EE 310 Signals and Systems

Project Guide

Spring Semester 2026

Department of Electrical Engineering

Important Information

- **Group Size:** 3 students per group
- **Duration:** 5 weeks
- **Project Choice:** Select ONE project from two options
- **Submission:** Code, Report (IEEE format), Demo Video, Viva
- **Weightage:** 15% of total course grade

Contents

1 Overview

This document presents Acoustic Echo Cancellation (AEC) project designed to help you apply the core concepts of EE 310 to real-world systems. Working in teams of three, you will design, implement, test, and demonstrate a complete signal processing solution, and evaluate its performance under realistic conditions.

Upon successful completion of the project, you will be able to:

- Apply core EE 310 concepts (e.g., filtering, correlation, system behavior, and frequency-domain analysis) to a practical problem
- Implement and test signal processing algorithms in MATLAB or Python
- Evaluate system performance using appropriate quantitative metrics
- Analyze practical limitations such as noise, reverberation, latency, and hardware constraints
- Work effectively in a group and communicate technical results professionally

2 Project: Acoustic Echo Cancellation

2.1 Problem Statement

In hands-free audio communication systems (e.g., laptops, phones, and video conferencing), the microphone often picks up sound from the loudspeaker, creating an acoustic echo. This echo is heard by the remote listener and significantly degrades communication quality.

Your task is to design and implement an Adaptive Echo Canceller (AEC) that estimates the echo path in real time and subtracts the estimated echo from the microphone signal.

The system should be robust to:

- Background noise
- Time-varying acoustic environments
- Double-talk scenarios (when both near-end and far-end speakers talk simultaneously)
- Unknown room impulse responses

2.2 Technical Background and Formulation

This project builds on several core concepts from EE 310, including discrete-time signals, LTI systems, convolution, system identification, and adaptive filtering. In AEC, the objective is to model the acoustic path between the loudspeaker and microphone, and then remove the resulting echo from the microphone signal.

1) Signal Model (What is happening physically?)

Let:

- $x(n)$ be the far-end signal played through the loudspeaker (reference signal)
- $h(n)$ be the unknown acoustic echo path (room/speaker/microphone path)
- $s(n)$ be the near-end speech (local speaker)
- $v(n)$ be background noise

The echo generated at the microphone is the convolution of the loudspeaker signal with the acoustic path:

$$y_{\text{echo}}(n) = (h * x)(n) \tag{1}$$

The microphone signal (desired signal for the adaptive filter) is:

$$d(n) = y_{\text{echo}}(n) + s(n) + v(n) \tag{2}$$

In words: the microphone contains *echo + near-end speech + noise*. The goal of AEC is to estimate and remove only the echo component.

2) Echo Path as an LTI System

The acoustic path can be approximated as an LTI system over short durations. Its impulse response $h(n)$ represents:

- direct path from speaker to microphone
- reflections from walls, table, and surrounding objects

- attenuation and delay effects

Because the exact room impulse response is unknown (and may change with movement), we estimate it using an adaptive FIR filter.

3) Adaptive FIR Filter Formulation

We model the echo path using an adaptive FIR filter with coefficient vector:

$$\mathbf{w}(n) = [w_0(n)\ w_1(n)\ \cdots\ w_{N-1}(n)]^T$$

Define the reference signal buffer (most recent N samples) as:

$$\mathbf{x}(n) = [x(n)\ x(n-1)\ \cdots\ x(n-N+1)]^T$$

The estimated echo is:

$$\hat{y}_{\text{echo}}(n) = \mathbf{w}^T(n)\mathbf{x}(n) \tag{3}$$

The AEC output (error signal) is:

$$e(n) = d(n) - \hat{y}_{\text{echo}}(n) \tag{4}$$

If the adaptive filter learns the echo path well, then $\hat{y}_{\text{echo}}(n) \approx y_{\text{echo}}(n)$, and the error signal mainly contains near-end speech and residual noise:

$$e(n) \approx s(n) + v(n)$$

4) Objective of Adaptation

The adaptive filter updates $\mathbf{w}(n)$ to minimize the echo residual in the error signal. A common objective is to minimize the mean squared error (MSE):

$$J = \mathbb{E}[e^2(n)], \tag{5}$$

where $\mathbb{E}$ is the expectation (you can treat it as an average). Since the true echo path changes over time (movement, changing room conditions), the filter must:

- *converge* quickly to a good estimate, and
- *track* changes in the acoustic path

This is why adaptive filtering (rather than fixed filtering) is required.

5) Practical Conditions in AEC

Your design must account for the following practical scenarios:

- Far-end only (single-talk): Best condition for adaptation; the filter can learn the echo path effectively.
- Near-end only: No useful reference-to-microphone echo relationship; adaptation should ideally be slowed or frozen.
- Double-talk: Both far-end and near-end are active. If adaptation continues aggressively, the filter may diverge.
- Background noise / reverberation: These reduce estimation accuracy and slow convergence.

