RANDOM MATRICES AND CODES FOR THE ERASURE CHANNEL

CHRIS STUDHOLME AND IAN F. BLAKE

Abstract. The design of erasure correcting codes and their decoding algorithms is now at the point where capacity achieving codes are available with decoding algorithms that have complexity that is linear in the number of information symbols. One aspect of these codes is that the overhead (number of coded symbols beyond the number of information symbols required to achieve decoding completion with high probability) is linear in k. This work considers a new class of random codes which have the following advantages: (i) the overhead is constant (in the range of 5 to 10), independent of the number of data symbols being encoded (ii) the probability of completing decoding for such an overhead is essentially one (iii) the codes are effective for a number of information symbols as low as a few tens (iv) the only probability distribution required is the uniform distribution. The price for these properties is that the decoding complexity is greater, on the order of $k^{3/2}$. However, for the lower values of k where these codes are of particular interest, this increase in complexity might be outweighed by their advantages. The parity check matrices of these codes are chosen at random as windowed matrices i.e. for each column an initial starting position of a window of length w is chosen and the succeeding w positions are chosen at random as zero or one. It can be shown that it is necessary that $w = O\left(k^{1/2}\right)$ for the probabilistic matrix rank properties to behave as a non-windowed random matrix. The sufficiency of the condition has so far been established by extensive simulation, although other arguments strongly support this conclusion. The properties of the codes described depend heavily on the rank properties of random matrices over finite fields. Known results on such matrices are briefly reviewed and several conjectures needed in the discussion of the code properties, are stated. The likelihood of the validity of the conjectures is supported through extensive experimentation. Mathematical proof of the conjectures would be of great value for their own interest as well of the particular coding application described here.

December 2, 2007

1. Introduction

The construction and analysis of good (capacity achieving) erasure codes have been intensively investigated over the past few years following the ground breaking work of Luby, Shokrollahi, Spielman, Mitzenmacher and many others. Such codes can be used very effectively for large downloads over the Internet in a multicast scenario in the presence of packet losses, where there is no feedback channel, or for distributing and reconstructing files in the presence of server crashes, among other applications. The end result of this work, due to Shokrollahi and Maymounkov, is that we know how to construct erasure codes that can

Date: October, 2007.
1991 *Mathematics Subject Classification.* Primary 54C40, 14E20; Secondary 46E25, 20C20.
Key words and phrases. random matrices, bipartite graphs, erasure codes.
The first author was supported in part by NSERC Grant A632.

be decoded in linear (in number of data symbols) time with high probability of successful decoding. Such results are asymptotic in nature requiring a large number of data symbols to achieve the promised performance. Additionally, the number of extra coded symbols required to ensure a high probability of decoding for block lengths in the few thousands, might typically be on the order of 3 to 4% of the number of data symbols.

One aspect of the work considered here, is to propose a new class of codes with a decoding algorithm that is simple Gaussian elimination. The class of codes will have an encoder complexity of $O(\log k)$ block operations per coded symbol and a decoder complexity of $O\left(k^{3/2}\right)$, where k is the number of data symbols. In return for this increased decoder complexity it is shown that several interesting and useful advantages accrue: (i) the overhead is constant (in the range of 5 to 10), independent of the number of data symbols being encoded and related strongly to the likelihood of complete decoding (ii) the probability of completing decoding for such a small overhead is quite precise and essentially one for the range of overhead mentioned (iii) the codes are effective for a number of information symbols as low as a few tens (iv) additionally, the parity checks for these codes can be obtained from a uniform distribution over subsets of the data symbols, rather than a more complicated distribution used in [12] and [13]. Thus while the codes have a higher decoding complexity, the advantages of small overhead and essentially guaranteed decoding completeness, as well as simplicity of generation, compared to the Online and Raptor codes, are of great practical interest, especially for the more modest block lengths for which they are applicable.

In the next section we briefly review the properties of LT, Raptor and Online codes as background to windowed codes. The properties of the codes described depend heavily on the rank properties of random matrices over finite fields. Known results from the literature on such matrices are briefly reviewed in the succeeding two sections and three conjectures needed in the discussion of the code properties, are stated. The validity of the conjectures is suggested from the results of extensive experimentation. Mathematical proof of the conjectures would be of great value, both for their own interest as well for the particular coding application described here. However, the conjectures will likely be difficult to establish theoretically since the precise functional form of the relations involved may be more subtle than the forms tried. However the conjectures are central to the purpose of presenting this work as any progress on establishing them would be of significant general interest. In particular the properties of the class of windowed codes introduced depends on rank properties of windowed matrices. At this point, the principal property required for the construction of the windowed codes can only be established as a necessary condition. The sufficiency of the condition is again suggested through extensive simulation. Section 5 gives the new encoding and decoding algorithm and relies on the work of section 3 to establish their properties. Note that all logarithms are natural logarithms.

The only other work related to the problem of interest here is contained in the patent [11]. The problem of interest there is the encoding of very large files. To prevent the entire file being required to be in main memory at once, a sliding window approach is taken and thus has a similar view to this work. However we are not aware of any performance or analytical results being available relating to that patent.

2. Coding for the erasure channel

A very brief review of the remarkable recent work on erasure correcting codes is given. While work on low density parity check codes originates from the sixties [6], it was only more recent work that showed how such codes on the binary symmetric channel with errors (a transmitted binary 1 received as a zero or vice-versa) were capable of achieving the elusive goal of capacity. However, on the binary erasure channel a further step was taken to actually show how codes that achieve capacity can be constructed [9, 10, 12, 13, 15]. Only this more limited case of coding for erasure channels is of interest here. Much of this work was concerned with designing (random) codes by constructing a bipartite graph whose incidence matrix is the parity check matrix of a linear code. A requirement of the construction [13] was to design two distributions on the degrees of the left and right vertices of this graph in such a way as to achieve capacity for a given decoding process.

