Volume 3, No. 3, March 2012

Journal of Global Research in Computer Science

ISSN-2229-371X

REVIEW ARTICLE

Available Online at www.jgrcs.info

A CLOSER LOOK INTO THE RECTANGULAR CODES: WHERE THE PARITY CHECK PREDOMINATES

Romi Banerjee*[1], Saptarshi Naskar[2] and Samar Sen Sarma[3]
*[1]Department of Computer Science and Engineering, University of Calcutta, Kolkata, West Bengal, India
rm.banerjee@gmail.com[1]
[2]Department of Computer Science, Sarsuna College, Kolkata, West Bengal, India
sapgrin@gmail.com[2]
[3]Department of Computer Science and Engineering, University of Calcutta, Kolkata, West Bengal, India
sssarma2001@yahoo.com[3]

Abstract: Information transmission in the presence of 'white noise' is error prone, but is not as elusive as any arbitrary noise prone channel. With respect to this restriction, the parity check is the only feasible error handling scheme that could be adopted in, at least, any intra-system digital communication (e.g. All existing computing systems use the ASCII codes); while the repetition of messages (e.g. The triplicate transmission scheme) is not a viable alternative. In this paper, we propose simple algorithms to clearly prove the greater practical efficiency of the 2-dimensional parity check based coding systems in the 'rectangular' framework – as opposed to the established notion of the 'square code' being more efficient. These algorithms are further corroborated by theorems, associated hypotheses and experimental results. The lower and the upper bound of our algorithms are clearly explained.

Keywords: 2-dimensional codes, efficiency, parity check, rectangular code, redundancy, square code

INTRODUCTION

Digital computers are based on binary codes and digital communication necessitates the error free transmission of bits. Shannon's theorem on entropy provides a theoretical perspective on the 'amount of information' carried by these codes as well as the lower bound on the average code length [1], [2].

The parity check plays the most important role of error handling of messages in, at least any, intra – system digital communication. A typical example of the use of the parity check is in the transmission of ASCII codes in all existing computing systems – ASCII codes encode the alphabets, as in the keyboard encoders, during communication between human beings and any computing system. Effective parity check systems employ the 2-dimensional coding scheme on account of their exceptional error handling (single error correction and double error detection) properties.

In this paper, we limit our discussion to a study of the efficiency of the 2-dimensional codes, in the square and the rectangular frameworks. While [1] and [2] clearly depict the square code as 'theoretically more efficient', our studies prove the rectangular code to be definitely the 'most practically efficient' – compared to all other multi-dimensional coding systems [reminiscent of the philosophy of the 2-dimensional RAM organizations in computer architecture].

We propose here, a simple, yet unique, logic to establish the greater practical efficacy of the rectangular code. Discussions on syndrome calculation, triplicate transmission and other such processes are trivially out of our arena.

The paper begins with an insight into the inspiration behind the work followed by a discussion on the mathematical basis of our study. The article then proceeds to an elaboration on our proposed logic.

INITIAL INSPIRATION

Checksums, syndrome evaluation and the parity check are typical error handling schemes in digital transmission systems. Checksums permit error detection but no error correction; while syndrome evaluation and parity checks permit both – the former, at the price of high redundancy for small messages.

Thus, in the light of error-free digital intra-system communication, where the length of messages is of the order of bytes, a study on the parity-check and the 2-dimensional codes stands obligatory. Prime focus lies on the evaluation of the redundancy involved in these coding systems. The redundancy leads to a measure of the efficiency of the code; greater the redundancy, lower is the efficiency and vice versa.

Our concentration, in this paper, is entirely on error-free digital intra-system communication and consequentially on the efficiency of the 2-dimensional square and rectangular codes.

MATHEMATICAL FOUNDATION

This section highlights the mathematical prerequisites for the subject under discussion.

Parity Bit Evaluation:

Given a message M of (n – 1) bits, the consequent parity bit is a '1' or a '0', appended to M, such that the total number of ones in the n-bit message is either an even number (even-parity) or an odd number (odd-parity). Thus, the evaluation of the parity bits calls for a count of the number of ones in the given message.

The popular algorithms for counting the number of ones in an n-bit message (say, B) are as follows [4]:

- ***a. The linear technique*:** involving a bit by bit check and a subsequent count of the number of occurrences of '1' in the message. This algorithm evidently has a time complexity of O(n) and a space complexity of O(1).
- ***b. The iterative intersection technique*:** involving an iteration of B = (B ∩ (B – 1)) until B reduces to 0. This algorithm has a time complexity of O(c) [c = number of '1's] and a space complexity of O(n).
- ***c. The divide and conquer summation technique:*** is based on the binary search method and the algorithm has a time complexity of O($\log_2 n$) and a space complexity of O(n).
- ***d. The look:*** *up table technique:* follows the content-addressing approach where an input decimal number acts as the address to the corresponding bit-sum entry in a look-up table. This algorithm has a time complexity of O(1) but a space complexity of O(2^n).