For this reason, practical AEC systems often include a basic double-talk detector and carefully chosen adaptation parameters.

6) Performance Measures (What you will evaluate)

To evaluate your AEC system, you will measure how much echo is reduced and how well the adaptive filter estimates the true echo path. Common measures include:

- ERLE (Echo Return Loss Enhancement): How much the echo power is reduced
- Misalignment: Difference between the true and estimated echo-path coefficients (simulation)
- Convergence behavior: How quickly the system reaches stable performance

Summary

In this project, you will use:

- convolution and LTI system modeling (echo path),
- FIR filtering and vector signal representation,
- adaptive filtering for system identification,
- performance analysis using signal-based metrics.

The next section introduces the required adaptive algorithm (NLMS) and the implementation parameters for your project.

2.3 Technical Requirements

Core Algorithm: You must implement the Normalized Least Mean Squares (NLMS) adaptive filter yourself (i.e., no built-in adaptive filter functions).

The NLMS update equation is given by:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\mu}{\|\mathbf{x}(n)\|^2 + \delta}\, e(n)\, \mathbf{x}(n), \tag{6}$$

where:

- $\mathbf{w}(n)$: adaptive filter coefficients ($N \times 1$ vector)
- $\mathbf{x}(n)$: far-end reference signal buffer (input vector)
- $d(n)$: microphone (desired) signal
- $e(n) = d(n) - \mathbf{w}^T(n)\mathbf{x}(n)$: error signal (AEC output)
- μ: step size parameter ($0 < \mu < 2$)
- δ: regularization parameter (prevents division by zero)

Recommended Implementation Parameters:

Note: You may tune these parameters based on your setup, but you must justify your final choices in the report.

Parameter	Recommended Range
Sample Rate	8000 – 16000 Hz
Filter Length (N)	512 – 2048 taps
Step Size (μ)	0.3 – 0.7
Block Size (real-time)	256 – 1024 samples
Regularization (δ)	10^{-6} – 10^{-4}

Table 1: Recommended NLMS Parameters

2.4 Deliverables

2.4.1 Part A: Simulation Study

Implement a simulation framework to test and evaluate your AEC system under controlled conditions.
Required components:

1. Synthetic Echo Path: Generate a realistic room impulse response (e.g., exponentially decaying FIR model)
2. Signal Generation: Create:
   - far-end signal (speech or speech-like signal),
   - near-end signal, and
   - background noise
3. Performance Evaluation: Compute and plot:
   - ERLE (Echo Return Loss Enhancement) vs. time
   - Misalignment between true and estimated echo path
   - Convergence behavior (including convergence time)
4. Robustness Testing: Vary SNR from 0 to 30 dB and analyze performance trends

ERLE Definition: (In practice, compute ERLE over a short sliding window (e.g., 100–500 ms) for a stable time-varying plot)

$$\mathrm{ERLE}_{dB} = 10\log_{10}\left(\frac{\sum_n |\mathrm{echo}(n)|^2}{\sum_n |\mathrm{echo}(n) - \hat{\mathrm{echo}}(n)|^2 + \epsilon}\right) \tag{7}$$

where ϵ is a small constant to avoid division by zero.

Target performance:

- ERLE $>$ 15 dB: Acceptable
- ERLE $>$ 25 dB: Excellent

What to present at this stage:

- Working simulation code
- Plots for all required metrics
- Brief analysis of parameter choices and observed behavior

2.4.2 Part B: Real-Time Demo

Develop a real-time AEC implementation and demonstrate audible echo reduction using actual audio hardware.

Required functionality:

1. Use actual hardware (laptop microphone/speakers or external USB microphone)
2. Process audio in blocks with acceptable latency (< 100 ms)
3. Demonstrate audible echo reduction
4. Include a basic double-talk detection mechanism (Geigel detector recommended)

Demo video (3–5 minutes) must include:

- Echo before cancellation
- Echo after cancellation
- System behavior during double-talk

What to present/submit:

- Real-time demo code
- Recorded demonstration video
- Brief note on hardware setup and observed latency/performance

2.4.3 Part C: Report and Analysis

Submit a professional IEEE-format report (template will be provided) documenting your design, implementation, and results.