Subsequent to this, the work of Luby [9, 10] introduced the notion of a *rateless* code. Again, this can be thought of in terms of a bipartite graph where the left vertices are identified with, say, k data or input symbols (which we may take as binary symbols for simplicity although the extension to strings of binary symbols is trivial). The right vertices are thought of as parity checks on the input or data symbols or as coded symbols. These are generated as follows: for a given degree distribution on the integers $\rho(i),\ i = 1, 2, \cdots k$, the distribution is sampled to give an integer $d,\ 1 \leq d \leq k$. The corresponding code symbol is formed by choosing d input symbols uniformly at random and summing them (usually via bitwise exclusive or). Information as to which data symbols are involved in forming a given coded symbol is encoded in the header of the coded symbol and this aspect is ignored in the following discussion. For decoding, if a sufficient number of coded symbols are obtained, the process starts by choosing a coded symbol of degree 1 i.e. a code symbol corresponding to a right vertex of degree 1. The value of the code symbol is transferred to the corresponding input symbol, whose value is then transferred to all coded symbols containing it, and all the corresponding edges are removed from the graph. If a new (modified by the above process) right vertex is now of degree 1, the process continues.

Clearly the decoding continues until it completes with all input symbols recovered or there are no right vertices of degree 1 at some stage, in which case the decoding fails. To minimize the probability of the latter, the distribution $\rho(i)$ is chosen carefully. Initially, it was suggested [9] to use the *soliton* distribution given by $\rho(i) = 1/i(i-1),\ i = 2, 3, \cdots, k,\ \rho(1) = 1/k$. The theoretical reasoning for this distribution is sound, but it turns out in practice that the probability, at each stage of the decoding, of having a vertex of degree 1 is too sensitive, giving too high a probability the decoding would not complete. To remedy the situation, a *robust* soliton distribution was suggested that proved more robust in practice. These codes are referred to as rateless since the decoder can continue to gather coded symbols from any source, on the same input data, until decoding completes – there is no fixed *a priori* rate associated with the code. In conventional coding, it would first be necessary to estimate the erasure probability to allow proper code design and this rateless feature is very useful.

For an original set of k data or input symbols, the number of extra (more than k) coded symbols required to give a high probability of decoding completion is referred to as the *overhead*. It is shown in [9] that if δ is the allowable decoding failure probability then by using the robust soliton distribution for the generation of the coded symbols, the overhead required to decode is $O\left(\sqrt{k}\log^2(k/\delta)\right)$ and the average degree of the right vertices is $D = \log(k/\delta)$. Since the complexity of decoding is proportional to the number of edges in the graph, this last result shows the decoding complexity to be $O\left(k\log(k/\delta)\right)$.

To improve the situation further Shokrollahi [13] and independently Maymounkov [12] introduced the idea of *precoding*. Here one uses a good erasure correcting code to code the original input symbols resulting in a certain number of *auxiliary* symbols. These are then added to the input symbols (on the left hand side of the bipartite graph) and the combined set of left hand side symbols are coded using a truncated soliton-like encoder. This latter (outer) code has a mean right hand side degree that is independent of k, giving linear time encoding and decoding. In exchange for this complexity reduction, overhead is increased to $O\left(k\right)$ extra coded symbols. For convenience, we note Theorem 2 from [12], using our terminology. A similar result is found in [13].

Theorem 2.1. *(Maymounkov): For any message of size k input symbols, and any parameter $\epsilon > 0$, there is a rateless locally encodable code (right distribution) that can recover the input symbols from any $(1+\epsilon)k$ coded symbols with high probability in time proportional to $k\log(1/\epsilon)$.*

In the presence of such a powerful result one might wonder why further work on this problem is of interest. One aspect of the above discussion is that the quoted results are essentially asymptotic in nature. In practice, with our simulations and others available in the references, a typical overhead for the LT decoding might be as much as 15% for a number of input symbols on the order of a thousand and this drops to perhaps 3% to 4% for a much higher number of input symbols. The Online or Raptor codes can also be constructed to have an overhead of 3% to 4%, of k for large k.

The class of codes presented here will have a decoding complexity of $O\left(k^{3/2}\right)$ for k input symbols. Although higher than the previous work, for low k the increase is not so large when all the constants in the estimates are taken into account. In return for this increased complexity, one obtains codes that are effective for values of k as low as a few tens of symbols, are easily generated requiring only a uniform distribution, and have a very high and quantifiable probability of decoding completion, depending only on the overhead i.e. the small number of extra coded symbols gathered that is a constant, independent of k. This is to be compared with the Raptor and Online codes which typically are effective for block lengths of several thousands and have a far more intricate coding and decoding procedure, and which utilize certain distributions. It is felt there will be many applications where the properties of the codes presented here are of interest. These results however depend on the experimental results obtained on randomly generated matrices and the conjectures that result from these experiments. These are discussed in the next two sections.

3. Rank properties of random matrices

The rank properties of random matrices over finite fields have been widely investigated as a problem in combinatorial graph theory, but also in other contexts such as coding ([2, 3]). Many of the important results are in the Russian literature and a few of these results are cited here through the English translations, where available. Of the more recent results of which the authors are aware, the papers of Cooper ([4, 5]) and the book [7] are the most relevant to this work. A few of these results are quoted below. We let $M_{k,k+m}(p,q)$ denote a random $k \times (k+m)$ matrix over $\mathbb{F}_q$ where each of the nonzero field elements are chosen equally likely and the probability of a zero element is $1-p$. In the case that $p=(q-1)/q$ all field elements are chosen equally likely. We are generally interested in the case that $m \geq 0$ but for many results allow $-k < m \leq 0$ as well. Denote by $\rho(M_{k,k+m}(p,q))$ the rank of the matrix. In cases where certain of the parameters are understood, they are omitted. The case of equally likely elements, especially in the binary case, will be referred to as the "purely random" case.

