Assignment 1: Speech Production and Models
EQ2320 Speech Signal Processing

2016-01-23

Instructions for the deliverables

Perform all (or as many as you can) of the tasks in this project assignment. Summarize your results in a presentation style document. You are going to present your work to the course responsible or one of the teaching assistants (TA) and it pays off to have a clear, well structured and brief document. Explicitly state, on the slides, the problem numbers. Include an algorithmic solution and/or numerical result and/or a graph illustrating your results.

Your Matlab code should be functioning and easy to read. You have to send us your Matlab source code at least a day before the scheduled time for your presentation. Use a naming convention for your files that is easy to understand and associate with a specific problem number. Be prepared to demo your own code on a personal or our computer. Archive your files before sending them to us!

You can find all necessary materials for successfully accomplishing the assignment on the course homepage: https://www.kth.se/social/course/EQ2320/.

Grading system

Each correctly solved task brings you a certain amount of points. The grade you get depends on the amount of points you accumulate and your ability to motivate your solutions during the presentation. A passing grade corresponds to having 30 % and more of the total amount of points, while an excellent grade requires 85 % and more. No points will be granted for a correct response if you cannot motivate your answer.

Keep in mind, when working in a group of two, that both people are expected to be able to make a complete presentation and answer questions related to all problems. You may be graded differently depending on your responses. The grade you get depends on the amount of correctly solved tasks and your ability to motivate the solutions during the presentation.

1 Introduction

The first assignment is perhaps the most important for two reasons: Firstly, the knowledge obtained in this exercise is the basis for the whole course, and the program code that you produce will be useful in coming exercises. Secondly, this exercise is relatively easy, and this is a chance for all who are not familiar with the MATLAB environment to catch up. The coming computer exercises will require more MATLAB knowledge (but will also be proportionally more exciting).

On the course homepage you can find the speech data which you will analyze in this computer assignment. Download and unzip the file `assignment1.zip` from the course webpage. Store it in a directory which is readable from the MATLAB prompt. To load the data, type `load assignment1.mat` in MATLAB. Several MATLAB variables are then loaded into memory. All the signals are sampled at 8 kHz. The signals are stored in double floating point format, but they originate from 16 bit AD conversion.

In this first assignment a lot of code is provided to you. Note that this is for your convenience only; if you prefer to write your own code from scratch we encourage you to do so, but it may be time consuming.

2 Bandwidth of Speech

In our first experiment we study how low-pass filtering affects the speech signal. Load the files `male44.waw` and `female44.wav` into MATLAB using the function `wavread`. The speech is sampled at 44.1 kHz. In the following, use the m-file (MATLAB function) `lowpass.m`.

TASKS/QUESTIONS

1. Do not listen to the full-band signals! Start with a very low cut-off frequency (say 500 Hz), and listen to the output. Increase the cut-off in steps of 500 Hz. At what cut-off frequency do you start hearing what is said? This is a test of intelligibility, i.e., how well we can understand what is spoken. (0.5 pts)

2. At what cut-off frequency do you start to hear a degradation in overall speech quality? (0.5 pts)

3. Why, do you think (you are allowed to speculate!), is a sampling frequency of 8 kHz used in plain old telephony services (POTS)? (1 pts)

HINTS/REMARKS

1. On most platforms[1], MATLAB's own playback functions `sound` and `soundsc` usually work fine. You should use the `soundsc` function to make sure the signal is properly scaled and does not cause overflow in the output sound device. See also hint about SPCLAB in the next section.

2. Two important commands in MATLAB are `help` and `lookfor`. With "help" you get detailed instructions about a certain MATLAB function. It requires that you know the name of the function. If you don't know the name of a function, sometimes "lookfor" finds it for you.

3 Voiced and Unvoiced Speech Sounds

Here, we study the variable "male short" which contains a short utterance of speech. From a time plot, find voiced and unvoiced regions. Study the harmonic structure and envelope of the speech spectra corresponding to voiced and unvoiced regions.

TASKS/QUESTIONS

1. Create a time plot with voiced and unvoiced regions marked with a pen. (0.5 pts)

2. In the time plot, mark the regions where the pitch is the highest and the lowest. What are the pitch frequencies in those regions? (0.5 pts)

3. Plot the DFT based spectrum for a voiced frame using,

```
% x is a vector of speech, N is the frame length (must be even),
% S is the first sample of the frame to be analyzed
xf = x(S:S+N-1).*hanning(N);
X = fft(xf);
figure(1); clf;
plot(10*log10(abs(X(1:N/2+1)).^2));
```

   Make sure you understand every line of the above piece of code. (0.5 pts)

[1] Under Linux on PC platforms the MATLAB-internal `soundsc` command might not work. In this case you can use the program `mysound` that is provided to you with the other files for this assignment.