Required report sections:

1. Introduction: Problem motivation and background
2. Theory: NLMS algorithm, formulation, and key properties
3. Methodology: Implementation details, parameter selection, and design choices
4. Results: Simulation and real-time performance (with plots and observations)
5. Discussion: Analysis of results, limitations, and possible improvements
6. Conclusion: Summary of achievements
7. References: At least 5 relevant sources (IEEE citation style)

Expected quality:

- Clear technical writing
- Properly labeled figures and axes
- Reproducible methodology and parameter settings
- Critical discussion (not just screenshots/plots)

2.5 Hardware Requirements

Minimum (available to all groups):

- Laptop with built-in microphone and speakers
- MATLAB R2020a+ or Python 3.8+

Recommended (optional):

- USB microphone (for more consistent input quality)
- Headphones (recommended during testing)

Important note: For real-time testing, disable built-in audio enhancements (if possible), such as automatic gain control (AGC), noise suppression, or built-in echo cancellation, as these can affect your AEC results.

2.6 Tools and Libraries

Allowed: Standard DSP operations and libraries for:

- FFT and filtering
- Audio input/output
- Plotting and analysis

Not Allowed: Built-in adaptive filter functions (you must implement NLMS yourself)

Suggested software tools:

- MATLAB: `audioDeviceReader`, `audioDeviceWriter`
- Python: `sounddevice`, `numpy`, `scipy`

3 Project Grading and Checkpoint Schedule

The project grade is based on checkpoint performance, final viva, final report quality, and overall code quality/documentation. Code quality and documentation will be assessed continuously across checkpoints and finalized in Week 12.

- All group members must be present at every checkpoint.
- Any group member may be asked questions during any checkpoint.
- Individual understanding will be assessed in detail during the final viva.

Note for Students: Parts in previous section serve as Requirements, Checkpoints here serve as Timeline. Parts describe *What*, and Checkpoints describe *When & How.*

3.1 Checkpoint 1: Algorithm Implementation (Week 9) — 15%

Focus: Core algorithm implementation and basic functionality
Requirements:

- Present a working implementation of the core algorithm: NLMS adaptive filter
- Test the implementation with synthetic/controlled signals
- Show at least one plot demonstrating basic functionality
- Explain initial parameter choices

Expected deliverables / evidence:

- Checkpoint code (current version)
- 1–2 plots showing basic operation
- Brief checkpoint summary (slides or notes)

Mapping: Part A (Started)

3.2 Checkpoint 2: Simulation Results (Week 10) — 20%

Focus: Simulation study, performance metrics, and analysis
Requirements:

- Complete the simulation framework
- Compute all required performance metrics
- Evaluate the system under multiple scenarios
- Present clear plots and a brief analysis of results
- Discuss parameter tuning and observations

Expected deliverables / evidence:

- Simulation code and generated results
- Metric plots and/or summary tables
- Draft report sections (Theory / Methodology / Initial Results)

Mapping: Part A (Complete)

3.3 Checkpoint 3: Real-Time Demo (Week 11) — 25%

Focus: Hardware-based real-time system demonstration
Requirements:

- Demonstrate a working real-time system using actual hardware
- Present a live demonstration during the checkpoint
- Show system behavior under realistic conditions
- Discuss implementation challenges (e.g., latency, noise, stability, calibration)

Expected deliverables / evidence:

- Real-time demo code
- Live demonstration during checkpoint
- Draft/final demo video recording
- Brief hardware setup summary

Mapping: Part B (Complete)

3.4 Checkpoint 4: Final Viva (Week 12) — 20%

Focus: Individual and team understanding of the complete project
Requirements:

- Defend the theory, implementation, and results
- Answer technical and design-related questions
- Explain design choices, limitations, and improvements
- Demonstrate individual understanding of the project

Expected deliverables / evidence:

- Oral defense (all group members)
- Technical discussion of the full project

Mapping: Part C (Complete)

3.5 Final Report (IEEE Format, Week 13) — 15%

Focus: Professional technical documentation
Requirements:

- Submit a complete IEEE-format report
- Include theory, methodology, results, and discussion
- Present clear figures, tables, and references

Expected deliverables / evidence:

- Final report PDF (IEEE format)

Mapping: All Parts (Complete)

3.6 Code Quality and Documentation (Assessed Across All Weeks, Finalized in Week 13) — 5%

Focus: Code organization, readability, and reproducibility
Requirements:

- Write modular, readable, and well-organized code
- Include meaningful comments and documentation
- Ensure reproducible execution with clear instructions

Expected deliverables / evidence:

- Final code package
- README with setup and execution instructions
- Supporting files/data needed to reproduce results

4 Submission Guidelines

4.1 Code Submission

Directory Structure:

```
ProjectX_GroupY/
 src/
    main_algorithm.m (or .py)
    simulation.m
    realtime_demo.m
    helper_functions/
 data/
    test_signals/
    results/
 docs/
    final_report.pdf
    checkpoint_presentations/
 videos/
    demo_video.mp4
 README.md
```

Code Requirements:

- Well-commented (explain non-obvious sections)
- Modular design (functions for each component)
- README with setup and execution instructions
- Include sample data for reproducibility