The paper of Cooper [5] considers the case of rectangular matrices over an arbitrary finite field $\mathbb{F}_q$. The following two theorems, where the notation is adapted to the above, contain perhaps the strongest results of which the authors are aware:

Theorem 3.1 ([5] theorem 1)**.** *Let $M_{k,k+m}(p=(q-1)/q,\ q)$ be a matrix where all elements of $\mathbb{F}_q$ are equally likely (each having probability $1/q$). Then for $m \geq 0$*

$$\begin{aligned}\lim_{k\to\infty} P(\rho(M_{k,k+m}(p,q)) = k-s) &= \pi_m(s,q) \\ &= \begin{cases} \prod_{j=m+1}^{\infty}\left(1-\frac{1}{q^j}\right), & s=0 \\ \prod_{j=s+m+1}^{\infty}\left(1-\frac{1}{q^j}\right)/\prod_{j=1}^{s}\left(1-\frac{1}{q^j}\right) q^{s(s+m)}, & s \geq 1 \end{cases}\end{aligned}$$

Theorem 3.2 ([5] theorem 2)**.** *For the finite field $\mathbb{F}_q$ let $q \geq 3$ and $q = O(\log\log(k+m))$. Let m be a nonnegative integer and $M_{k,k+m}(p,q)$ a $k \times (k+m)$ random matrix over $\mathbb{F}_q$ with entries independently and identically distributed (zero element has probability $1-p$ and nonzero elements equally distributed with probability $p/(q-1)$) and $k/(k+m) \longrightarrow 1$ as $k \longrightarrow \infty$. Let $c_q = \pi_m(0,q)$ be the asymptotic probability, as $k \to \infty$, the matrix has full rank (k). Let $p(k) = (\log(k)+d(k))/k$ where $d(k) \geq -\log(\log(k/9q))$. Then:*

(i)

$$\lim_{n\to\infty} P(M_{k,k+m}(p=(q-1)/q,q) \text{ is nonsingular}) = \begin{cases} 0 & d(k) \to -\infty \\ c_q e^{-2e^{-d}} & d \text{ constant} \\ c_q & d(k) \to \infty \end{cases}$$

(ii) Let $\mathcal{F}$ be the event there are no zero rows or columns in the matrix. For any nonnegative integer s

$$\lim_{k\to\infty} P(M_{k,k+m}(p=(q-1)/q,q) \text{ has rank } k-s \mid \mathcal{F}) = \pi_m(s,q),$$

and in particular

$$\lim_{n\to\infty} P(M_{k,k+m}(p=(q-1)/q,q) \text{ is nonsingular} \mid \mathcal{F}) = c_q.$$

These two results are generalizations of the following well known result to be found in [7] for binary matrices with equally likely and independently chosen elements. Again, the notation has been adapted to the above.

Theorem 3.3 ([7], page 126)**.** *Let $M_{k,k+m}$ be a binary random $k \times (k+m)$, $m > -k$, matrix with entries chosen equally likely and $\rho(M_{k,k+m})$ its rank (over $\mathbb{F}_2$). Then for $k-s \leq \min(k, k+m)$, $k \geq s \geq 0$, (hence $s+m \geq 0$) as $k \to \infty$ we have*

$$P\left(\rho(M_{k,k+m}) = k-s\right) \to 2^{-(s)(m+s)} \prod_{i=s+1}^{\infty} \left(1 - \frac{1}{2^i}\right) \prod_{i=1}^{m+s} \left(1 - \frac{1}{2^i}\right)^{-1}, \tag{3.1}$$

where the last product is 1 if $m+s=0$ (i.e. the matrix is of full rank).

It follows from this last result that the probability a random $k \times (k+m)$ binary matrix with elements equally likely, $M_{k,k+m}$ is of full rank k for $m \geq 0$ is

$$Q_m = \prod_{i=m+1}^{\infty} \left(1 - \frac{1}{2^i}\right), \quad m = 0, 1, \cdots, \tag{3.2}$$

and it is noted in [1] that this result is credited to [8]. For $m < 0$ the probability of a $k \times (k+m)$ random binary equally likely matrix being of full rank $(k+m)$ is

$$Q_m = \prod_{i=1}^{k+m} \left(1 - \frac{1}{2^{k-i+1}}\right).$$

A useful rapidly converging form ([1]) of this equation is

$$\log Q_m = \sum_{i=m+1}^{\infty} \log(1 - 2^{-i}) = \sum_{j=1}^{\infty} \frac{-2^{-mj}}{j(2^j - 1)} \quad s \geq 0.$$

In particular it is noted that

$$Q_0 = \prod_{j=1}^{\infty} \left(1 - \frac{1}{2^j}\right) = 0.2887880951 \cdots \tag{3.3}$$

i.e. the probability a square random binary matrix is nonsingular as its dimension tends to infinity. As will be noted later, this expression is very accurate even for k as low as 10.

Denote by P_m, $m \geq 0$, the probability that exactly m columns beyond k are needed to obtain rank k for the $k \times (k+m)$ equally likely random binary matrix, then

$$P_m = Q_m - Q_{m-1}$$

and the average number of extra columns needed for full rank is

$$\bar{m} = \sum_{m=0}^{\infty} m P_m = \sum_{i=0}^{\infty} (1 - Q_i) = 1.60669515 \cdots$$

This result will be of value for the windowed codes to be discussed later.

The Theorem (3.2) indicates that the rank properties of (both binary and non-binary) $k \times (k+m)$ random matrices are relatively insensitive to the probability p of a nonbinary element, as long as p behaves asymtotically as in Theorem 3.2 and, in particular, $p >$

$\log(k)/k$. Informally, this threshold can be seen to arise from consideration of the probability of zero rows or columns appearing. Thus if the probability of a one in such a binary random matrix is given by $p = c\log(k)/k$ then the probability of an all zero column is given by

$$(1-p)^k = \left(1 - \frac{c\log(k)}{k}\right)^k \longrightarrow e^{-c\log(k)} \longrightarrow \frac{1}{k^c}$$

and as $c < 1$ decreases the probability becomes significant for fixed k. The condition is related to part (i) of Theorem 3.2. Arguing informally, if one allows the quantity $d = \pm\log(k)$, then

$$c_q e^{-2e^d}$$

tends to c_q or 0 depending on whether $+$ or $-$ is chosen.