4. What is the mathematical expression of the MATLAB variable X(3), i.e., the third element in the FFT vector? What analog frequency (between 0-8000 Hz) does X(3) correspond to? (1 pts)

5. What is the fundamental frequency (pitch) in your particular case? (0.5 pts)

6. What frame length is appropriate? What compromise do you have to make when choosing frame length? (1 pts)

7. Replace the `hanning` window function with a rectangular window and compare the result. (1 pts)

8. Extend the code above with the following lines to plot also the LP envelope of the short-time speech spectrum:

```
% M is the prediction order
c = xcorr(xf, xf, M);
[a, e]= levinson(c(M+1:2*M+1)); % a is a vector always starting
with a 1.
a = a(:); % Make a a column vector
% The commented code below is essentially equivalent to the two
lines above
% r = c(M+1:2*M+1);
% R = toeplitz(r(1:M),r(1:M)); % A Toeplitz matrix with first
row r(1:M)
% a = - R / r(2:M+1);
% a = [1; a]; % Add the leading 1
% e = sum(a*r);
h = zeros(N,1);
for k=0:N-1
h(k+1) = 1 / (a'*exp(-i*2*pi*k/N*(0:M)') );
end
% h = freqz(1, a, N, 'whole'); % Equivalent to the above for-loop!
hold on
plot(10*log10(e*abs(h(1:N/2+1)).^2), 'r');
```

   Make sure you understand every line in the code above or use your own code which you do understand. (0.5 pts)

9. What is the mathematical expression of the MATLAB variable c(3)? Why do we multiply `abs(h(1:N/2+1)).^2` by `e` before plotting? (1 pts)

10. What prediction order do you recommend? (1 pts)

11. Repeat the spectrum plotting for an unvoiced frame. (0.5 pts)

HINTS/REMARKS

1. You can use the zoom in the figure window in MATLAB to study details of the signals.

2. A much more convenient tool is SPCLAB that is free to use and can be accessed from the course web page. SPCLAB provides a lot of useful functionality for the study of speech signals and we recommend its use. Especially useful here is the tool to zoom in on a region corresponding to a certain speech sound and then saving that region to a vector in the MATLAB workspace for further processing.

3. The statement "Plot the spectrum" is vague. There are many alternatives, e.g. plotting the real, and imaginary part of the FT separately. Here we focus on the squared magnitude of the DFT coefficients, and hence disregard the phase. Also it is common to plot the spectrum in a logarithmic scale, i.e., in decibels (dB), since this better reflects what we actually hear.

4. In all MATLAB functions related to linear prediction and filtering, the polynomial A(z) is defined $A(z) = 1 + \sum a_k z^{-k}$. This means that if you use for example levinson to calculate the polynomial coefficients they will have opposite sign compared to what we get based on our definition in the course book.

4 Formants

In the speech variable "male short", try to identify the vowels by comparing the formants from an LP analysis to the formant frequencies of template vowels in Table 1. Note that the table contains only the three first formants; you can often see up to four formants.

TASKS/QUESTIONS

1. In the time plot from the previous section, add the formant frequencies and the vowels you come up with. (1 pts)

2. How are pitch and formant frequencies related? (1.5 pts)

vowel	F1	F2	F3	example
iy	270	2290	3010	beet
ih	390	1990	2550	bit
eh	530	1840	2480	bet
ae	660	1720	2410	bat
ah	520	1190	2390	but
aa	730	1090	2240	hot
ao	570	840	2410	bought
uh	440	1020	2240	foot
uw	300	870	2240	boot
er	490	1350	1690	bird

Table 1: Formant frequencies of vowels

HINTS/REMARKS

It more fun if you do not listen to the speech sample beforehand. Afterwards you can check if your formant analysis was "correct". Use the zoom in the figure window in MATLAB to find segments where the signal can be regarded as stationary. Then copy that segment to a new vector and perform an LP analysis.

5 Phonemes and Allophones

Vowels, which we studied in the previous section, constitute one class of phonemes. Here we identify what other phonemes are contained in "male short". This time, it is OK to listen to the speech!

