BER Comparison Between Convolutional, Turbo, LDPC, and Polar Codes

Bashar Tahir*, Stefan Schwarz†, and Markus Rupp‡
Institute of Telecommunications
Technische Universität (TU) Wien
Vienna, Austria
Email: {*bashar.tahir, †stefan.schwarz, ‡markus.rupp}@tuwien.ac.at

***Abstract*—Channel coding is a fundamental building block in any communications system. High performance codes, with low complexity encoding and decoding are a must-have for future wireless systems, with requirements ranging from the operation in highly reliable scenarios, utilizing short information messages and low code rates, to high throughput scenarios, working with long messages, and high code rates.**

We investigate in this paper the performance of Convolutional, Turbo, Low-Density Parity-Check (LDPC), and Polar codes, in terms of the Bit-Error-Ratio (BER) for different information block lengths and code rates, spanning the multiple scenarios of reliability and high throughput. We further investigate their convergence behavior with respect to the number of iterations (turbo and LDPC), and list size (polar), as well as how their performance is impacted by the approximate decoding algorithms.

I. Introduction

In 1948, Shannon [1] showed that an error-free communication over a noisy channel is possible, if the information transmission rate is below or equal to a specific bound; the *Channel Capacity* bound. Since then, enormous efforts were put into finding new transmission techniques with the aim to get closer and closer to the channel capacity.

Channel coding is one of the fundamental techniques that make such near-capacity operation possible. By introducing a structured redundancy at the transmitter (*encoding*), and exploiting it at the receiver (*decoding*), wide possibilities of error detection and correction can be achieved. We consider four coding schemes: *convolutional*, *turbo*, *Low-Density Partiy-Check* (LDPC), and *polar* codes. These schemes were selected as candidates for 5th generation wireless communications (5G), due to their good performance, and low complexity state-of-the-art implementation.

Convolutional codes were introduced by Elias in 1955 [2]. They are a class of linear codes in which the input information bits are encoded in a bit-by-bit (stream) fashion, in such way that the input bits are convolved with (or slided against) predefined polynomials, hence the name "convolutional". The common decoding algorithms for convolutional codes, are the *Viterbi* algorithm [3], and the *BCJR* algorithm [4].

The financial support by the Austrian Federal Ministry of Science, Research and Economy and the National Foundation for Research, Technology and Development is gratefully acknowledged.

†Stefan Schwarz is with the Christian Doppler Laboratory for Dependable Wireless Connectivity for the Society in Motion.

A major breakthrough happened in 1993 when turbo codes were introduced [5]. They represent a class of codes that can perform very close to the capacity limit. In its common form, turbo encoding is done using two recursive convolutional encoders. The input stream is passed to the first encoder, and a permuted version is passed to the second one. At the receiving side, two decoders are used, each one decodes the streams of the corresponding encoder. By exchanging probabilistic information, the two decoders can iteratively help each other in a manner similar to a turbo engine.

Another class of capacity-approaching codes, are the LDPC codes. They were first proposed by Gallager in 1960 [6]. At that time, they were considered too complex for practical implementation. In 1996 [7], LDPC codes were rediscovered, and obtained a steady interest further on. As the name implies, LDPC codes are block codes with a sparse *parity check* matrix. Such sparsity allows for low complexity decoding using the iterative *Belief Propagation* algorithm [8], also called *Sum-Product Algorithm* (SPA), and when designed with a specific structure, low-complexity encoding can also be performed.

A fairly recent type of codes, called polar codes, were introduced by Arıkan in 2008 [9]. They are constructed using the *channel polarization* transform. Aside from their great performance, they are the first practical codes that are proven to achieve the channel capacity at infinite length. Arıkan also showed that a polar code of length N, can be encoded and decoded with a complexity of $O(N \log N)$ each. The encoding is performed using the *Generator* matrix obtained from the polarization transform, and the decoding can be achieved by a *Successive Cancellation* (SC) technique [9].

Similar partial comparisons between those schemes have been carried out in previous publications, such as [10]–[16], but they provided only a limited set of results. We present here, a Bit-Error-Ratio (BER) comparison between the aforementioned coding schemes for different block lengths and code rates, representing the multiple scenarios of reliability and high throughput. We also examine their convergence behavior, and the effect of using the approximate decoding algorithms.

In Section II to V, we present an overview of the encoding, and decoding process of those schemes. In Section VI, we explain our methodology for the comparison and present our results. In Section VII, we provide concluding remarks.

978-1-5386-0643-8/17/$31.00 ©2017 IEEE

II. Convolutional Codes

A. Encoding

Convolving the inputs bits with the code polynomials can be done efficiently using a simple combination of memory elements and XOR operations. Fig. 1 shows the rate 1/3 convolutional encoder used in LTE [17], where the polynomials (G_i) are described in *Octal* form. Understanding the *states' transitions*, is important later on for the decoding. Also, knowing the starting and ending states of the encoder is needed at the decoder, otherwise performance loss might exist.