Extensive experimentation was done on the hypothesis of the relative independence of the rank formulae of the previous theorems with the value of p, for $p > \log(k)/k$. In figure 1 the probability of full rank curves for values of $p = .1, .2, \cdots, .5$ are shown for $k = 100$ and 500. The results are indistinguishable from the equally likely case, even for the relatively low values of k. Note that even for $k = 100$ we have $\log(100)/100 < .1$. We further support this by the Figure 2 where, for $k = 100$ and 500, probabilities $p = c\log(k)/k$ are used for various values of $c < 2$.

As already commented on, when p falls much below $2\log(k)/k$ the probability of zero rows and columns begins to become significant and the probability of full rank given by Equation 3.2 no longer holds. An approximate way of correcting for this is as follows. We first enumerate full rank matrices on the number of zero columns – the probability a column is all zeros is $(1-p)^k$, and then condition on there being no zero rows. Hence the probability a $k \times (k+m)$ matrix is of full rank is

$$(1-(1-p)^{k+m})^k \sum_{j=0}^{m} \binom{k+m}{j} (1-p)^{kj} (1-(1-p)^k)^{k+m-j} Q_{m-j} \tag{3.4}$$

where the first term is the fraction of all $k \times (k+m)$ matrices that have no all-zero rows, and the term after the binomial coefficient is the probability of exactly j all-zero columns. and Q_{m-j} is the fraction of all $k \times (k+m-j)$ matrices that are nonsingular. The overall expression is an approximation since it assumes the probabilities of all-zero rows and columns are independent. For values of p greater than $2\log(k)/k$ experimental results for such probabilities are virtually identical to those of the previous expression Equation 3.2. Interestingly, as shown in Figure 2, Equation 3.4 remains valid even for probabilities that are considerably less than $\log(k)/k$.

With these observations the following conjecture is formulated:

Conjecture 3.4. *For sufficiently large k, if the elements of a random binary $k \times (k+m)$, $m \geq 0$ matrix $M_{k,k+m}$ are chosen independently and identically at random to be 1 with probability p, then their rank properties are indistinguishable from the purely random case ($p = 1/2$), as long as $1/2 > p > 2\log(k)/k$.*

Notice that the conjecture is somewhat imprecise. It is certainly true asymptotically since in that case it is covered by Theroem 3.2. What the conjecture is intended to convey is that for "large" but finite values of k the rank properties of the random binary $k \times (k+m),\ m \geq 0$ matrices are relatively insensitive to the precise values of p and that as long as this probability satisfies the values noted, they behave precisely as for the purely random case. The value of $p = 2\log(k)/k$, from extensive simulation, does seem to be relatively accurate since even slightly lower values showed deviations from the purely random case for all values of k and m tried. However this seems not sufficient evidence to draw such a strong conclusion since the exact form of p could easily involve more subtle terms and it is likely to be a difficult exercise to verify this conjecture theoretically. Nonetheless the authors felt the problem was of great interest for further examination.

Note that Theorem 3.2 establishes asymptotic properties that hold provided that $p > \log(k)/k$. The indistinguishability from the purely random case requires the higher value noted. The conjecture is closely related to previous results of [4, 5, 7] although the perspective here seems sufficiently distinct to report on it separately.

It has been observed that the average number of columns, beyond k, for a purely random $k \times (k+m)$ matrix to achieve full rank with high probability, is $1.60669515\cdots$. Based on the above observations, the following conjecture is suggested:

Conjecture 3.5. *If the elements of a random binary $k \times (k+m)$ matrix $M_{k,k+m}$ are chosen independently and identically at random to be 1 with probability p, then the expected number of columns beyond k required to achieve full rank (k), $\bar{m}$, is, asymptotically as $k \to \infty$*

$$\bar{m} = 1.60669515\cdots \text{ (as in section 3) for } 2\log(k)/k < p < 1 - 2\log(k)/k. \tag{3.5}$$

The behaviour of $\bar{m}$ for $p = c\log(k)/k$ for $1 < c < 2$ would be of interest. It is conjectured further that values of p in this range give $\bar{m}$ as in the equation.

Notice that from the work of Cooper [4, 5] that $\bar{m} = \infty$ for $p < \log(k)/k$.

4. Rank properties of windowed random matrices

For application to the construction of codes, to be discussed in section 5, we will be interested in *windowed binary matrices*, where the nonzero elements of each column are restricted to fall within a window of length w, beginning at a randomly chosen row. Specifically, to add a column to the matrix, choose a row number at random place a one in that position and fill the succeeding w elements at random, with the probability of a 1 being p. If the initial row is chosen within w of the bottom row, the column will wrap around to the top of the matrix. The item of interest is to relate the window size, probability of a one p, and the rank properties.

