Improving the Decoding of LDPC Codes for the Packet Erasure Channel with a Hybrid Zyablov Iterative Decoding/Gaussian Elimination Scheme

Mathieu Cunche, Vincent Roca

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Improving the Decoding of LDPC Codes for the Packet Erasure Channel with a Hybrid Zyablov Iterative Decoding/Gaussian Elimination Scheme

Mathieu CUNCHE — Vincent ROCA

N° ????

March 2008

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--????--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
GRENOBLE - RHÔNE-ALPES

Improving the Decoding of LDPC Codes for the Packet Erasure Channel with a Hybrid Zyablov Iterative Decoding/Gaussian Elimination Scheme

Mathieu CUNCHE , Vincent ROCA

Thème COM — Systèmes communicants
Équipe-Projet Planète

Rapport de recherche n ???? — March 2008 — 16 pages

Abstract: This work focuses on the decoding algorithm of the LDPC large block FEC codes for the packet erasure channel, also called AL-FEC (Application-Level Forward Error Correction). More specifically this work details the design and the performance of a hybrid decoding scheme, that starts with the Zyablov iterative decoding algorithm, a rapid but suboptimal algorithm in terms of erasure recovery capabilities, and, when required, continues with a Gaussian elimination algorithm. For practical reasons this work focuses on two LDPC codes for the erasure channel, namely LDPC-staircase and LDPC-triangle codes. Nevertheless the decoding scheme proposed can be used with other LDPC codes without any problem.

The performance experiments carried out show that the erasure recovery capabilities of LDPC-triangle codes are now extremely close to that of an ideal code, even with small block sizes. This is all the more true with small code rates: whereas the Zyablov iterative decoding scheme becomes unusable as the code rate decreases, the Gaussian elimination makes the LDPC-triangle codes almost ideal. In all the tests, when carefully implemented, the LDPC-triangle codec featuring the proposed decoding scheme is fast, and in particular always significantly faster than the reference Reed-Solomon on GF(2^8) codec.

The erasure recovery capabilities of LDPC-staircase codes are also significantly improved, even if they remain a little bit farther from an ideal code. Nevertheless, a great advantage is the fact that LDPC-staircase codes remain significantly faster than LDPC-triangle codes, which, for instance, enables their use with larger blocks.

All these results make these codes extremely attractive for many situations and contradict the common belief that using Gaussian elimination is not usable because of a prohibitive processing load. Moreover the proposed approach offers an important flexibility in practice, and depending on the situation, one can either choose to favor erasure recovery capabilities or the processing time.

Key-words: FEC codes for the packet erasure channel, AL-FEC codes, LDPC codes, LDPC-staircase codes, LDPC-triangle codes, Gaussian elimination, Zyablov iterative decoding

This work is supported by the ANR/RNRT 2006 contract number 06TCOM01901.

Centre de recherche INRIA Grenoble – Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Amélioration du Décodage de Codes LDPC pour le Canal à Effacement de Paquets avec une Approche Hybride Décodage Itératif de Zyablov/Pivot de Gauss

Résumé : Ce travail aborde les algorithmes de décodage des codes FEC de type grand bloc pour le canal à effacement de paquets, aussi appelés AL-FEC (Application-Level Forward Error Correction). Plus précisement, ce travail détaille la conception et les performances d'une approche de décodage hybride, qui débute avec l'algorithme de décodage itératif de Zyablov, un algorithme rapide mais sous-optimal en terme de capacités de correction d'erreurs, et se poursuit, lorsque ceci est nécessaire, avec un pivot de Gauss. Pour des raisons pratiques ce travail considère deux codes AL-FEC pour le canal à effacement de paquets, les codes LDPC-staircase et LDPC-triangle. Cependant le schéma de décodage proposé peut s'appliquer à d'autres codes LDPC sans aucun problème.

Les tests éffectués montrent que les codes LDPC-triangle atteignent des performances en terme de capacité de correction d'effacements extrêmement proches de celles d'un code idéal, ceci même avec de petites tailles de blocs. Ceci est encore plus vrai avec de petits code rates: alors que le décodage itératif de Zyablov devient inutilisable au fur et à mesure que le code rate diminue, le pivot de Gauss rend les codes LDPC-triangle quasiment optimaux. Dans tous les tests, lorsqu'il est soigneusement implémenté, le codec LDPC-triangle doté du schéma de décodage proposé est rapide, et en particulier toujours significativement plus rapide que le codec Reed-Solomon sur GF(2^8) de référence.

Les capacités des correction d'effacements des codes LDPC-staircase sont également significativement améliorées, même si elles restent un peu plus éloignées d'un code idéal. Cependant un gros avantage est le fait que les codes LDPC-staircase sont significativement plus rapides que les codes LDPC-triangle, ce qui permet, par exemple, de les utiliser avec des tailles de blocs plus importantes.

Tous ces résultats rendent ces codes extrêmement attractifs pour de nombreuses situations et contredisent une idée largement répendue selon laquelle le pivot de Gauss ne serait pas utilisable du fait de coûts de calcul prohibitifs. De plus l'approche proposée offre une très grande flexibilité à l'usage, et suivant la situation, on pourra choisir de privilégier soit les capacités de correction, soit le temps de calcul.

Mots-clés : Codes FEC pour canaux à effacement de paquets, codes AL-FEC, codes LDPC, codes LDPC-staircase, codes LDPC-triangle, Pivot de Gauss, décodage itératif de Zyablov

Contents

1 Introduction

This work focuses on the decoding algorithm of the LDPC large block FEC codes for the packet erasure channel. These codes usually operate within the transport or application layer (hence their name, AL-FEC), and sometimes within the MAC layer (e.g., in the MPE-FEC layer of DVB-H systems).

One benefit is that AL-FEC codes are easily implemented in software codecs, which provides a lot of flexibility: they can operate on very large blocks, whereas physical layer FEC codes are often limited by the available chipset memory; they can be tailored to the application constraints by dynamically varying the size of the encoded block, the amount of redundancy added, or even by providing several layer of protections.

Codes for this type of channel are becoming increasingly important since they are the cornerstone of many content delivery protocols (CDP) and systems. For instance, AL-FEC codes are a key building block of the FLUTE/ALC protocol stack integrated in the DVB-H/SH IP Datacasting service for reliable file delivery. For the same reasons, the Reliable Multicast Transport (RMT) working group of the IETF is standardizing several such AL-FEC codes, including Reed-Solomon codes, Raptor codes, and the LDPC-staircase and LDPC-triangle codes, two LDPC codes that will be considered in this work.

The performance of LDPC AL-FEC codes, both in terms of erasure recovery capabilities and decoding speed, is largely dependent on the decoding algorithm used. A commonly used decoding scheme has been described by Zyablov and Pinsker in 1974 in [11] (see Section 3.2). If this algorithm features a very high decoding speed, it remains sub-optimal in terms of erasure recovery capabilities.

