FACULTY OF ENGINEERING AND ARCHITECTURE

DEPARTMENT OF INFORMATION TECHNOLOGY
RESEARCH GROUP WAVES

Prof. Dr. Verhulst
s.verhulst@ugent.be
waves.intec.ugent.be/hearing-technology

Lab Exercise 4

Speech intelligibility is crucial for efficient communication with others and can be influenced by the sound source (e.g., the gender of the speaker or pronunciation), the medium (e.g., home, office, theatre) and the receiver (e.g., our ears, attention state, fatigue).
In this assignment, we will explore objective models of speech intelligibility, which are useful in predicting how speech intelligibility changes when the medium or receiver changes, i.e. the speech intelligibility index (SII). The SII ranges from 0 (non-intelligible) to 1 (highly intelligible) and is defined as the product of the band importance function and band audibility function, summed over the total number of frequency bands:

$$SII = \sum_{i=1}^{N} I_i A_i$$

Where *N* is the total number of computational frequency bands i, I the band-importance functions and *A* the band audibility functions. The computations of the SII are described in an ANSI standard and will be used in this exercise to study how varying the frequency content of speech impacts the SII. To this end, you will process speech material which is part of a Flemish standardized speech intelligibility test (Matrix test), and for a small number of conditions, you will test whether the predicted SII value corresponds to your experimental observations. The sounds will be processed using Python and the experiments on speech intelligibility will be done using the AFC toolbox for MATLAB that can be run either on your local computer or using the MATLAB app available on Athena.

The current assignment consists of two parts: 4.1 Intelligibility of Speech and 4.2 AM detection threshold. In lab exercise 4.1, parts 1 and 2 ("Spectral properties of speech material" and "Speech processing"), a Jupyter Notebook (Python3 kernel) is provided. In lab exercise 4.1, part 3, a speech intelligibility (in noise) test, the Flemish matrix test, will be run using the AFC toolbox for MATLAB. In lab exercise 4.2, an amplitude-modulation (AM) detection threshold test will be run using the AFC toolbox for MATLAB.

The following materials are provided:

1. **HearingLab4.ipynb**: Jupyter Notebook where the assignment (lab exercise 4.1 Parts 1 and 2) has to be filled in. The sounds to be processed are included in the 'sounds/vlmatrix/' folder (AFC toolbox).
2. **helper.py**: Python functions to assess the speech intelligibility index (SII), read and write wav files. See the tutorial at the end of this document.
3. **AFC toolbox (AFC_for_lectures_2022)**: MATLAB toolbox to conduct psychoacoustic experiments. In the 'experiments' folder, The Flemish Matrix (VlMatrix) speech test is included in the folder 'SRT_task' and the AM detection threshold test in the folder 'AM_detection_ptp'. The VlMatrix sounds of three lists (60 sentences) are provided in the folder "../sounds/vlmatrix/00-original". Two other empty folders are foreseen where your processed speech materials will have to be placed: Folders "01-LP" and "02-HP."

Start by placing all these files (unzip if necessary) in your computer. You require to use the location of the AFC toolbox as input for the Jupyter notebook.

Please answer the questions in the Jupyter Notebook or in an extra file that you can upload on Ufora.

Lab Exercise 4.1: Intelligibility of Speech

Part 1: Spectral properties of speech material

First, we will investigate the spectral properties of speech material and noise signals which are often adopted to study how well listeners can perceive speech in challenging (noisy) listening conditions. The sentences we use are part of a standard speech reception threshold (SRT) test. You will have to read and process the background noise that is provided with the Flemish Matrix (VlMatrixnoise_ltass.wav) and the sentences of the materials. Follow step 1 (below) to get acquainted with reading and visualizing signals from the Flemish Matrix. Please note that you have to start indicating the absolute location of the AFC toolbox, re-assign the variable "dir_where_AFC" (cell [2] of the notebook), assuming you work on the Athena desktop (on Windows):