To begin, we give an argument that suggests a lower bound on the window size for the rank properties to be similar to those of a random unwindowed $k \times k$ matrix. Let us divide the matrix into *top columns*, where the initial position is chosen among the top $k/2$ rows, and *bottom columns*. Suppose the number of top columns is m_0 and bottom columns m_1 ($m_0 + m_1 = k$). It is clear that if either m_0 or m_1 exceeds $k/2 + w$ the matrix cannot achieve full rank, since in that case, either the columns of the top half or bottom half, cannot achieve full rank. Furthermore, if this happens for one, the other will be less than

$k/2-w$, and therefore, a necessary condition for full rank is that

$$\frac{k}{2}-w \;\leq\; m_0 \;\leq\; \frac{k}{2}+w.$$

The random variable m_0 is binomially distributed with mean $k/2$ and variance $\sigma=\sqrt{k}/2$. For k large, we use the normal approximation to the binomial, let

$$z=\frac{m_0-(k/2)}{\sqrt{k}/2}\;,$$

and note that

$$k/2-w\leq m_0\leq k/2+w \;\;\Rightarrow\;\; \frac{-2w}{\sqrt{k}}\leq z\leq\frac{2w}{\sqrt{k}}.$$

If we choose $w=\sqrt{k}$ the probability needed is the probability a zero mean unit variance normal variate falls in the interval $(-2,2)$ which is approximately .95. In this case the probability the windowed matrix achieves full rank is upper bounded by $.95\times .288788\cdots$. It is this consideration that leads the authors to believe that the window size must necessarily be of size $w=O\left(k^{1/2}\right)$ in order for the resulting matrix to achieve full rank. In general we have:

Theorem 4.1. *For sufficiently large k, the probability that a $k\times k$ random, windowed binary matrix with window length $w=\delta\sqrt{k}/2$ has rank k is at most $2\Phi(\delta)Q_0$, where $\Phi(z)$ is the normal distribution function and $Q_0=.288788\cdots$ as given by equation 3.2.*

We describe some experiments on windowed binary random matrices. The extensive experimentation done on this case suggests that the window size must be at least $2\sqrt{k}$ and the mean weight of the columns must be (as before, in the unwindowed case) at least $2\log(k)$.

In Figures 3 and 4 values of k vary from 100 to 10000 and $p=1/2$. With the possible exception of $k=100$, the rank behaviour is as for a random (unwindowed) matrix with impressive accuracy. For a random matrix with $p=2\log(k)/k$, it has been observed the rank behaviour is as for a purely random matrix – and hence has an average column weight of $2\log(k)$. To emulate this situation for windowed matrices, the case where mean column weight of $2\log(k)$ is included in Figure 4. Again, with the small exception of $k=100$, which appears to be too small to be accurately modeled by these questions, the agreement with purely random case for window sizes of at least $2\sqrt{k}$ and mean column weight of at least $2\log(k)$ is very good. Figures 5 and 6 further support this statement and more extensive results are given in [14].

The conclusions of this experimental evidence suggests that windowed matrices with window size at least $w\geq 2\sqrt{k}$ and mean column weight greater than $2\log(k)$ give probabilities of full rank as those of purely random matrices. The following conjecture is suggested:

Conjecture 4.2. *The rank properties of a binary windowed $k\times(k+m)$, $m\geq 0$ random matrix, for sufficiently large values of k, behave as a purely random binary matrix if the window length is at least $2\sqrt{k}$ and the probability of an element being 1 is at least p, where p is chosen to give a mean column weight of at least $2\log(k)$.*

The simulation done on this windowed matrix problem has strongly supported this conjecture in that window sizes of slightly smaller widths, of the form $2\sqrt{k}$ less a small constant, show markedly lower probabilities of full rank. It has to be admitted however that the actual required window size for full rank may have a functional form that is more subtle than reported here. Again, a theoretical investigation of this problem would be of considerable interest.

One interpretation of the conjecture is that the extra condition of restricting the nonzero elements in a row to be in a window, does not seem to be as significant as requiring the overall probability the element is nonzero, in the binary case, to be at least $2\log(k)/k$.

These matrices will be used in the next section to construct a class of codes that are efficient in terms of coding and decoding complexity while maintaining very low overhead. The results of more extensive simulations on these questions are available in an unpublished report [14]

5. Windowed matrices for erasure codes

The windowed random binary matrices discussed in the previous section can be used as the basis for an efficiently encodable and decodable erasure code. The code we describe here has an encoder complexity of about $2\log k$ block sums per output block and a decoder complexity of $1.3k^{3/2}$ block sums. This complexity is achieved while keeping the overhead constant at approximately 2 extra blocks.

5.1. **Encoding.** The encoder has k data symbols (input blocks) that need to be transmitted to the decoder. To create an output block, the encoder first generates a windowed column as described in the previous section. We suggest the encoder generate fixed weight columns with $\sigma = \lceil 2\log k \rceil_{odd}$. This notation is used to refer to the lowest odd integer greater than or equal to $2\log k$. Note that k columns each having even weight cannot have rank k, and therefore, σ must be odd or the decoder will always fail.

Along with generating a windowed column, the encoder sums (using bitwise exclusive or) the σ input blocks corresponding to the 1's in the column. The column (or some efficient encoding of it) along with the sum are then transmitted to the decoder.

Although setting $w = 2\sqrt{k}$ will work reasonably well, we suggest one instead set $w = 2(\sqrt{k}-1)(\sigma-1)/(\sigma-2)$. This slightly larger value ensures that the expected number of rows between the first and last 1 (inclusive) in each column is $2\sqrt{k}$.

5.2. **Decoding.** The decoding algorithm is simply *Gaussian Elimination*; however, to ensure our discussion of decoder complexity is precise, we describe a specific decoding algorithm. Decoding has two phases:

(1) **Column collection.** During column collection, matrix columns, along with their associated data blocks, are inserted into a hash table and reduced as necessary to ensure they are independent.
(2) **Back filling.** At the conclusion of column collection, we have a lower triangular matrix with 1's along the diagonal. This matrix is non-singular. With a series of data block sums, the matrix is made diagonal and decoding is complete.

The hash table constructed during the first phase is to have exactly k bins and each column received hashes to the bin whose index is the index of the first row containing a 1. For the purposes of this algorithm, columns are not considered to wrap from bottom to top, and as a result, the first 1 in a column may not coincide with the initial 1. When this table is full, the k columns comprise a lower triangular matrix with 1's along the diagonal.

Hash collisions will occasionally happen during column collection. To resolve such a collision, simply add the two columns (and their associated data blocks) together to get a column that hashes to a different bin. A subtle but important detail of this algorithm is the choice of columns to keep after collision resolution. Obviously, the sum is to be kept. The other column to keep is the *shorter* of the two colliding columns. Here, the length of a column is the number of rows between the first 1 and the last 1 (inclusive). If the two columns are of equal length, either may be kept. Two identical columns indicate a dependency; one is simply discarded and an extra column must be collected.

