EECS 3213 : Assignment 1

In this assignment, you will write code that implements a CRC check, and simulate your code to demonstrate how it works. Perfectly completing parts 1 and 2 will give you enough marks for an A (16/20); also completing the (challenging) part 3 will earn you full marks (20/20).

Part of your grade is dedicated to precisely following the directions, and to the proper commenting and formatting of your code for readability.

Part 1 – CRC check implementation (12 marks). You will use the generator polynomial $g(x) = x^5+x^3+1$, with (n,k) = (20,15). Using MATLAB, Python, or the language of your choice, write the following two functions:

- crc_encode(x): This function encodes x with the generator polynomial g(x), following the procedure given in class. Parameter x is a length-15 vector of bits {0,1}. Returns a length-20 CRC-encoded vector of bits {0,1}.

- crc_decode(b): This function determines whether b passes the CRC, given the generator polynomial g(x). Parameter b is a length-20 vector of bits {0,1}. Returns 1 if b passes the CRC, 0 if not (i.e., 1 if b contains errors, 0 if not).

Part 2 – CRC check simulation (4 marks). Using the functions from part 1, write two simulation scripts as follows:

- sim_single(p, x): This script takes a 15-bit vector x as input and encodes x using the CRC encoder from part 1. Bits in the encoded vector are flipped (0 to 1 or 1 to 0) with probability p (see note below; since this is a random process, it is possible that there will be no flipped bits). The CRC decoder is run to determine whether the CRC detects the error. Finally, the script prints the following to the screen: original encoded vector; encoded vector with bit flips; whether or not there were errors present; and whether or not the CRC detected the errors. Example output below.

```
>> sim_single(0.05,[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0])
Original encoded vector
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Encoded vector with bit flips
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Errors present: YES
CRC detects errors: YES
```

- sim_multiple(p, j): Similar to sim_multiple, but performs j iterations. In each iteration, a random 15-bit sequence of bits is generated, where 0 and 1 are equally likely. The sequence is encoded and errors are introduced, as above. The script then counts the number of sequences where the CRC detected an error, and the number of sequences

where errors were present but the CRC did not detect the error. At the end, these counts are printed to the screen. **Nothing else should be printed.** Example output below.

```
>> sim_multiple(0.05,1000)
Sequences with errors: 550
Sequences with undetected errors: 0
```

Note: The following statement in MATLAB will generate a random bit in {0,1} where the probability of 1 is p, storing the result in b:

```
b = (rand(1) < p);
```

Part 3 – Arbitrary generator matrix (4 marks). Modify your encoder and decoder functions from part 1 to accept an arbitrary generator matrix. The functions should now be crc_encode(x,g) and crc_decode(b,g), where g is the generator matrix. The generator matrix should be specified as a vector, e.g. [1 0 1 1 1] = x^4+x^2+x+1. The code should be flexible enough for any pair (n,k), and should get these values from the lengths of x, b, and g.

Deliverables: Properly commented and formatted code as described above. If you implement part 3, submit functions that satisfy **both** parts 1 and 3.

Due date: November 7, 2017, at 11:59 PM, by e-mail: aeckford@yorku.ca