TASKS/QUESTIONS

1. In the time plot from the previous sections, mark the regions of the other phonemes and label each region with the correct phonetic symbol. (1 pts)
2. Can consonants have formant frequencies? (1 pts)
3. What is a diphthong? (0.5 pts)
4. How many phonemes are there in English? (0.5 pts)
5. What is a phone? What is an allophone? How many allophones are there? (0.5 pts)

HINTS/REMARKS

1. In chapter 2 of the course book, you can learn what the difference between phonemes, allophones, and phones is. This is material which makes linguistics happy. Basic knowledge of this is also useful to the speech signal processing expert.

6 The Spectrogram

Program a spectrogram function in MATLAB. A spectrogram shows the energy in speech as a function of time and frequency. The result is often displayed with time on the horizontal axis, and frequency on the vertical axis. The energy (squared magnitude, and in dB) of the DFT coefficients are illustrated by e.g. a grayscale, where black corresponds to high energy, and white represents low energy. To obtain a smooth evolution over time, we want to use overlapping analysis windows, see Figure 1. Test your function on the speech samples provided for this lab. For the presentation have the result for "male_short" available.

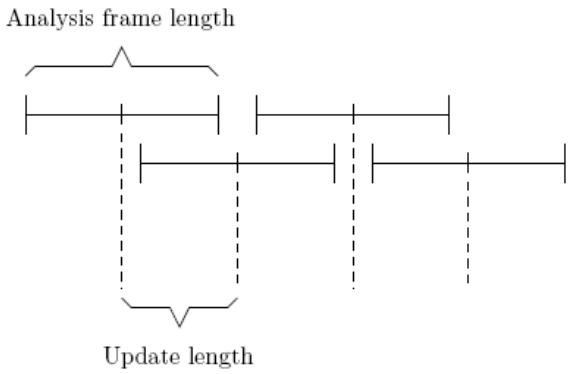

Figure 1: Overlapping analysis frames

TASKS/QUESTIONS

1. First program a function that extracts overlapping frames and plots each frame. Input arguments should be update length, and analysis length:

```
function myspectrogram(x, alen, ulen)
% x is the speech vector
```

```
% alen is the analysis frame length, ulen is the update length

 N = length(x);
naf = floor((N-alen+ulen)/ulen); % Number of analysis frames

n1 = 1;
n2 = alen;
for n=1:naf % Counter over analysis frames
xf = x(n1:n2);
figure(1); clf; plot(xf); axis([1 alen min(x) max(x)]); pause(0.05)
n1 = n1 + ulen;
n2 = n2 + ulen;
end
```

Call `myspectrogram(male_short, 256, 32)` from the MATLAB prompt and view the "animation"! (0.5 pts)

2. Next add frequency analysis of each frame. Use the code for the DFT based spectrum from section 3 (dont forget to window before DFT). If you want to plot both the time domain signal, and the spectrum in the same plot, but in different parts of the window, see `subplot`. Illustrate the function as in the previous step. (0.5 pts)

3. Now we are ready to implement the classical spectrogram. Instead of plotting the squared magnitude of the DFT, you put it into the columns of a matrix S. Preallocate the matrix by `S = zeros(alen/2+1, naf);`. In the loop you write to column n by `S(:,n) = ....` In this assignment it is important that the right hand side is a column vector (try and see what MATLAB says otherwise). To plot the spectrogram, use the following code

```
colormap(gray) % The classical spectrogram is gray,
% type help gray for other colormaps

imagesc(flipud(-S)); % flipud flips S along the frequency axis
so that
% frequencies increase when we move up the vertical axis
% -S makes black correspond to high energy!
```

(1 pts)

4. Produce a narrow-band spectrogram. Be prepared to indicate the fundamental frequency track, and the boundaries of the phonemes. (0.5 pts)

5. Produce a wide-band spectrogram. Here the spectrogram tends to become blocky if smoothing is not performed in the frequency domain. Incorporate smoothing by zero-padding before applying the DFT. Can you see the formant trajectories? (0.5 pts)

7 Speech Parameter Estimation

In this section, you will analyze (estimate) the parameters of a vocoder model. In the next section you will synthesize speech based on the estimated parameters. The parameters we will estimate are frame energy, pitch, vocal tract filter coefficients, and voiced/unvoiced classification.

The estimation should be done on a frame-by-frame basis and we will use overlapping analysis frames. Just as in the case of the spectrogram the choice of analysis frame length is a compromise between having long enough frames to get reliable estimates, but not too long so that rapid events are averaged out[2]. Also as in the spectrogram case, the update length controls the smoothness of the parameter trajectories over time.

