RLL Codes: Constructions and Extensions

Akash S. Doshi 140010008
Debarnab Mitra 140070037

References

- Roth, Ronald M., Paul H. Siegel, and Jack K. Wolf. "Efficient coding for a two-dimensional runlength-limited constraint." *Proc. of the SPIE-The Intl. Soc. for Optical Engineering*. Vol. 3802. 1999.
- Sankarasubramaniam, Yogesh. *New Capacity-Approaching Codes for Run-Length-Limited Channels*. Diss. Georgia Institute of Technology, 2006.
- Bender, Paul E., and J. K. Wolf. "A universal algorithm for generating optimal and nearly optimal run-length-limited, charge-constrained binary sequences." *Information Theory, 1993. Proceedings. 1993 IEEE International Symposium on*. IEEE, 1993.
- Norris, Kermit, and D. Bloomberg. "Channel capacity of charge-constrained run-length limited codes." *IEEE Transactions on Magnetics* 17.6 (1981): 3452-3455.
- Immink, Kees A. Schouhamer. *Codes for mass data storage systems*. Shannon Foundation Publisher, 2004.

Introduction

- A communication channel is said to be Run-Length-Limited (RLL), if it imposes constraints on runs of consecutive input symbols.
- RLL channels are found in digital recording systems like the Hard Disk Drive (HDD), Compact Disc (CD), and Digital Versatile Disc (DVD). The length of time usually expressed in channel bits between consecutive transitions is known as the runlength. For instance, the runlengths in the word ’0111100111000000’are of length 1, 4, 2 3, and 6.
- The future of RLL sequences is very bright as also in new mass storage products, such as BluRay Disc, a rate 2/3, parity preserving word assignment(parity of both source word and its assigned codeword are the same), dc-free RLL code has been adopted.

Some Prerequisites

- A dk-limited binary sequence, has a **d** constraint - two logical ones are separated by a run of consecutive ’zero’s of length at least d, and a **k** constraint - any run of consecutive ’zero’s is of length at most k.
- A (dk) sequence is converted to a runlength-limited channel sequence in the following way:The logical ’one’s in the (dk) sequence indicate the positions of a transition $1 \rightarrow -1$ or $-1 \rightarrow 1$ of the corresponding RLL sequence. The (dk) sequence 0 1 0 0 0 1 0 0 1 0 0 0 1 1 0 1 ... would be converted to the RLL channel sequence 1 -1 -1 -1 -1 1 1 1 -1 -1 -1 -1 1 -1 -1 1 ... . The mapping of the waveform by this coding step is known as **precoding.**

Definition of RLL Sequences

- Runlength-limited (RLL) sequences are characterized by two parameters, **(d + 1)** and **(k + 1)**, which stipulate the minimum (with the exception of the very first and last runlength) and maximum runlength, respectively, that may occur in the sequence.
- The parameter **d** controls the **highest transition frequency** and thus has a bearing on intersymbol interference when the sequence is transmitted over a bandwidth-limited channel.
- The **k** constraint is important for **timing recovery**. In the transmission of binary data it is generally desirable that the received signal is self-synchronizing or self-clocking. Timing is commonly recovered with a phase-locked loop which adjusts the phase of the detection instant according to observed transitions of the received waveform.

Capacity of (d,k) sequences

- An encoder translates arbitrary user (or source) information into a sequence that satisfies given (dk) constraints. On the average, m source symbols are translated into n channel symbols.
- Rate of the code , $R_{dk} = m/n$
- Capacity of a binary constrained (d.k) sequence C(d,k) is defined as

$$C(d,k) = \lim_{n\to\infty} \frac{1}{n} \log_2 N_{dk}(n).$$

- $N_{dk}(n)$ is the number of (dk) sequences of length n.
- C(d,k) upper bounds the rate R_{dk} of any (d,k) constrained coding scheme

Counting (d,∞) sequences

- Let N_d (n) denote the number of distinct (d,∞) sequences of length n(where ∞ denotes the absence of a constraint on the maximum possible run length)
- Define:

  $N_d(n) = 0, n < 0,$

  $N_d(0) = 1.$
- $N_d()$ is found to follow the recursive relations

  $$N_d(n) = n + 1, 1 \leq n \leq d + 1,$$

  $$N_d(n) = N_d(n - 1) + N_d(n - d - 1), n > d + 1$$
- Solution is of the form

  $N_d(n) = cz^n$, where $c \neq 0$;
- Consequently, z must be a root of the equation

  $$z^{d+1} - z^d = 1$$ (characteristic equation of the (d,∞) code)

Counting (d,∞) sequences

- The most general solution for $N_d(n)$ is of the form

$$N_d(n) = \sum_{i=1}^{d+1} a_i \lambda_i^n,$$

  Where λ_i, $i = 1,\ldots,d+1$ are the distinct d+1 roots of the characteristic equation (for the case it exists).

- For large enough n, we obtain the following

  $N_d(n) \propto \lambda^n$, where λ is the largest root

Capacity of (d,∞) sequences

- Thus we find the capacity as

$$C(d,\infty) = \lim_{n\to\infty} \frac{1}{n} \log_2 N_d(n) = \log_2 \lambda.$$

- The capacity of some well (d,∞) sequences:

d	$C(d,\infty)$
1	0.694
2	0.551
3	0.465
4	0.406

- We see that as d increases capacity decreases.

