DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
R.V. COLLEGE OF ENGINEERING, BENGALURU -560059
(Autonomous Institution Affiliated to VTU, Belgaum)

ASSIGNMENT PROJECT REPORT ON

Submitted by

Name: UJWALAKAVYA J **USN: 1RV17CS435**

Submitted to

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
R.V.College of Engineering, Bangalore-59
R.V. COLLEGE OF ENGINEERING, BANGALORE – 560059

(Autonomous Institution Affiliated to VTU, Belgaum)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the Self Study report titled Error-detection-and-correction-using-2D-Parity-check is carried out by Ujwalakavya J **(1RV17CS435)** who is bonafide students of R.V College of Engineering, Bengaluru, in partial fulfillment for the award of degree of BE of the Visvesvaraya Technological University, Belgaum during the year **2018-2019**. It is certified that all corrections/suggestions indicated for the internal Assessment have been incorporated in the report deposited in the departmental library. The report has been approved as it satisfies the academic requirements in respect of assignment work prescribed for the course by the department for the said degree.

Maximum marks	**Marks obtained**
10	

Signature of Course In charge

TABLE OF CONTENTS

1. INTRODUCTION

Using multiple redundancy is expensive especially when huge amount of data needs to be transferred between the systems. Also when we require certain degree of error correction. So we adapt the idea of parity bits (which was cheaper) to not only detect but correct errors the idea is 2D parity check where we arrange the stream of bits we want to send in a two dimensional MxN array.

```
            N
   ---------------------
   | 1 1 0 1 .... 0 | 1
   | 0 0 0 1 .... 1 | 1
 M | ...            | .
   | ...            | .
|___________________|___
   | 1 1 0 0 .... 0 | 1
```

2D Parity

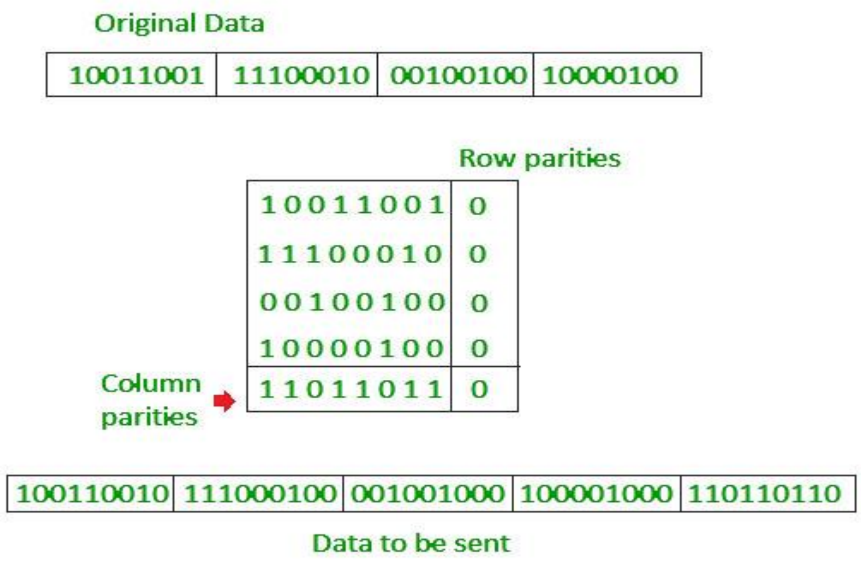
We make the same assumption as in 1D parity checking, namely that errors are rare, so that the chance of two errors occurring the same row or column is extremely low. In this scheme, we add a parity sum for each row and column. Now, if an error occurs, we detect a parity error in both a row and a column. This allows us to localize the bit which is in error, using far fewer bits. Since it is only a bit which is in error, we can simply flip the bit to correct the error.

This suffers from the same problems as with ordinary parity, i.e. that we cannot detect even numbers of errors in a row or column. On the other hand it has a huge advantage in efficiency compared to the multiple redundancy approach. If we define redundancy by the below formula:

$$R = \frac{\text{Number of bits used in full}}{\text{Number of bits in message}}$$

Then, as the length of the rows increases the redundancy becomes small. But we can't make the rows too large, or the change of errors increases.

A Sample diagram has been given below depicting how the 2D parity works and how the codewords are transmitted to the receiver.



