Class Project: EE2800 – Digital Signal Processing

January-May 2025

Instructor: Sundar Vanka

Due: April 19, 2025, 11:59 pm

Talk and Demo: April 26, 2025 in A-220 (9:30 am - 1:30 pm)

INSTRUCTIONS

Expectations and Ground Rules

- **This project contributes 30% to your overall score**.
- The project involves the **design and implementation of DSP algorithms** to solve real-life problems, albeit in a vastly simplified form.
- To mimic real world design problems:
  - The problem statement is framed almost completely in words, not as equations or formulas.
  - The problem is **open-ended**, encouraging **experimentation**, **independent learning**, with more than one "correct" way and several "wrong" ways to approach the design.
- Teams are free to look up any books/papers or other resources they may require. **However, only textbooks and peer-reviewed research papers are acceptable final references.**
- Each team must be able to demonstrate their effort and understanding by **answering detailed technical questions** about their design. This includes, but is not limited to, **explaining their code step-by-step, justifying their design choices and the trade-offs considered, and answering followup questions**.
- **Failure to do so may be considered a case of academic misconduct and the team will be awarded a failing grade in the course per the academic plagiarism clause in the institute rules.**
- Teams can approach the instructor to get inputs about the project and/or their design. The usual means of contacting apply. No penalties for such advice.

Project Deliverables

- A **three-slide deck** summarizing your solution approach, design, along with justifications/tradeoffs involved in design choices.
- **Matlab code that implements this design** (may include one or more m-files, should work on institute-supplied Matlab installations)

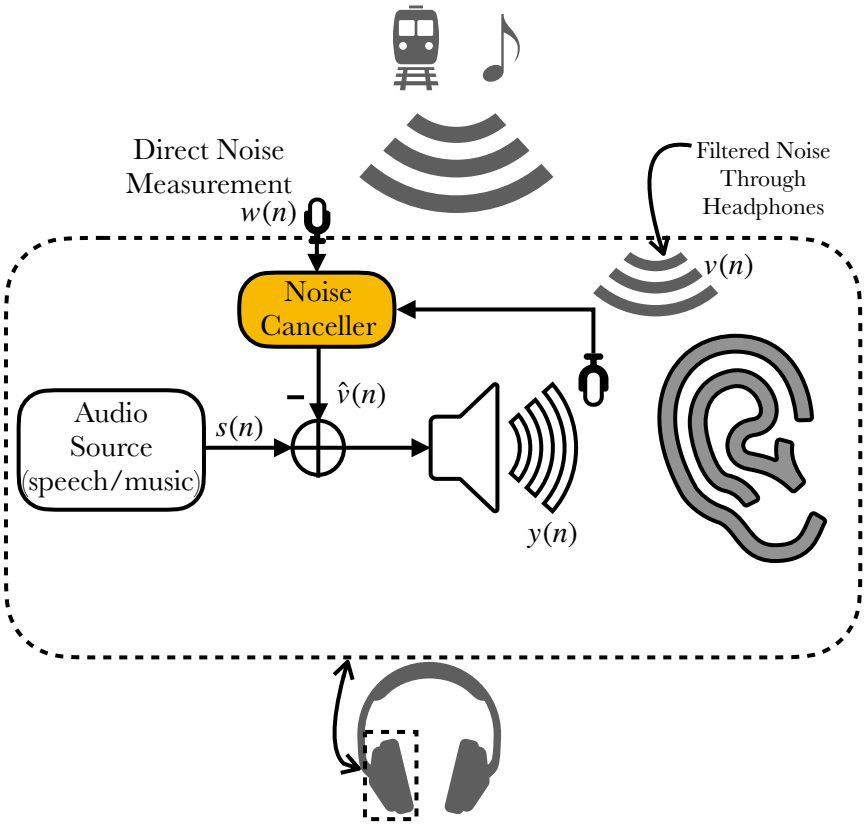
- **No Matlab built-in functions must be used in the code, except to implement static FIR/IIR filters.**
- **The above material should be uploaded as a SINGLE tarball per team**. The name of the tarball should be named as `TEAM<team number>.tar`. For example, team 1's submission should appear as `TEAM1.tar`.
- **There is only ONE project submission per team, and exactly ONE of the team members must submit that assignment.** If there are multiple submissions against the same team number, only the first submission will be evaluated.

Evaluation

- Time & Venue: **9:30 am - 1:30 pm on April 26 (Sat) in A-220.**
- Each team gets 15 min:
  - **Presentation + Q&A: 4 min. + 6 min.**
  - **Demo (on TA's computer running institute-supplied Matlab): 5 min.**
- Slots listed in the xls shared with this project on Google Classroom. **Submissions of teams that do not show up during the evaluation slot will not be graded.**

Project Theme: DSP Systems that Learn, aka Adaptive DSP: This course has focused primarily on *linear, time-invariant* systems. As discussed in the class, there are several practical scenarios where time-invariance does not hold, e.g., in systems that *learn* an unknown (and often time-varying) vector (e.g., the impulse response of an unknown system). We will explore this idea through *noise cancellation* algorithms widely deployed in noise-canceling ear/headphones. You will design simple DSP algorithms for such noise cancelers, exploring classical methods (e.g., fixed filters) and/or *adaptive* filters (loosely put, "linear" machine learning methods) as needed to achieve the design objective below.