Once the number of '1's (say m) in the message is counted, the parity bit equals m modulus 2.

Of all the four algorithms:

a) The linear technique, though is the slowest, is the simplest and is acceptable for all practical purposes;

b) The iterative intersection technique is ideal for sparse strings and can only be used if sufficient memory be available;

c) The divide and conquer summation technique, though elegant, has a rather complex implementation strategy and depends on the available memory.

d) The look – up table technique is the fastest algorithm but requires an extravagant amount of memory.

Note: Use of parallel algorithms to count the number of ones, reduces the parity check evaluation to an O(1) operation.

***Redundancy in the 2 – dimensional rectangular and square Codes* [1], [2]:**

In the rectangular code representation of an n – bit message, the bits of the message are arranged in a (p x q) rectangle, where (p * q) = n. A parity check bit is appended to each of the rows as well as to each of the columns, transforming the original (p x q) rectangle into a [(p + 1) x (q + 1)] rectangle. The redundancy (R_{rect}) thus equals:

$$R_{rect} = \frac{\text{Number of bits used in the complete message}}{\text{Number of bits in the original message}}$$

$$\text{or, } R_{rect} = \frac{(p+1)(q+1)}{pq} \qquad (1)$$

and,

the total number of redundant bits = number of parity check bits = p + q + 1 (2)

Now, for the case that the rectangle approaches a square (where, p = q), i.e. n is a perfect square, the redundancy (R_{sq}) clearly equals [using Equation 1]:

$$R_{sq} = \frac{(p+1)(p+1)}{pp}$$

$$\text{or, } R_{sq} = \frac{(p+1)^2}{p^2} \qquad (3)$$

and,

the total number of redundant bits = number of parity check bits= 2p + 1 (4)

Evidently, for a given value of 'n', where 'n' is a perfect square and has multiple factors (besides 1, 'n' and $\sqrt{n}$), Equations 1 and 2 attain minimum value when they equate to Equations 3 and 4 respectively.

The redundancy associated with the square and the rectangular codes, as evaluated in Equations 2 and 4, is visualized through the plots shown in Fig. 1. The square code indeed shows minimum redundancy for all values of 'n' – where 'n' is a perfect square. In Fig. 1, the rectangular code plot is formed using any random factor pair of 'n' and this code equates to the square code only when 'n' has no other factor besides 1, 'n' and $\sqrt{n}$.

The rectangular codes and the square codes are responsible for the [1]-[3]:

a. Detection of an even number of bits in error, across either a single row (where each column has a single bit in error) or a single column (where each row has a single bit in error).

b. Correction of any number of odd bits in error, across either a single row (where each column has a single bit in error) or a single column (where each row has a single bit in error).

Relevant Theorems, Observations and Hypothesis:

Theorem 1: Given a message M of n – bits, where 'n' is a composite number, let there be k pairs of factors of 'n', {(p_1, q_1), (p_2, q_2), …, (p_k, q_k)}. The redundancy is the minimum, if the factor pair (p, q) selected satisfies:

$$(p+1)(q+1) = \min_{1\le i\le k}[(p_i+1)(q_i+1)] \qquad (5)$$

Proof: From Equation (1), the redundancy for the i^{th} (1≤ I ≤ k) factor pair equals:

$$R_{rect_i} = \frac{(p_i+1)(q_i+1)}{pq} \qquad (6)$$

Now, the denominator of the fraction in Equation 6 being equal $\forall i$, R_{rect_i} is undoubtedly minimum for the factor pair that satisfies Equation 5.

Trivial Observation 1: The distance between i^2 and $(i + 1)^2$, where 'i' is a natural number and i ≥ 1,

= (2i + 1) (7)

Note: This observation is in accordance with the sparse distribution of perfect square integers in the natural number space. For instance, there exist three perfect square integers {1, 4, 9} in the range [1, 10], one perfect square integer {16} in the range [11, 20], no perfect square integers in the [51, 60] range and so on.

Trivial Observation 2: The number of zeroes that needs to be appended to a given n-bit message M, such that n is raised to its nearest perfect square integer is:

$$\left(\lceil\sqrt{n}\rceil^2 - n\right) \qquad (8)$$

Note: Zeroes appended to a given message M do not affect the parity of M.