Capacity of (d,k) sequences

- For n >> k , the ratio $N_{dk}(n+l)/N_{dk}(n)$ should tend to a number z between 1 and 2 , in RLL codes. Thus $N_{dk}(n) = z^n$, and $C(S) = \log_2 z$.
- $N_{dk}(n)$ follows the recursion:

$$\begin{aligned}
&N(n) = n + 1, \ 1 \le n \le d + 1, \\
&N(n) = N(n-1) + N(n-d-1), \ d + 1 \le n \le k, \\
&N(n) = d + k + 1 - n + \sum_{i=d}^{k} N(n-i-1), \ k < n \le d + k, \qquad (4.4) \\
&N(n) = \sum_{i=d}^{k} N(n-i-1), \ n > d + k.
\end{aligned}$$

- Invoking recursion relation (4.4), we can write down the characteristic equation :

$$z^{k+2} - z^{k+1} - z^{k-d+1} + 1 = 0$$

- $C(d,k) = \log_2\lambda$, where λ is the largest real root of the characteristic equation

Capacity of (d,k) sequences

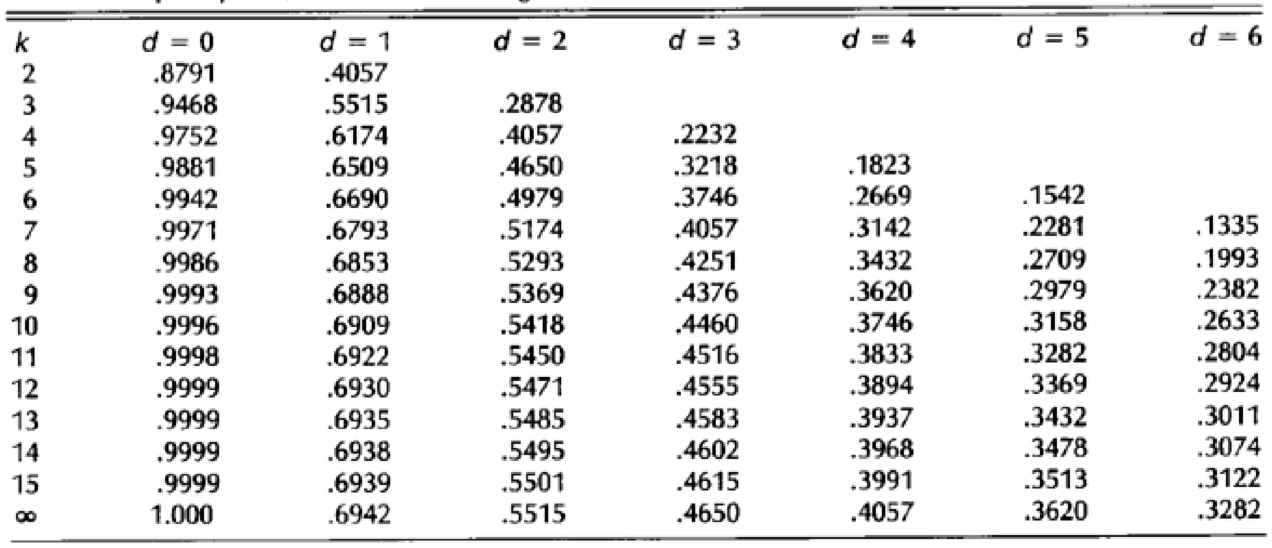


Table 3 Capacity $C(d, k)$ versus Runlength Parameters d and k

k	$d = 0$	$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 6$
2	.8791	.4057					
3	.9468	.5515	.2878				
4	.9752	.6174	.4057	.2232			
5	.9881	.6509	.4650	.3218	.1823		
6	.9942	.6690	.4979	.3746	.2669	.1542	
7	.9971	.6793	.5174	.4057	.3142	.2281	.1335
8	.9986	.6853	.5293	.4251	.3432	.2709	.1993
9	.9993	.6888	.5369	.4376	.3620	.2979	.2382
10	.9996	.6909	.5418	.4460	.3746	.3158	.2633
11	.9998	.6922	.5450	.4516	.3833	.3282	.2804
12	.9999	.6930	.5471	.4555	.3894	.3369	.2924
13	.9999	.6935	.5485	.4583	.3937	.3432	.3011
14	.9999	.6938	.5495	.4602	.3968	.3478	.3074
15	.9999	.6939	.5501	.4615	.3991	.3513	.3122
∞	1.000	.6942	.5515	.4650	.4057	.3620	.3282

- What are the inferences?

Charge Constrained RLL Codes

- These have three constraining parameters (d,k,c). The first two constraints, d and k, put a lower and an upper bound on the runlengths, as before. The third parameter, c, puts an upper bound on the absolute accumulated charge.

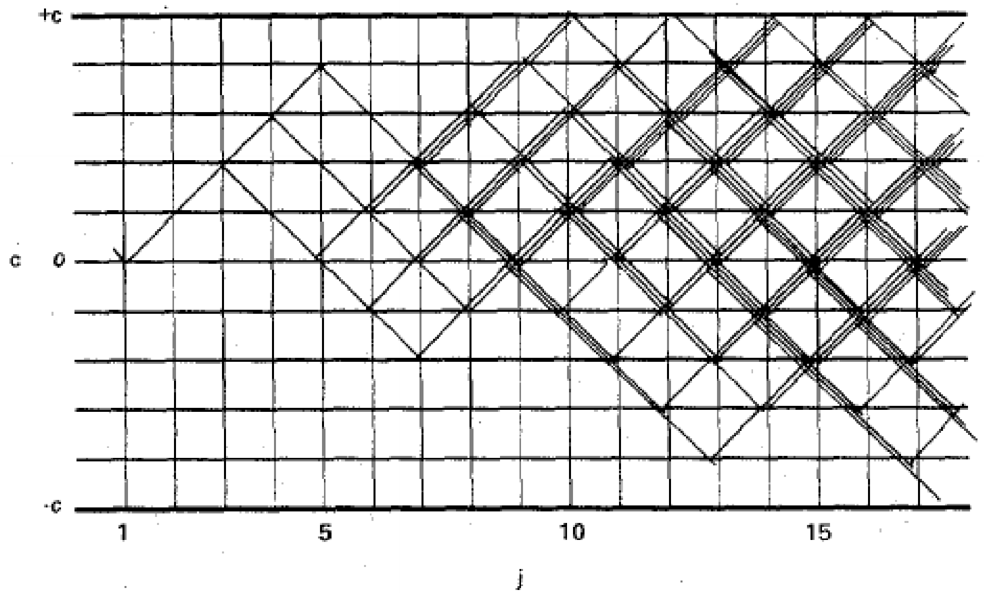


Fig. 3. Charge-trellis diagram showing growth from single upward transition in d, k, c sequence of 1, 3, 5.