2D Parity Example

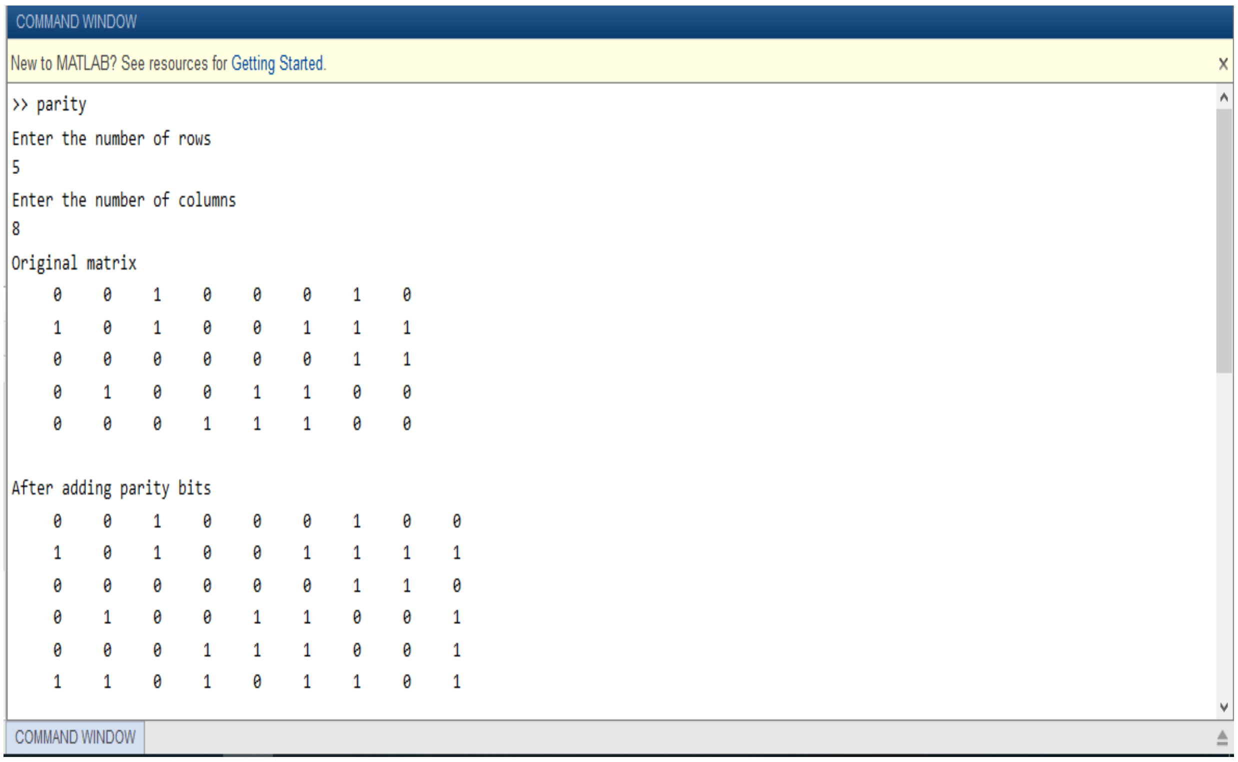
2. SOURCE CODE OF MATLAB PROGRAM

```
prompt = 'Enter the number of rows';
m = input(prompt);
prompt = 'Enter the number of columns';
n = input(prompt);
a =zeros(m+1,n+1);
for i= 1:(m)
   for j= 1:(n)
      a(i,j) = mod(randi(100),2);
   end
end
disp('Original matrix')
disp(a(1:m,1:n))
%Add parity bit for rows(even parity )
for i= 1:(m)
            count = 0;
            for j= 1:(n)
                        if a(i,j)==1
                                    count=count+1;
 end
 end
   %disp(count)
                        if mod(count,2) == 0
                                    a(i,(n+1)) = 0;
                        else
                                    a(i,(n+1)) = 1;
 end
end
```