The goal of the present work is to go beyond this decoding algorithm and to analyze the performance of the two LDPC variants mentioned above with a hybrid Zyablov Iterative decoding/Gaussian elimination scheme. The contributions of this work are twofold: (1) it proves that very simple LDPC codes feature *very good performance in terms of erasure recovery capabilities*, sometimes very close to that of an ideal code, and (2) it shows that, when carefully implemented, *the proposed decoding scheme is rather fast*. In particular the two LDPC codes are always significantly faster than the software reference Reed-Solomon codec working on $GF(2^8)$.

The remaining of this paper is organized as follows. We first describe related works. Then we give more information on the two LDPC codes considered and the Zyablov iterative decoding algorithm in Section 3. We detail our proposal in Section 4. We introduce and discuss performance analyzes in Section 5. Finally we conclude.

2 Related Works

An improved approach for decoding Raptor codes (that are a certain form of LDPC codes) is presented in the following patent description [9] and a similar one for LDPC codes in [1]. More precisely, the two methods aim to reduce the complexity of the decoding by reducing the size of the system on which the Gaussian elimination is applied. Let us consider an LDPC code of length n and dimension k and its parity check matrix H of dimension $L \times n$, and a BEC of loss probability δ. In order to recover all the symbols of the transmitted codeword Y we have to solve a linear system with δn variables and L equations. Thanks to a procedure called symbol deactivation in [9] and reference variables declaration in [1], all the δn unknown symbols are expressed as a function of a subset of αn unknown symbols. The symbol deactivation (respectively reference variables declaration) procedure translates the original system involving δn variables into a smaller system involving only αn reference variables. The smaller system can now easily be solved by Gaussian elimination. Finally the remaining unknown symbols are deduced by a substitution of the values of the reference variables. Several methods for choosing the reference variables are presented in [1] and the authors claim an overall computation gain of at least $(\delta/\alpha)^2$. [9] introduces a specific code construction scheme that improves the efficiency of the inactivation decoder of Raptor codes.

Our proposal differs from these two approaches, since we restrict ourselves to using two known techniques for decoding a system of linear equations, rather than proposing a new one. Thanks to this choice our proposal does not infringe the patent [9] which is a great practical asset. We will see in the remaining of this paper

that for reasonable object sizes, our proposal, applied to the two LDPC variants chosen, yields excellent results.

In [2] a scheme called In-place algorithm is presented, which uses a Gaussian elimination for decoding LDPC codes and other binary linear block codes on the Binary Erasure Channel (BEC). As in our scheme the Gaussian elimination procedure is applied on a simplified parity check matrix and the operation are limited to the rows. The performance of this scheme is evaluated and compared with other decoding scheme on LDPC, BCH, and quadratic residue codes of small length ($n \leq 341$). A limitation of the work is that it does not provide any complexity evaluation of the presented scheme nor decoding time analysis. Our work is therefore very similar to this one but focuses on different LDPC variants, and aims at providing an extensive performance analysis, using on on-the-shelf, high performance, codecs.

3 LDPC-staircase and LDPC-triangle Codes and the Zyablov Iterative Decoding Scheme

3.1 Introduction to LDPC-staircase and LDPC-triangle AL-FEC Codes

The LDPC-staircase [4] and LDPC-triangle codes are variants of the well known LDPC codes introduced by Gallager in the 60s [3]. Thanks to an appropriate parity check matrix structure, the LDPC-staircase and LDPC-triangle codes feature a very high encoding and decoding speed. These codes therefore belong to the class of large block codes, which means they can easily operate on blocks that are composed of a huge number of source symbols (up to several 100,000s). If each symbol is 1 kilobyte long, then a file of several hundreds of megabytes can be encoded in a single pass. This is a great advantage over so called small-block FEC codes, like the popular Reed-Solomon codes over GF(2^8) for the erasure channel [5].

The parity check matrix H of each LDPC variant considered, of size $n-k$ rows (i.e., the equations or constraints) for n columns (i.e., the source and repair symbols), is built as follows. H is the concatenation of sub-matrix H_1, of size $n-k$ rows for k columns (i.e., the source symbols), and of sub-matrix H_2, of size $n-k$ rows for $n-k$ columns (i.e., the repair symbols). For both LDPC variants, the H_1 sub-matrix is filled in a *fully regular way*, with three "1s" per columns and at least two "1s" per row. The variants only differ by their H_2 sub-matrix, which has either a "staircase" (also called double diagonal) or a "lower triangle" structure. In the latter case, the triangle is filled-in thanks to a dedicated algorithm. The interested reader is invited to refer to [7][8] for further details.

The remaining of this work is based on the LDPC-staircase and LDPC-triangle AL-FEC codes. No other LDPC codes will be considered, because these codes are already known to yield good performances [7], because an on-the-shelf, high performance, open source codec is available [6], and because these codes are currently being standardized at IETF [8].

3.2 Decoding with the Zyablov Iterative Decoding Algorithm

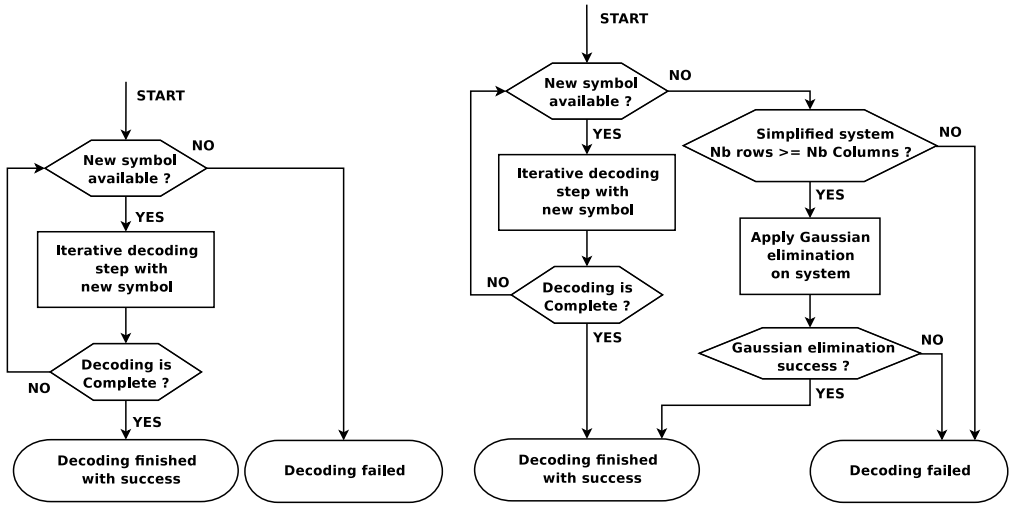
Decoding follows a trivial iterative decoding algorithm, described by Zyablov and Pinsker in [11]: given a set of linear equations, if one of them has only one remaining unknown variable, then the value of this variable is that of the constant term. So, replace this variable by its value in all the remaining linear equations and reiterate. The value of several variables can therefore be found recursively.