- Charge is plotted vertically against message bit and all possible branchings are shown. In order to permit run lengths of k zeroes, the total charge excursion 2c must satisfy:

$$2c \geqslant \begin{cases} k+1, & k \text{ odd} \\ k+2, & k \text{ even.} \end{cases}$$

Channel Capacity of Charge Constrained RLL Codes

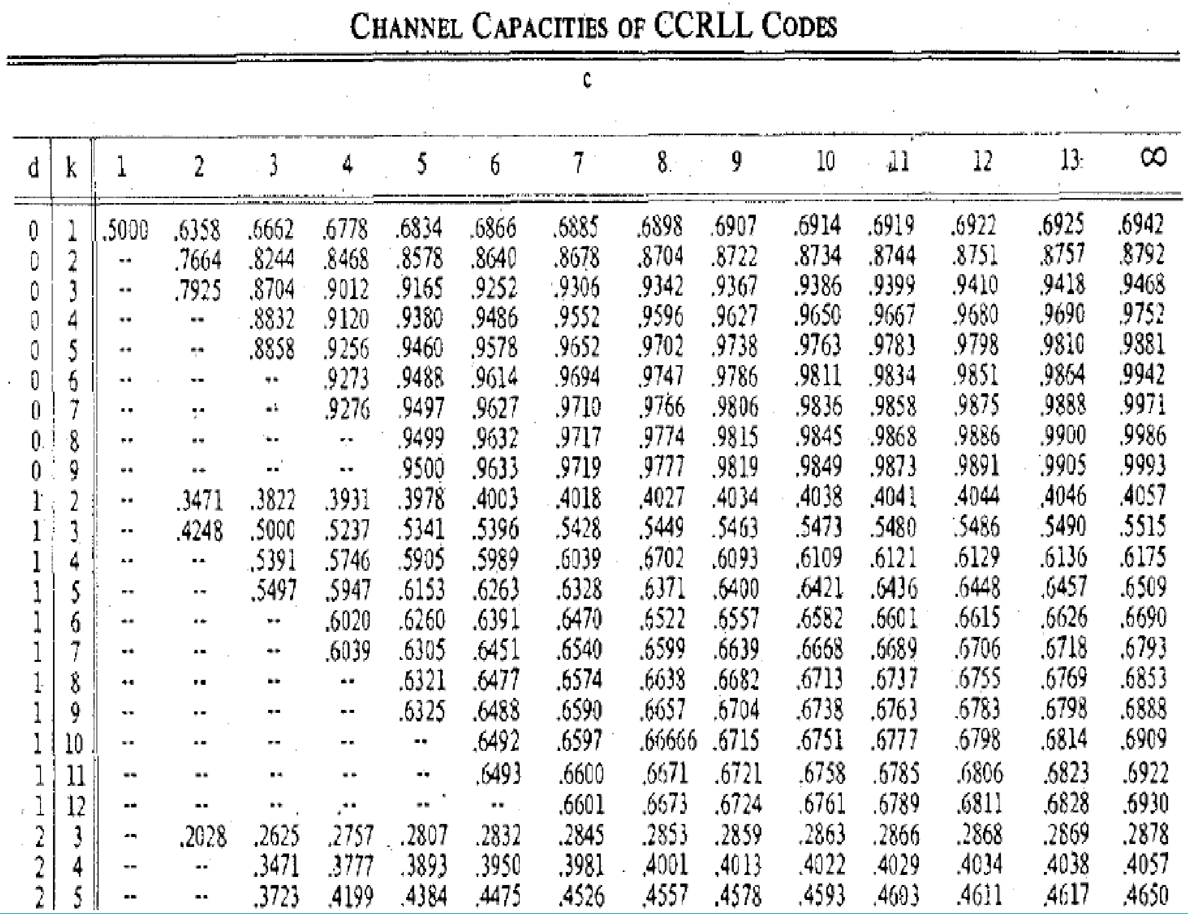
- Let the **number of PSM** with upward and downward directed **transitions** at (c, j) be U(c,j) and D(c,j) ,respectively. For the d, k , c = 1,3, 5 code in Fig. 3, inspection of the diagram gives D(c,j) = U(c-2,j-2) + U(c-3,j-3) + U(c-4,j-4) and U(c,j) = D(c+2,j-2) + D(c+3,j-3) + D(c+4,j-4).
- For sufficiently large j , symmetry about c = 0 yields D(c,j) = U(-c,j). And, as before, we can write D(c,j) = z^iD(c,j-i) and U(c,j) = z^i U(c,j-i).
- Using these facts, dropping reference to j and generalizing to any (d,k), we can write:

$$D(c) = z^{-(d+1)}D(-c+d+1) + z^{-(d+2)}D(-c+d+2)$$

$$\cdot + \cdots + z^{-(k+1)}D(-c+k+1), \quad c = -c, -c+1, \cdots c.$$

- This is a set of N = 2c + 1 homogeneous linear equations. The requirement of a vanishing determinant gives an algebraic equation for z. As before, the channel capacity is the logarithm of the largest root.