dir_where_AFC= '**H:\\Desktop\\HearingLab4_2022\\AFC_for_lectures_2022'**

1. Use the `audioread` function (see the Jupyter Notebook example) to load the .wav file `VlMatrixnoise_ltass.wav`. (listen to it, you can do it externally with your computer player or within Python). This is a stationary noise which has a spectral density equal to that of all the sentences in the speech test, it hence has the same long-term spectrum as the speech sounds. Therefore, this noise is often used as an **energetic** masker. It "competes" with the speaker on a spectral basis, without providing meaningful speech content. This masker is often used to determine the SRT (i.e., the signal-to-noise ratio in dB needed to identify 50% of the words in a sentence test correctly). Another commonly used masker is an **informational masker**. This masker contains speech-like information such as temporal gaps, frequent rises and falls, or even a clear formant structure (not shown in this demo, one reference: Dreschler et al., 2001).
2. To compare the spectral content of the `VlMatrixnoise_ltass.wav` noise masker and that of the sentences, make a spectrum plot of the noise (given as example in cell [5] in the Jupyter Notebook), and compare it with the spectrum of a single chosen sentence (fill it in in cell [6]), as well as to the spectrum of the 20-concatenated sentence signal (fill it in in cell [7]). Label your figure and compare your results. Why is it meaningful to use 20 sentences in for further analyses (the SII prediction), rather than using a single sentence?

Part 2: Speech processing – High- and lowpass filtered speech

Next, we will generate speech material which band-limits the frequency content of the sound and calculate how these signal modifications impact the predicted speech intelligibility (SII).To generate degraded speech samples, two types of processing will be used: low-pass (LP) and high-pass (HP) filtering.

3. Choose a range of cut-off frequencies you will use in the filtering process and that will yield different SII values. Choose **20 cut-off frequencies** in the range between 100 Hz and 10000 Hz and motivate your choice. You will use these 20 cut-off frequencies for both LP and HP filtering. In cells [7] and [8] you can fill in the variable f2test (you can use the same f2test variable in both cells. You may want to use the numpy function arange and please reflect about a relevant spacing of frequencies (linear or logarithmic scale, or others?).
4. Calculate **SII values** for the LP-processed sentences using the provided `SII` function and use filtered versions of the 20-concatenated sentences. The function requires as input a speech spectrum for 18 frequencies spaced at one-third (1/3) octave-band intervals (given in the variable "freq_i", given by us already in cell [3]). To generate this, you need to: *(i)* load the sounds; *(ii)* apply the filtering (LP or HP): use the function `butter` (use order 4 filters) to find the coefficients "a" and "b" that will be used for the function `filter`; *(iii)* apply an FFT: use the function `fft` and convert the output amplitudes to dB; and *(iv)* use the function called `Oct3Smooth`, which will return the 18 amplitudes in dB that can be used as input for the `SII` function. Compute the SII values and plot the results as a function of the 1/3 octave band frequencies. Have a look at the pre-coded lines in the notebook before starting this step.
5. Compute SII values for the HP-processed sentences. Use the same set of 20 concatenated sentences as for the LP-processing. Follow the same step-by-step processing as in 4, making sure that you use the filter coefficients for a high-pass filter.
6. From your SII-figures, choose the cut-off frequency that produce SII values of 0.75 for both LP- and HP-processed sounds, and fill in the values fc_low and fc_high, respectively. You will assess the experimental speech intelligibility for these two cut-off frequencies.
7. Finally, use these cut-off frequencies to filter all of the speech material. Store all of the files in different folders without changing their names. Some pre-coded lines are provided to read all the files under "dir_where" in a for loop. Fill in fc_low_norm and fc_high_norm according to the required input format of the "butter" function. Save the processed wav files in "dir_where_LP" and "dir_where_HP", respectively. Use the function audiowrite.