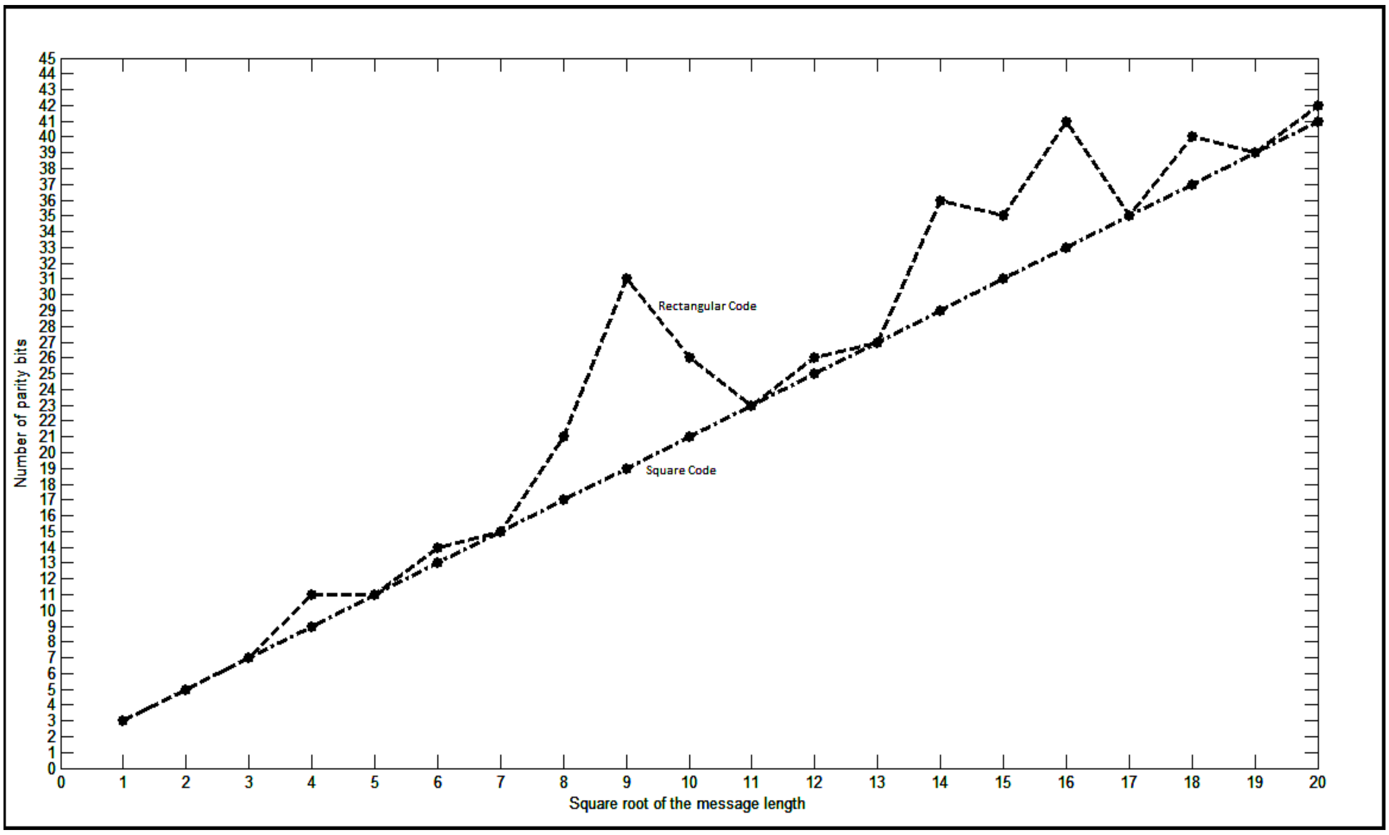


Figure 1. Plot representing the number of parity (or redundant) bits, with respect to Equations 2 and 4, in the square and the rectangular code of a given n – bit message; ‘n’ is factored with respect to Equation 5. The square code indeed shows minimum redundancy for all values of ‘n’ – ‘n’ is a perfect square.

Deductive Hypothesis: Using Equations 7 and 8, we arrive at the following relation,

$$0 \leq \left(\left\lceil \sqrt{n} \right\rceil^2 - n \right) \leq 2i \qquad (9)$$

where, $i^2 \leq n < (i + 1)^2$, ‘i’ is a natural number and $i \geq 1$.

DETAILS OF THE PROPOSED LOGIC

The results of the preceding section theoretically and figuratively depict the square code to be more efficient than the rectangular code. Evidently, these results assume: 1) ‘n’ to be a perfect square and thus to be directly convertible to the consequent rectangular and square forms, and 2) the parity bit(s) is the sole contributor to the redundancy. However, on a deeper examination it is obvious that the convertibility of the n-bit message (M) to these 2-dimensional codes depends entirely on whether ‘n’ is a perfect square, prime or composite-non-square integer.

This observation suggests the incorporation of a second factor of redundancy – the number of ‘0’s that need to be appended to M to raise ‘n’ to a number directly transformable to the square or the rectangular codes.

If ‘n’ is not a perfect square, the conversion of M into a square code relies on its distance to the nearest perfect square integer, as is shown in Equation 8. Whereas, the conversion of M into the rectangular code requires appending, at most, a single ‘0’ to the message only if ‘n’ is a prime number (n > 2) [Two ‘0’s need to be appended only if n = 2].