Applied to LDPC codes working on an erasure channel, the parity check matrix defines a set of linear equations whose variables are the source symbols and repair symbols. Receiving or decoding a symbol is equivalent to having the value of a variable. Since there are initially $n-k$ linear equations of n variables (i.e., source and repair symbols), this system cannot be solved and we need to receive symbols from the network. Each received symbol contains the value of the associated variable, so we replace this variable in all linear equations in which it appears. We then apply the above algorithm and see if decoding can progress by one or more steps. As we approach the end of decoding, received symbols tend to trigger the decoding of several symbols, until all of the k source symbols have been recovered.

In a previous work [7] we have shown, for instance, that for a code rate 2/3, the Zyablov iterative decoding applied to LDPC-staircase and LDPC-triangle codes requires on average respectively 6.8% and 5.5% symbols in addition to k, whereas k symbols are sufficient with an ideal code.

4 Our Proposal: a Hybrid Zyablov Iterative Decoding/Gaussian Elimination Scheme

4.1 Principles



(a) Zyablov-only iterative decoding. (b) Hybrid Zyablov/Gaussian elimination decoding.

Figure 1: Comparison of the two decoding schemes.

Our proposal works as follows. The receiver collects source and repair symbols (e.g., from the network) and submits them to the Zyablov iterative decoding scheme as they arrive. If the object can be decoded, i.e., if a sufficiently high number of symbols is available for the decoding to finish, the receiver marks the corresponding block as decoded. If the object cannot be decoded, and if the receiver knows he will no longer receive any additional symbol for this block, then he switches to the Gaussian elimination scheme. The system of linear equations upon which the Gaussian elimination works is the simplified system that results from the partial decoding with the iterative scheme. This simplified system can be significantly smaller than the initial system, i.e., it can be composed of a number of equations and variables that are significantly smaller than respectively $n - k$ and n. If we detect that the system cannot be solved with a Gaussian elimination, typically because there is a insufficient number of equations W.R.T. the number of variables, the decoder immediately stops. Otherwise a Gaussian elimination is tried and, if successful, the block is marked as being decoded.

The two flow charts of Figure 1 compare the decoding process with a Zyablov decoding scheme only (on the left) and the hybrid decoding process that couples the two decoding schemes (on the right).

Decoding with the Gaussian Elimination Algorithm: Let us detail how decoding can be finished with a Gaussian elimination, after several steps of the iterative Zyablov algorithm. Let X be the vector containing the set of n source and repair symbols and H be the parity matrix. Decoding is equivalent to solving the system:

$$HX = 0$$

As explained in Section 3.2 the linear system is simplified as the symbols are received or rebuilt. Suppose that after t decoding steps, n_r^t symbols have been received and n_b^t symbols have been rebuilt, and that among the received symbols, n_u^t have been ignored because they have been already rebuilt ($n_u^t \leq n_b^t$). We have: $n_r^t + n_b^t - n_u^t = t$. So t variables and n_b^t equations have been removed from the initial system. The linear system now has $(n-k) - n_b^t$ equations and $n - t$ variables:

$$H_t X_t = Y_t$$

where: H_t is a sub-matrix of the initial H matrix, with $(n-k) - n_b^t$ rows and $n - t$ columns; X_t is the variable vector of size $n - t$ representing the unknown symbols; and Y_t is the (no longer null) vector of size $(n-k) - n_b^t$ containing the sum of the already known symbols for each equation. Of course, if $t = 0$, we have $H_t = H$, $X_t = X$ and $Y_t = 0$.

Let us assume that the decoder switches to the Gaussian elimination scheme after decoding step T. If H_T has a higher (or equal) number of rows than columns, that is to say if $(n-k) - n_b^T \geq n - T$, then the system may be solvable. Note that the previous condition can be fulfilled only if $n_r^t \geq k$, and therefore the maximum size of this system is $(n-k) \times (n-k)$. In that case, a Gaussian elimination is applied. In order to keep the correspondence between the coordinates of the variable vector X_T and the values of the symbols, the Gaussian elimination operates only on the rows. If H_T has a full rank, then the following system is obtained:

$$\begin{pmatrix} Id \\ 0 \end{pmatrix} X_T = Y'_T$$

and the unknown symbols contained in X_T are immediately found.

A key point that enabled us to achieve good results is the following: during all the Zyablov iterative decoding, the sparse parity check matrix H is stored using a sparse representation, where each "1" entry in H is associated to pointers to the next (respectively previous) "1" in the row/column. Yet, during the Gaussian elimination, the resulting system is no longer sparse. Therefore, before beginning the Gaussian elimination, the sparse representation of the parity check matrix is converted to a dense representation, where each row is associated to a bit field of the appropriate size. Because of the algorithmic simplifications this dense matrix enables (e.g., when XORing two rows), this conversion helped us to greatly reduce the decoding time.

4.2 When and How Should the Hybrid Decoding Scheme Be Used?

As we will see in Section 5, the processing load of a Gaussian elimination is not necessarily prohibitive. For instance the LDPC codec featuring an hybrid decoding scheme remains significantly faster than a software Reed-Solomon codec working over GF(2^8). However a Gaussian elimination decoding has an intrinsic cost that should perhaps be avoided in some situations. This is what we now discuss.

Two transmission scenarios must be considered since they will deeply impact the strategy a receiver should follow:

- transmissions within a carousel of the encoding object(s), over a period that largely exceeds the minimum time required to download and decode the object(s);

- single transmission of the encoded object(s);

The first case is typical of a file transmission scenario within, e.g., a DVB-H/SH IP Datacasting session, using FLUTE/ALC as the transmission protocol. Here a popular content is broadcast to a large set of users during, lets say, a few days. A user can select and download the content during this whole period. Once selected, the terminal receives a continuous set of symbols for this object. As long as new packets arrive, a good strategy can be to use only the Zyablov iterative decoding scheme.

The second case is typical of many transmission scenarios where a receiver has only one opportunity to download a content since the source and repair symbols are sent once. Depending on the context, this content might be a file (e.g., with FLUTE/ALC) or a block of bits or bytes in a transmission channel (e.g., in the DVB-H/SH MAC layer). Here, as soon as the receiver detects that the sender will no longer transmit

additional symbols for this block, if the Zyablov iterative decoding scheme did not succeed, then he switches to the Gaussian elimination scheme.

The receiver may also have additional strategies. For instance a receiver might decide that a Gaussian elimination will only be used if the simplified system resulting from the Zyablov iterative decoding scheme has a size inferior to a certain threshold. This threshold can be defined after taking into account the processing capabilities of the terminal, or the remaining battery capacity, or the estimated decoding time.

A receiver might also decide that once the number of (non duplicated) symbols received exceeds a certain threshold (e.g., $1.02 * k$ with LDPC-triangle codes), then he switches to a Gaussian elimination scheme in order to deliver the object to the application in a timely manner, even if additional symbols for this block may still be received in the future.