Part 3: Test your speech reception threshold

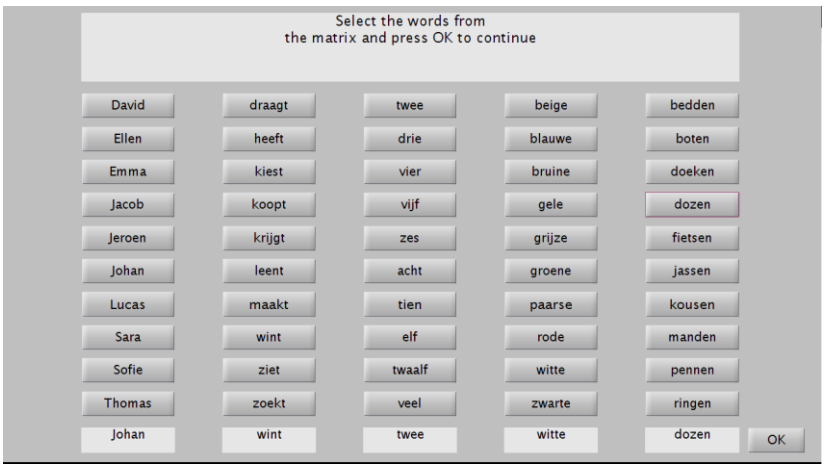
In this part, you will test your own speech intelligibility for the stimuli you generated in part 2. Place the filtered speech materials with SII values of 0.75 in the folders "01-LP" and "02-HP" for the low-pass and high-pass filtered signals respectively. Please note that a noise (the same as in the broadband speech) is already provided, so don't overwrite it. For running the experiments you need headphones or loudspeakers connected to your computer (set the reproduction level to a comfortable volume before starting the tests), and follow the instructions at the end of this document. The speech tests should run without any problems

across platforms (Windows, Mac, or Linux) and it was also tested to be run on Athena but you need to have MATLAB installed.

Using a standardized speech intelligibility test (a closed set 5-word Dutch sentence test, Matrix), you will determine the Speech Reception Threshold (SRT). This is the signal-to-noise ratio for which you perceive 50% of the words in a sentence test correctly. Your task is to identify the words you hear presented to you over headphones (or loudspeakers), while the signal level changes adaptively to challenge your intelligibility in an adaptive tracking procedure. Do this test in broadband speech, low-pass filtered speech and high-pass filtered speech.

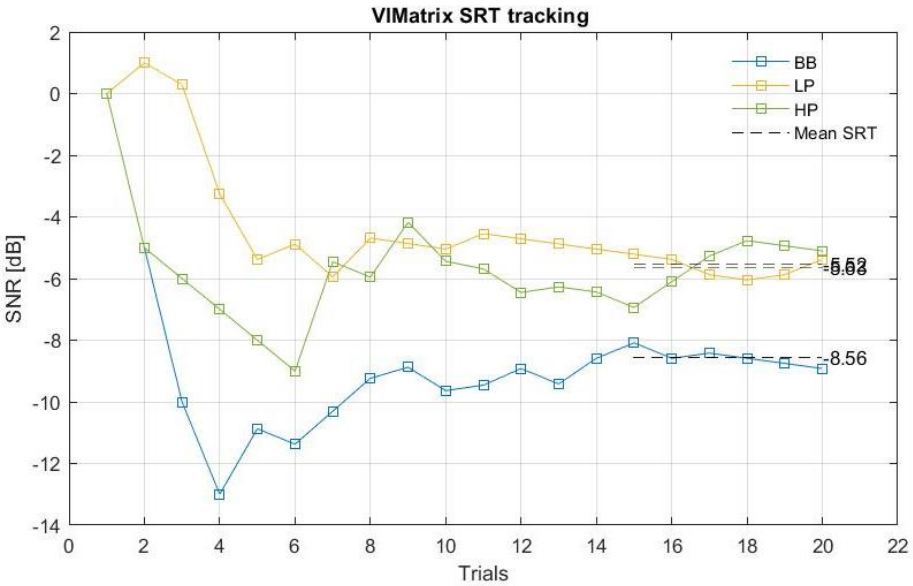
1. If you have set up everything according to the tutorial, set the folder 'AFC_for_lectures_2022' as your current MATLAB directory and type on the MATLAB command line "afc('main','exampleVlMatrix','S01','00')" ('00' for broadband speech, '01' for low-pass speech, '02' for high-pass speech; 'S01' is just the subject name so you can change this). Immediately after you are asked to indicate a list to test enter 1, 2, or 4 (no brackets needed). Make sure you use a different list for every test (broadband, low-pass and high-pass). To start you have to press "any key." You are presented with a list of twenty Dutch (Flemish) sentences. However, being a fluent speaker of Dutch is not necessary for the test, as all of the possible answers are given. Cross every word that you believe was said and press the button "OK" (see a screenshot of one trial below). The test will become more difficult after every correct decision.

```
>> afc('main','exampleVlMatrix','S01','00')
```