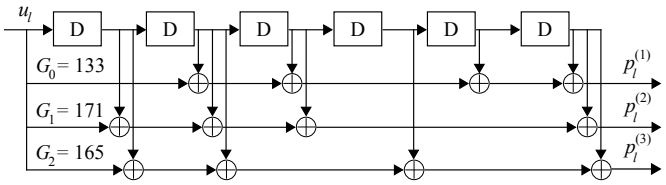

Fig. 1. LTE rate 1/3 convolutional encoder [17].

Due to the simple structure, convolutional codes enjoy low complexity encoding, and combined with the fast clock speeds of the state-of-the-art systems, encoding latency is not an issue.

B. Decoding

We consider the Bit-wise *Maximum A Posteriori* (MAP) decoder, which is utilized using the BCJR algorithm [4]. For information bit u_l at time l, received codeword $\mathbf{y}$, and decoded bit $\hat{u}_l$, the *Log-Likelihood Ratio* (LLR) of u_l is given by

$$L_{u_l} = \log\left(\frac{P\{u_l = 0|\mathbf{y}\}}{P\{u_l = 1|\mathbf{y}\}}\right). \tag{1}$$

Due to the *Trellis* structure of convolutional codes, these probabilities can be written as [18]

$$L_{u_l} = \log\left(\frac{\sum_{U_0} P\{s_{l-1} = s', s_l = s, \mathbf{y}\}}{\sum_{U_1} P\{s_{l-1} = s', s_l = s, \mathbf{y}\}}\right), \tag{2}$$

where s_l is the state at time l, U_0 is the set of pairs (s', s) for the state transition $s' \to s$ when $u_l = 0$, and U_1 is the set of pairs (s', s) for the transition when $u_l = 1$. Using the BCJR algorithm, these probabilities can be factorized as

$$P\{s_{l-1} = s', s_l = s, \mathbf{y}\} = \alpha_{l-1}(s')\gamma_l(s', s)\beta_l(s). \tag{3}$$

where $\gamma_l(s', s)$ is the *Branch Metric*. The probabilities α_l, and β_l are calculated recursively [4]. Processing this in the log-domain, the final expression for the LLR is given by [18]

$$\begin{aligned} L_{u_l} = &\max_{U_0}{}^*[\alpha_{l-1}(s') + \gamma_l(s', s) + \beta_l(s)] \\ &-\max_{U_1}{}^*[\alpha_{l-1}(s') + \gamma_l(s', s) + \beta_l(s)], \end{aligned} \tag{4}$$

The $\max^*$ function is given by

$$\max{}^*(a, b) = \max(a, b) + \log(1 + e^{-|a-b|}). \tag{5}$$

An approximation can be made by neglecting the $\log$ term, yielding the *Max-Log-MAP* algorithm [19].

III. Turbo Codes

Turbo codes are usually constructed by a parallel concatenation of two recursive convolutional encoders separated by an *Interleaver*. The task is then to design the code polynomials for the individual encoders, and to use an appropriate interleaver.

A. Encoding

Since the individual encoders are basically convolutional, the same discussion we had in the previous section carries on to here. The only new element is the interleaver. Fig. 2 shows the turbo encoder used in LTE [17] [20], where a *Quadratic Permutation Polynomials* (QPP) interleaver is used. The outputs of the first encoder are a *systematic* stream u_l, and a *parity* stream $p_l^{(1)}$, while the second encoder generates a parity stream $p_l^{(2)}$ only. This makes it a rate 1/3 turbo code.

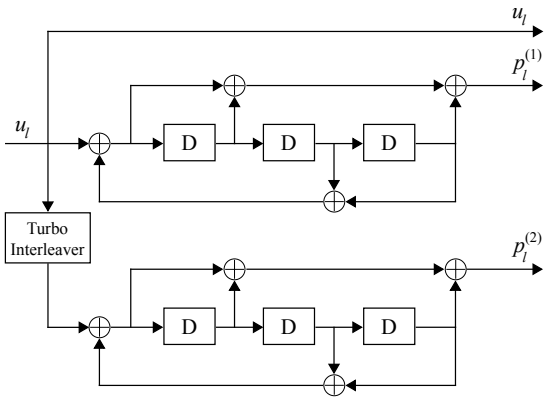

Fig. 2. LTE rate 1/3 turbo encoder [17].

Similarly here, knowing the starting and ending states of the of the encoder at the decoder is important to avoid performance loss. This is handled via *trellis termination*.

B. Decoding

The turbo decoder consists of two *Soft-Input Soft-Ouput* (SISO) decoders. Those decoders are similar to the convolutional decoder, except of some modifications. The systematic stream and the first parity stream are fed to the first decoder, while an interleaved version of the systematic stream, and the second parity stream are fed to the second one. The first decoder starts, and instead of generating a final LLR, it generates a cleared up version, called *extrinsic* information. This is interleaved, and sent to the second decoder. It performs decoding, which is more reliable compared to the case where it does not have the additional information from the first decoder. In a similar manner, it generates extrinsic information for the first decoder, and instead of interleaving, it performs deinterleaving, and at this point, an iteration is completed.

