

# Rapid coding algorithm for Low Density Codes based on Reed-Solomon Codes over GF (q)\*

F.V. Groshev

Institute for Information  
Transmission Problems,  
Russian Academy of Sciences  
19 B. Karetny Lane, GSP-4,  
127994, Moscow, Russia  
e-mail: groshev@iitp.ru

V.V. Zyablov

Institute for Information  
Transmission Problems,  
Russian Academy of Sciences  
19 B. Karetny Lane, GSP-4,  
127994, Moscow, Russia  
e-mail: zyablov@iitp.ru

## Abstract

In this paper a rapid coding algorithm for low density codes (LDCode), based on Reed-Solomon codes over GF(q) is considered. We are aiming at finding encoder complexity estimates and upper bounds for code distance. We use simulation for this purpose.

## I. INTRODUCTION

The complexity of traditional encoders and decoders for long codes is very high. To solve this problem its typical to replace long codes into a set of shorter codes spread according to the preassigned rule over the code length of the long code. This code construction is known as low density code (LDCode). The above mentioned codes are in great demand now for the following reasons: in the ensemble code distance growth is linear with code length, up to day level of digital technology let us realize an encoder for this ensemble.

## II. MAIN PART

We are aiming at finding easily implemented coding algorithms for the codes from LDCodes ensemble based on Reed-Solomon codes over GF(q) and estimation of the algorithms' characteristics.

### A. The LDCodes ensemble description

Actually, it's typical to define an LDCode with the sparse of non systematic check matrix. There is a fixed number of non zero elements in each row and each column of this matrix. The required amount of codes should be spread in the check matrix H (see Table 1) according to the Rule 1. The coding is then implemented by choosing component RS codes according to the Rule 2. Let us call the component RS codes sequence defined by the Rule 2 the rapid coding trajectory.

**Rule 1:** check matrix H generation

- the matrix consists of  $N$  columns and  $lm$  rows, where  $N=mn$  is the LDCode length,  $lm$  is the number of RS codes;
- the matrix cell contents a column of  $r$ -length from RS code check matrix;
- the matrix consists of  $l$  horizontal stripes with  $m$ -width, that is why each stripe consists of about  $n$  squares with of  $m \times m$  dimensions;
- each matrix stripe contents strictly  $N - n$  zero cells and  $n$  non zero cells, each of them is a column of  $r$ -length from RS code check matrix;
- there is only one non zero cell in each  $m \times m$  dimension square;

\* This work was supported in part by the State Contract No. 02.514.11.4025

- the non zero cells order in the first stripe is block-diagonal. In other stripes non zero cells order is randomized.

**Rule 2:** the rapid coding trajectory design

- we should take the RS code with maximum number of symbols, given or calculated in previous iteration of the trajectory design. If there are several RS codes satisfying this condition, choose a random code from this set;
- the following check is required: whether there are less than  $r$  unoccupied positions in  $l$  codes associated with the code under consideration from other stripes or not;
- for the included RS code  $n - r - p$  information and  $r$  checking symbols are the corresponding symbols of the LDCode (here  $p$  is the number of RS code symbols, assigned in the previous steps);
- check the overall number of the informational symbols in the trajectory. If this number is less than  $K = m(n - lr)$ , then go to the next stripe. If this number is more than  $K$ , then we decline the trajectory and restart the algorithm; If this number is equal to  $K$ , then operation stops – the trajectory design is completed.

Different forms of check matrixes, equivalent RS codes can be used in different rows of LDCode. In this paper we used an extended RS code. There is an all-one row in the check matrix of the code. It leads to check matrix rank decrease. The latter is undesirable since it results in trajectory length decrease. We used generalized extended RS codes with random chosen multiplies since there is no all-one row in its check matrix.

Let us denote canonical check matrix of the extended RS code by  $H_0$ . Then the check matrix of the generalized extended RS code is  $H_+ = H_0 D$  (here  $D$  is a diagonal matrix with random non zero elements from  $GF(q)$ ).

### B. Rapid algorithm definition

Matrix inversion is the most consuming part of the encoding algorithm, from the point of view of computation. The issue point of rapid encoding algorithm is to decrease the size of the matrix to be inverted in order to minimize the overall computation complexity. The latter could be done by representing the matrix under consideration in the appropriate form (see fig.1). To transform the matrix into the appropriate form (which allows us to solve corresponding equations without matrix inversion) we will use elementary operations only (i.e. rows and column transpositions). The remained of the matrix ( $H_{11}$ ) is to be invertible. Nevertheless it is much smaller than matrix  $H$ , thus it's inversion require less computation. Hence the matrix  $H$  is to be represented in the following form:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{00} & \mathbf{0} \\ \mathbf{H}_{01} & \mathbf{H}_{11} \end{bmatrix}$$

, here  $H_{11}$  is the invertible remainder.