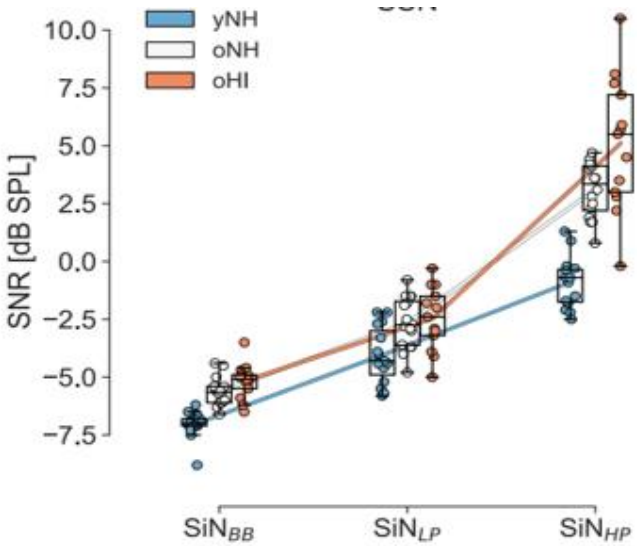
2. When you finish, you have to press the key 'e' to exit the test. The result file was already saved in the 'experiments > SRT_task > results' folder. Inspect the result files of all three conditions (in the 'exampleVlMatrix… .dat' file, which is a regular txt file, showing the used Testlist as exppar1 (list 1, 2 or 4), followed by the predicted SRT in dB, and the percent of correct speech recognition which is 50% in this test.) and compare them to the SII predictions. What was your final speech-reception-threshold for the low-pass and high-pass condition? Is it different to the SRT for the original broadband speech test material? You can do the test a couple of times per condition, it is known that due to the 'training effect' the resulting SRT is more reliable after a training phase with 20 sentences per condition.

3. Now you can visualize the adaptive tracking of the SRT. Set the folder 'AFC_for_lectures_2022' as your current MATLAB directory and type on the MATLAB command line "afc_SRT_extraction('S01')" with 'S01' the subject name you used to perform the broadband, high-pass and low-pass filtered VlMatrix test. This function will plot the SRT tracking and also shows the mean SRT calculated over the last 6 trials (sentences). Upload the resulting SRT tracking plot as a jpg or png file. Can you observe the SRT tracking? Compare again the results of the BB, LP and HP conditions. An example of such an SRT tracking is shown in the figure below.




```
Mean of last 6 trials - exampleVlMatrix
Condition BB : -8.56
Condition LP : -5.63
Condition HP : -5.52
```

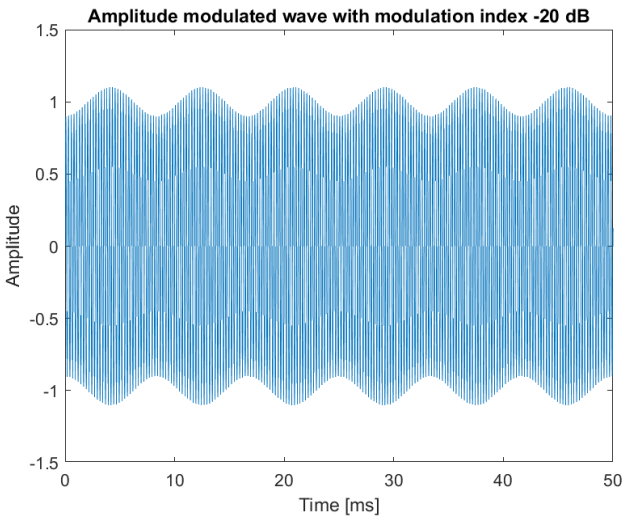
4. Compare your results with the SRT values for BB, LP and HP VlMatrix test (ipsilateral noise) data in literature shown in the figure on the right, for young normal hearing (yNH), old normal hearing (oNH) and old hearing impaired (oHI) subjects. How is your datapoint situated?