Channel Capacity of Charge Constrained RLL Codes



CHANNEL CAPACITIES OF CCRLL CODES

		c													
d	k	1	2	3	4	5	6	7	8	9	10	11	12	13	∞
0	1	.5000	.6358	.6662	.6778	.6834	.6866	.6885	.6898	.6907	.6914	.6919	.6922	.6925	.6942
0	2	--	.7664	.8244	.8468	.8578	.8640	.8678	.8704	.8722	.8734	.8744	.8751	.8757	.8792
0	3	--	.7925	.8704	.9012	.9165	.9252	.9306	.9342	.9367	.9386	.9399	.9410	.9418	.9468
0	4	--	--	.8832	.9120	.9380	.9486	.9552	.9596	.9627	.9650	.9667	.9680	.9690	.9752
0	5	--	--	.8858	.9256	.9460	.9578	.9652	.9702	.9738	.9763	.9783	.9798	.9810	.9881
0	6	--	--	--	.9273	.9488	.9614	.9694	.9747	.9786	.9811	.9834	.9851	.9864	.9942
0	7	--	--	--	.9276	.9497	.9627	.9710	.9766	.9806	.9836	.9858	.9875	.9888	.9971
0	8	--	--	--	--	.9499	.9632	.9717	.9774	.9815	.9845	.9868	.9886	.9900	.9986
0	9	--	--	--	--	.9500	.9633	.9719	.9777	.9819	.9849	.9873	.9891	.9905	.9993
1	2	--	.3471	.3822	.3931	.3978	.4003	.4018	.4027	.4034	.4038	.4041	.4044	.4046	.4057
1	3	--	.4248	.5000	.5237	.5341	.5396	.5428	.5449	.5463	.5473	.5480	.5486	.5490	.5515
1	4	--	--	.5391	.5746	.5905	.5989	.6039	.6702	.6093	.6109	.6121	.6129	.6136	.6175
1	5	--	--	.5497	.5947	.6153	.6263	.6328	.6371	.6400	.6421	.6436	.6448	.6457	.6509
1	6	--	--	--	.6020	.6260	.6391	.6470	.6522	.6557	.6582	.6601	.6615	.6626	.6690
1	7	--	--	--	.6039	.6305	.6451	.6540	.6599	.6639	.6668	.6689	.6706	.6718	.6793
1	8	--	--	--	--	.6321	.6477	.6574	.6638	.6682	.6713	.6737	.6755	.6769	.6853
1	9	--	--	--	--	.6325	.6488	.6590	.6657	.6704	.6738	.6763	.6783	.6798	.6888
1	10	--	--	--	--	--	.6492	.6597	.66666	.6715	.6751	.6777	.6798	.6814	.6909
1	11	--	--	--	--	--	.6493	.6600	.6671	.6721	.6758	.6785	.6806	.6823	.6922
1	12	--	--	--	--	--	--	.6601	.6673	.6724	.6761	.6789	.6811	.6828	.6930
2	3	--	.2028	.2625	.2757	.2807	.2832	.2845	.2853	.2859	.2863	.2866	.2868	.2869	.2878
2	4	--	--	.3471	.3777	.3893	.3950	.3981	.4001	.4013	.4022	.4029	.4034	.4038	.4057
2	5	--	--	.3723	.4199	.4384	.4475	.4526	.4557	.4578	.4593	.4603	.4611	.4617	.4650

As a simple example, consider the biphase (or frequency modulation (FM)) code, with d, k, c = 0, 1, 1. The equations are: $D(-1) = 0$, $D(0) = z^{-1}D(1)$, $D(1) = z^{-1}D(0) + z^{-2}D(1)$. Hence the determinant is:

$$\begin{vmatrix} -1 & 0 & 0 \\ 0 & -1 & z^{-1} \\ 0 & z^{-1} & z^{-2}-1 \end{vmatrix} = 0$$

This has $z = \sqrt{2}$ as the largest root. The channel capacity is exactly 1/2

Bit Stuffing Algorithm for Encoding of CCRLL Codes

- The encoder for the (d,k,c) bit stuff algorithm uses two variables to keep track of the information need to correctly insert the extra bits.
- The first variable, k', keeps track of the current run-length, where a run is a string of consecutive 0's. If k' is ever equal to k , then the encoder inserts a 1 to avoid a possible violation of the k constraint.
- The second variable, c' , keeps track of the accumulated charge. If c' is ever equal to -c + 1 or c -1, then the encoder inserts a 1 to avoid a possible violation of the c constraint.
- Then, after every 1 the encoder inserts d 0's to avoid a possible violation of the d constraint.
- It is shown that the algorithm is optimal for (d,∞,∞), (d,d+1,∞), and (2c-2,∞,c) constraints and nearly optimal for all other (d,k,c) constraints.

Is charge buildup really a concern?

Some communication channels such as perpendicular recording channels may inherently include a DC component in the read back signal. The DC component may complicate and degrade the decoding of the signal requiring tracking of the DC offset. In some cases, the performance of DC offset tracking circuits may degrade by as much as two dB in comparison to the average case

Thus the build-up of DC magnetism or "charge" can be detrimental to a system that cannot support DC. The **c** constraint gives the maximum magnetization or "charge" that can be accumulated along the track.

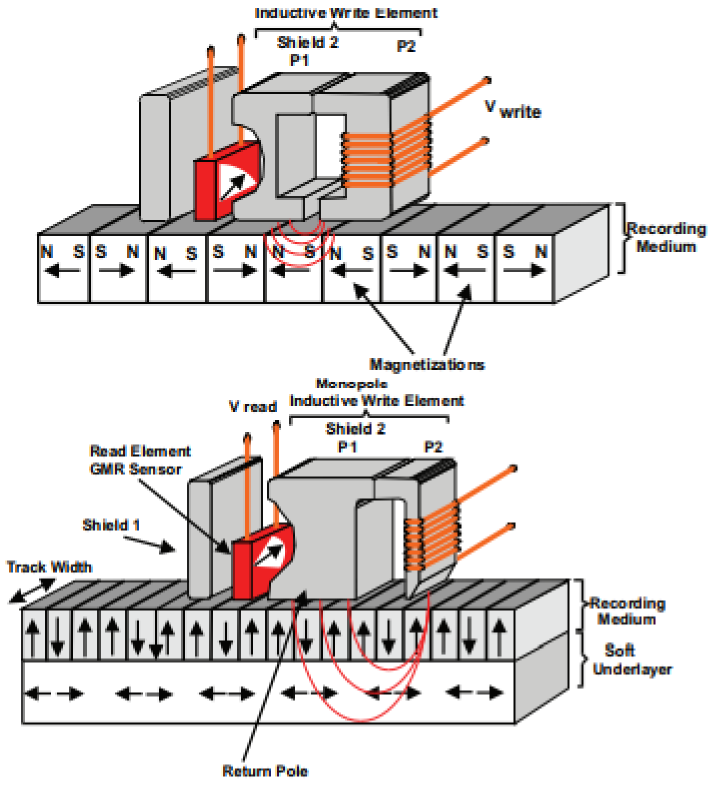


Figure 1: Longitudinal recording diagram (top) and perpendicular recording diagram (bottom)