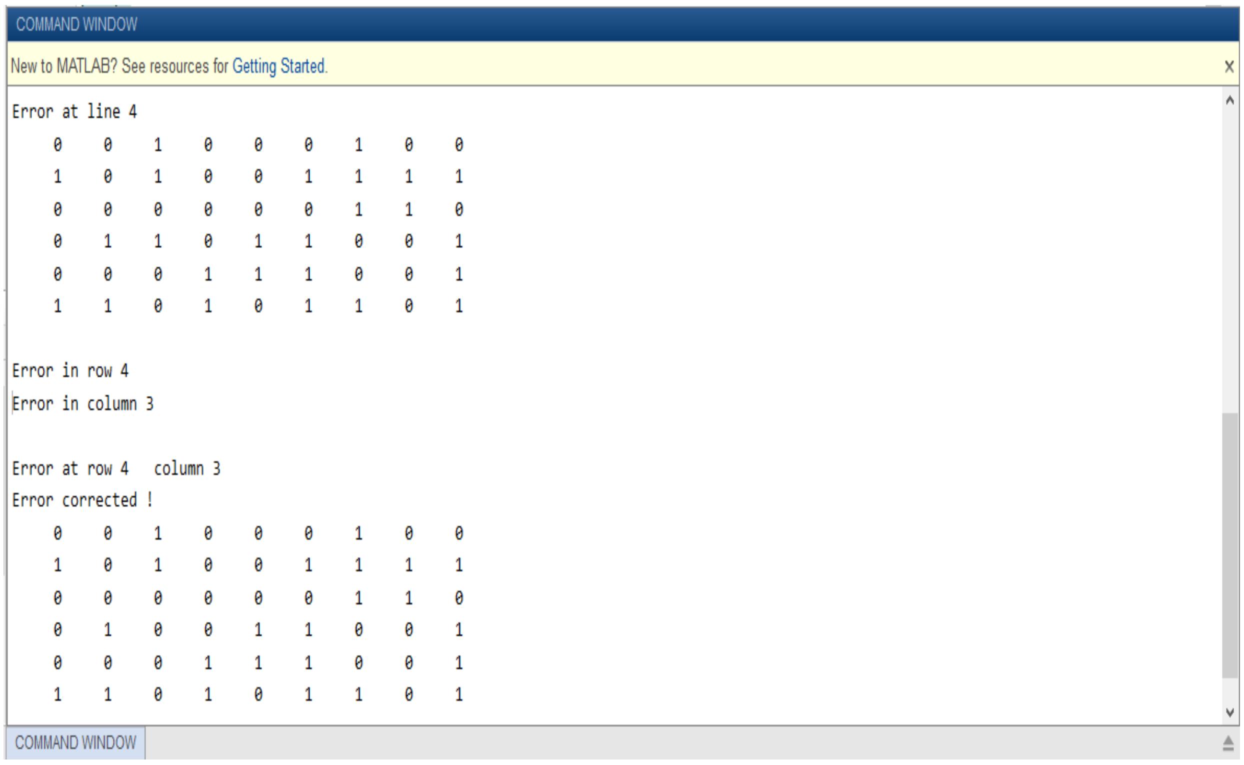
```
%Add parity bit for columns (even parity)
for i= 1:(n+1)
            count = 0;
            for j= 1:(m)
                        if a(j,i)==1
                                    count=count+1;
  end
 end
    %disp(count)
                        if mod(count,2) == 0
                                    a((m+1),i) = 0;
                        else
                                    a((m+1),i) = 1;
  end
end
disp('After adding parity bits')
disp(a)
% 1 or 2 bit error %
for i= 1:randi(2)
k1 = randi(m);
k2 = randi(n);
    if a(k1,k2)==0
         a(k1,k2)=1;
    else
        a(k1,k2)=0;
    end
 x = sprintf('Error at line %d',k1);
 disp(x)
end
 disp(a)

% Error detection code
row_er=0;
colm_er=0;
for i=1:m
 rcount = 0;
            for j=1:n
                        if a(i,j) == 1
                                    rcount=rcount+1;
  end
 end
 %x= sprintf('i= %d \n rcount= %d \t ccount=%d',i,rcount,ccount);
 %disp(x)
                if ((mod(rcount,2) == 0 && a(i,(n+1)) ~= 0) ||(mod(rcount,2) == 1 &&
a(i,(n+1)) ~= 1))
                    row_er=row_er+1;
                    r=i;
                        x=sprintf('Error in row %d',i);
                    disp(x);
                end
```

```
end
for i=1:n
 ccount = 0;

            for j=1:m
                if a(j,i) == 1
                    ccount= ccount+1;
                end
 end
                   if ((mod(ccount,2) == 0 && a((m+1),i) ~= 0) || (mod(ccount,2) == 1 &&
a((m+1),i) ~= 1))
                       colm_er=colm_er+1;
                       c=i;
                        x=sprintf('Error in column %d',i);
                        disp(x);
                    end
end
% 1 bit Error correction code
if( row_er==1)&&(colm_er==1)
    y=sprintf('\nError at row %d \t column %d',r,c);
    disp(y);
   if a(r,c)==1
       a(r,c)=0;
   else
       a(r,c)=1;
   end
   disp("Error corrected !")
    disp(a)
else
    disp('Number of bits corrupted = 2');
end
```