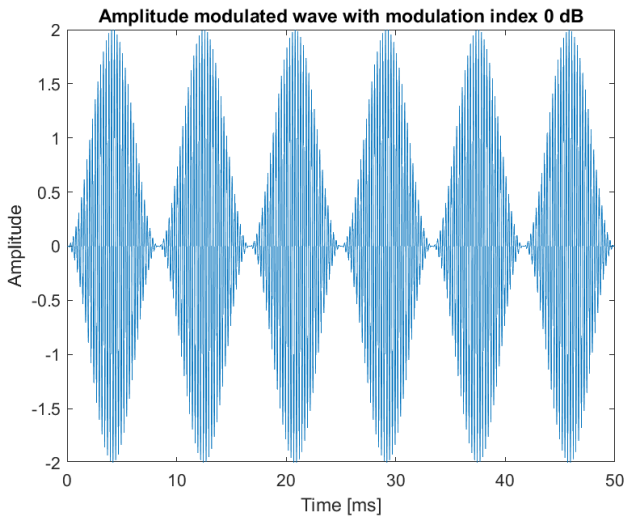


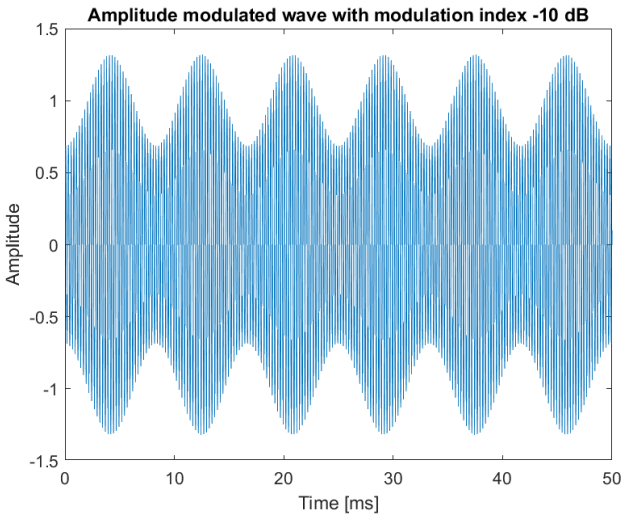
Lab Exercise 4.2: AM detection threshold

In the amplitude modulation (AM) detection threshold experiment, it is investigated how much amplitude modulation you require in order to be able to distinguish the pure tone from the AM tone. The experiment of this lab exercise is a 1-up, 2-down 3-AFC procedure (see lecture slides) and it determines the modulation index at which you have, in the case of this 1-up, 2-down procedure, 70.71% correctness at the equilibrium point. The modulation frequency is 120Hz, the carrier frequency 4kHz. The figures below show AM tones for different modulation indices. The AM tones are constructed using this MATLAB code:

```
% modulation wave: cosine of 120Hz, length 24000 samples, sample frequency
48000Hz
cosine_wave = cos([0:24000-1]'*2*pi*120/48000+pi);
% carrier wave: sine of 4000Hz
sine_wave = sin([0:24000-1]'*2*pi*4000/48000);
% amplitude modulated tone
modulation_index = 0; % to be determined during the experiment
AM_tone = sine_wave .* (1 + (10^(modulation_index/20) * cosine_wave));
```