Efficient Coding for 2D RLL Constraints

- Page-oriented storage technologies, such as holographic storage,employ multidimensional storage. Among the constraints of theoretical and possible practical interest are 2-D, RLL (d, k) constraints, in which the 1-D RLL (d, k) constraint is satisfied both horizontally and vertically.
- Holographic data storage records information throughout the volume of the medium and is capable of recording multiple images in the same area utilizing light at different angles. It is capable of recording and reading millions of bits in parallel, enabling data transfer rates greater than those attained by traditional optical storage.
- We consider coding schemes that map unconstrained binary sequences into two-dimensional, runlength-limited (d, ∞) constrained binary arrays, in which 1's are followed by at least d 0's in both the horizontal and vertical dimensions.

Known bounds on $C_2(1,\infty)$

- If N(m, n : S) are the number of sequences that satisfy the (d, ∞) constraints, the capacity $C_2(d,\infty)$ is

$$C(\mathcal{S}) = \lim_{m\to\infty, n\to\infty} \frac{1}{mn} \log_2 N(m, n : \mathcal{S}).$$

- Calkin and Wilf used a transfer matrix method to derive close lower and upper bounds for the 2-D (d,k) = (1,∞).

$$0.5879 \le C_2(1,\infty) \le 0.5883$$

- These bounds were further improved by Weeks and Blahut, and further improved by Nagy and Zeger, these lower and upper bounds now agree upto 9 decimal places

$$0.587891161775 \le C_2(1,\infty) \le 0.587891161868.$$

Bit stuffing Algorithm in 2-d

- The bit stuffing construction technique falls within 1% of capacity of the constraint.
- Binary digits are inserted into a 2-dimensional array specified by parallelogram $\Delta_{m,n}$
- A parallelogram $\Delta_{m,n}$ is a subset of the integer plane defined by

$$\Delta_{m,n} = \{(i,j) \in \mathcal{Z}^2 : 0 \le i < m,\ 0 \le i+j < n\}$$

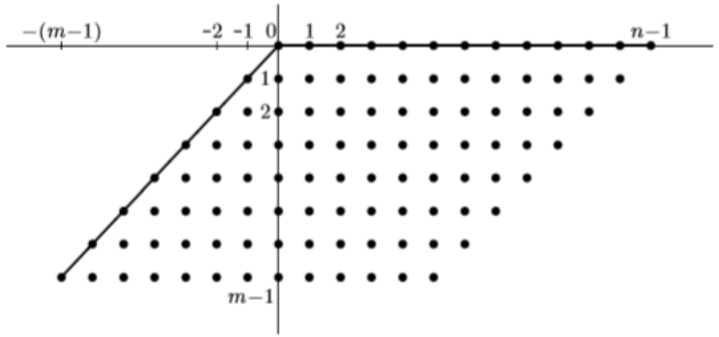


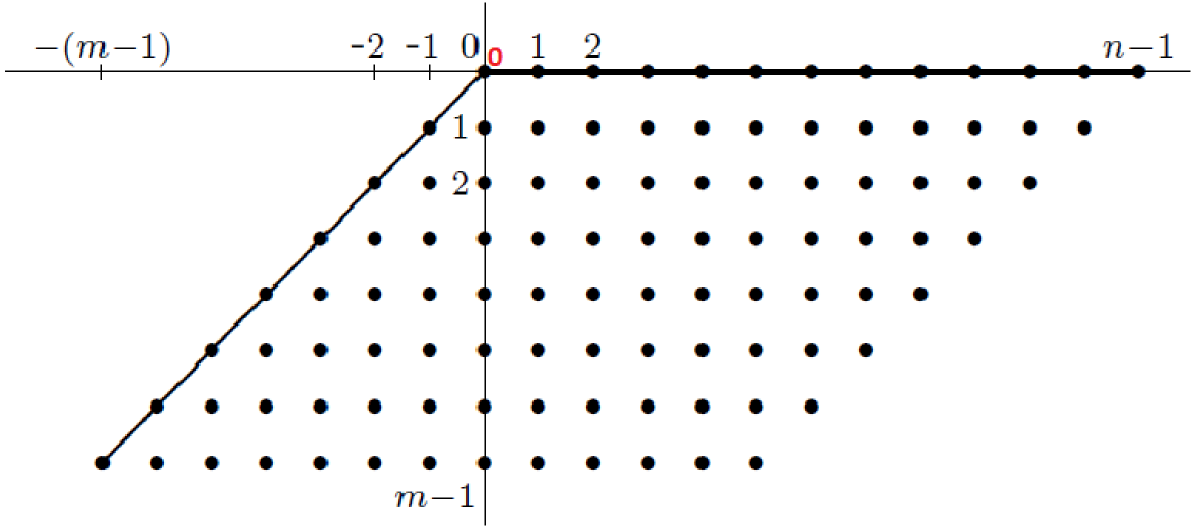
Figure 1. Parallelogram $\Delta_{m,n}$.

