Mathematisch-Naturwissenschaftliche
Fakultät

Fachbereich Informatik
Arbeitsbereich Visual Computing

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Grundlagen der Signalverarbeitung (Prof. Schilling)

SS 2024

Assignment 2

Remarks

Please submit your exercises in ILIAS before 23:55 on the closing date. At least one member of the group must be present at our biweekly tutorial, beeing prepared to explain *each* exercise. Random groups will be asked to present their solutions. Stick to the submission procedure described below.
Reminder: If there is a built-in function for an algorithm, you are supposed to implement, do *not* use it. If you are asked to explain your results, hand your answers in as a pdf-document.

Python

- **Working on the programming assignments requires a python environment with the packages numpy scipy and matplotlib installed.**
- **Use the script file that is provided on ILIAS. Use the exact filenames for your functions and scripts that are specified in the framework.**
- **Please put the answers to comprehension questions into a separate PDF file.**
- **If there is a built-in function for an algorithm you are supposed to implement, do *not* use it (unless you are told so).**

How to work on these exercises: The programming assignments in this course consist of multiple exercise files named `exercise_XX.py`. These files contain some incomplete definitions of python functions which you should complete. Running an exercise script with `python exercise_XX.py` (where `XX` refers to the two-digit exercise number) from your anaconda environment plots the results of your implementation. There are also some test cases logged to the console to provide feedback to you. Please ensure that your implementation passes all test cases. Note that we will grade your submissions using further test cases with multiple different inputs.

Submission: Create a folder `lastname1_..._lastnameN` with your team members lastnames in alphabetical order. Copy your `exercise_XX.py`-files into this folder. Create a zip file `lastname1_..._lastnameN.zip` from the folder. Submit the zip-file to Ilias.

Exercise 3: SSB Modulation [10 points]

This exercise (`exercise_03.py`) is about implementing the single side band (ssb) modulation and demodulation.

a) Hilbert Transformation [2 points]: Complete the function `hilbert_trafo` that applies the hilbert transformation to a given input signal. You are allowed and encouraged to use the `numpy` functions `np.fft.fft`, `np.fft.ifft`, `np.fft.fftshift` and `np.fft.ifftshift`.

b) SSB Demodulation [4 points]: In this task you are given the file `antenna_signal_ssb.wav` which contains multiple modulated channels in the high frequency spectrum. Complete the function `ssb_demodulation` that receives a ssb-modulated signal and performs demodulation to extract the low-frequency signal from a given channel. When executing `exercise_03.py` the demodulated signal will be written to the file `received_signal_ssb_3b.wav`. Listen to this file to verify your implementation's correctness. You may change the arguments `carrier_freq` and `upper_side_band` in the `main` function to demodulate all channels. Note that there are lower and upper side band channels present in the antenna signal. Hint: Use your implementation of the function `hilbert_trafo` from the previous task.

c) SSB Modulation [4 points]: Complete the function `ssb_modulation` that receives a low-frequency audio signal and modulates it to a given frequency band. Apply the bandpass filter to prevent your signal from exceeding the desired bandwidth.
If your implementation is correct, the file `bach_demodulated_ssb_3c.wav` should sound similar to the file `bach.wav` located in the `audio` directory.

Exercise 4: Frequency Modulation [8 points]

This exercise (`exercise_04.py`) is about implementing the frequency modulation and demodulation.

a) Frequency Demodulation [4 points]: In this task you are given the file `antenna_signal_fm.wav` which contains multiple modulated channels in the high frequency spectrum. Complete the function `frequency_demodulation` that receives a frequency-modulated signal and performs demodulation to extract the low-frequency signal from a given channel. When executing `exercise_04.py` the demodulated signal will be written to the file `received_signal_fm_4a.wav`. Listen to this file to verify your implementation's correctness. You may change the argument `carrier_freq` in the `main` function to demodulate all channels. Hint: You may use the function `ideal_bandpass` from `utils.py`.

b) Frequency Modulation [4 points]: Complete the function `frequency_modulation` that receives a low-frequency audio signal and modulates it to a given frequency band. Apply the bandpass filter to prevent your signal from exceeding the desired bandwidth.
If your implementation is correct, the file `bach_demodulated_fm_4b.wav` should sound similar to the file `bach.wav` located in the `audio` directory.