4.2 Report Format

IEEE Two-Column Format:

- Use IEEE conference template (to be provided)
- Length: 8-12 pages (including figures and references)
- Figures: High quality, properly captioned
- Equations: Numbered and referenced in text
- References: IEEE citation style, minimum 5 sources

Report Sections:

1. Abstract (150-200 words)
2. Introduction with motivation
3. Theoretical Background
4. System Design and Implementation
5. Experimental Setup and Methodology
6. Results and Discussion
7. Conclusion and Future Work
8. References

4.3 Demo Video

Requirements:

- Duration: 3-5 minutes
- Resolution: Minimum 720p
- Content: System overview, live demo, results discussion
- Audio: Clear narration explaining what is happening
- Format: MP4 or AVI

4.4 Submission Method

- **Platform:** Upload to designated course portal
- **Deadline:** End of Week 12, 11:59 PM (*Tentative*)
- **Late Policy:** 10% deduction per day (max 3 days)
- **File Size:** Maximum 100 MB (compress if needed)

5 Group Formation and Collaboration

5.1 Group Registration

- Form groups of exactly 3 students
- Group Confirmation **End of Week 7**
- **Recommended:** Designate one group leader

5.2 Collaboration Guidelines

Within Group:

- Divide work equitably among members
- Suggested roles: Algorithm Specialist, System Integration, Documentation
- Hold regular team meetings (document in meeting logs)
- Use version control (Git recommended)

Between Groups:

- Discussion of concepts and theory: **Allowed**
- Sharing code or implementation details: **NOT Allowed**
- Helping with debugging: **Limited** (instructor approval required)

Use of External Resources:

- Textbooks, papers, online tutorials: **Allowed** (cite properly)
- Library functions for basic operations: **Allowed**
- Copy-pasting large code blocks: **NOT Allowed**
- Using built-in adaptive filters or DOA estimators: **NOT Allowed**

5.3 Academic Integrity

Warning

Plagiarism or unauthorized collaboration will result in:

- Zero grade on the project
- Report to academic affairs
- Possible course failure

We use plagiarism detection tools. Your code and report will be checked against:

- Online repositories (GitHub, Stack Overflow, etc.)
- Published papers and textbooks

6 Frequently Asked Questions

6.1 General

Q1: Can we use programming languages other than MATLAB/Python?
A: No. Projects must be in MATLAB or Python for consistency in grading.

Q2: What if we finish early?
A: Great! Add optional extensions like frequency-domain processing, multiple source localization, or machine learning approaches. Document these in your report.

6.2 Project Specific

Q3: How do we test AEC without annoying echo?
A: Use headphones during development. Record reference and microphone signals separately, then process offline. Real-time testing comes later.

Q4: My ERLE is only 10 dB. Is that acceptable?
A: For Checkpoint 2, yes. For final submission, aim for >15 dB. Check filter length and step size if performance is poor.

Q5: Do we need to handle double-talk perfectly?
A: No. Basic Geigel detector that freezes adaptation during double-talk is sufficient.

6.3 Technical

Q6: Can we use GPU acceleration?
A: Yes, if you document it. Not required or expected.

Q7: What sample rate should we use?
A: 16 kHz is recommended. Lower (8 kHz) is acceptable but may reduce quality.

Q8: How do we handle real-time buffer underruns?
A: Increase buffer size or reduce processing load. Document any latency issues.

Success Tips

1. **Start Early:** Don't wait until Week 8 or so to start coding
2. **Test Incrementally:** Verify each component before moving on
3. **Commit Regularly:** Use Git and commit frequently if you want
4. **Ask Questions:** Use office hours and TA support
5. **Document as You Go:** Don't leave report writing for the end
6. **Practice Demo:** Rehearse your presentation and demo

Appendix A: Quick Start Guides

A.1 MATLAB Setup

```
% Test audio I/O
deviceReader = audioDeviceReader(16000, 512);
deviceWriter = audioDeviceWriter(16000, 512);

% Record 1 second
audioIn = deviceReader();
deviceWriter(audioIn);

release(deviceReader);
release(deviceWriter);
```

A.2 Python Setup

```
# Install libraries
pip install numpy scipy matplotlib sounddevice soundfile

# Test audio I/O
import sounddevice as sd
import numpy as np

# Record 1 second
fs = 16000
audio = sd.rec(int(1 * fs), samplerate=fs, channels=1)
sd.wait()
sd.play(audio, fs)
sd.wait()
```

A.3 Extra

MATLAB:

```
info = audiodevinfo;
% Look for InputChannels >= 2
```

Python:

```
import sounddevice as sd
print(sd.query_devices())
# Look for max_input_channels >= 2
```

For questions or clarifications, please use project channel on Slack.
Last Updated: February 2026