[redacted] Programming Assignment on Linear Codes

This is a programming assignment which requires you to encode and decode binary message bits using linear block codes.

Input:

- A text file of size 1 KB of your choice.

Ingredients:

1) **Linear code 1**: Repetition code of size n with a $1 \times n$ generator matrix $G = [1\ 1\ 1\ \ldots\ 1]$ with $n = 3$.
2) **Linear code 2**: A $(2^m - 1, 2^m - 1 - m)$ binary Hamming code with $m = 3$.

Computing Environment:

- Matlab

Experiment 1:

1) Read the input text-file and convert it to a binary string, say of length M bits.
2) Generate a random binary error pattern of length M with hamming weight d such that the non-zero entries are **uniformly distributed** across M bits.
3) XOR the above error pattern with the message bits to obtain a new sequence denoted by y.
4) Using y, retrieve the text-file.
5) In the reconstructed file, compute the percentage of modified characters with respect to the input file.
6) Repeat the above experiment by varying the value of $d = \{10, 100, 200, 500, 5000\}$.

Experiment 2:

1) Read the input text-file and convert it to a binary string, say of length M bits.
2) Divide the input string into several chunks such that each chunk is of size k bits.
3) Encode each chunk into a sequence of n bits by using one of the above codes. After encoding, let the total number of bits generated from the entire text-file be M'.
4) Generate a random binary error pattern of length M' with hamming weight d such that the non-zero entries are **uniformly distributed** across M' bits.
5) XOR the above error pattern with the encoded bits to obtain a new sequence denoted by y.
6) Using y, retrieve the text-file by using the (i) bounded distance decoder, and (ii) the complete decoder.
7) Compute the number of errors you could detect, and the number of errors you could correct.
8) In the reconstructed file, compute the percentage of modified characters with respect to the input file.
9) Repeat the above experiment by varying the value of $d = \{10, 100, 200, 500, 5000\}$.
10) Compare the results between **Linear code 1** and **Linear code 2**.

Experiment 3:

1) Read the input text-file and convert it to a binary string, say of length M bits.
2) Divide the input string into several chunks such that each chunk is of size k bits.
3) Encode each chunk into a sequence of n bits by using one of the above codes. After encoding, let the total number of bits generated from the entire text-file be M'.
4) Generate a random binary error pattern of length M' with hamming weight d such that the non-zero entries are **appear in burst** (in successive positions).

5) XOR the above error pattern with the encoded bits to obtain a new sequence denoted by y.
6) Using y, retrieve the text-file by using the (i) bounded distance decoder, and (ii) the complete decoder.
7) Compute the number of errors you could detect, and the number of errors you could correct.
8) In the reconstructed file, compute the percentage of modified characters with respect to the input file.
9) Repeat the above experiment by varying the value of $d = \{10, 100, 200, 500\}$.
10) Compare the results between **Linear code 1** and **Linear code 2**.

Experiment 4:

1) Read the input text-file and convert it to a binary string, say of length M bits.
2) Divide the input string into several chunks such that each chunk is of size k bits.
3) Encode each chunk into a sequence of n bits by using one of the above codes. After encoding, let the total number of bits generated from the entire text-file be M'.
4) Use a random permutation P on the encoded bits to rearrange the order of the bits.
5) Generate a random binary error pattern of length M' with hamming weight d such that the non-zero entries **appear in burst** (in successive positions).
6) XOR the above error pattern with the encoded bits to obtain a new sequence denoted by y.
7) Apply the inverse of the permutation operation P to obtain a new sequence denoted by y'
8) Using y', retrieve the text-file by using the (i) bounded distance decoder, and (ii) the complete decoder.
9) Compute the number of errors you could detect, and the number of errors you could correct.
10) In the reconstructed file, compute the percentage of modified characters with respect to the input file.
11) Repeat the above experiment by varying the value of $d = \{10, 100, 200, 500\}$.
12) Compare the results between **Linear code 1** and **Linear code 2**.