These facts lead to the conclusion that the rectangular code, almost always, supersedes the square code in terms of efficiency, given any arbitrary n-bit message. It is this deduction that forms the basis of this paper. Moreover, considering an intra-system digital communication system and its constraints on message length, incorporation of any irrelevant redundancy is undeniably unwanted. Therefore, all studies made in the context of redundancy are undoubtedly of importance.

This section describes our studies – the subtle mathematical details, algorithms and experimental results obtained – in the purview of the supposition made on the rectangular codes.

Given a message of n bits –
Assumption 1. The system is assumed to utilize any of the standard algorithms to test if ‘n’ is prime or is a perfect square and also to evaluate the factors of ‘n’ [5], [6]. The factors of ‘n’ follow Equation 5. These tests are essentially pre-processing steps, and the complexities of these algorithms, though affect the execution time of the entire 2-dimensional coding system, do not particularly affect the areas of our concern – the parity check and the efficiency of the 2-dimensional codes. Thus, the complexities of these algorithms do not feature in the subsequent discussion.

Assumption 2. The system utilizes the linear of counting the number of ‘1’s to evaluate the parity bits in a given message.

Assumption 3. The system takes constant time to append a single zero to the message.

Proposed Algorithm for the Formation of the Square Code, Given an n – bit Message:

Algorithm I:

Input: Message M of n – bits
Output: The square code of M
Steps:

a. Find square root ('root') of n
   [root = $\left\lceil \sqrt{n} \right\rceil$]
b. Is n a perfect square?
   [If $(\text{root})^2 > n$,
   then append $((\text{root})^2 – n)$ redundant zeroes to M so that the modified M contains $(\text{root})^2$ bits]
c. Arrange M into the (root x root) square code organization
   [The first set of 'root' consecutive bits of M forms row_1 of the organization, the next set of 'root' consecutive bits of M form row_2, and so on till row_{root} is formed.]
d. Evaluate the parity bit for every row and column, of the (root x root) organization, and append the parity bits to the corresponding row and column respectively.
   [After the inclusion of the parity bits, the square code of M is obtained. The dimension of the formed square code is: (root + 1) x (root + 1).]
e. Stop

Computation of the Redundancy Introduced into the Message by Algorithm I:

i. Computation of the square root (sqr) of n,
   where, sqr = $\left\lceil \sqrt{n} \right\rceil$
ii. If $[(\text{sqr})^2 > n]$
   then
   a. $r = (\text{sqr})^2$, where r = new message length.
   b. $s = r – n$, where s = number of zeroes appended to the original message, such that $(s + n) = r$.
   Else
   $s = 0$
iii. Using Equation 3, redundancy (R_{sq_q}) equals

$$R_{sq\_q} = \frac{(\text{sqr}+1)^2}{n} \qquad (10)$$

and using Equation 4,
the total number of actual redundant bits
= s + total number of parity check bits
$= s + (2 * \text{sqr} + 1)$ (11)
where, s lies in the range stated in the Inequality 9.

Analysis of Algorithm I:

Using Assumptions 1, 2 and 3, we arrive at the following results:

a. Time to append the redundant zeroes
= time to append a single zero * number of redundant zeroes
= (1 * s) i.e., O(s) (12)

b. Time to evaluate the parity bits
= [time to evaluate the parity bits for the rows of the square code + time to evaluate the parity bits for the columns of the square code]
= (time to evaluate the parity bit for a single row * number of rows) + (time to evaluate the parity bit for a single column * number of columns)
= (number of bits in a single row * number of rows) + (number of bits in a single column * number of columns) (13)

Now, after the parity bits for the rows (columns) are calculated, the parity bits for the columns (rows) requires evaluating the parity bits over (sqr + 1) columns (rows). Thus, Equation 13 equates to:

$$= (\text{sqr})^2 + [\text{sqr} * (\text{sqr} + 1)] = [\text{sqr}(2\text{sqr} + 1)]$$

$$\leq \left\lceil n^{1/2}+1 \right\rceil^3 \text{ i.e., } n^{3/2} + O(n^2) \qquad (14)$$

c. Using Equations 12 and 14, the total time complexity of the algorithm
= time to append the redundant zeroes + time to evaluate the parity bits = $n^{3/2} + O(s + n^2)$ (15)

In the best case, s = 0 and the time complexity of the algorithm reduces to that in Equation 14. In the worst case, when s = 2i [as in Inequality 9], the time complexity of the algorithm is inclusive of all the terms and equates to [$n^{3/2} + O(2i + n^2)$].