We see that thanks to its high flexibility, the hybrid decoding scheme proposed can easily be tailored to the use-case needs and the current situation.

5 Performance Evaluation

This section first describes the experimental setup and then gives an account of the various tests carried out to assess the performances achieved with the proposed decoding scheme, both in terms of erasure recovery capabilities and in terms of decoding time.

5.1 Experimental Setup

5.1.1 Software Codecs

The LDPC tests use the high performance, open-source, LDPC C++ codec, version 2.1 [6], for which we added a Gaussian elimination scheme in addition to the existing Zyablov iterative decoding.

The Reed-Solomon tests rely on the excellent open-source C codec [5] designed by L. Rizzo, which has been widely used in the community during the last ten years. This codec will serve as a reference during our tests.

The erasure recovery capability tests also refer to an ideal code, i.e., an MDS code which has no restriction on the maximum source block size or encoded block size (i.e., k and n can be chosen arbitrarily large). Of course, no codec is associated to ideal codes, the results are purely theoretical.

5.1.2 Performance Analysis Application

We designed a dedicated performance analysis application on top of these codecs, derived from `eperf_tool` ([6]). Given an object and a code rate, the application first performs FEC encoding: to k source symbols of a block, the codec produces $n - k = k(\frac{1}{code\_rate} - 1)$ repair symbols. The size of the source and repair symbols is set to 1024 bytes for all the tests.

Once encoding has been performed, the source and repair symbols are transmitted in a fully random order. A loss probability is then applied, leading to the removal (i.e., erasure) of a certain number of symbols. The receiving application submits each received symbol to the decoder. This application stops either when the decoding finishes or when all symbols have been received and submitted to the decoder. The decoding status (success or failure) is then determined.

Note that there is no explicit loss model (e.g., random or per burst erasures) because shuffled transmissions are not affected by this model. The loss probability (for a given code rate) is the only parameter that needs to be considered.

5.1.3 Block Partitioning of the Object with Reed-Solomon and LDPC codes

Because of the intrinsic limitations of a Reed-Solomon codec working on GF(2^8), a large object has to be partitioned into several blocks such that: $k \leq n \leq max\_n = 255$. Determining the block structure of an object requires two things. The first key parameter is the maximum source block size possible, $max\_k = max\_n * code\_rate = 255 * code\_rate$. The second key feature is the block partitioning algorithm.

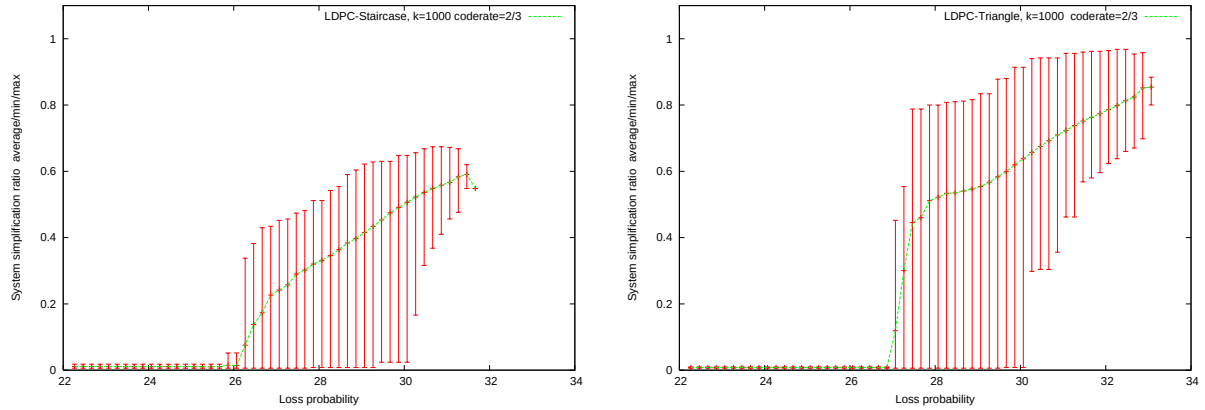
We chose the one specified by the IETF [10], which is used in current standards, like FLUTE/ALC in DVB-H. With this algorithm, all blocks are either of size A_large symbols or $A\_large - 1$ symbols, with $A\_large \leq max\_k$. Taking care of these two aspects is of high importance for reliably comparing the performances of the Reed-Solomon and LDPC codes.

Using Reed-Solomon codes over GF(2^{16}) is another possibility. Indeed, this Galois Field removes the above constraint on the maximum block size, since now: $k \leq n \leq max\_n = 65535$. All the objects of size inferior to this threshold will be encoded in a single pass, within a single block as it is the case with the LDPC codes considered. However the price to pay for that is much higher decoding times, whereas the decoding time is already a limitation with GF(2^8) as we will see in Section 5.4. Therefore we deliberately chose to restrict ourselves to Reed-Solomon codes over GF(2^8).

With the two LDPC variants considered, the object is always encoded as a single block in the following tests. If practical limits exist, they are beyond the object sizes considered in this work, that amount to at most 10,000 symbols.

5.2 Evaluation of the System Simplification Made Possible the Zyablov Iterative Decoding

We first start by analyzing the efficiency of the Zyablov iterative decoding scheme as a way to simplify the system of linear equations before applying a Gaussian elimination.



(a) LDPC-staircase.

(b) LDPC-triangle.

Figure 2: Simplification ratio of the system as a function of the loss ratio (object size=1,000 symbols and code rate=2/3).

Figure 2 shows the simplification ratio of the system as a function of the loss probability. This simplification ratio is defined as the ratio between the simplified system size to the initial system size (i.e., if Zyablov iterative decoding is not used). The closer to zero this ratio is, the better. We see that with small loss probabilities, the average system simplification ratio is almost null, meaning that the Zyablov iterative decoding succeeds. Then as the loss probability increases, the average ratio increases too, meaning that the Zyablov iterative decoding is less and less efficient in its role of simplifying the system. This phenomenon is more visible with LDPC-triangle codes than it is with LDPC-staircase codes. In other words, with LDPC-staircase codes, even if the object cannot be fully decoded, a significant subset of it is nonetheless rebuilt.

5.3 Evaluation of the Erasure Recovery Capabilities

The erasure recovery capabilities of the codes are the main performance criteria. This feature is measured with the *inefficiency ratio metric*, which is defined as the ratio of the number of symbols that must be

received in order to decode the object (or all blocks of the object in case of Reed-Solomon) to the source block size. Of course, an ideal code has an inefficiency ratio always equal to 1.0.