The submatrix  $H_{00}$  contains RS codes included in to the trajectory. The submatrix  $H_{00}$  is block-diagonal. Each column contains not more than  $l$  non zero elements. Each column of submatrix  $H_{01}$  contains not more than  $l - 1$  non zero elements.

Submatrix  $H_{11}$  is always a square matrix:

Let the length of the trajectory be  $L$  blocks. Then the number of rows of the submatrix  $H_{11}$  is  $lm - L$  (in blocks) or  $(lm - L)r$  (in symbols). On the other hand, RS blocks included in to the trajectory denote  $Lr$  positions of the check symbols and  $K = N - lmr$  of the LDCode.

		n squares m x m dimensions									
		1	...	m	1	...	m	...	1	...	m
1	1	h <sub>11</sub> ... h <sub>1n</sub>									
	m			h <sub>21</sub> ... h <sub>2n</sub>					h <sub>(4m-2) 1</sub> ... h <sub>(4m-2) n</sub>		
				h <sub>31</sub> ... h <sub>3n</sub>					h <sub>(4m-1) 1</sub> ... h <sub>(4m-1) n</sub>		
				h <sub>41</sub> ... h <sub>4n</sub>					h <sub>(4m) 1</sub> ... h <sub>(4m) n</sub>		
				h <sub>51</sub> ... h <sub>5n</sub>							
				h <sub>61</sub> ... h <sub>6n</sub>							
m	1			h <sub>2n</sub>			h <sub>41</sub>		h <sub>(4m) 1</sub>		
	m	h <sub>11</sub>		h <sub>1n</sub>					h <sub>(4m) n</sub>		
		...	...	...			...	...	h <sub>(4m-2) 1</sub>		
				h <sub>31</sub>					h <sub>(4m-1) n</sub>		
				h <sub>3n</sub>					h <sub>(4m-1) 1</sub>		
				h <sub>21</sub>					h <sub>(4m-2) n</sub>		
m	1	h <sub>11</sub>		h <sub>31</sub>			h <sub>41</sub>		h <sub>(4m-1) 1</sub>		
	m	...	...	...			...	...	h <sub>(4m-1) n</sub>		
				h <sub>1n</sub>					h <sub>(4m) n</sub>		
				h <sub>21</sub>					h <sub>(4m) 1</sub>		
				h <sub>3n</sub>					h <sub>(4m-2) 1</sub>		
				h <sub>2n</sub>					h <sub>(4m-2) n</sub>		
				h <sub>3n</sub>					h <sub>(4m) 1</sub>		
				h <sub>5n</sub>					h <sub>(4m) n</sub>		
				h <sub>6n</sub>					h <sub>(4m-2) 1</sub>		
m	1	h <sub>11</sub>		h <sub>2n</sub>			h <sub>41</sub>		h <sub>(4m) 1</sub>		
	m	...	...	...			...	...	h <sub>(4m-2) n</sub>		
				h <sub>1n</sub>					h <sub>(4m-1) 1</sub>		
				h <sub>21</sub>					h <sub>(4m-1) n</sub>		
				h <sub>31</sub>					h <sub>(4m-1) 1</sub>		
				h <sub>3n</sub>					h <sub>(4m-1) n</sub>		
				h <sub>5n</sub>					h <sub>(4m-1) 1</sub>		
				h <sub>6n</sub>					h <sub>(4m-1) n</sub>		
l-m	1			h <sub>51</sub>					h <sub>(4m-1) n</sub>		

TABLE I  
 $H_{ij}$  - J-TH COLUMN OF LENGTH  $r$  FROM RS CODE CHECK MATRIX

Hence, there are  $N - K - Lr = N - N + lmr - Lr = r(lm - L)$  checking symbols in  $H_{11}$ . This number is the number of columns of the submatrix  $H_{11}$ . This number is also the number of rows of the submatrix  $H_{11}$ .

$H_{00}$  contains all positions corresponding to the informational symbols and some positions corresponding to the checking symbols, while the submatrix  $H_{11}$  contains positions corresponding to all the rest  $tr$

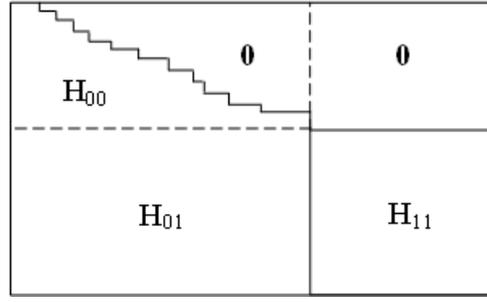


Fig. 1. Matrix H structure

checking symbols.