- Note: The binary data sequence is first converted by a distribution transformer E to a sequence of statistically independent binary digits with the probability of a 1 equal to p and the probability of a 0 equal to (1−p).

$$R(p) = H_2(p)*\frac{no.of\ unbalanced\ digits}{total\ no.of\ digits} \quad (total\ no.of\ digits = no.of\ unbalanced\ digits + stuffed\ digits)$$

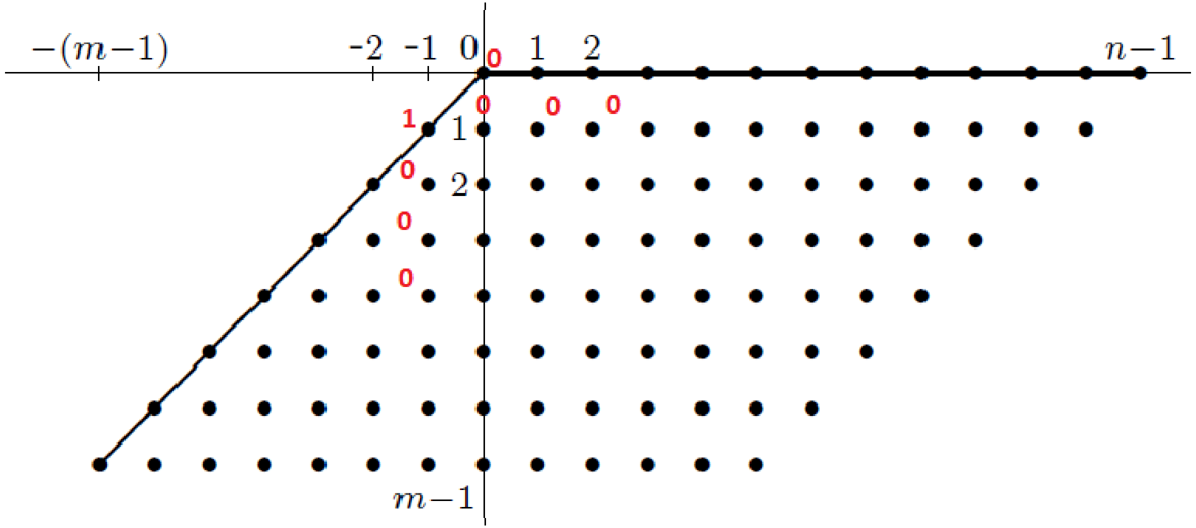
Bit stuffing Algorithm in 2-d - Encoding

- Whenever a 1 in the unbalanced source sequence is written, d 0’s are inserted – or “stuffed”– in the d positions to the right of it and in the d positions below it.
- In writing the unbalanced sequence down diagonals, any position already filled by a previously stuffed 0 is skipped.



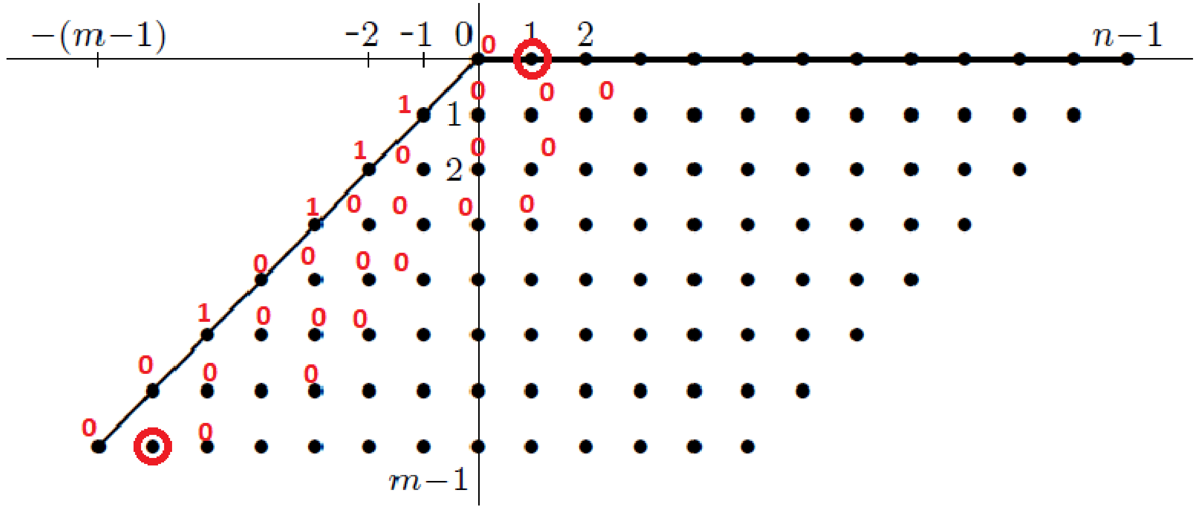
Bit stuffing Algorithm in 2-d - Encoding

- Whenever a 1 in the unbalanced source sequence is written, d 0’s are inserted – or “stuffed”– in the d positions to the right of it and in the d positions below it.
- In writing the unbalanced sequence down diagonals, any position already filled by a previously stuffed 0 is skipped.



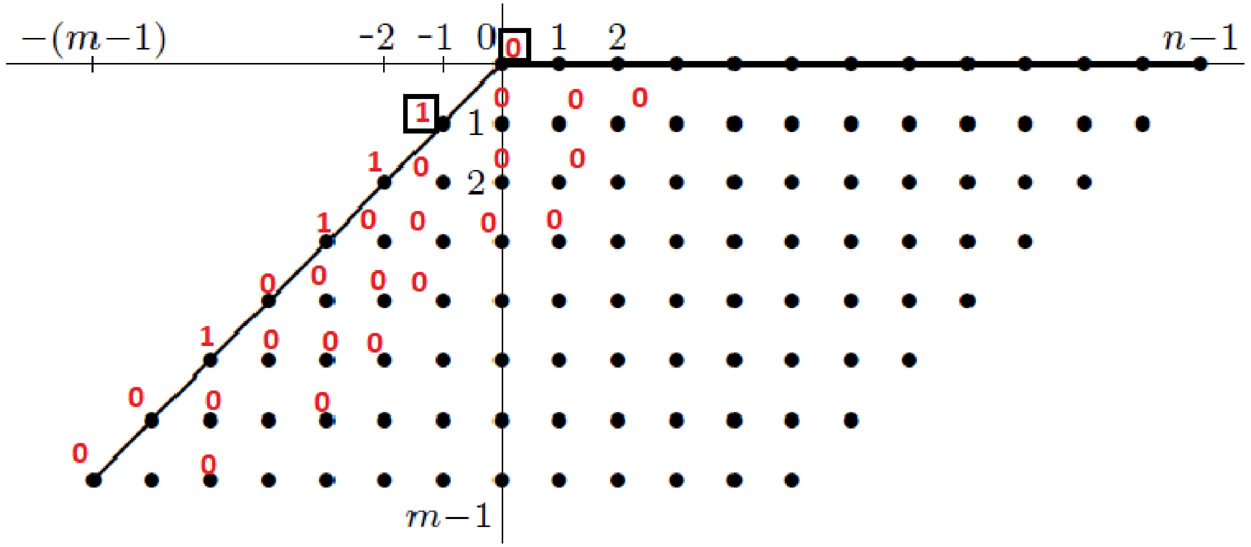
Bit stuffing Algorithm in 2-d - Encoding

- Whenever a 1 in the unbalanced source sequence is written, d 0’s are inserted – or “stuffed”– in the d positions to the right of it and in the d positions below it.
- In writing the unbalanced sequence down diagonals, any position already filled by a previously stuffed 0 is skipped.