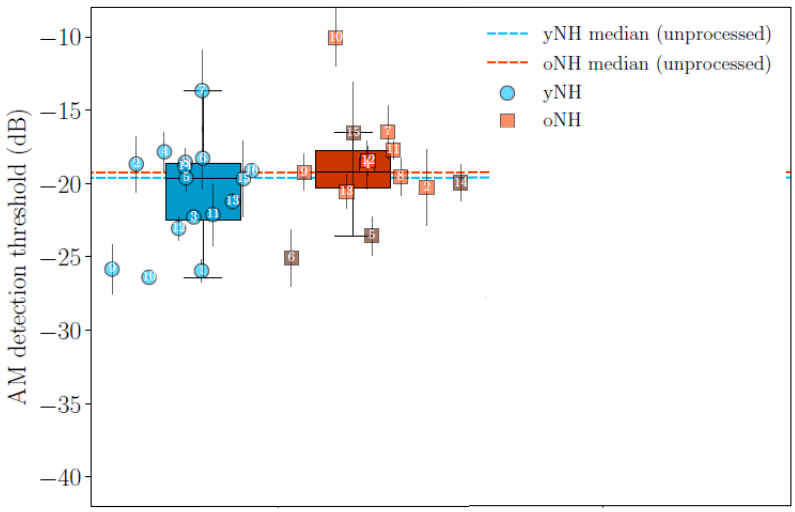


We will perform the AM detection threshold experiment in the MATLAB AFC Toolbox.

1. Set the folder 'AFC_for_lectures_2022' as your current MATLAB directory and type on the MATLAB command line "afc('main','AM_detection_ptp','S01')" with 'S01' the subject name. For every subject that you run this a message will pop-up asking you to create a new subject folder (def.control_path does not exist. Do you want to create it?), press "Yes". For running the experiment you need headphones or loudspeakers connected to your computer (set the reproduction level to a comfortable volume before starting the tests).

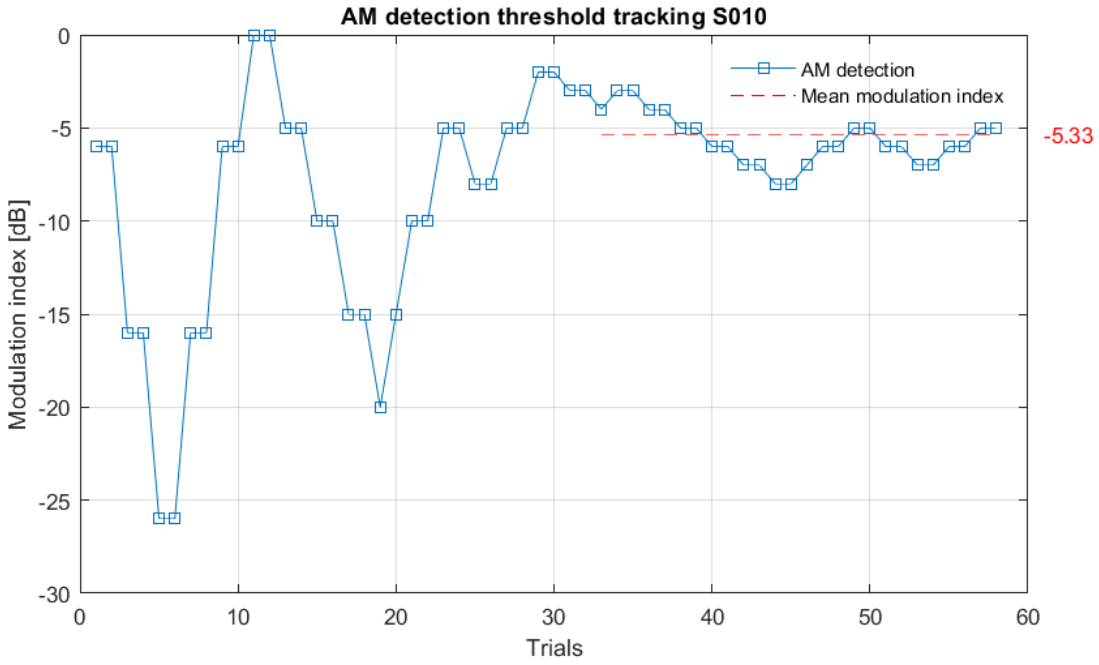
```
>> afc('main','AM_detection_ptp','S01')
```