On the next iteration, the first decoder starts the same as before, but now it has extrinsic information from the second decoder, and therefore a more reliable output is calculated. The decoding continues until a stopping criterion is satisfied, or the maximum number of iterations has been reached.

After any iteration, the total LLR is calculated as [18]

$$L_{u_l(total)} = L_{u_l(channel)} + L^{e(2\to1)}_{u_{\text{Deint}(l)}} + L^{e(1\to2)}_{u_l}, \quad (6)$$

where $L_{u_l(channel)}$ is the channel LLR, $L^{e(1\to2)}_{u_l}$ is the extrinsic information sent from first decoder to the second one. Deint(l) is the deinterleaved position of u_l.

If the interleaver is designed appropriately, then it would appear as if the original and interleaved streams are uncorrelated. This is essential for the turbo gain, since it will be unlikely that the original stream and its interleaved counterpart undergo the same encoding, transmission, and/or decoding conditions.

IV. LDPC Codes

An LDPC code is characterized by its sparse parity check matrix. Such sparsity facilitates low complexity encoding and decoding. An example code is the following

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}, \quad (7)$$

which is given here just for demonstration. LDPC codes can be represented by a *Tanner graph* [21]. Each row is represented by a *Check Node* (CN), and each column is represented by a *Variable Node* (VN). The "1"s in the matrix represent the connections between the CNs and VNs. Fig. 3 shows the Tanner graph of the example code.

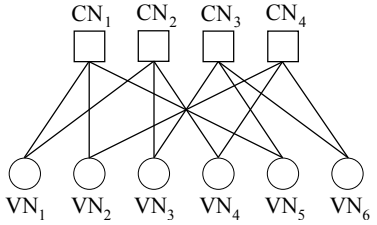

Fig. 3. Tanner graph of the example code.

A. Encoding

The encoding can be described in the following form

$$\mathbf{c} = \mathbf{uG}, \quad (8)$$

where $\mathbf{c}$ is the output codeword, $\mathbf{u}$ is the input block, and $\mathbf{G}$ is the generator matrix.

For LDPC codes, the parity check matrix $\mathbf{H}$ is the design parameter, and not the generator matrix $\mathbf{G}$. However, the generator matrix can still be obtained from a given parity check matrix. This is usually done by putting $\mathbf{H}$ into systematic form using *Gauss-Jordan Elimination*, and then the generator matrix is found directly [18].

Two problems exist, first, the parity check matrix is designed for a specific input block length, and therefore using other lengths is not possible. The second problem lies in the transformation of $\mathbf{H}$ into systematic form, since it can get too complicated for long block lengths. The first problem is handled using *Quasi-Cyclic* (QC) LDPC codes, and those can easily support variable input sizes through *Lifting* [22]. The second problem can be mitigated by utilizing a structure similar to *Repeat-Accumulate* (RA) codes [23], which allows direct encoding from the parity check matrix through back-substitution [24].

B. Decoding

Decoding of LDPC codes is performed with the Sum-Product Algorithm (SPA) [8]. This is based on message passing between the CNs, and VNs in the Tanner graph.

At the start, the VNs send the channel LLRs L_j to the connected CNs. The CNs then perform their calculation, and pass new messages to their connected VNs according to [18]

$$L_{i\to j} = 2\tanh^{-1}\left[\prod_{j'\in N(i)-\{j\}} \tanh(L_{j'\to i}/2)\right], \quad (9)$$

where $L_{i\to j}$ is the message passed from the ith CN to jth VN, $L_{j\to i}$ is the message passed from the jth VN to the ith CN, and $N(i)$ is the set of VNs connected to the ith CN. The VNs receive these messages, process them, and then pass new messages to the connected CNs according to

$$L_{j\to i} = L_j + \sum_{i'\in N(j)-\{i\}} L_{i'\to j}, \quad (10)$$

where $N(j)$ is the set of CNs connected to the jth VN. At this point, one iteration is finished, and the total LLR can be calculated as

$$L_{j(total)} = L_j + \sum_{i\in N(j)} L_{i\to j}. \quad (11)$$

The sequence in which the nodes are scheduled can affect the performance. The one described above, in which all the CNs, and then all the VNs update their messages in parallel, is called the *Flood* schedule. An improved performance can be achieved if *serial* scheduling is performed. This is called *Layered Belief Propagation* (LBP) [25], [26], and it offers almost double the convergence speed (in terms of iterations) to that of the flood schedule.

