COMMUNICATIONS RELIABILITY CHALLENGE DOCUMENTATION

Name: Harshvardhan Gaggar
Reg No: 245891390

GitHub Link: https://github.com/its-harsh-here/CubeSat

Technical Report is uploaded on github (under "/docs" folder) along with source file and decoded bitstream.

Resources Used:

I mostly used gpts for research rather than google as they were faster and just as accurate thus I havnt cited many resources, but when required I could attach the resources the ai agents used.

- ChatGPT: https://chatgpt.com/share/68baeae5-50cc-8003-ae2e-ea4eaa14b821
- Perplexity AI: https://www.perplexity.ai/search/my-understanding-i-am-supposed-ELUbBXvKSjiV1k_HzdJfOQ
- BFSK(GeeksforGeeks): https://www.geeksforgeeks.org/electronics-engineering/bpsk-binary-phase-shift-keying/s

Approach:

What I understood after reading the document: I am supposed to take bitstreams from rx.npy, decode it with the help of meta.json and save the decoded bits in decoded_bits.npy while only using scipy and numpy to implement the code. Then I am supposed to use run evaluation script to generate BER/FER metrics.

After this I used Perplexity AI to verify by understanding and how to implement it. Do note that I have vibe coded the whole project(I believe that was the whole point of the project since we didn't have enough time + question wasn't really very easy).

Do note that this Document along with technical report have been made by me with whatever I understood and learned while implementing this project.

So we have 4 phases with different types of noise in the received bit stream. Since we had to preserve the hierarchy of the dataset, and I wanted to keep my answer clean, I made a src directory where I kept all the codes. I recursively went into each folder and recovered bitstream by saving it in decoded_bits.npy as specified by the question.

COMMUNICATIONS RELIABILITY CHALLENGE DOCUMENTATION

Challenges/Solutions:

- **Not Understanding how to implement this answer:**
  I thought about how anyone would/could implement it without coding feasibility. So I had a rough I idea and was able to understand the question and the steps I would need to follow to be able to solve this question
- **How to make one code work for all files:**
  So my first working code faced an issue of only working for phase 1,2 and 4, not for phase 3 because it had one extra directory. I was able to solve this problem by thinking of directories as trees and recursively go in each folder until you find rx.npy and meta.json.
- **Code from GPT didn't work as expected**
  Followed up with many questions and suggestions and tried to debug the solution on my own -> I was able to understand what each function does and how the whole program works on a broad level, unfortunately not the inner workings of the code.
- **Time**
  I didn't have a lot of free time mostly due to lab mid sems(+ record file) as well as club orientation(I am in the cc of codex).

Implementation:

For phase 1: I could have used Gardner Timing Error or Mueller-Muller timing recovery algorithm. I choose to go with Gardner Timing Error since it is said to have better performance. Very honestly I didn't have enough time to go into the dept of this algorithm(because of lab mid sems + club orientation) but I did read a little about it.

For phase 2: Was built on phase 1 with signal/noise power estimation and scaling.

For phase 3: I found that there are two ways I could correct bitstream here. Either by Reed-Solomon RS(15, 11) over GF(16) along with Viterbi decoding.

For phase 4: I used FFT-based estimation cause that's what was the best.