We conducted three kinds of tests:

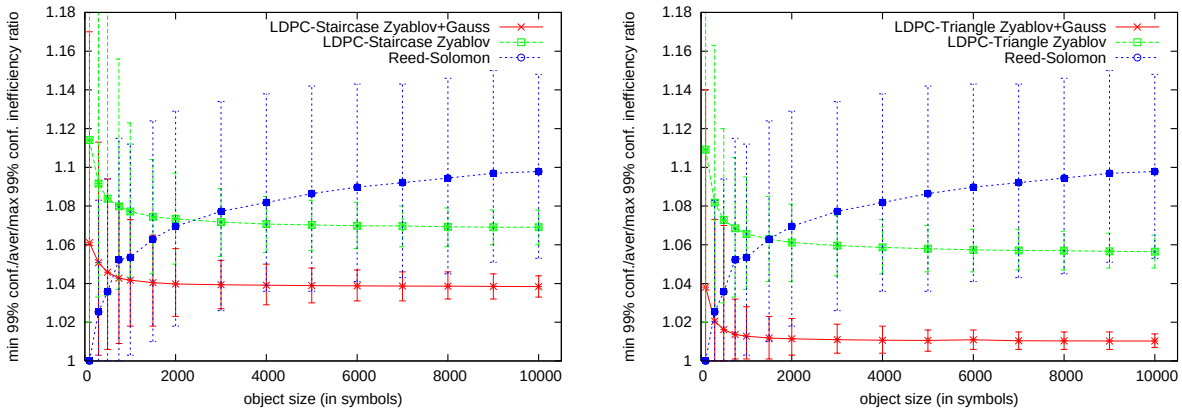
- *fixed code rate and different object sizes:* the code rate is set to 2/3 and the object size is progressively increased. The goal is to evaluate the asymptotic behavior of the codes, as well as their behavior with small objects where LDPC codes traditionally perform badly;

- *fixed object size and different code rates:* the object size is set to 1,000 symbols (≈1 MBytes) and the code rate is progressively decreased. The goal is to evaluate the behavior of the two LDPC codes as they approach rateless codes.

- *fixed object size and code rate, different loss probability:* The goal is to evaluate the detailed behavior of the codes as the loss probability approaches the maximum supportable loss rate made possible by the code rate.

With the first two kinds of tests the methodology is the following. For a given {object size; code rate} tuple, we chose a loss probability (or symbol erasure probability) and perform 1,000 tests, determining each time whether decoding succeeds or not. Then we reiterate with a different loss probability. Finally, from the whole set of tests for a given {block size; code rate} tuple, we derive the inefficiency ratio and the corresponding 99% confidence intervals for the values lower (respectively higher) than this average.

5.3.1 Fixed Code Rate and Different Object Sizes



(a) LDPC-staircase.

(b) LDPC-triangle.

Figure 3: Inefficiency ratio of Reed-Solomon, LDPC-staircase and LDPC-triangle codes as a function of the object size, for a code rate=2/3.

Object size	100	500	1,000	2,000	5,000	10,000
LDPC-Staircase Gauss	6.10%	4.59%	4.17%	3.97%	3.89%	3.84%
LDPC-Triangle Gauss	3.82%	1.62%	1.28%	1.14%	1.06%	1.03%
Reed-Solomon	0%	3.58%	5.35%	6.95%	8.65%	9.79%

Table 1: Average inefficiency ratio as a function of the object size, for a code rate=2/3.

Figure 3 and Table 1 show the performance of the codes for various object sizes and a code rate set to 2/3. We see that the average inefficiency ratio of LDPC-triangle codes falls below 1.6% very quickly,

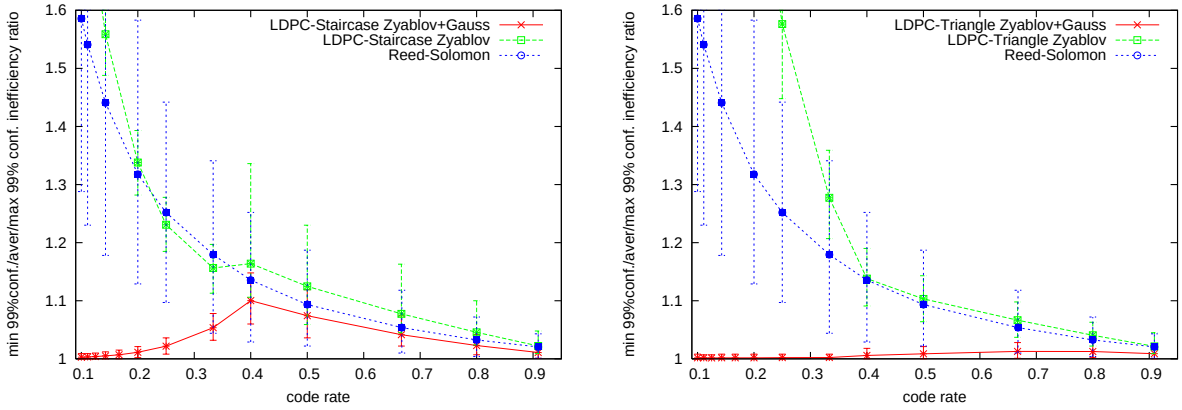
for objects of size 500 symbols and then stabilizes around 1.0%. On the opposite, with a Zyablov iterative decoding scheme, LDPC-triangle codes have an inefficiency ratio that asymptotically stabilizes around 5.5%. Not only does the Gaussian elimination scheme enable to reach lower inefficiency ratios with LDPC-triangle codes, but it does it more quickly and with a better confidence interval than the Zyablov iterative decoding scheme.

The benefits of the Gaussian elimination are also significant with LDPC-staircase codes, but do not enable to reach as good inefficiency ratios as with LDPC-triangle codes. The average inefficiency ratio quickly achieves a value around 4.0% when Gaussian elimination is used, to be compared to the asymptotic value of 6.8% with a Zyablov iterative decoding scheme.

Finally, we see that Reed-Solomon code are only interesting for very small objects, respectively below 750 symbols with LDPC-staircase codes and 255 symbols with LDPC-triangle codes. Above these values, the two LDPC variants featuring a Gaussian elimination are always more efficient in terms of erasure recovery capabilities.

Let us remind that the bad behavior of Reed-Solomon codes over GF(2^8) is related to the so-called "coupon collector problem": when several blocks are needed, the probability that a repair symbol chosen randomly be useful to recover from an erasure is inversely proportional to the number of blocks. As the object size increases, this "coupon collector problem" becomes more and more significant, as can be seen in the figures.

5.3.2 Fixed Object Size and Different Code Rates



(a) LDPC-staircase.

(b) LDPC-triangle.

Figure 4: Inefficiency ratio of Reed-Solomon, LDPC-staircase and LDPC-triangle codes as a function of the code rate, for an object size=1,000 symbols.

Code Rate	1/10	1/9	1/8	1/7	1/6	1/5	1/4
LDPC-Staircase Gauss	0.28%	0.32%	0.40%	0.50%	0.70%	1.14%	2.18%
LDPC-Triangle Gauss	0.17%	0.16%	0.15%	0.17%	0.16%	0.16%	0.20%
Code Rate	1/3	2/5	1/2	2/3	4/5	10/11	
LDPC-Staircase Gauss	5.35%	10.01%	7.42%	4.15%	2.28%	1.08%	
LDPC-Triangle Gauss	0.23%	0.58%	0.86%	1.28%	1.25%	0.88%	

Table 2: Average inefficiency ratio as a function of the code rate, for an object size=1,000 symbols.