When the hash table is full, back filling can begin. Back filling is done starting at the last row and working up through the matrix. First, the data block associated with the 1 on the diagonal is added to all of the data blocks associated with 1's in the last row. Then, the second to last row is processed in a similar manner. At the completion of back filling, the data blocks will be the original input blocks.

Theorem 5.1. *Worst case decoder complexity is $\bar{\ell}k$ data block additions, where $\bar{\ell}$ is the mean column length. Column length, ℓ, as mentioned earlier in this section, is the number of rows between the first* 1 *and the last* 1*, inclusive.*

Proof. During the column collection phase, one data block addition is required each time there is a hash table collision. If two columns, one of length x and the other of length y, $x \leq y$, collide, their sum will be a column whose length is no greater than $y-1$. Since $\bar{\ell}k$ is the sum of the lengths of the columns and each collision reduces this total length by at least 1, there can be at most $\bar{\ell}k$ collisions.

During the back filling phase, the number of data block additions needed is exactly the weight of the matrix (after column collection) less k. Also, the weight of the matrix is no greater than the total length, and the total length after column collection no greater than the total length before column collection less the number of collisions. Therefore, the sum of the weight of the matrix after column collection and the number of collisions resolved during column collection is at most $\bar{\ell}k$. ☐

The average case complexity is $\bar{\ell}k/2$. This follows from the fact that when columns of length x and y, $x \leq y$, are added, the expected length of the resulting column is $y-2$. Furthermore, the expected weight of the matrix after column collection is half the total length.

To see how the complexity may be calculated from w, first notice that for columns that do not wrap, $\ell \leq w+1$; however, for columns that do wrap, ℓ may be as large as k. Since the probability of generating a column that wraps is w/k, mean column length can be estimated as $\bar{\ell} = (w/k)k + (1-w/k)(w+1) \approx 2w$. When using $w \approx 2\sqrt{k}$ as suggested, expected decoder complexity is $wk \approx 2k^{3/2}$.

Figure 8 shows that for k between 100 and 10,000 we achieve a decoder complexity of $1.3k^{3/2}$ while maintaining an overhead of about 2 extra symbols. For an idea of real world performance we performed two tests involving a 32 MiByte file on an AMD Athlon XP 1800+. Decoding 8192 blocks (4 kiB each) requires about 10 seconds of CPU time, while decoding 65536 blocks (512 bytes each) takes 2 minutes.

Finally, we note that allowing columns to wrap from bottom to top increases decoder complexity by a factor of 2. We have developed a strategy to avoid such wrapping; however, we find that to maintain a high probability of full rank, w must be as large as $4\sqrt{k}$ for some columns. This negates most of the benefit of avoiding wrapping, and therefore, we do not describe the strategy here.

6. Comments

The paper has introduced a class of windowed random binary erasure codes. These have an increased decoding complexity of $O\left(k^{3/2}\right)$ compared to recently constructed codes which have a linear complexity. However they have certain advantages; namely they achieve complete decoding with probability arbitrarily close to one, with overhead of only a few symbols, independent of the number of data symbols and using only a uniform distribution for code symbol generation, and these properties remain valid for even very short block lengths of a few tens. This is to be compared to the other codes mentioned that require block lengths of several thousands for effectiveness. The properties described make the windowed codes very attractive for applications. It should be noted that the properties of the shorter length codes do not depend on the validity of the conjectures - they have been shown to be valid for code lengths of to around 10,000 by the simulations. The asymptotic validity of the conjectures would verify the code properties at very long lengths, but in this region the other codes mentioned might well be more attractive.

The properties of the matrices used rely on rank properties of random matrices which were discussed in some detail. These rank properties were investigated with extensive simulation which led to the formulation of three conjectures. The mathematical proof of these conjectures would be of considerable independent interest. As noted further experimental results are available in the unpublished report [14].

Acknowledgement: The second author would like to thank Michael Luby of Digital Fountain Inc. for a useful discussion of the approach taken in this work and for pointing out reference [11]. The authors would like to thank a reviewer of the first version of this paper for their perceptive comments.

References

1. E.R. Berlekamp, The technology of error correcting codes, *Proc. IEEE*, vol. 68, pp. 564-593, 1980.
2. Johannes Blömer and Richard Karp and Emo Welzl, The rank of sparse random matrices over finite fields, *Random Structures and algorithms*, vol. 10, pp 407–419, 1997.
3. Béla Bollobas, *Random Graphs*, 2nd edition, Cambridge, 2001.
4. C. Cooper, On the rank of random matrices, *Random Structures and Algorithms*, vol. 16, no. 2, pp. 209-232, 2000.

5. C. Cooper, On the distribution of rank of a random matrix over a finite field, *Random Structures and Algorithms*, vol. 17, pp. 197-212, 2000.
6. R.G. Gallager, *Low Density Parity Check Codes*, MIT Press, 1963.
7. V.F. Kolchin, *Random Graphs*, Cambridge University Press, 1999.
8. G. Landsburg, Uber eine Anzahlbestimmiung und eine damit zusammenhangende Reihe, *J. Reine Angew. Math.* vol. 111, pp. 87-88, 1893.
9. Michael Luby, Michael Mitzenmacher, M. Amin Shokrollahi, Daniel Speilman and Volker Stemann, Practical loss-resilient codes, *STOC*, pp. 150–159, 1997.
10. M. Luby, LT codes, *STOC*, 2002.
11. M. Luby et al, *On demand encoding with a window*, United States Patent and Trademark Office, Patent No. 6,486,803, November 26, 2002.
12. P. Maymounkov, Online Codes, Technical Report TR2002-833, New York University, October 2002.
13. Amin Shokrollahi, Raptor Codes, *IEEE Trans. Information Theory*, vol. 52, pp. 2552-2567, 2006
14. Chris Studhome and Ian F. Blake, Properties of random matrices and applications, Unpublished report available at `<http://stoa.dhs.org/~cvs/coding/>`.
15. Omid. Esmaeli and Amin Shokrollahi, Raptor codes for the binary memoryless symmetric channels, *IEEE Trans. Information Theory*, vol. 52, pp. 2033-2051, 2006