2. In this experiment, each time you will be presented sequentially with three tones labelled 1, 2, 3, where two of them are the same and one is different (the AM tone). Your task is to select the one that sounds different than the other two by clicking the corresponding number. After you click one option you will hear the next three sounds. Based on your choice, the sound is adapted every time. You will have to repeat this task several times, with the experiment getting more difficult each time, until the point where you can barely detect the different tone is estimated. Every time, just answer what you think is correct (it doesn't matter if you are not completely sure).
3. Check in the folder 'experiments > AM_detection_ptp > results' in the dat-file 'AM_detection_ptp_ … .dat' your resulting modulation index: the exppar1 is the modulation frequency of 120Hz, exppar2 is the unprocessed material (default), the expvar is the mean modulation index in dB and its standard deviation is given next. Compare your resulting modulation index to the data below from literature, how is it situated compared to the NH data? The more negative the modulation index, the better you performed on the test.



Unprocessed conditions

4. Now visualize the tracking of the modulation index by using the command "afc_AM_extraction('S01')". An example is shown in the figure below. Indicate on the figure the last 6 turning points (reversals), upload this as a jpg or png file. Calculate the mean over these last 6 reversals (see lecture slides), this should correspond to the resulting modulation index threshold.


AM detection threshold tracking S010
AM detection
Mean modulation index
-5.33
Modulation index [dB]
Trials
0
-5
-10
-15
-20
-25
-30
0
10
20
30
40
50
60

Python tutorial: Functions in helper.py

SII

The function SII calculates the Speech Intelligibility Index according to ANSI S3.5-1997 (1997). This function requires a number of input parameters. Only one of them is mandatory: the speech spectrum level 'E':

- `'E'`: Stands for Speech Spectrum Level (section, 3.6 in the standard) and corresponds to the spectrum level of the speech in 18 bands spaced at steps of 1/3 octaves between 160 and 8000 Hz.
- 'VE': Vocal effort, indicates whether the voice is `'normal'` (default) `'raised'`, `'loud'`, or `'shout'`. Use the vocal effort you think is more suitable for the speech material you have.

The function can be called (minimalistically) in the following way:

```
SII_value = SII(E)  # assumes a normal vocal effort
SII_value = SII(E,VE='normal')
```

Oct3Smooth

The function `Oct3Smooth` takes the amplitudes `vLin` (in dB) that have frequencies `vFin` and combines together all the amplitudes that would correspond to the frequencies specified by `vFout`. If `vFout` is the vector with 18 one-third octave-band frequencies, and `vFin` and `vLin` are obtained from an FFT, then the FFT spectrum is converted to one-third octave band amplitudes, as required for the SII script. An example of the function usage is:

```
E = Oct3Smooth(vFin,vLin,vFout)
```

Its input arguments are:

- `vFin` [Hz]: The frequency vector describing the input spectrum
- `vLin` [dB]: The input spectrum
- `vFout` [Hz]: The frequency vector for the smoothed output spectrum

audioread

Reads the file specified by the file name "fname" (see cell [4] in the Jupyter notebook). It returns the input signal "insig" with amplitudes between -1 and 1 and the sampling frequency "fs". "fname" should be visible to Python, in the example the full absolute path is used.

```
fs, insig = audioread(fname)
```

audiowrite

Stores in signal "insig" with amplitudes between -1 and 1, sampled at a sampling frequency "fs" in Hz to the file name "fname," as a wav file with an amplitude resolution of 16 bits.

```
audiowrite(insig,fs,fname)
```

SpeechArray

Reads all the wav files within the folder "dir_where" (please make sure there are no subfolders in "dir_where" otherwise the function gives an error), and chooses a number of sentences ("nr_of_sentences", default = 10) at random and puts them all together in one long column array, assuming that all these sentences belong to the same audio file and are "played together". Make sure the dir_where contains more files as specified by "nr_of_senteces". This function excludes the speech shaped noise "VIMatrixnoise_ltass.wav".

```
Insig_all_sentences = SpeechArray(dir_where,20) # to choose 20 sentences
```