Proposed Algorithm for the Formation of the Rectangular Code, Given an n – bit Message:

Algorithm II:

Input: Message M of n – bits
Output: The rectangular code of M
Steps:

a. Is n a composite number?
   [If n is prime and n = 2,
   then append two redundant zeroes to M, so that the modified M contains 4 bits and n = 4
   Else
   if n is prime and $n \geq 3$,
   then append one redundant zero to M, so that the modified M contains (n + 1) bits and n = (n + 1)]
b. Factorize n into its factors p and q
   [p and q follow Equation 5]
c. Arrange M into the (p x q) rectangular code organization
   [The first set of 'q' consecutive bits of M form row_1 of the organization, the next set of 'q' consecutive bits of M form row_2, and so on till row_p is formed]
d. Evaluate the parity bit for every row and every column of the (p x q) organization and append the parity bits to the corresponding row and column respectively.
   [After the inclusion of the parity bits, the rectangular code of M is obtained. The dimension of the formed rectangular code is: (p + 1) x (q + 1).]
e. Stop

Computation of the Redundancy Introduced into the Message by Algorithm II:

i. If n is a prime number
   a. If $(n \geq 3)$,
   then s = 1, where s equals to the number of zeroes appended to the message.
   b. If (n = 2),
   then s = 2,
   Else
   s = 0

ii. $r = (n + s)$, where r is the new message length and r is composite.

iii. Factorization of r into its factors p and q, as per Equation 5.

iv. Using Equation 1, redundancy (R_{rect_q}) equals

$$R_{rect\_q} = \frac{(p+1)(q+1)}{n} \qquad (16)$$

and using Equation 2,
the total number of actual redundant bits
= s + total number of parity check bits

$$= s + (p + q + 1) \qquad (17)$$

where, s = 0 (if, n is composite), 1 (if n is a prime number ≥ 3), 2 (if n = 2).

Analysis of the Algorithm II:

Using assumptions (1), (2) and (3), we arrive at the following results:

a. The time to append the redundant zeroes
= time to append a single zero * number of redundant zeroes
= (1 * s) i.e., O(s) (18)

b. The time to evaluate the parity bits
= [time to evaluate the parity bits for the rows of the rectangular code + time to evaluate the parity bits for the columns of the rectangular code]
= (time to evaluate the parity bit for a single row * number of rows) + (time to evaluate the parity bit for a single column * number of columns)
= (number of bits in a single row * number of rows) + (number of bits in a single column * number of columns) (19)

Now, after the parity bits for the rows (columns) are calculated, the parity bits for the columns (rows) requires evaluating the parity bits over (q + 1) columns [(p + 1) rows]. Thus, Equation 19,

$= (q * p) + [p * (q + 1)]$ [or, $(p * q) + [q * (p + 1)]$]

$= [p(2q + 1)]$ [or, $q(2p + 1)$]

$\leq 3n$ i.e., $O(n^2)$ (20)

c. Using Equations 18 and 20, the total time complexity of the algorithm
= time to append the redundant zeroes + time to evaluate the parity bits = $O(s + n^2)$ (21)

In the best case, s = 0 and the time complexity of the algorithm reduces to Equation (20). In the worst case, s = 2 [when n = 2], and the time complexity of the algorithm equates to $O(2 + n^2)$.

Inference:

For a given n-bit message M, we infer the following from Equations 10-21:

a. Equations 10 and 11 acknowledge the sparse distribution of perfect square integers across the natural number space. Raising any arbitrary value 'n' to its nearest perfect square integer, is indeed an expensive operation.

b. Equations 10, and 16 yield equal results only if p = q = sqr, i.e., when 'n' is a perfect square. [This coincides with the results in [1] – explained in the preceding section].

c. Equations 16 and 17 yield lower values than those of Equations 10 and 11 respectively, when 'n' is not a perfect square, i.e. $(sqr^2) > (n)$.

d. Equations 15 and 21 underline the time complexities of the algorithms proposed for the square code and the rectangular code formations, respectively. The 2-dimensional codes, in the rectangular framework, are clearly optimal.

The studies described in this section, essentially generalize the formation of the square and the rectangular codes to that for any given n-bit message – irrespective of the numeric character of 'n'.

All of the aforementioned consequential deductions, thus, formidably prove the rectangular code to be the coding scheme that incorporates minimum redundancy and is thus, the most 'practically efficient' 2-dimensional code. The inferences are further supported by Fig. 2 and Fig. 3**.**

Fig. 2 depicts plots corresponding to the number of extra zeroes that need to be appended to any n-bit message to evaluate the square and the rectangular codes. Fig. 3 depicts plots corresponding to the total number of redundant bits for both the square and the rectangular code formations given any n-bit message. In both the figures, the rectangular code achieves considerably lower values, for all values of 'n'.
Table 1 represents a tabular summarization of Fig. 2 and Fig. 3.

Note: An extension of the described logic describes the efficiency of the Rectangular codes in comparison to all the other multi – dimensional code formats.

