Specifications

Digital Signal Generator and Visualizer

Due Date: 23 October 2025
Deliverables: Source Code + Specification Report (uploaded separately to Gradescope)

1. Overview

This project implements a **Digital Signal Generator** capable of producing, encoding, modulating, decoding, and visualizing digital and analog signals.
It supports **line encoding schemes** (NRZ-L, NRZ-I, Manchester, Differential Manchester, AMI) and **scrambling techniques** (B8ZS, HDB3).
Additionally, **PCM (Pulse Code Modulation)** and **Delta Modulation (DM)** are integrated for analog-to-digital conversion.
A **C++ MathGL-based visualizer** displays the encoded signal waveform.

2. Languages and Libraries Used

Languages:

- Python 3.10+ — for signal generation, encoding, modulation, and decoding.
- C++ (MathGL + FLTK) — for waveform visualization.

Python Libraries:

- numpy — numerical operations and signal generation.
- subprocess — executing external visualization.
- math — mathematical utilities for decoding.

C++ Libraries:

- mgl2/mgl.h, mgl2/fltk.h — MathGL and FLTK for plotting and GUI visualization.
- Standard Template Library (<vector>, <fstream>, <iostream>).

3. Assumptions Considered

- Sampling time for one bit is uniform.
- AMI encoding alternates polarity for logic '1' and maintains zero level for logic '0'.
- PCM quantization uses **8 uniform levels** between ±Amplitude.
- Delta Modulation uses a fixed step size (2*amp/samples).
- Scrambling applies only when AMI is selected.
- Input validation ensures binary bits or analog waveform parameters are valid.
- Visualization expects signal.txt and encoding_type.txt generated by Python.

4. How to Run the Code

Step 1: Compile the Visualizer (C++)

```
g++ visualizer.cpp -lmgl -lmgl-fltk -o visualizer
```

Step 2: Run signal generator(python)

python3 signal_generator.py(ubuntu)
python signal_generator.py(windows)