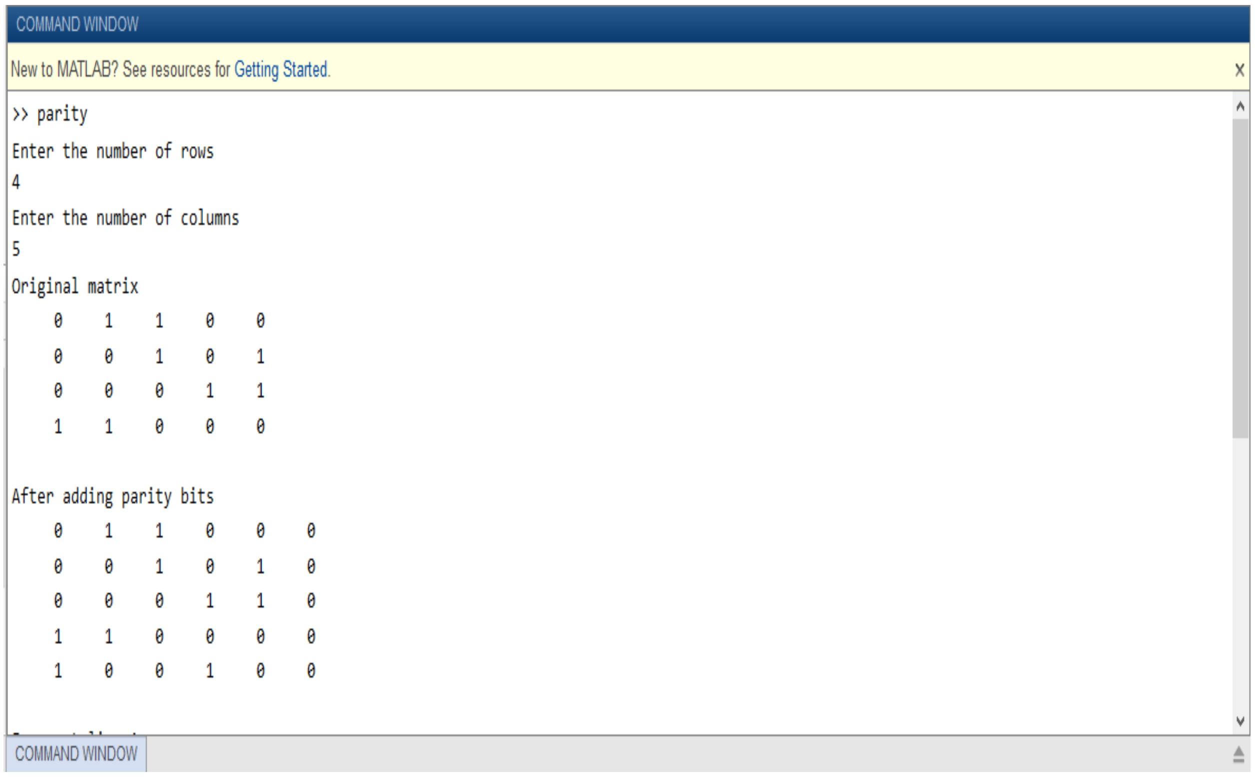
3. OBSERVATIONS, GRAPHS, RESULTS



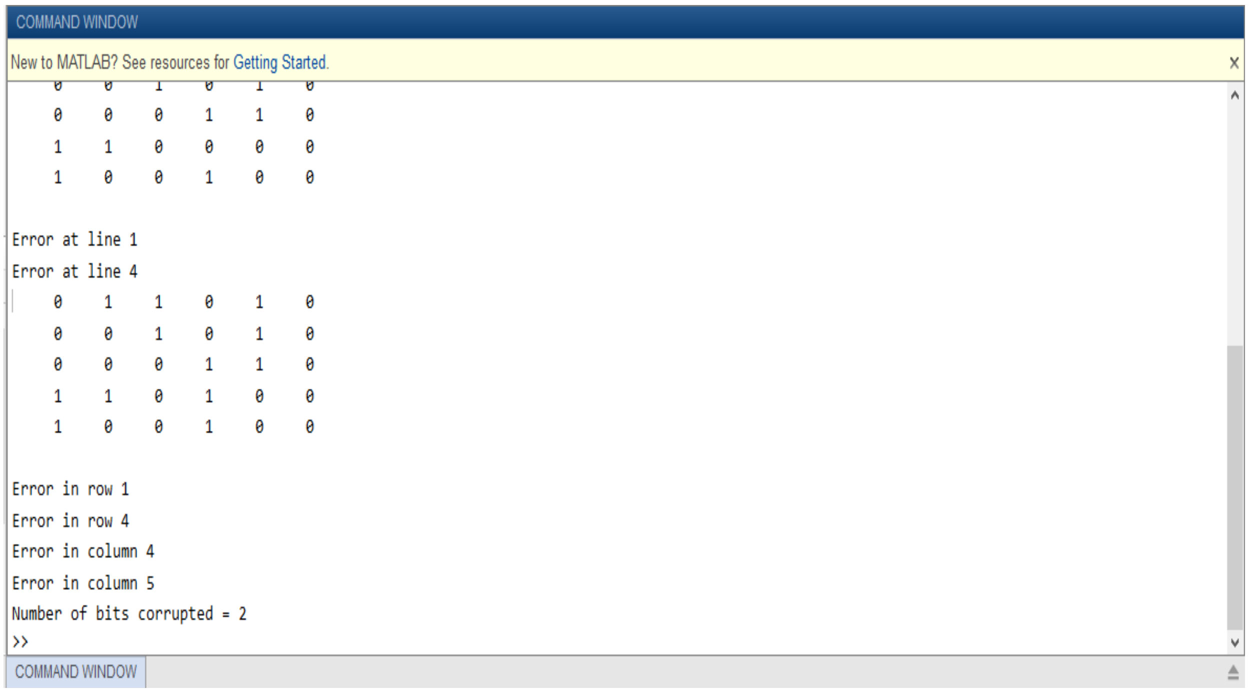
```
>> parity
Enter the number of rows
5
Enter the number of columns
8
Original matrix
     0     0     1     0     0     0     1     0
     1     0     1     0     0     1     1     1
     0     0     0     0     0     0     1     1
     0     1     0     0     1     1     0     0
     0     0     0     1     1     1     0     0

After adding parity bits
     0     0     1     0     0     0     1     0     0
     1     0     1     0     0     1     1     1     1
     0     0     0     0     0     0     1     1     0
     0     1     0     0     1     1     0     0     1
     0     0     0     1     1     1     0     0     1
     1     1     0     1     0     1     1     0     1
```



```
Error at line 4
     0     0     1     0     0     0     1     0     0
     1     0     1     0     0     1     1     1     1
     0     0     0     0     0     0     1     1     0
     0     1     1     0     1     1     0     0     1
     0     0     0     1     1     1     0     0     1
     1     1     0     1     0     1     1     0     1

Error in row 4
Error in column 3

Error at row 4   column 3
Error corrected !
     0     0     1     0     0     0     1     0     0
     1     0     1     0     0     1     1     1     1
     0     0     0     0     0     0     1     1     0
     0     1     0     0     1     1     0     0     1
     0     0     0     1     1     1     0     0     1
     1     1     0     1     0     1     1     0     1
```



```
>> parity
Enter the number of rows
4
Enter the number of columns
5
Original matrix
     0     1     1     0     0