An approximation can be made to (9) in the form

$$L_{i\to j} = \left(\prod_{j'\in N(i)-\{j\}} \alpha_{j'\to i}\right) \cdot \min_{j'\in N(i)-\{j\}} \beta_{j'\to i}, \quad (12)$$

where $\alpha_{j'\to i}$ and $\beta_{j'\to i}$ are the sign and magnitude of $L_{j'\to i}$, respectively. This is the *Min-Sum* approximation [27], and offers lower complexity decoding at the cost of some performance loss.

V. Polar Codes

Polar codes are those constructed as a result of the *channel polarization* transform [9]. The idea is that by *channel combining* and *splitting*, and at infinite length, the channels (bits' positions) will polarize in the sense that some of channels will be highly reliable, and the rest will be unreliable. If the information bits are put only into the reliable channels,

and foreknown bits (usually zeros) are put into the unreliable channels, then the channel capacity can be achieved.

The task of polar code construction is to find this set of the most unreliable channels, which is usually called the *Frozen Set*. There exist multiple construction algorithms [28], with varying complexity, and due to the non-*universal* behavior of polar codes, those algorithms require a parameter called *Design-SNR*. However, universal constructions also exist.

A. Encoding

The encoder is basically the polarization transform, which is given by the kernel [9]

$$\mathbf{F} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}. \tag{13}$$

The transform for a larger input size is obtained via the *Kronecker* product of this kernel with itself, causing polar codes to have lengths that are powers of 2. For a code of length N, and $n = \log_2(N)$, the encoder is given by

$$\mathbf{G} = \mathbf{F}^{\otimes n}, \tag{14}$$

where $\mathbf{F}^{\otimes n}$ is the Kronecker product of $\mathbf{F}$ with itself n times. The encoding is then carried out as in (8), which is shown in Fig. 4 for a code of length 4.

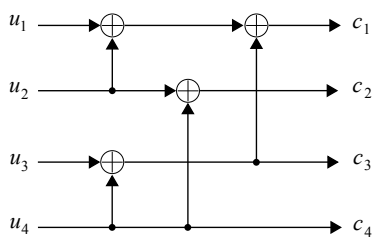

Fig. 4. Polar encoder of length 4.

B. Decoding

Although polar codes can be decoded through Belief Propagation, the standard decoding algorithm is Successive Cancellation (SC). The SC decoder can be found directly from the encoder, where the XOR, and connection nodes are represented by the probabilistic nodes f and g, respectively. In the LLR domain, the f and g nodes perform the following calculations for given input LLRs a and b [29]

$$\begin{aligned} f(a,b) &= \log\left(\frac{e^{a+b}+1}{e^a+e^b}\right), \\ g(a,b,s) &= (-1)^s a + b, \end{aligned} \tag{15}$$

where s is the *Partial Sum*, which is the sum of the previously decoded bits that are participating in the current g node. An approximation can be applied to f, yielding

$$f(a,b) = \text{sign}(a)\text{sign}(b)\min\left(|a|,|b|\right). \tag{16}$$

The LLRs are propagated from the right to the left in Fig. 4. The first bit u_1 can be decoded directly by passing the LLRs through the appropriate f nodes. Once it is decoded, u_2 can be decoded using a g node, which requires the corresponding partial sum. Since only u_1 is participating, then the partial sum is equal to u_1. If u_1 is *frozen*, then the decoder already knows its value, and can directly set it to 0. Therefore, the channel LLRs and u_1 are used to decode u_2, leading to a higher decoding reliability of u_2. The decoding continues until all the nodes are processed.

The performance can be improved, if a List Decoder [30] is used, yielding the List-SC decoder. For each decoded bit, the two possibilities of being decoded as 1 or 0 are considered. This is achieved by splitting the current decoding path into two new paths, one for each possibility. The total number of possibilities across the decoding tree is limited by the *List size.* After the decoding is finished, the path with the smallest *Path Metric* is chosen [29]. A further improvement can be achieved by performing a *Cyclic Redundancy Check* (CRC) on the surviving paths, and the one satisfying it, is the correct one.

VI. Performance Comparison

In this section, we compare the coding schemes in terms of the BER for different information block lengths, and code rates. We check their convergence behavior, and the impact of using the approximate decoding algorithms described above.

A. Setup and Code Construction

We transmit using *Binary Phase Shift Keying* (BPSK) over the *Additive White Gaussian Noise* (AWGN) channel. For convolutional and turbo codes, we chose those of LTE [17]. The interleaver parameters for information length of $K = 8192$ were obtained from [31]. For LDPC, we used the IEEE 802.16 codes [32], and since it does not support codes of rate 1/3, an extension method has been applied to the rate 1/2 code in a fashion similar to [33]. As for polar codes, they were constructed using the *Bhattacharya* bounds algorithm [28], and by searching for a suitable Design-SNR. The constructed codes are available from the author on request.