CONCLUSION

In this paper, we algorithmically, mathematically and experimentally prove the rectangular code to be the most 'practically efficient' code.

We begin with a discussion on the established notion of the efficiency of the square code, to later point out the sparse distribution of the perfect square integers to be detrimental to the notion. Conversion of a message to its square code is indeed inefficient and expensive. The paper then presents studies that establish the intuitive, highly efficient and relatively cost effective character of the rectangular code. Moreover, the rectangular code inherits the exceptional error handling potential of the 2-dimensional coding systems. It can thus be rightly concluded that the rectangular code is definitely the most practically efficient scheme to enhance the robustness of the existing intra-system digital communication systems.

Further studies on the rectangular codes are alluring, and we are in the process of its excavation.

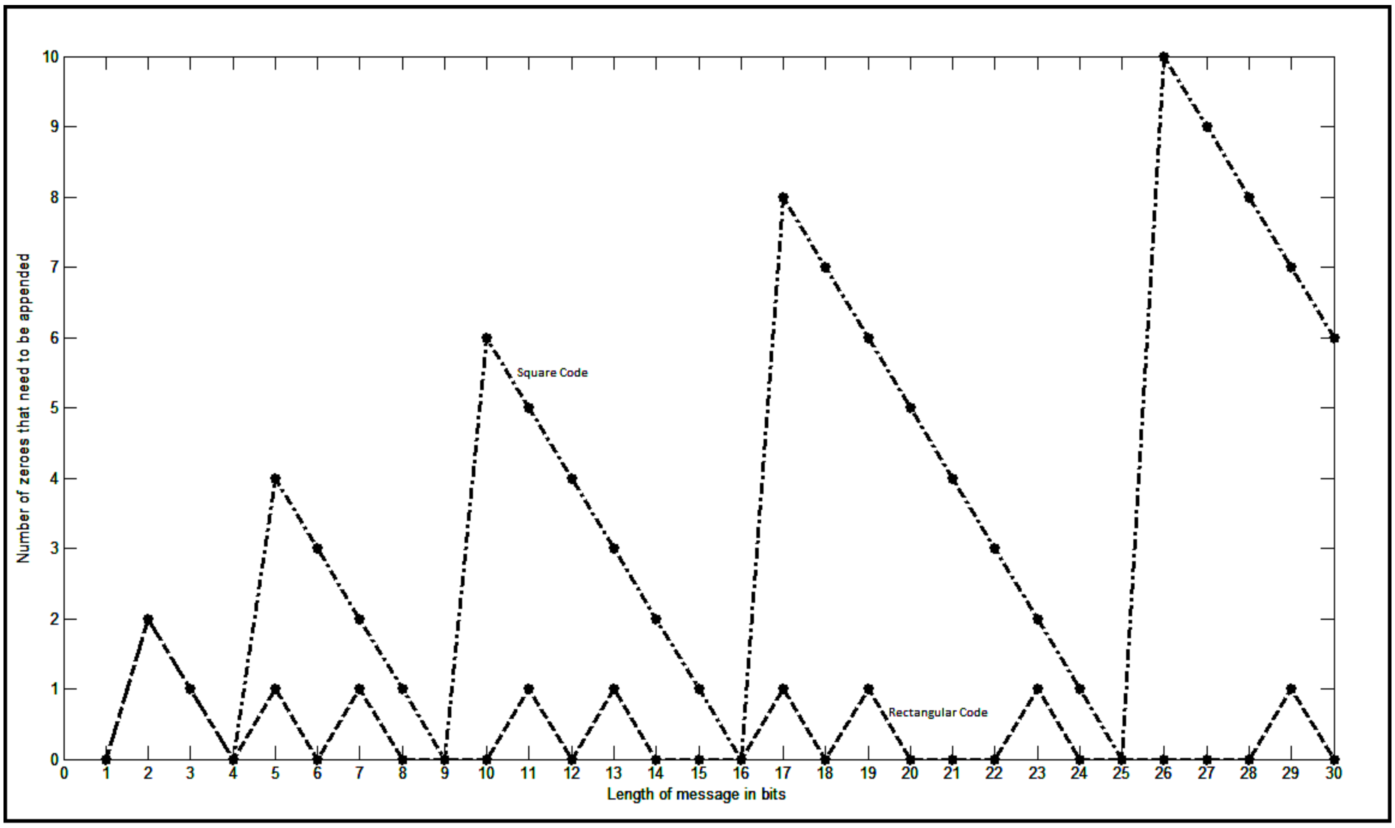


Figure 2. Plot representing the number of zeroes to be appended to the given n – bit message, where 'n' is an arbitrary integer, for the conversion to the square and the rectangular codes. The rectangular code achieves minimum results for all values of 'n', and coincides with the square code only when 'n' is a perfect square.

