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 2: Hearing Aid and Cochlear Implant Signal Processing

Activate the environment that you used in the previous exercise:
`source activate <name_environment>`

The pyaudio and GPyT python modules need to be installed for subsection 1.1:
`conda install pyaudio`

To install GPyT:
`git clone https://github.com/jabeim/AB-Generic-Python-Toolbox.git`
or download ZIP from https://github.com/jabeim/AB-Generic-Python-Toolbox and extract it
Go to the newly created AB-Generic-Python-Toolbox folder and run the command:
`pip install -e .`

Go to the AudCompLab2 folder via the terminal and open a jupyter notebook:
`jupyter notebook`

Part 1: Cochlear Implant Signal Processing
From a first, basic perspective, CI signal processing sends electrical pulses over the electrodes which are positioned along an electrode array inserted inside the cochlea. Current pulses are sent out at a fixed rate (pulses per second, pps) along each of the electrodes, with the intensities of the current pulses following the envelope of the signal.

1.1 Make a new python file in which you load the `example.wav` file, and make a basic CI algorithm which approximates the signal by a positive polarity pulse train that follows the envelope of the original speech waveform. You can omit the negative amplitudes of the original signal in your algorithm. The pulse width should be 1 ms and there should be 900 pulses per second. The following functions might come in handy to make your algorithm:
`np.tile, scipy.signal.hilbert, np.zeros, np.ones, np.concatenate`

You can compare the output of your CI algorithm to the multichannel HiRes120 algorithm [1], used in the Advanced Bionics cochlear implants. The `CompAudLab2_CI` notebook has simulated the electrical pulses generated over time by the HiRes120 algorithm for the `example.wav` file, across 16 electrodes that are normally used in an Advanced Bionics CI.
The pulses were also passed to a vocoder function to simulate the sound perceived by a cochlear implant user. The vocoded audio signal was saved to the `vocoded_example.wav` file, you can listen to it to experience how a cochlear implant sounds like.

1.2 In the next part of this exercise, you can compare the performance of a paradigm multichannel CI algorithm [2] against the auditory-nerve performance of the normal human

auditory system. To help you with this analysis, `CompAudLab2_CI` has simulated the response from the auditory-nerve (AN) to two inputs: (i) the audio signal processed using the custom CI algorithm, and (ii) the audio signal processed using the transmission-line cochlear model (CoNNear), followed by an inner-hair-cell model. The outputs of this procedure are `AN_CI` and `AN_NH`, corresponding to the auditory nerve firing rate [spikes/s] for the model with CI or auditory periphery as input, respectively. The sampling frequency is 16 kHz, the dimensions (N x filter channel), `cf_tl` has the center frequencies corresponding to the filters. In case your computer cannot run the model code, you can load in the `AN_responses.mat` file into Python using the following command:
`AN_responses=scipy.io.loadmat('AN_responses.mat')`
`AN_responses['AN_NH']`, `AN_responses['AN_CI']` & `AN_responses['cf']` are arrays with the auditory nerve responses for the simulated cf channels, which you require for the remainder of the exercise.

1.3 Expand on this code to investigate the following aspects: Compare the AN outputs for a center frequency well above the phase-locking limit (e.g. 3 kHz), and for a center frequency below the phase-locking limit (e.g. 1kHz). Are there notable differences between the waveforms in the periphery model, and in the CI algorithm?

1.4 Sum up the AN waveforms across all 16 channels over time, to which degree can you reconstruct the original speech signal waveform? Compare your results to the summed AN responses of the standard HiRes120 algorithm from subsection 1.1, simulated at the last section of the `CompAudLab2_CI` notebook. In your opinion, which CI algorithm better emulates the AN responses of the normal human auditory system? By examining the generated pulses across channels (e.g. for a low and a high frequency), which factor could justify the performance difference between the standard HiRes120 strategy and the paradigm CI algorithm?

1.5 Now change the parameters from the paradigm CI-algorithm to produce monophasic pulses. What is the effect of this change on the algorithm and the AN outputs?

1.6 Cochlear implant users have extreme difficulties in localizing interaural time differences (ITDs) for very basic stimuli (e.g. low-frequency pure-tones with a different interaural phase). Can you, with help of your knowledge of the coding differences in the normal-hearing model and the CI coding strategy, find out what the origin of this perceptual deficit can be? Do you think CI users are able to code interaural level differences (ILDs) well?

Part 2: Hearing-Aid Signal Processing

In this part of the exercise, you will study the effects of outer-hair-cell damage on auditory speech processing, as well as apply a dynamic compression hearing-aid algorithm to speech [2], after which we can study to which degree this improves/modifies the hearing-impaired signal processing.