Figure 4 and Table 2 show the performance of the codes for various code rates and an object of size 1,000 symbols. We see here the *exceptional behavior of LDPC-triangle codes that are extremely close to an ideal*

code. For instance, with a code rate of 1/3 or lower, the average inefficiency ratio of LDPC-triangle codes remains below 0.23% thanks to the Gaussian elimination scheme. On the opposite, the Zyablov iterative decoding scheme performs very badly as the code rate decreases, and becomes almost unusable below a code rate 0.4.

The benefits of the Gaussian elimination with LDPC-staircase codes becomes really significant essentially below code rate 0.4. Below this code rate, the performance of LDPC-staircase codes asymptotically approaches the erasure recovery capabilities of LDPC-triangle codes. Here also the Zyablov iterative decoding scheme performs very badly as the code rate decreases, and becomes almost unusable below code rate 0.33.

Finally, with such a small object (1,000 symbols), we see that LDPC codes need to use a Gaussian elimination to be more efficient than Reed-Solomon codes, and this is all the more true as we approach rateless codes.

The LDPC-staircase curves highlight a potential issue in the design of these codes, since there is an inflection at code rate 2/5. This value corresponds to the point where there are exactly three "1"s per column and two "1"s per row in the left parity check sub-matrix. Since the goal of the present work is to assess the performances of the decoding algorithm on existing LDPC codes, we will not discuss this phenomenon in this paper. Nevertheless we believe that there is potentially here a possibility to further optimize LDPC-staircase codes, for instance by making the density of "1"s in this left sub-matrix dependent on the code rate.

5.3.3 Fixed Object Size/Code Rate and Different Loss Probabilities

Figure 5 exhibits the decoding success probability W.R.T. the erasure probability of LDPC codes with/without Gaussian elimination, Reed-Solomon codes over GF(2^8), and an ideal code. The object size is set to 1,000 symbols, and the code rate is set either to 2/3 or 1/3. This code rate also defines an upper bound on the loss probability above which decoding becomes totally impossible.

The closer to the rectangular curve of an ideal code, the better. We see from this point of view the excellent performance of both LDPC codes with code rate 1/3. In particular, the LDPC-triangle codes curve is almost impossible to distinguish from that of an ideal curve, which is a major achievement. The fact that the slope of the curves of all LDPC codes using Gaussian elimination be closer to a vertical than the curves of LDPC codes using Zyablov iterative decoding only and the curves of Reed-Solomon codes is also excellent. It shows that once Gaussian elimination starts, the success probability is rather high.

5.4 Evaluation of the Decoding Time

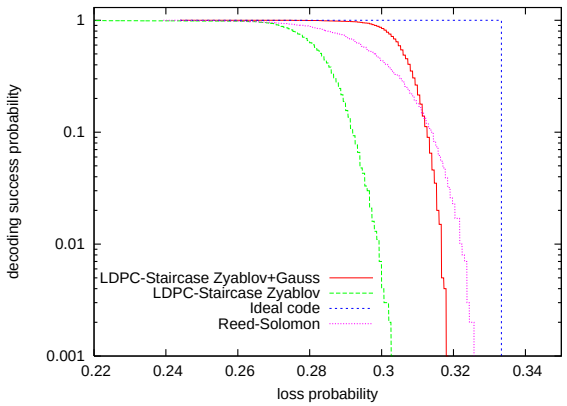
We have seen so far that the two LDPC codes considered feature very good performances in terms of erasure recovery capabilities. The next question is naturally: what is the price to pay for that? Does the Gaussian elimination scheme lead to prohibitive processing load or not?

We carried out several experiments meant to analyze the performance of the codec in terms of decoding time. These experiments have been performed on a Linux PC using a 2.6.18-6/64 bit Debian operating system and featuring a Dual Core Intel Xeon 5120/1.86GHz (1066MHz) processor. While this machine is not (yet?) representative of portable terminals, these tests enable us to compare the decoding speed of both LDPC variants with that of the Reed-Solomon software codec that serves as a reference point.

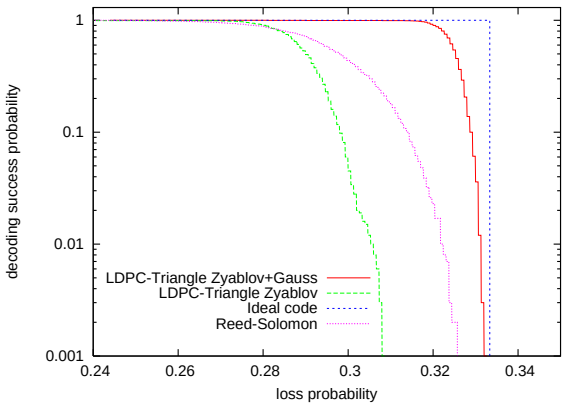
Two kinds of experiments are carried out:

- decoding time measurements, when the Zyablov iterative decoding does not succeed, and therefore a Gaussian elimination is needed. This is the worst case from a decoding point of view;

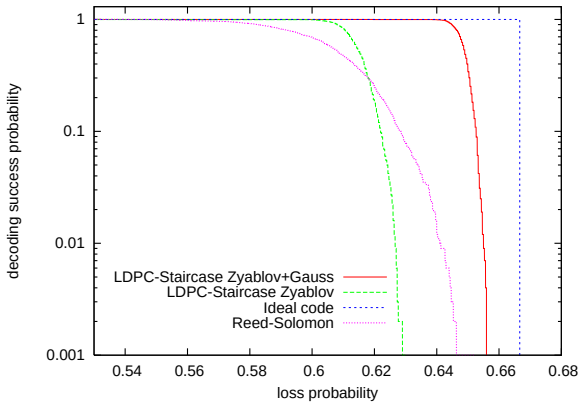
- decoding time measurements as a function of the loss probability, in order to compare situations where the Zyablov iterative decoding is successful and situations where Gaussian elimination is needed. These tests are more representative of real situations.



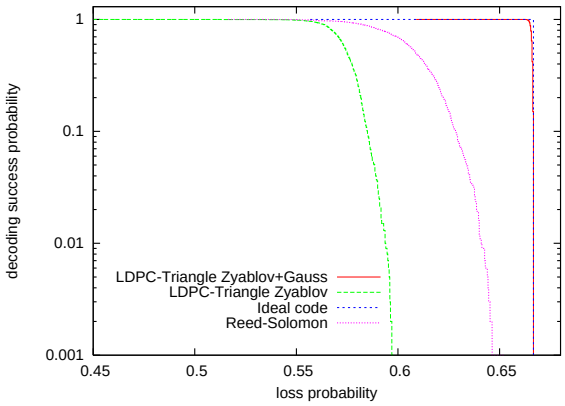
(a) LDPC-staircase, code rate 2/3.