     0     0     1     0     1
     0     0     0     1     1
     1     1     0     0     0


After adding parity bits
     0     1     1     0     0     0
     0     0     1     0     1     0
     0     0     0     1     1     0
     1     1     0     0     0     0
     1     0     0     1     0     0
```



```
     0     0     0     1     1     0
     1     1     0     0     0     0
     1     0     0     1     0     0


Error at line 1
Error at line 4
     0     1     1     0     1     0
     0     0     1     0     1     0
     0     0     0     1     1     0
     1     1     0     1     0     0
     1     0     0     1     0     0


Error in row 1
Error in row 4
Error in column 4
Error in column 5
Number of bits corrupted = 2
>>
```

We can observe that the program on taking the input of size of rows and column which is nothing but the data word appends an extra row and column initially filled with zero. It finds the even parity of each row and each column to find the code word this is a matrix of M+1 X N+1 assuming M,N to be initial row and column size of data word respectively. This program mimics the natural communication by randomly inducing error in the matrix and trying to correct that. This program detects two bit errors and corrects one.

4. REFERENCES AND APPENDIX

The main reference is
https://www.mathworks.com/help/comm/ug/error-detection-and-correction.html

randi – It is used to induce error in the data

https://github.com/gmendonca/gaussian-elimination-pthreads-openmp/blob/master/gauss.c

https://github.com/newOnahtaN/CS312-PrinciplesOfProgrammingLanguages/tree/master/Parallelized-Gaussian-Elimination

https://github.com/mghojal/Edge-detection-using-laplacian-operator