The rate adaption for convolutional and turbo codes was obviously done by *puncturing*. For polar codes, since the encoder size is limited to powers of 2, the extra positions were handled by applying zeros to the encoder bottom positions, and since their corresponding outputs do not depend on the upper positions, then they are also equal to zero and can be removed from the output codeword. At the decoder, the LLRs of these positions are set to a very high positive value, reflecting a positive infinity LLR.

B. Convergence

We examine here how the performance is affected with respect to the number of iterations (turbo and LDPC), and list size (polar). The decoding algorithms used are Max-Log-MAP, Layered Min-Sum, List-SC with approximation for turbo, LDPC, and polar codes, respectively. The results are given for a rate R of 1/2, with information block length K of 2048 (2052 for LDPC). These are shown in Figs. 5, 6 and 7, for iterations (list sizes) of 32, 16, 8, 4, 2, and 1.

Given the results, it is apparent that going for more than eight iterations for turbo codes, does not provide that much

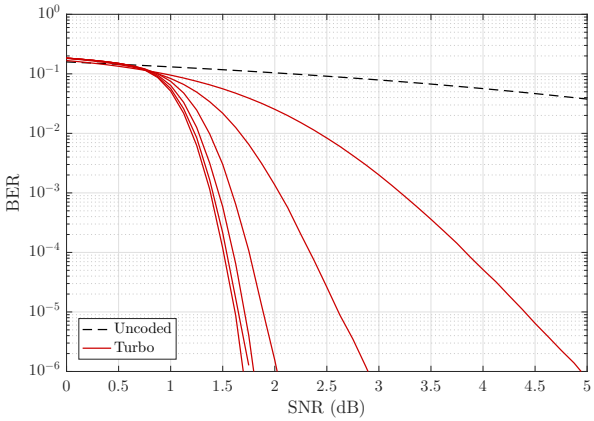

Fig. 5. Turbo code convergence, K = 2048, R = 1/2, Iterations = 32, 16, 8, 4, 2, 1 from left to right.

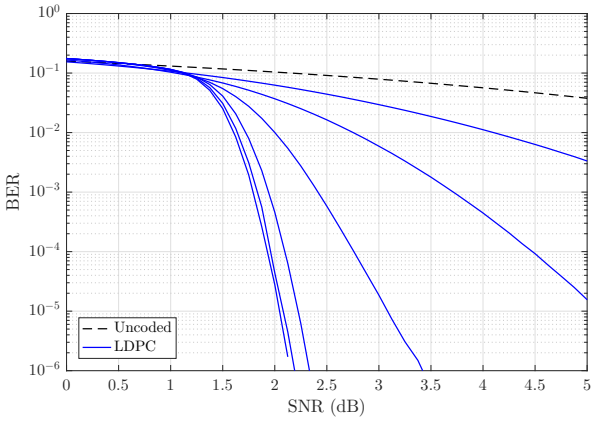

Fig. 6. LDPC code convergence, K = 2052, R = 1/2, Iterations = 32, 16, 8, 4, 2, 1 from left to right.

gain, especially if we consider the relatively high cost of a single turbo iteration. For the LDPC code, 16 iterations appear to be sufficient, and there is very little gain if we go for 32 iterations. An integral part of this fast convergence is due to the usage of layered decoding. As for polar codes, better performance is obtained for larger list sizes, and at the list size of 32, the limitation in the low BER region due to the *Maximum Likelihood* (ML) bound [30], starts to appear.

For the rest of simulations, we choose 8 iterations for turbo, 16 iterations for LDPC, and a list size of 8 for polar codes.

C. Decoding Algorithms

Fig. 8 shows the performance of the exact algorithms (dashed) against their approximations (solid) in (5), (12), and (16). For convolutional and polar codes, the difference is almost nonexistent. However, for turbo and LDPC, there is a considerable difference of approximately 0.37 to 0.47 dB. There are modified algorithms that help close the gap to the exact ones, some of them use look-up tables, offsetting, or low complexity functions, which might require some extra processing, or additional memory usage.

For the simulations in the next section, MAX-Log-MAP is used for convolutional and turbo codes, Layered Min-Sum for LDPC codes, and List-SC with approximation for polar codes.

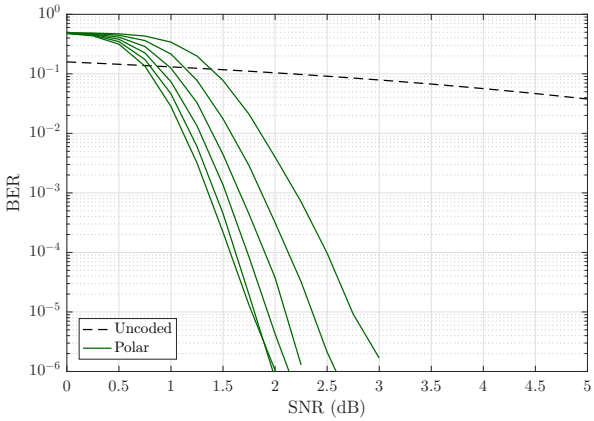

Fig. 7. Polar code convergence, K = 2048, R = 1/2, List size = 32, 16, 8, 4, 2, 1 from left to right.