(b) LDPC-triangle, code rate 2/3.



(c) LDPC-staircase, code rate 1/3.



(d) LDPC-triangle, code rate 1/3.

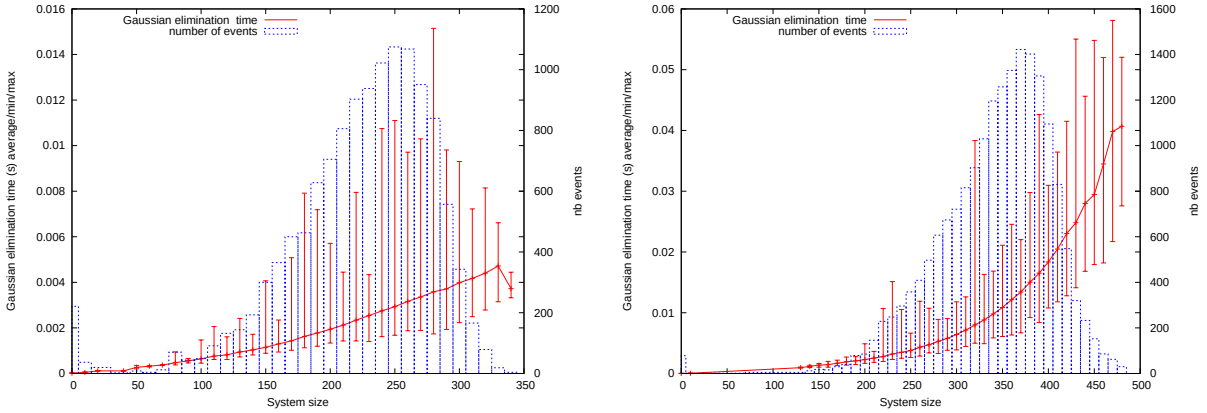
Figure 5: Decoding success probability of Reed-Solomon, LDPC-staircase and LDPC-triangle codes (k=1,000 symbols, code rate 2/3 and 1/3).

5.4.1 Decoding Time When Gaussian Elimination is Needed (Worst Case)

Decoding Time for a Fixed Object Size and Code Rate Figures 6 show the decoding time during the Gaussian elimination process of both LDPC codes for an object of size 1,000 symbols and a code rate set to 2/3. By varying the symbol erasure probability, the resulting average simplified system size changes, which in turn impacts the decoding times. These figures show this time as a function of the simplified system size, and the corresponding histogram. With LDPC-staircase codes, the histogram shows that most simplified systems are composed of 250 variables, corresponding to an average decoding time around 0.003 seconds. We also see that if the average decoding time never exceeds 0.005 seconds, the maximum time remains below 0.015 seconds which is acceptable.

LDPC-triangle codes are known to be slower with the Zyablov Iterative decoding scheme. This is confirmed here also. We see that most simplified systems are composed of 370 variables, corresponding to an average decoding time around 0.014 seconds. Here the maximum decoding time remains below 0.057 seconds.

Decoding Time Comparison for Various Object Sizes When the object size increases, Figure 7 shows that the Gaussian elimination decoding time increases, following a $O(k^3)$ law. With LDPC-triangle codes, an object of size 4,000 source symbols (it leads to a simplified matrix which is almost always composed of a number of variables below but close to $k \times (3/2 - 1) = 2000$), the maximum decoding time remains



(a) LDPC-staircase. (b) LDPC-triangle.

Figure 6: Histogram of the Gaussian elimination decoding time of LDPC-staircase and LDPC-triangle codes W.R.T. the simplified system size (object size=1,000 symbols and code rate=2/3).

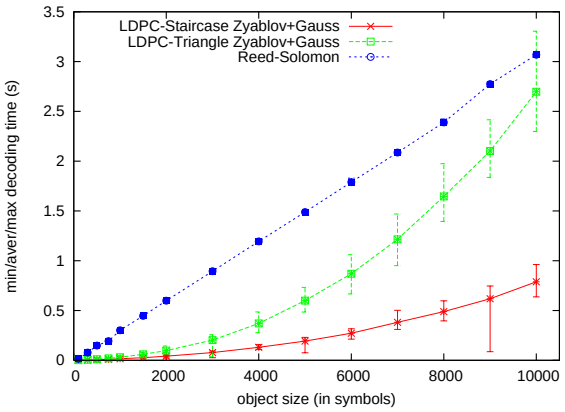


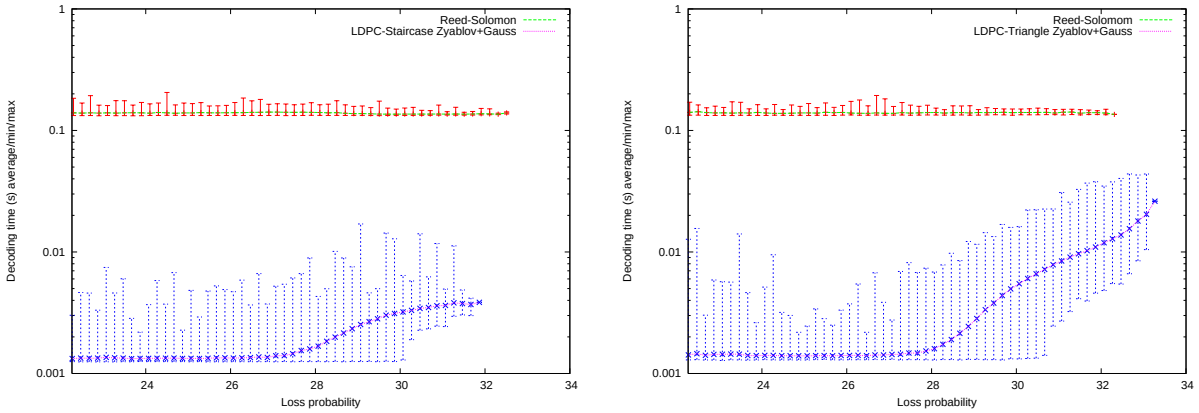
Figure 7: Decoding time of LDPC-staircase/LDPC-triangle codes with a Gaussian elimination and Reed-Solomon codes as a function of the object size, for a code rate=2/3.

below 0.5 second. With LDPC-staircase codes, the same level of performance is achieved with an object of size 7,000 symbols instead of 4,000. If the decoding time of objects larger than this threshold quickly increase, conversely smaller objects are very easily decoded. In all cases, in the range of objects considered, these decoding times are significantly below the ones achieved with Reed-Solomon codes.

5.4.2 Decoding Time of the Hybrid Decoding Scheme

We have seen so far the decoding time of the two LDPC codes when Gaussian elimination is needed. In practice, if the loss probability is sufficiently low (i.e., the reception quality is good enough), the Zyablov iterative decoding algorithm will succeed. The results of previous section are therefore rather pessimistic. We now study the whole decoding time as a function of the loss probability.