The main aim of the algorithm is to maximize  $H_{00}$  size (i.e. to maximize the rapid coding trajectory length).

### C. Rapid LDCode RS based encoding

We used an RS systematic check matrix  $\mathbf{H}_{RS} = [\mathbf{P}, \mathbf{I}_r]$ , here  $\mathbf{I}_r$  corresponds to checking positions of the LDCode. Thus, RS encoding is simply multiplication of an information vector and submatrix  $\mathbf{P}$ .

The first step of the encoding procedure is the multiplication of the information vector and submatrix  $H_{00}$ . Since the systematic check matrix is used the encoding is implemented rowwise according to the rapid encoding trajectory. The complexity of the first step is linear to the LDCode length since the trajectory length is proportional to  $m$ .

The second step of the encoding procedure is the equation system solving

$$\mathbf{H}_{11}\mathbf{c}_1^T = \mathbf{H}_{01}\mathbf{c}_0^T = \mathbf{s}^T$$

To solve this system the inverse of the submatrix  $\mathbf{H}_{11}$  is to be precalculated using the following equation

$$\mathbf{c}_1^T = \mathbf{H}_{11}^{-1}\mathbf{H}_{01}\mathbf{c}_0^T = \mathbf{H}_{11}^{-1}\mathbf{s}^T$$

The complexity of the step is considered to be the square of the submatrix  $\mathbf{H}_{11}$  size.

### D. Simulation

The simulation algorithm consists of the following steps: at first, we design a number of trajectories with different random number generator seed values. Then we choose the code corresponding to the trajectory of maximum length. This code is the code with maximum computation complexity payoff (compared with the full matrix inversion technique). The complexity of the proposed algorithm has been compared with that of the full matrix inversion algorithm ( $O(R^2)$ , here  $R$  is the number of checking symbols).

The code distance estimation algorithm is based on the fact that if there exists a code word with  $w$ -weight, the code distance can not be more than  $w$ . Thus we are aiming at finding the code word of the minimal weight for the algorithm proposed. The latter was done by encoding a vector, just  $N_{\text{inf}}$  last symbols of which may be non zero (only  $\nu$  randomly chosen from  $N_{\text{inf}}$  symbols are non zero at each experiment).

We are finally presenting the simulation results.

Rapid encoding algorithm complexity estimation:

$m=208$ , check matrixes number is 1000, trajectories number is 50000 (for each matrix)

Maximal trajectory length is 283 RS block;

Minimal submatrix  $\mathbf{H}_{11}$ (which is to be inverted):

$(4*208)-283=549$  RS blocks \*2 = 1098 symbols;

Average maximal (for each check matrix) trajectory length: 278.9 RS blocks;

Payoff (compared to the full matrix inversion technique)  $\frac{1664^2}{1098^2} = 2.3$  times.  
 $m=256$ , check matrixes number is 1000, trajectories number is 200000 (for each matrix)  
 Maximal trajectory length is 349 RS block;  
 Minimal submatrix  $H_{11}$ (which is to be inverted):  
 $(4*256) - 349 = 675$  RS blocks  $*2 = 1350$  symbols;  
 Average maximal (for each check matrix) trajectory length: 342.90 RS blocks;  
 Payoff (compared with the full matrix inversion technique)  $\frac{2048^2}{1350^2} = 2.3$  times.  
 Upper bound for the LDCode distance:  
 $m=256$ , check matrixes number is 1000, trajectories number is 50000,  $N_{inf}=60$ ,  $\nu=3$   
 Minimal code weight is 1180 symbols.

### III. CONCLUSION

The proposed algorithm provides complexity decrease in the 2.3 times if to compare it with the complexity of the full matrix inversion algorithm. The code distance of the LDCode is upper bounded by 1180 q-ary symbols.

### IV. ACKNOWLEDGMENT

The author are eager to express there profound gratitude to V. B. Afanassiev, A.A. Davydov, D.S. Osipov for their help.

### REFERENCES

- [1] R. G. Gallager, Low Density Parity-Check Codes. Cambridge, MA:MIT Press, 1963.
- [2] S. Freundlich, D. Burshtein, S. Litsyn, Approximately lower triangular ensembles of LDPC PC codes with linear encoding complexity, , ISIT 2006, Seattle, USA, July 9-14, 2006.
- [3] T. Hoholdt, J. Justesen, Graph codes with Reed-Solomon Component codes, ISIT 2006, Seattle, USA, July 9-14, 2006