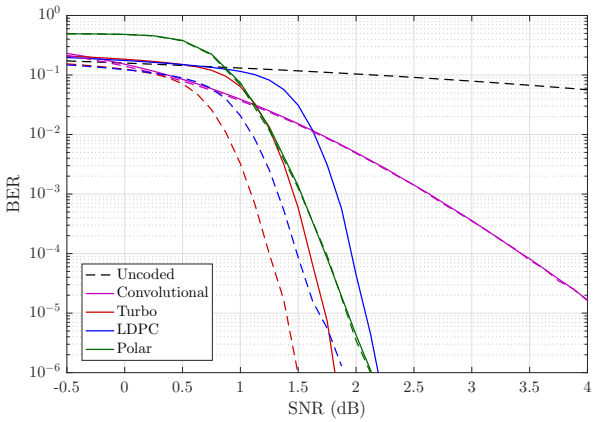

Fig. 8. Exact (dashed) vs. Approximate (solid) decoding algorithms, K = 2048 (2052 for LDPC), $R = 1/2$, 8 iterations turbo, 16 iterations LDPC, list size 8 polar.

D. Results for different block lengths and code rates

In Figs. 9 to 14, the performance of the coding schemes for information block lengths of K = 256, 512, 1024, 2048, 4096, and 8192, and code rates of R = 1/3, 1/2, 2/3, and 5/6 are given. For LDPC, the information length is slightly different, since the dimensions of the used base matrices do not allow such extension. The number of iterations (list size) and decoding algorithms follow the previous two subsections.

Turbo, LDPC, and polar codes perform somehow close to each other, especially at long block lengths. The convolutional code, as expected, performs the worst. But, it still provides a low complexity alternative. The results for polar codes are quite interesting, considering that we used only a list size of 8, and without CRC. This shows how good a polar code can perform when it is constructed appropriately. Nonetheless, some of the polar code curves exhibited a relative saturation in the low BER region. However, potential improvement through better construction should be possible.

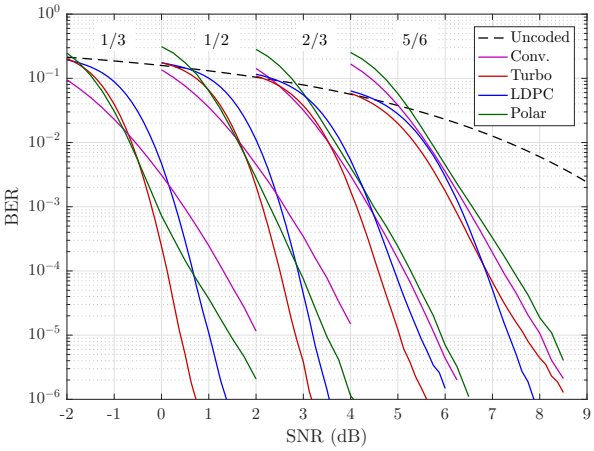

Fig. 9. BER comparison for different code rates, $K = 256$ (For LDPC, $K = 252$ for $R = 1/2,$ and $1/3$, and $K = 260$ for $R = 5/6$.)

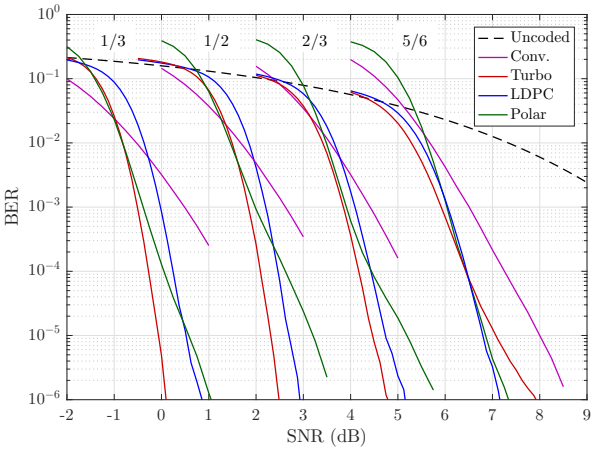

Fig. 10. BER comparison for different code rates, $K = 512$ (For LDPC, $K = 516$ for $R = 1/2,$ and $1/3$, and $K = 520$ for $R = 5/6$.)