2.1 To start this exercise, the `example.wav` file was pre-run in 3 cochlear transmission-line models in which the degree of OHC damage was varied in a frequency specific way. The `CompAudLab2_HA` notebook has simulated the outputs of the TL cochlear model (CoNNear)

for the normal-hearing (NH) reference model (`nh_out`), the Slope25 model with a sloping high-frequency hearing loss (`hislope_out`), and the Flat25 model in which the outer-hair-cells were equally damaged at all frequencies (`hiflat_out`). Both latter models are referred to as hearing-impaired (HI) models.
First, you should estimate how much dB gain loss is present at the different cochlear center frequencies (cfs) of the HI models. For a cf corresponding to approx. 4 kHz, compare the time-domain waveforms between the different models. Do you see a difference?
Now plot `20*log10(abs(X))` [in dB], and compare the results again. Visually, is there a gain loss [in dB] between the different models? Is that loss equally big at small and large amplitude sections of the waveform?

2.2 Gain loss is important for the low stimulus levels as this may reduce audibility, so you can estimate the dB gain loss difference between the normal-hearing model and the two hearing-impaired models as the root-mean-square difference over the 0.55-0.95 second window (26400-45600 in samples) for each cf by writing a `for-loop`. Plot the gain loss function (i.e., root-mean-square difference [in dB: 20*log10(NHreference)-20*log10(HImodel)]) as a function of cf for the two models. You have will then have derived a gain loss function from the model simulations which you can use to base your hearing-aid fitting onto. Alternatively, and with real people, you could use a measured audiogram for this purpose also (i.e. pure-tone detection threshold difference from reference normal hearing in dB HL for octave frequencies between 0.25 and 8 kHz).

2.3 Now run the dynamic compression algorithm with a gain prescription which you think will work well for each type of hearing impairment and the `example.wav` file. The compressor algorithm requires a gain table as input, and it will deliver you a modified stimulus waveform for which the hearing-aid algorithm was applied. An example of a level-independent gain prescription is provided in the `CompAudLab2_HA` file. You can also refer to the `README` file of the algorithm (included in the `hearing_aid_processing.zip`) or the header of the `dynamic_compressor` script for more information on how you need to define the inputs to this model.
Apply for each of your HI models, *(i)* a level-independent half-gain rule for the HI algorithm, and *(ii)*, a compressive level-dependent algorithm for which the low stimulus levels follow the half-gain rule and less gain is applied for the higher stimulus levels. Plot the waveforms of the algorithms you will apply and validate roughly that they applied the gain rule as you desired (have a look at the power spectrum also).

2.4 Use the processed signals as inputs to the HI cochlear models to get the basilar-membrane response. For each gain rule, estimate in the window between 0.55 and 0.95 seconds of the responses to evaluate how much gain your algorithm has restored of the basilar-membrane vibration, by comparing the model outputs and restoration in the HI model that was run with original speech to the model outputs of the same HI model which was run with the HI algorithm-processed speech. Take another time window of the same duration at a stimulus maximum and perform the same analysis. Does the algorithm provide equal amounts of gain at all instantaneous level differences? Are there other aspects in which the waveforms of the HI model, NH model and hearing-aid HI model responses are different?

References:

[1] Koch, D. B.; Osberger, M. J.; Segal, P. & Kessler, D. (2004). HiResolution and Conventional Sound Processing in the HiResolution Bionic Ear: Using Appropriate Outcome Measures to Assess Speech Recognition Ability, Audiology and Neurotology, Vol.9, No.4, (July/August 2004), pp. 241-223, ISSN 1420-3030

[2] Nogueira, W., Litvak, L., Edler, B., Ostermann, J., & Büchner, A. (2009). Signal processing strategies for cochlear implants using current steering. *EURASIP Journal on Advances in Signal Processing*, *2009*, 15.

[3] Grimm, G., Herzke, T., Ewert, S., & Hohmann, V. (2015). Implementation and evaluation of an experimental hearing aid dynamic range compressor.

The CI processing and dynamic compression code was written by Fotios Drakopoulos, UGent, 2020 (fotios.drakopoulos@ugent.be). The auditory model code provided in this exercise cannot be used for other purposes than this lecture. Should you wish to use the CoNNear models for your own research, you can find the model code and academic license here:

https://github.com/HearingTechnology/CoNNear_cochlea
https://github.com/HearingTechnology/CoNNear_IHC-ANF

Baby, D., Van Den Broucke, A., & Verhulst, S. (2021). A convolutional neural-network model of human cochlear mechanics and filter tuning for real-time applications. *Nature Machine Intelligence*, 1-10.

Van Den Broucke, A., Baby, D., & Verhulst, S. (2020). Hearing-Impaired Bio-Inspired Cochlear Models for Real-Time Auditory Applications. In *Proc Interspeech* (Vol. 2020, pp. 2842-2846).

Drakopoulos, F., Baby, D., & Verhulst, S. (2020). A neural-network framework for modelling auditory sensory cells and synapses. *bioRxiv*.