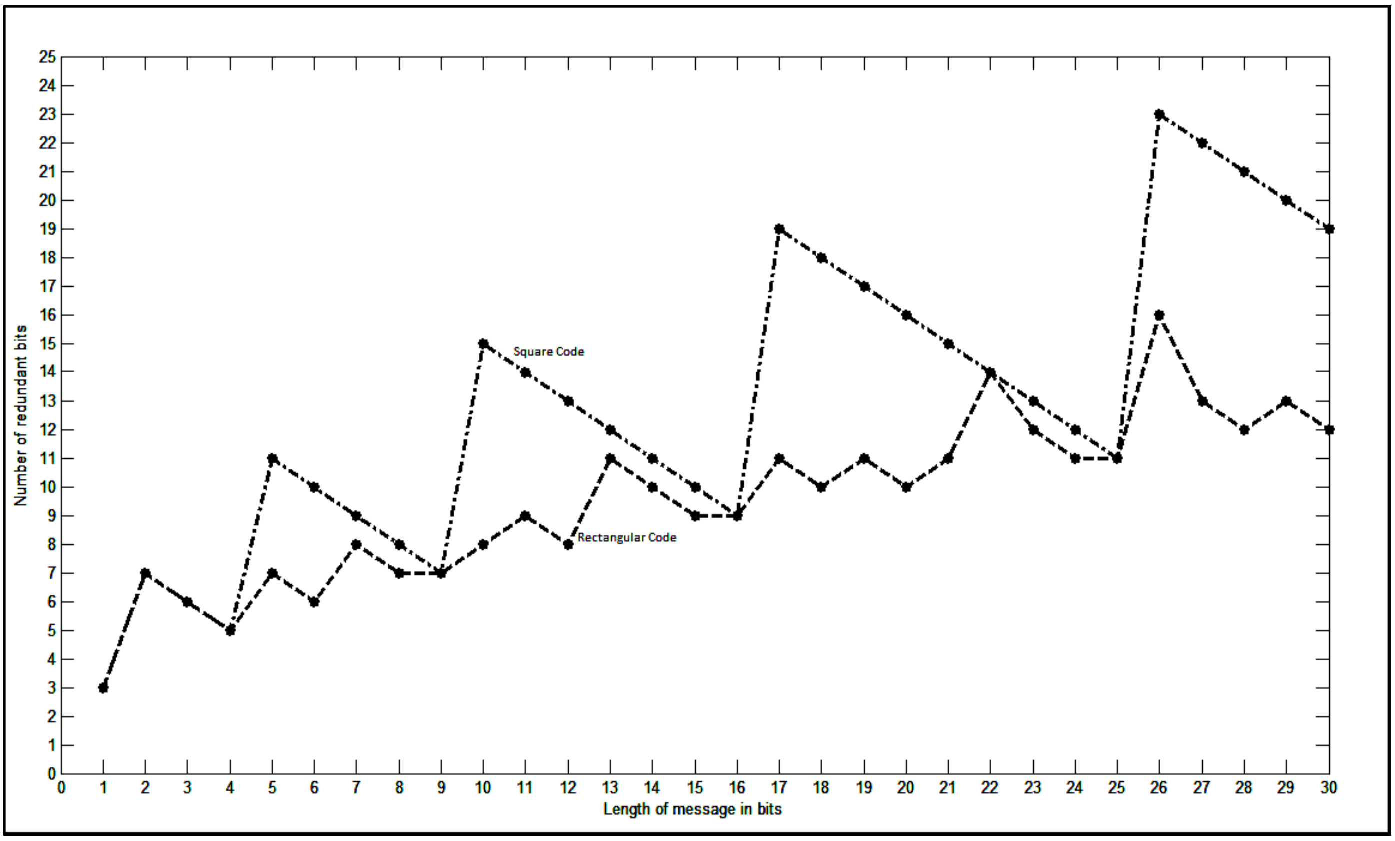
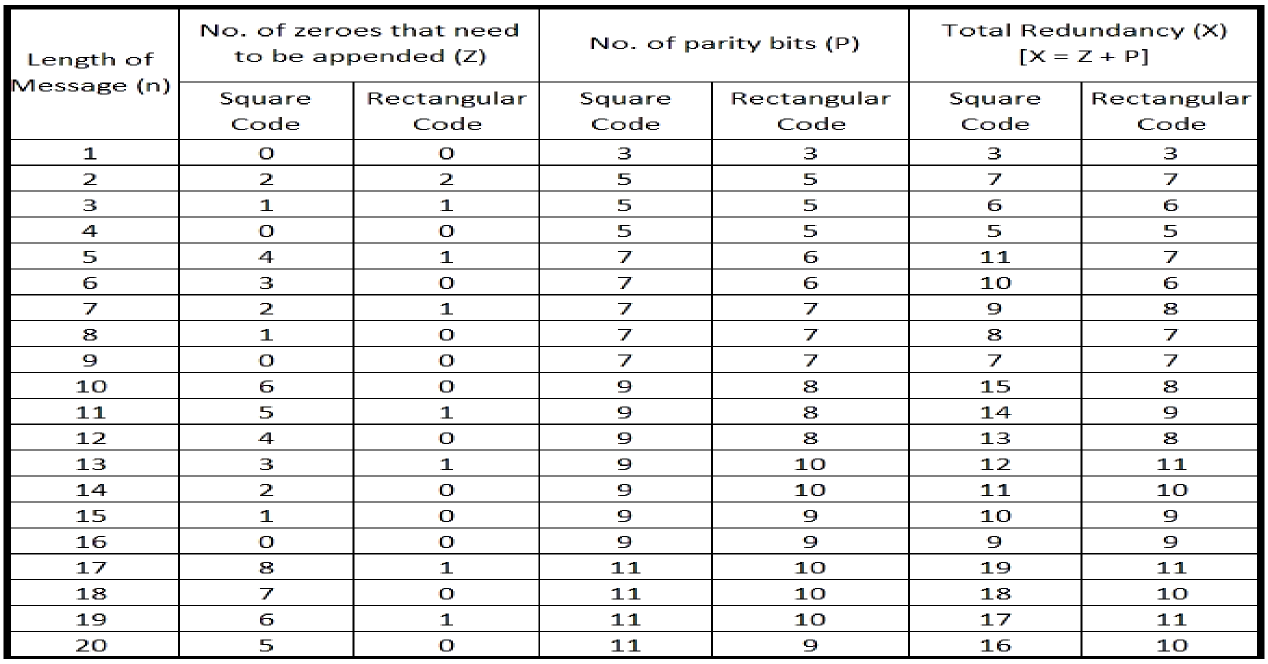