Problem Statement: You are given a file `external_noise.txt` containing the sound sequence $w(n)$ reaching the headphone cup (see figure). The file `noisy_speech.txt` consists of a clean speech signal $s(n)$ contaminated by the leakage noise $v(n)$ coming through the cup. Using just these two signals, design a **programmable**, **selective noise canceller** (see figure, orange). Its output $\hat{v}(n)$ is needs to contain undesirable noise components that are to be "pre-subtracted" from $s(n)$ and fed to a speaker. The leakage noise $v(n)$ adds back this undesired noise, resulting in completely removing undesirable noise!



A block diagram showing how noise suppression works. When the noise canceler (orange) is turned off, $\hat{v}(n) \equiv 0$, so the user hears $s(n) + v(n)$. With the canceler turned on, the sound entering the user's ear is $s(n) + v(n) - \hat{v}(n)$, where $v(n) - \hat{v}(n)$ is the residual noise. In full suppression, the canceler learns to approximate $v(n)$ using the direct noise measurements $w(n)$ and its previous approximation errors. The design should ensure $\hat{v}(n) \approx v(n)$, so the noise reaching the ear is very small. With partial suppression, this approximation applies to the undesired components of $v(n)$.

Noise Composition:

- The external sound $w(n)$ and its leakage $v(n)$ may contain one or more **tone-like noise sources** with **time-varying amplitudes** and **programmable frequencies**. The user supplies these frequencies.
- The external sounds may contain **non-tonal** sounds that may be non-stationary.
- The user should be able to enable or disable tonal noise suppression separately from other external sounds suppression.

Design Objective:

- The residual noise sequence is difference between leakage noise $v(n)$ and the cancelled noise $\hat{v}(n)$ (see figure). It is this residual noise needs to meet certain criteria.
- In the *full suppression mode*, $\hat{v}(n)$ must cancel all components of $v(n)$ while preserving the speech signal $s(n)$ as much as possible.
- In the *partial suppression mode*, the tonal noise components of $v(n)$ are to be retained post-cancellation, while the non-tonal external sounds are to be suppressed.
- For full suppression, the algorithm should maximize post-cancellation Signal-to-Noise Ratio (SNR) (as measured over the entire duration of the speech sample) with respect to the pre-cancellation SNR. The clean speech signal $s(n)$ in a file `clean_speech.txt` is provided for SNR measurement purposes. SNR gains of $\sim$15 dB or better are possible depending on your design in the full suppression mode.
- For partial suppression, you are required to show the near absence of non-tonal noise after the learning algorithm converges. You are expected to propose one or more metrics to measure this and justify their use.

Design Guidelines:

- All signals are sampled at 44.1 kHz.
- You may want to assume that the tonal noise power is the same or more about the non-tonal noise power. Both of them are typically larger than the average clean speech power.
- To mimic real-time processing, you cannot assume the all the samples in the file are available at once; rather, the design should ideally buffer some input samples to produce each output sample in $\hat{v}(\cdot)$. The size of this input buffer should be as small as possible to reduce latency and computation burden, while providing significant noise suppression.
- After a change in noise conditions, noise suppression should return to the target suppression level as soon as possible.

Note: The files mentioned above are only to assist you in system design. Your final design should:

1. Have programmable frequencies for tonal interference. The number of locations should be at least one, with the upper limit depending on your design.
2. Not make any assumptions about the frequency content of the clean speech signal or the non-tonal noise signals, except that 44.1 kHz is above their Nyquist rate.
3. Function with other speech and/or external sound files.

Useful Concepts:

- Basics of: Discrete Time Stochastic Processes, Wiener Filter, the Gradient Descent Algorithm, Least Mean Square (LMS) and the Normalized LMS Algorithms.
- Basics of discrete time stochastic processes will be covered in classroom lectures. You are encouraged to read up the rest using the very accessible resources below.
- Some topics such as gradient descent and optimality in the MSE sense may be familiar to you from a machine learning course. LMS is very close to stochastic gradient algorithms encountered in basic ML courses.

Useful Book References:

- M. H. Hayes, "Statistical Digital Signal Processing and Modeling", John Wiley & Sons, 1996. Chapter 3 (3.1-3.4), Chapter 7 (7.1, 7.2), Chapter 9 (9.1-9.2.6).
- Paulo S. R. Diniz, "Adaptive Filtering: Algorithms and Practical Implementation", 5/e, Springer, 2020. Chapters 1, Chapter 2 (2.1-2.4), Chapter 3 (3.1-3.4).