Department of Computer Science, University of Toronto, Toronto, Ontario M5S 3G4
E-mail address: `cvs@cs,utoronto.ca`

Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario M5S 3G4
E-mail address: `ifblake@comm.utoronto.ca`

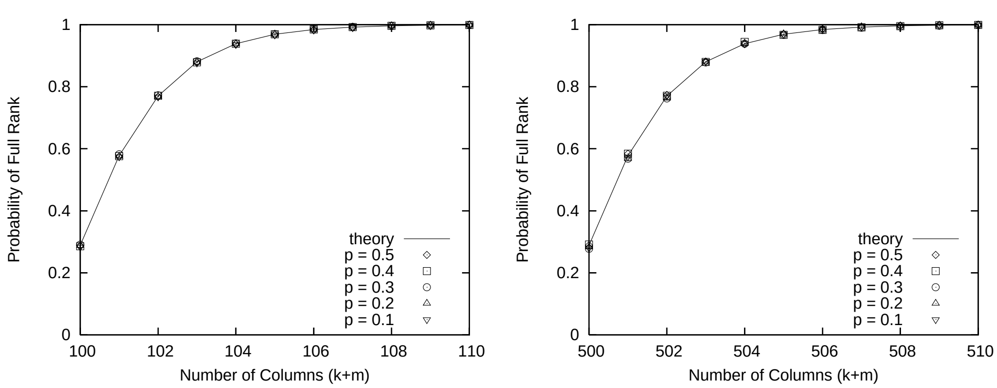

FIGURE 1. Probability of rank k for a $k \times (k+m)$ matrix. The left graph shows the results for $k = 100$, while the right is for $k = 500$. In both graphs, the theoretical expression of Equation 3.2 is shown versus experimental results for the probability of a nonzero element $p = 0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5$. Each point of the experimental results is for $50,000$ trials for $k = 100$ and $5,000$ trials for $k = 500$.

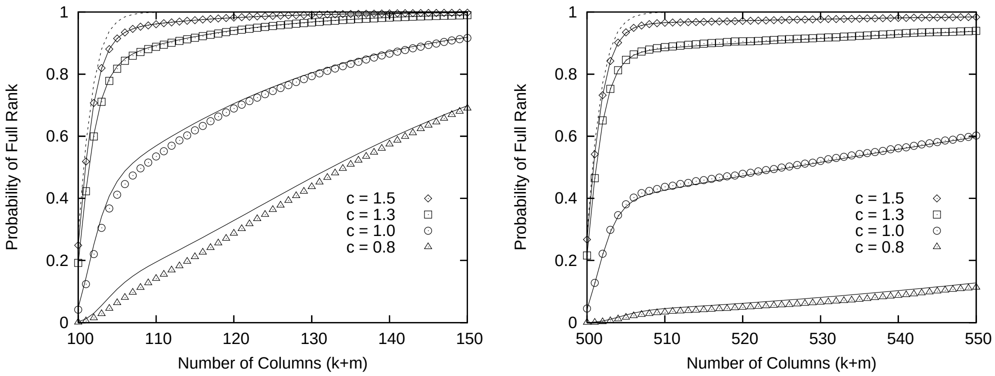

FIGURE 2. Probability of full rank for $p = c\log(k)/k$ as the number of extra columns increases. Each data point is the result of $50,000$ trials (for $k = 100$) and $5,000$ trials (for $k = 500$). The solid line is computed using Equation 3.4 and the dotted line (purely random matrix with constant $p > 2\log(k)/k$) for Equation 3.2.

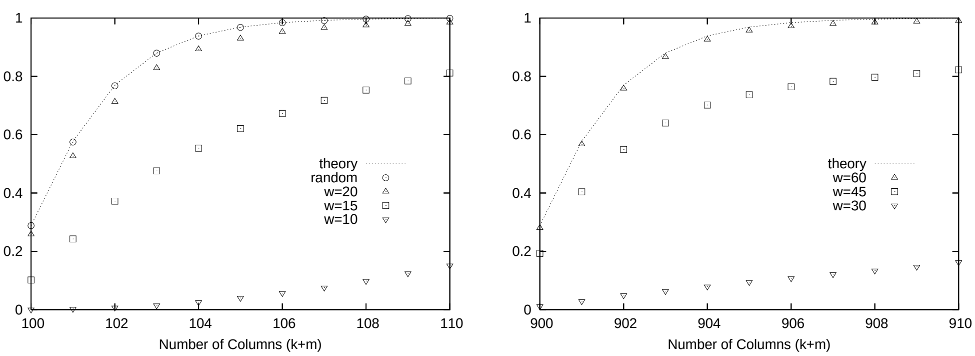

FIGURE 3. Probability of rank k for a $k \times (k+m)$ matrix. The left graph shows the results for $k = 100$, while the right is for $k = 900$. In both graphs, the dotted line is Q_m (Equation 3.2), the open circles show results from random ($p = 1/2$) matrices, and the other symbols show results for windowed matrices (with the specified window size). For clarity, no open circles are shown on the right graph.