TASKS/QUESTIONS

1. Let us warm up with estimation of frame energy (normalized per sample)

$$E = \frac{1}{N_\alpha} \sum_{n=0}^{N_\alpha - 1} x^2(n) \tag{1}$$

   where N_α is the analysis frame length and the sum is over the samples in one analysis frame. Write a function that returns the frame energies in a vector. Start with the function skeleton from the first task in section 6. Before the loop over all frames, allocate space for the vector of energies:

```
function E = analysis(x, alen, ulen)
% Initialization
E = zeros(naf, 1);
% Inside loop
E(n) = ...
```

[2]For the spectrogram we consciously average out the pitch pulses in the narrow-band case, whereas we consciously sacrifice accuracy in the spectral domain in the wide-band case.

(0.5 pts)

2. Next let us look at voiced/unvoiced detection. This is a difficult problem, but here we resolve to a simple solution based on zero-crossings. A zero crossing occurs when $x(n)x(n-1) < 0$. By counting all those occurrences within a frame, and normalizing by the frame length, a (near) continuous parameter is obtained. To make a binary decision a threshold can be used (by normalizing with the frame length, the threshold becomes independent (almost) of the analysis frame length). Extend your analysis function like:

```
function [E, ZC, V] = analysis(x, alen, ulen)
% Initialization
E = zeros(naf, 1);
ZC = zeros(naf, 1);
V = zeros(naf, 1);

% Inside loop
E(n) = ...
ZC(n) = ...  % The normalized number of zero crossings
V(n) = ...  % Equal to 1 if voiced, 0 if unvoiced.
```

(1 pts)

3. Next extend your analysis function by incorporating code for vocal tract filter estimation via LP analysis. The code from Section 3 should work fine! Store the filter parameters in the rows of a matrix like:

```
function [E, ZC, V, A] = analysis(x, alen, ulen, M)
% Initialization
E = zeros(naf, 1);
ZC = zeros(naf, 1);
V = zeros(naf, 1);
A = zeros(naf, M+1); % M is the prediction order.
% M+1 allows space for the leading 1

% Inside loop
E(n) = ...
ZC(n) = ...  % The normalized number of zero crossings
V(n) = ...  % Equal to 1 if voiced, 0 if unvoiced.
A(n,:)  = ...  % Make sure the polynomial coefficients are in
```

`a row vector!` (0.5 pts)

4. Finally extend the function with pitch analysis. Many speech coding (compression) systems depend on accurate pitch analysis. Here we base our estimation on the correlation function of the frame. You may choose if you wish to use the ACF or the normalized cross-correlation function (slightly more difficult to program). The problem you have to solve is how to find the lag of the peak in the ACF that corresponds to one pitch period. This is usually easy to do "manually", i.e., by looking at the ACF, but you have to make the computer do it automatically! The function will finally look something like

```
function [E, ZC, V, A, P] = analysis(x, alen, ulen, M)
% Initialization
E = zeros(naf, 1);
ZC = zeros(naf, 1);
V = zeros(naf, 1);
A = zeros(naf, M+1); % M is the prediction order.
% M+1 allows space for the leading 1
P = zeros(naf, 1);
% Inside loop
E(n) = ...
ZC(n) = ...  % The normalized number of zero crossings
V(n) = ...  % Equal to 1 if voiced, 0 if unvoiced.
A(n,:)  = ...  % Make sure the polynomial coefficients are in
a row vector!
P(n) = ...  % Pitch period in samples
```
(0.5 pts)

5. Plot the output of your analysis function with the following code (or include the code below in the analysis function):

```
figure(1);clf;

subplot(3,2,1)
plot(x) % Plot the input waveform
axis([1 length(x) min(x) max(x)]);

subplot(3,2,2)
plot(sqrt(E)) % Plot the standard deviation
```

```
axis([1 length(E) min(sqrt(E)) max(sqrt(E))]);

subplot(3,2,3)
plot(V, .) % Plot voiced/unvoiced decision
axis([1 length(V) 0 1]);

subplot(3,2,4)
plot(ZC) % Plot the normalized number of zero-crossings
axis([1 length(ZC) min(ZC) max(ZC)]);

subplot(3,2,5)
F = 8000./P;
plot(F) % Plot the fundamental frequency in Hz
axis([1 length(F) 0 600]);

subplot(3,2,6)
S = zeros(512, naf);
for n=1:naf
S(:,n) = 20*log10(abs(freqz(1,A(n,:),512)));
end
S = flipud(S);
colormap(gray);
imagesc(S); % Illustrate the vocal tract envelope in a spectrogram
style!
```