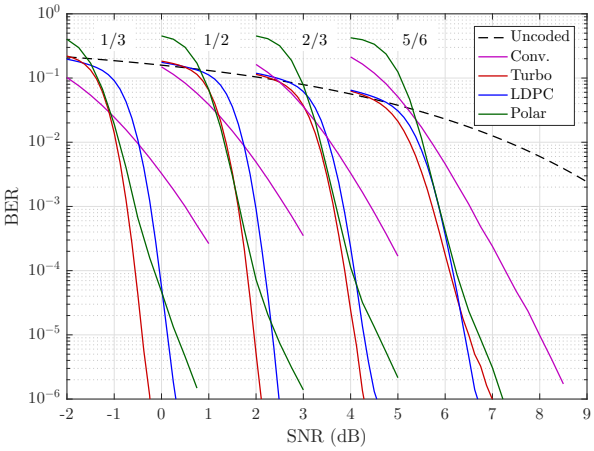

Fig. 11. BER comparison for different code rates, $K = 1024$ (For LDPC, $K = 1020$ for $R = 1/2, 1/3,$ and $5/6$.)

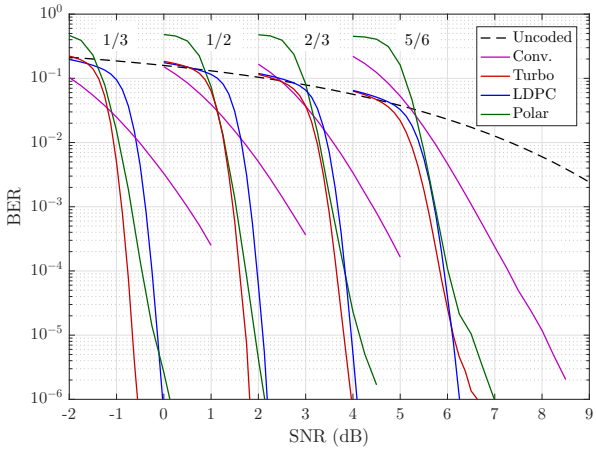

Fig. 12. BER comparison for different code rates, $K = 2048$ (For LDPC, $K = 2052$ for $R = 1/2,$ and $1/3$, and $K = 2040$ for $R = 5/6$.)

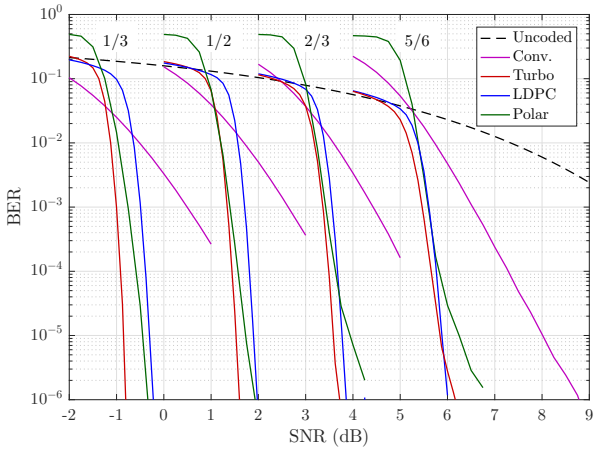

Fig. 13. BER comparison for different code rates, $K = 4096$ (For LDPC, $K = 4092$ for $R = 1/2,$ and $1/3$, and $K = 4100$ for $R = 5/6$.)

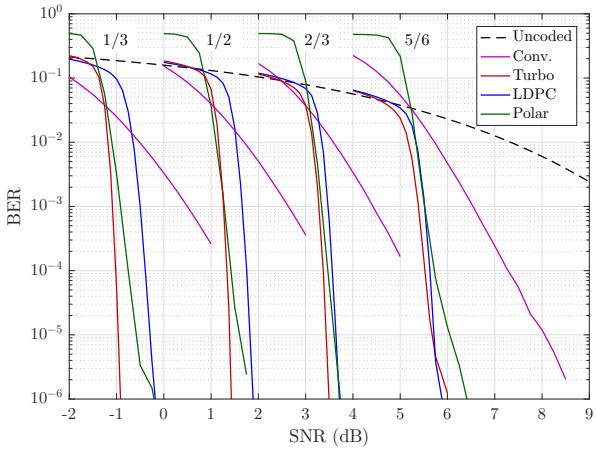

Fig. 14. BER comparison for different code rates, $K = 8192$ (For LDPC, $K = 8196$ for $R = 1/2,$ and $1/3$, and $K = 8200$ for $R = 5/6$.)

VII. Conclusion

In this paper, we provide a BER comparison between the coding schemes: convolutional, turbo, LDPC and polar codes for different scenarios. We consider the current state-of-the-art practical codes, except of polar codes, where a custom construction as described in Section VI.A was applied.

With exception of the convolutional code, the other schemes perform close to each other, which is the more true the larger the information block length is.

References

[1] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, July 1948.

[2] P. Elias, "Coding for noisy channels," *IRE Convention Record*, pp. 37–46, 1955.

[3] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, April 1967.

[4] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (corresp.)," *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.

[5] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *IEEE International Conference on Communications, 1993. ICC '93 Geneva. Technical Program, Conference Record*, vol. 2, May 1993, pp. 1064–1070 vol.2.