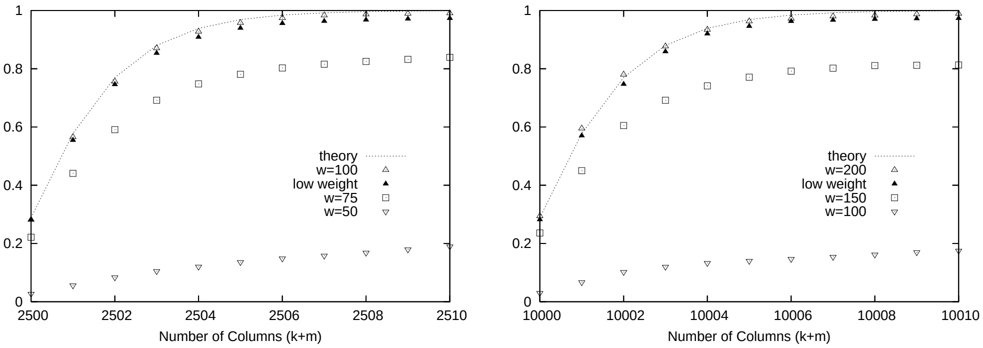

FIGURE 4. Probability of rank k for a $k \times (k+m)$ matrix. The left graph shows the results for $k = 2500$, while the right is for $k = 10000$. In both graphs, the broken line is Q_m (Equation 3.2), the open symbols show results for random ($p = 1/2$) matrices with various window sizes, and the *low weight* closed triangles are the results for a window size of $w = 2\sqrt{k}$ but with the mean column weight fixed at $2 \log k$.

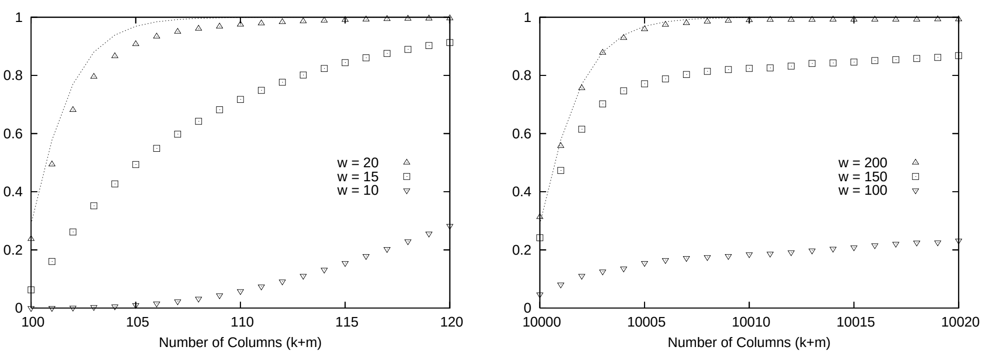

FIGURE 5. Probability of rank k for a $k \times (k+m)$ matrix. The left graph shows the results for $k = 100$, while the right is for $k = 10000$. In both graphs, the broken line is Q_m, and the other symbols show results for windowed matrices (with the specified window size), for $p = 1/2$ within a window.

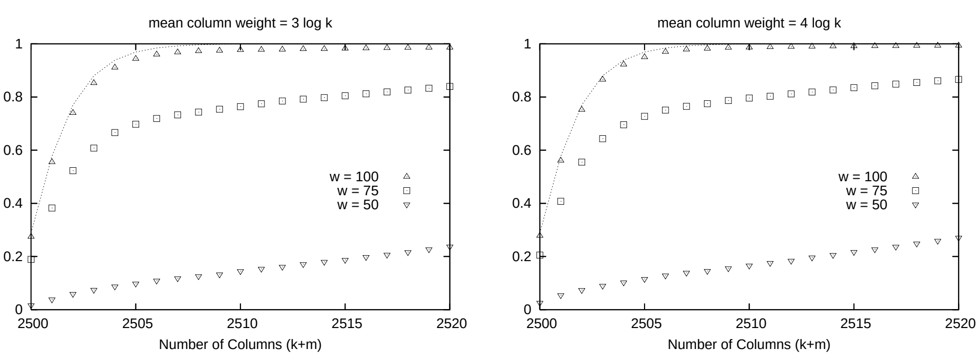

FIGURE 6. Probability of rank k for a $k \times (k+m)$ matrix. Similar to previous graphs with $k = 2500$ and higher mean column weight. In both graphs, the broken line is Q_m.

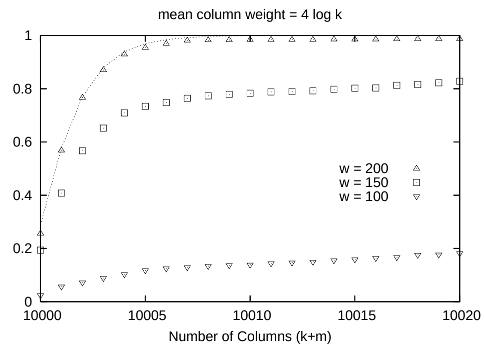

FIGURE 7. Probability of rank k for a $k \times (k+m)$ matrix. Similar to previous graphs with $k = 10000$ and higher mean column weight as indicated. The broken line is Q_m.

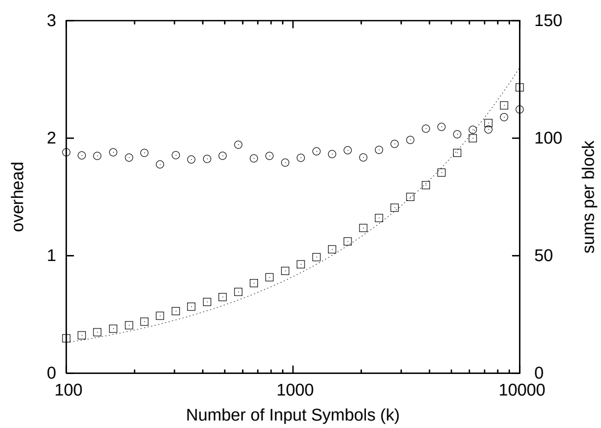

FIGURE 8. Overhead and decoder performance for a windowed erasure code. The circles show the mean number of extra columns needed (left axis) while the squares show the mean number of column operations needed per input symbol (right axis). Each data point is the mean from 10,000 runs. The broken line is $1.3\sqrt{k}$.