Test and tune your analysis function on the files "male long", and "female long". Be prepared to provide plots with smooth temporal evolution, i.e., with `ulen = 1`. (0.5 pts)

6. What could make the voiced/unvoiced detection go wrong? (1 pts)
7. In the pitch estimation, what can cause pitch doubling? What can be the reason for pitch halving? (1 pts)

8 The Vocoder

In this section we will synthesize speech based on the parameters from the analysis. Speech coding (compression) systems based on this principle are called vocoders and can reach bit rates as low as 2 kbits/second when the parameters

are quantized. This can be compared with 64 kbits/second for the PCM system used in fixed telephone networks. The quality of the vocoder is lower though. Here we will synthesize speech based on unquantized parameters.

TASKS/QUESTIONS

1. Let us start by only incorporating the vocal tract filter in the synthesis. Write a function like

```
function s = synthesis1(E, ZC, V, A, P, ulen)

% We have included all the estimated parameters as input arguments
% but here we only use A!

n_frames = size(A,1); % Assuming filter coefficients are stored
row-wise

% Create a pulse train excitation:
cp = ...; % Constant pitch period in samples

pexc = zeros(n_frames*ulen, 1);
pexc(1:cp:end) = 1;

% Create noise excitation:
nexc = ...;

n1 = 1;
n2 = ulen;
Z = [];
s = zeros(n_frames*ulen,1);
for n=1:n_frames
% Filter the excitation through the production (vocal tract) filter:
[s(n1:n2), Z] = varifilter(1, A(n,:), pexc(n1:n2), Z);

n1 = n1+ulen;
n2 = n2+ulen;
end
```

   Make sure you use the same update length (ulen) in the synthesis as in the analysis! Make sure you understand why ulen samples are generated

in the synthesis for each analysis frame[3].

Test with a pulse train excitation (use a constant pitch corresponding to 100 Hz). Then test with noise excitation. Can you hear what is said based on only vocal tract parameters (you may need to rescale the whole output before playing back)? (1 pts)

2. Make sure the energy contour of the synthesized speech matches that of the original speech. A crude but simple way is to (re)normalize each synthesized frame to have frame energy $E(n)$[4]. Write a new function `synthesis2` (naturally based on `synthesis1`!). Test with the pulse and noise excitations from the previous task. How much does the energy contour contribute to the intelligibility? (1 pts)

3. Switch between pulse excitation and noise excitation based on the voiced/unvoiced decision, and write `synthesis3`! Does this increase intelligibility? Our vocoder model assumes that all speech sounds are either voiced or unvoiced. Is this true? Discuss how our vocoder could be modified? (0.5 pts)

4. Create a function `synthesis4` by adding time varying pitch! Now you cannot use the variable pexc anymore, but have to create the pulse train "on the fly". One way is to keep a counter that is incremented for each sample during voiced frames. When the counter equals or exceeds the current pitch value, a pulse is inserted into the excitation, and the counter is set to zero. The quality of the synthesized speech depends a lot on the quality of the pitch estimate. Make sure the pitch contour is smooth in voiced regions. If necessary you may median filter the pitch contour prior to synthesis. (1 pts)

5. Tune your analysis and synthesis functions so that the produced speech

[3]Instead of `varifilter` you can also use the MATLAB function `filter` to implement the synthesis. But then you have to make sure that the final filter state after one frame is used as initial state in the next frame. Unfortunately, the filter structure in MATLAB is not designed for time-varying coefficients and audible clicks may result. To remedy this, see the function `filtic`, and `varifilter`. Another option is to implement the filter yourself. In that way you have full control over the filter memory. The downside is that your filter will probably execute slower than MATLAB's `filter` which is a built-in function.

[4]In the most straightforward energy normalization, a problem occurs when the synthesis frame contains no pitch pulses (this will happen frequently if your update length is short). Can you suggest a fix? It may require estimating the prediction residual energy (thus changing also your analysis function slightly), and re-normalizing the excitation (before filtering) during synthesis.

has as high quality as possible. We want to hear an exciting result! Please make sure the vocoder output is readily available in a separate file (.mat; see help save) when you come to present your results. (1.5 pts)