Using the AFC toolbox for MATLAB

The AFC toolbox for MATLAB (Ewert, 2013) is available in the zip file "**AFC_for_lectures_2022**"

1. Unzip this file in any accessible folder on your local computer or place it in your Athena folder. The folder name is "AFC_for_lectures_2022".
2. Run MATLAB and change the current directory to the main folder by either navigating through the MATLAB explorer (window named "Current Folder") or by typing the following on the MATLAB command line (assuming you placed the AFC toolbox in the Athena desktop:

   cd('H:\Desktop\HearingLab4_2022\AFC_for_lectures_2022\')
3. Run the script "afc_addpath.m", you can type on the command line:

   afc_addpath;
4. To run the AFC toolbox using the Flemish Matrix material (UGent AFC scripts) you have to type (you will be asked on the MATLAB screen which sentence list you want to test, by entering 1, 2, or 4 – only three lists are provided):

   afc('main','exampleVlMatrix','S01','00')

   where the arguments 'main', and 'exampleVlMatrix' should not be changed (remember that MATLAB is case sensitive), the third argument is the subject ID, it can be any string in between brackets, here 'S01' is to respect the nomenclature "Subject 01". The last input parameter is very important because it sets the condition you want to test:
   - '00' will use the original Flemish Matrix material (provided by us, files in ".. \AFC_for_lectures_2022\sounds\vlmatrix\00-original\")
   - '01' will use the low-pass filtered speech, that you have generated in Part2 of this assignment. Make sure you use the same filenames. Please note that the background noise is provided (the same background noise is used for all the experimental conditions "VlMatrixnoise_ltass.wav").
   - '02' will use the high-pass filtered speech, that you have generated in Part2 of this assignment. Again, make sure you use the same filenames and that the same background noise is already included.
5. All results are stored in ".. \AFC_for_lectures_2022\experiments\SRT_task\results\, where after you run every condition you will obtain three dat files. If you used 'S01' as the participant ID:
   - exampleVlMatrix_S01_00.dat for the original (broadband) speech
   - exampleVlMatrix_S01_01.dat for the low-pass speech
   - exampleVlMatrix_S01_02.dat for the high-pass speech

   These are txt files that contain the signal to noise ratio (SNR) at your 50% threshold (SRT), where the initial SNR was 0 dB (noise level = speech level) and then the noise level was adjusted adaptively using the tracking rule described by Brand & Kollmeier (2002).

Note that after you completed each experimental condition you cannot test it again using the same subject ID. If you, by mistake or to test completed the task and want to do it again, you should remove manually the corresponding control file in the results folder, which would be something like "control_exampleVlMatrix_S01_00.dat." To avoid confusion in this case, also remove the corresponding result file, which is for this example the file "exampleVlMatrix_S01_00.dat."

You are now ready to run the experiments from home.

References:

ANSI, A. (1997). "S3. 5-1997, methods for the calculation of the speech intelligibility index," New York: American National Standards Institute 19, 90–119.

Brand, T., & Kollmeier, B. (2002). Efficient adaptive procedures for threshold and concurrent slope estimates for psychophysics and speech intelligibility tests. J. Acoust. Soc. Am., 111(6), 2801–2810. https://doi.org/10.1121/1.1479152

Dreschler, W., Verschuure, H., Ludvigsen, C., & Westermann, S. (2001). ICRA noises: Artificial noise signals with speech-like spectral and temporal properties for hearing instrument assessment. Int. J. Audiol., 40(3), 148–157.

Ewert, S. (2013). AFC - A modular framework for running psychoacoustic experiments and computational perception models. In Proceedings of the International Conference on Acoustics AIA-DAGA (pp. 1326–1329).

Houben, Rolph, et al. "Development of a Dutch matrix sentence test to assess speech intelligibility in noise." International Journal of Audiology 53.10 (2014): 760-763

This exercise guide was developed by Tijmen Wartenberg, Alejandro Osses, Sarah Verhulst, 2020, UGent. Adapted by Marjoleen Wouters, 2022, UGent.