Figure 8 compares the decoding time of the LDPC-staircase and LDPC-triangle codes W.R.T. that of Reed-Solomon as a function of the loss probability, for an object of size 1,000 symbols and a code rate set to 2/3. With LDPC codes, as long as the loss probability remains below ≈ 0.27, the high speed Zyablov algorithm succeeds to decode the object. Then, as the loss probability increases, the Gaussian elimination must be used. The decoding time progressively increases because (1) the Gaussian elimination is more and



(a) LDPC-staircase.

(b) LDPC-triangle.

Figure 8: Total decoding time of LDPC-staircase and LDPC-triangle codes versus Reed-Solomon codes, as a function of the loss probability, for an object size=1,000 symbols and a code rate=2/3.

more likely to be required and (2) in that case, the size of the system that has been simplified by the Zyablov scheme increases too (said differently the Zyablov iterative decoding scheme behaves more and more badly). Then above a certain threshold that depends on the code rate and erasure recovery capabilities, decoding becomes impossible. If we compare with the Reed-Solomon decoding time, we see that LDPC-triangle decoding is always an order of magnitude faster, even when Gaussian elimination is needed. This result is even more true with LDPC-staircase codes for which the Zyablov and Gaussian elimination schemes are both faster than with LDPC-triangle codes (Figure 6).

6 Conclusions

This work focuses on the decoding of the LDPC AL-FEC codes for the erasure channel. More precisely, this work studies the erasure recovery capabilities of these codes when a Gaussian elimination scheme is used in addition to the Zyablov iterative decoding scheme. For the need of the exercise, two LDPC flavors are considered, namely LDPC-staircase and LDPC-triangle, but similar results are expected with other LDPC codes.

A first contribution is to show that the erasure recovery capabilities of LDPC-triangle codes are now extremely close to that of an ideal code. With a code rate 2/3, the LDPC-triangle inefficiency ratio is between 1.6% and 1.0% only compared to the 5.5% when the iterative decoding scheme alone is used. This result is all the more significant as it is not achieved asymptotically as it is often the case, but very quickly, with blocks of size 500 symbols. The results W.R.T. small code rates are even more impressive. Whereas the iterative decoding scheme becomes unusable as the code rate decreases, the Gaussian elimination makes the LDPC-triangle codes almost ideal with an inefficiency ratio lower than 0.2%. Said differently, *the Zyablov decoding algorithm does not exhibit the near-MDS capabilities of the LDPC-triangle codes as a Gaussian elimination does.*

A second contribution is to show that such simple codes as LDPC-staircase codes can achieve good results, even if their erasure recovery capabilities remain a little bit farther from an ideal code. Here also the use of Gaussian elimination largely improves the situation, even with small source blocks or small code rates. Besides *LDPC-staircase codes are significantly faster* than LDPC-triangle codes, which, for instance, enables their use with larger blocks. For instance LDPC-staircase and triangle codes feature the same worst case decoding time with blocks of size 7,000 and 4,000 symbols respectively (i.e., 0.5 seconds in our tests).

A third contribution is that our results contradict a common belief that using Gaussian elimination is not usable in practice because of a prohibitive processing load. On the contrary, we show that with block sizes

that amount to a few thousands of symbols, the joint use of the two decoding algorithms (since the Gaussian elimination no longer works on the initial system but on a simplified one) and a careful implementation make the processing load reasonable. For instance, with an object of size 1,000 symbols and a code rate 2/3, the worst case decoding of LDPC-triangle codes with Gaussian elimination is an order of magnitude faster than the decoding of the reference Reed-Solomon codec. And if the erasure rate is sufficiently low for the Zyablov iterative decoding to succeed, decoding is two orders of magnitude faster than with Reed-Solomon. Even for objects of size 10,000 symbols, both LDPC variants remain faster than Reed-Solomon codes.

All these results are likely to change the way we consider some simple yet efficient LDPC codes for the erasure channel. They also open new perspectives since these codes feature better erasure recovery capabilities and lower decoding times than the software Reed-Solomon codec we used. Additionally *the hybrid Zyablov iterative decoding/Gaussian elimination scheme can easily be tailored to match the use-case and operational requirements*. For instance a receiver might decide that a Gaussian elimination will only be used if the simplified system has a size inferior to a certain that can take into account the processing capabilities of the terminal, or the remaining battery capacity, or the estimated decoding time. This flexibility is another key benefit of our proposal.

7 Acknowledgments

The authors would like to acknowledge Laurent Fazzio from STMicroelectronics who wrote the first version of the Gaussian elimination code, as well as Pascal Moniot from STMicroelectronics for his support. The authors also want to acknowledge Jérôme Lacan and Valentin Savin for their valuable comments.

References

[1] D. Burshtein and G. Miller. An efficient maximum-likelihood decoding of ldpc codes over the binary erasure channel. *IEEE Transactions on Information Theory*, 50(11):2837–2844, 2004.

[2] J. Cai, C. Tjhai, M. Tomlinson, M. Ambroze, and M. Ahmed. *A New Non-Iterative Decoding Algorithm for the Erasure Channel : Comparisons with Enhanced Iterative Methods*, Mar. 2005. http://arxiv.org/abs/cs/0503006.

[3] R. G. Gallager. Low density parity check codes. *IEEE Transactions on Information Theory*, 8(1), Jan. 1962.

[4] D. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, ISBN: 0521642981, 2003.

[5] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM Computer Communication Review*, 27(2), Apr. 1997.

[6] V. Roca, M. Cunche, C. Neumann, and J. Labouré. *An Open-Source LDPC Large Block FEC Codec*. URL: http://planete-bcast.inrialpes.fr/.

[7] V. Roca and C. Neumann. Design, evaluation and comparison of four large block fec codecs, ldpc, ldgm, ldgm staircase and ldgm triangle, plus a reed-solomon small block fec codec. Research Report 5225, INRIA, June 2004.

[8] V. Roca, C. Neumann, and D. Furodet. *Low Density Parity Check (LDPC) Staircase and Triangle Forward Error Correction (FEC) Schemes*, Jan. 2008. IETF RMT Working Group, Work in Progress: <draft-ietf-rmt-fec-bb-ldpc-08.txt>.

[9] A. Shokrollahi, S. Lassen, and R. Karp. *Systems and Processes for Decoding Chain Reaction Codes Through Inactivation*, Feb. 2005. U.S. Patent Number 6,856,263.

[10] M. Watson, M. Luby, and L. Vicisano. *Forward Error Correction (FEC) building block*, Aug. 2007. IETF Request for Comments, RFC5052.

[11] V. Zyablov and M. Pinsker. Decoding complexity of low- density codes for transmission in a channel with erasures. *Translated from Problemy Peredachi Informatsii*, 10(1), Jan. 1974.

Centre de recherche INRIA Grenoble – Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399