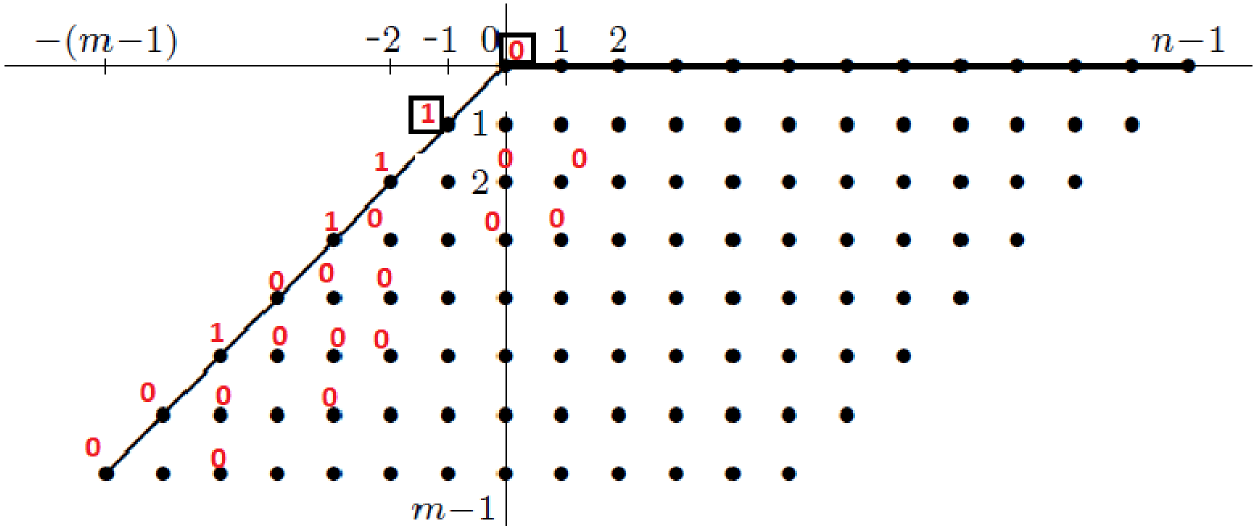
Bit stuffing Algorithm in 2-d - Decoding

- The unbalanced binary digits are read successively from the array, with certain 0 bits being ignored.
- Whenever a 1 is read from the array, the stuffed 0's to the right of it and below it are normally deleted



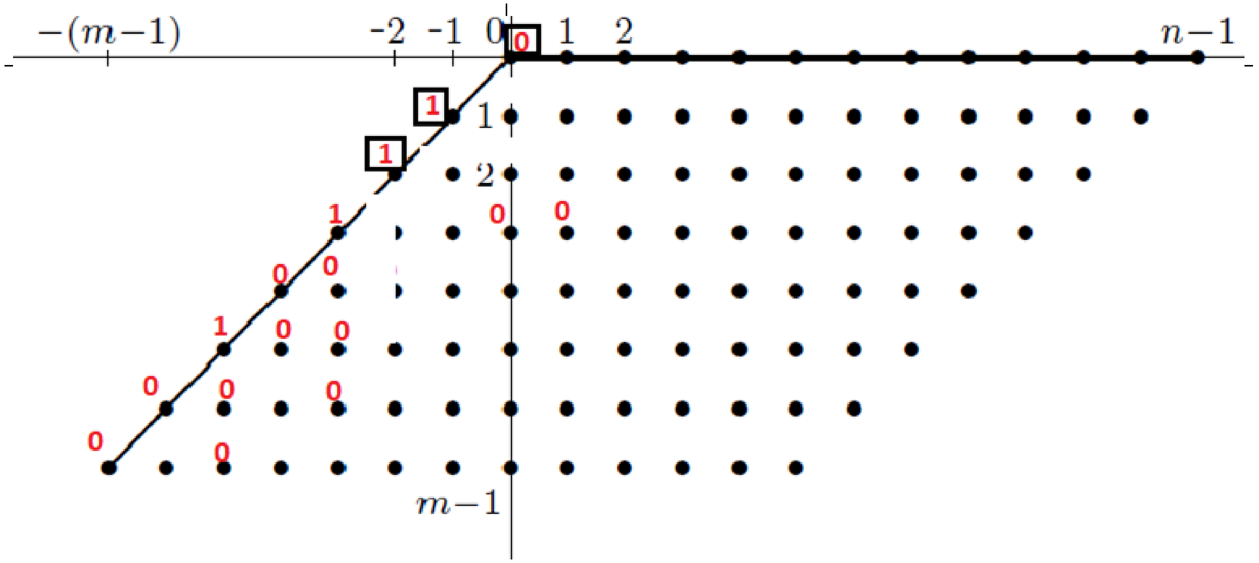
Bit stuffing Algorithm in 2-d - Decoding

- The unbalanced binary digits are read successively from the array, with certain 0 bits being ignored.
- Whenever a 1 is read from the array, the stuffed 0's to the right of it and below it are normally deleted



