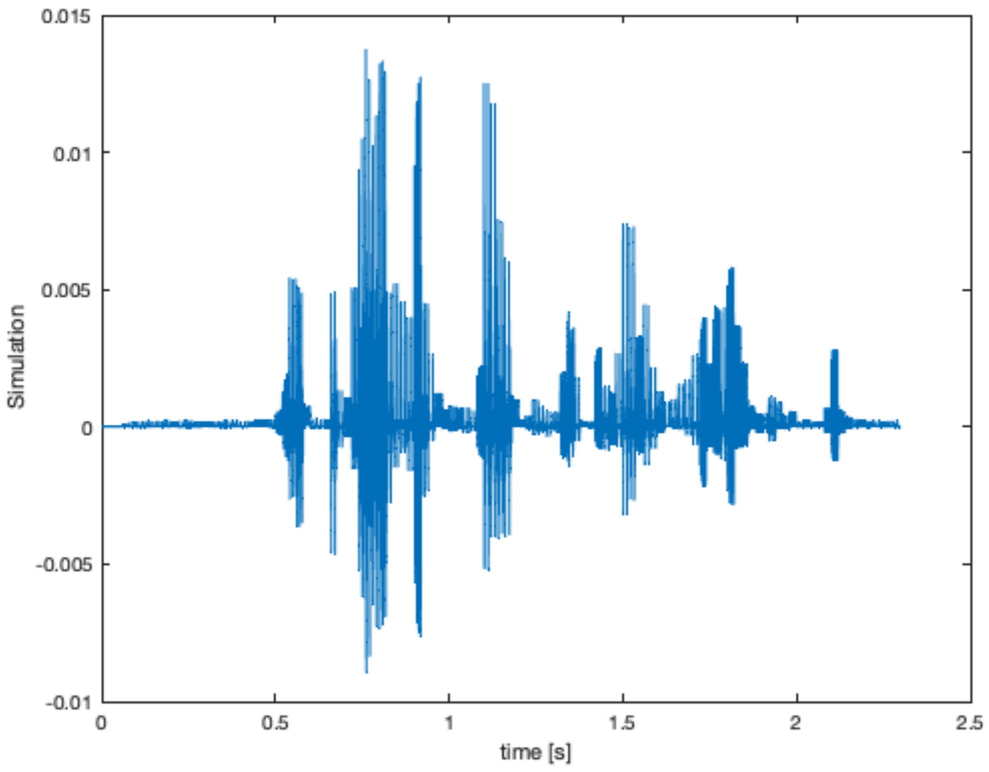


Published with MATLAB® R2020a

```
% 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