Figure 3. Plot representing the total number of redundant bits in the square and the rectangular code of a given n – bit message, where 'n' is an arbitrary integer. The rectangular code shows minimum redundancy for all values of 'n'.

Table 1. Tabular summarization of the plots in Fig. 1, Fig. 2 and Fig. 3. The rectangular code shows minimum overall redundancy requirements for all values of 'n' – irrespective of the numeric nature of 'n' and is thus, evidently the most efficient code.



Length of Message (n)	No. of zeroes that need to be appended (Z)		No. of parity bits (P)		Total Redundancy (X) [X = Z + P]	
	Square Code	Rectangular Code	Square Code	Rectangular Code	Square Code	Rectangular Code
1	0	0	3	3	3	3
2	2	2	5	5	7	7
3	1	1	5	5	6	6
4	0	0	5	5	5	5
5	4	1	7	6	11	7
6	3	0	7	6	10	6
7	2	1	7	7	9	8
8	1	0	7	7	8	7
9	0	0	7	7	7	7
10	6	0	9	8	15	8
11	5	1	9	8	14	9
12	4	0	9	8	13	8
13	3	1	9	10	12	11
14	2	0	9	10	11	10
15	1	0	9	9	10	9
16	0	0	9	9	9	9
17	8	1	11	10	19	11
18	7	0	11	10	18	10
19	6	1	11	10	17	11
20	5	0	11	9	16	10

REFERENCES

[1]. R. W. Hamming, "Coding and Information Theory", 2nd Edition, Prentice Hall, U.S.A, 1986

[2]. R. P. Feynman, "Lectures on Computation", Perseus Books Groups, U.S.A, 1996

[3]. G. A. Jones and J. M. Jones, "Information and Coding Theory", Springer, Great Britain, 2000

[4]. E. Reingold, J. Nievergelt, N. Deo, "Combinatorial Algorithms : Theory and Practice", Prentice Hall, U.S.A, 1977

[5]. D. E. Knuth, "The Art of Computer Programming Volume 2 : Seminumerical Algorithms", 2nd Edition, Addison – Wesley, U.S.A, 1981

[6]. B. A. Forouzan, "Cryptography and Network Security", Tata McGraw Hill, India, 2007

Short Bio Data for the Author

Romi Banerjee: Is currently a Master of Technology (MTech) student of the Department of Computer Science and Engineering, at the University of Calcutta, India. She is

currently working on Machine Intelligence under the guidance of Professors Samar Sen Sarma (University of Calcutta) and Sankar K. Pal (Former Director, Indian Statistical Institute)

Saptarshi Naskar: Is a member of the faculty of the Department of Computer Science at Sarsuna College, India. He is currently registered for his PhD on Graph Theory and Cryptology under the University of Calcutta.

Professor Samar Sen Sarma : Is the founder member and senior professor of the Department of Computer Science and Engineering at the University of Calcutta (the first multidisciplinary modern university in South Asia and the oldest institution to have the 'University' status). His current research interests include Graph Theory, Quantum Computing and Algorithms.