Bit stuffing Algorithm in 2-d - Decoding

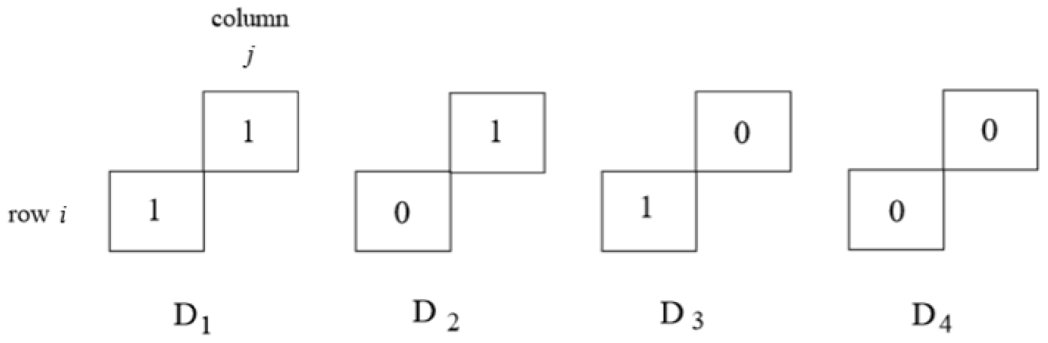
- The unbalanced binary digits are read successively from the array, with certain 0 bits being ignored.
- Whenever a 1 is read from the array, the stuffed 0's to the right of it and below it are normally deleted



Information rate of the bit-stuffing encoder

- Average information rate of the bit-stuff encoder is given by

$$\mathcal{R}(p) = H_2(p)Pr(D_4) = H_2(p)z.$$



$x = Pr(D_1)$ $y = Pr(D_2) = Pr(D_3)$ $z = Pr(D_4)$

- In RLL sequence information is contained is writing a '1' which correspond to writing (0,0) on the diagonals.

$$R(p) = H_2(p) * \frac{no.of\ unbalanced\ digits}{total\ no.of\ digits} \quad (total\ no.of\ digits = no.of\ unbalanced\ digits + stuffed\ digits)$$

- If I(A) is the indicator random variable for (0,0) on a diagonal, then the average information rate

$$R(p) = H2(p) * \frac{E[no.of\ unbalanced\ digits]}{total\ no.of\ digits} = \frac{\Sigma_{all\ digits} I(A)}{total\ no.of\ digits} = \frac{P((0,0\ on\ diagonal) * total\ no\ of\ digits}{total\ no.of\ digits}$$

Information rate of the bit-stuffing encoder

- Now, $\Pr(x_{i,j} = 1) = z - zq$, $P(x_{i,j} = 0) = 1 - z + zq$ and $x+2y+z=1$

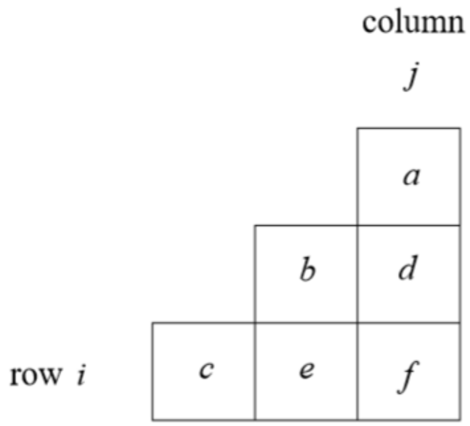
- $z = \Pr(b = 1) + \Pr(b = 0) . [\Pr(a = 1| b = 0)\Pr(c = 1| b = 0) + \Pr(a = 0 | b = 0)\Pr(c = 1 | b = 0) q + \Pr(a = 1|b= 0)\Pr(c = 0|b= 0)q + \Pr(a = 0| b= 0)\Pr(c = 0|b =0)q^2]$

- Using $Pr(a=1|b=0) = \frac{Pr(a=1,b=0)}{Pr(b=0)} = \frac{y}{1-(z-zq)}$

$$Pr(c=1|b=0) = \frac{Pr(a=1,b=0)}{Pr(b=0)} = \frac{y}{1-(z-zq)}$$

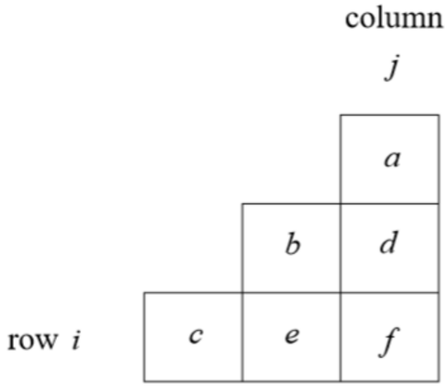
$$Pr(a=0|b=0) = Pr(c=0|b=0) = \frac{z}{1-(z-zq)}$$

- On substitution we get, $y^2 + (2zq)y + z(z-1)q = 0.$



Information rate of the bit-stuffing encoder

- Now the probability of 1 in an entry of any diagonals is x +y, thus x+y = z-zq. Using this and the fact that x+2y + z = 1, we get
$$z = x + y + \frac{(y+zq)^2}{1-(x+y)}.$$



- **Pr(f = 0) = Pr(d = e =1) + Pr(d = 1, e = 0) + Pr(d = e = 0)q** or equivalently **1 - x - y = x + 2y + zq**
- On simplification : $z^2(1-q)(4-3q) - z(4-3q) + 1 = 0.$
- Thus we get the average information rate as

$$\mathcal{R}(p) = H_2(p)\frac{(4-3q)+\sqrt{(4-3q)^2-4(1-q)(4-3q)}}{2(1-q)(4-3q)}.$$

- The value of q which maximizes R(p) is 0.644400 which yields a rate = 0.583056 which is within 1% of capacity of $C_2(1,\infty)$

Recent Developments

- In [Roth, Siegel, 2004] they considering bounds on the entropy of measures induced by bit-stuffing encoders to yield improved lower bounds on the capacity for all $d > 1$.
- In [Roth, 2013], they consider 2D constrained coding based on tiling. For certain constraints, such as the $(0, 2)$-RLL and $(3,\infty)$-RLL constraints, the technique is shown to improve on previously-published lower bounds on the capacity of the constraint.