[6] R. G. Gallager, *Low Density Parity Check Codes,*. Sc.D. thesis, MIT, Cambridge, 1960.

[7] D. J. C. MacKay and R. M. Neal, "Near shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 33, no. 6, pp. 457–458, Mar. 1997.

[8] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.

[9] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, July 2009.

[10] J. V. Wonterghem, A. Alloum, J. J. Boutros, and M. Moeneclaey, "Performance comparison of short-length error-correcting codes," *CoRR*, 2016. [Online]. Available: http://arxiv.org/abs/1609.07907

[11] T. Hehn and J. B. Huber, "LDPC codes and convolutional codes with equal structural delay: a comparison," *IEEE Transactions on Communications*, vol. 57, no. 6, pp. 1683–1692, June 2009.

[12] S. V. Maiya, D. J. Costello, T. E. Fuja, and W. Fong, "Coding with a latency constraint: The benefits of sequential decoding," in *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sep 2010, pp. 201–207.

[13] K. Fagervik and A. S. Larssen, "Performance and complexity comparison of low density parity check codes and turbo codes," *Norwegian Signal Processing Symposium*, 2003.

[14] Üstün Özgür, "A Performance Comparison of Polar Codes with Convolutional Turbo Codes," Master's thesis, Bilkent University, Turkey, 2009.

[15] N. Andreadou, F. N. Pavlidou, S. Papaharalabos, and P. T. Mathiopoulos, "Quasi-Cyclic Low-Density Parity-Check (QC-LDPC) codes for deep space and high data rate applications," in *Satellite and Space Communications, 2009. IWSSC 2009. International Workshop on*, Sep. 2009, pp. 225–229.

[16] E. Arikan, N. ul Hassan, M. Lentmaier, G. Montorsi, and J. Sayir, "Challenges and some new directions in channel coding," *Journal of Communications and Networks*, vol. 17, no. 4, pp. 328–338, Aug. 2015.

[17] "Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding," 3rd Generation Partnership Project (3GPP), TS 36.212, 2016.

[18] W. E. Ryan and S. Lin, *Channel Codes: Classical and Modern*. Cambridge University Press, 2009.

[19] W. Koch and A. Baier, "Optimum and sub-optimum detection of coded data disturbed by time-varying intersymbol interference [applicable to digital mobile radio receivers]," in *Global Telecommunications Conference, 1990, and Exhibition. 'Communications: Connecting the Future', GLOBECOM '90., IEEE*, Dec. 1990, pp. 1679–1684 vol.3.

[20] J. C. Ikuno, S. Schwarz, and M. Simko, "LTE Rate Matching Performance with Code Block Balancing," in *17th European Wireless 2011 - Sustainable Wireless Technologies*, April 2011, pp. 1–3.

[21] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, Sep. 1981.

[22] S. Myung, K. Yang, and Y. Kim, "Lifting methods for quasi-cyclic LDPC codes," *IEEE Communications Letters*, vol. 10, no. 6, pp. 489–491, June 2006.

[23] D. Divsalar, H. Jin, and R. McEliece, "Coding theorems for turbo-like codes," *Proc. 36th Annual Allerton Conf. on Communication, Control, and Computing*, pp. 201–210, Sep. 1998.

[24] T. J. Richardson and R. L. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 638–656, Feb. 2001.

[25] H. Kfir and I. Kanter, "Parallel versus sequential updating for belief propagation decoding," *Physica A Statistical Mechanics and its Applications*, vol. 330, pp. 259–270, Dec. 2003.

[26] M. M. Mansour and N. R. Shanbhag, "Turbo decoder architectures for low-density parity-check codes," in *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, vol. 2, Nov. 2002, pp. 1383–1388 vol.2.

[27] M. P. C. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Transactions on Communications*, vol. 47, no. 5, pp. 673–680, May 1999.

[28] H. Vangala, E. Viterbo, and Y. Hong, "A comparative study of polar code constructions for the AWGN channel," July 2015. [Online]. Available: https://arxiv.org/abs/1501.02473v1

[29] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "LLR-based successive cancellation list decoding of polar codes," *IEEE Transactions on Signal Processing*, vol. 63, no. 19, pp. 5165–5179, Oct. 2015.

[30] I. Tal and A. Vardy, "List decoding of polar codes," in *2011 IEEE International Symposium on Information Theory Proceedings (ISIT)*, July 2011, pp. 1–5.

[31] "Contention-free Interleaver designs for LTE Turbo Codes," Motorola, R1- 070054, 2007.

[32] "IEEE Standard for Air Interface for Broadband Wireless Access Systems," Institute of Electrical and Electronics Engineers (IEEE), IEEE Std 802.16, 2012.

[33] H. J. Joo, S. N. Hong, and D. J. Shin, "Design of rate-compatible ra-type low-density parity-check codes using splitting," *IEEE Transactions on Communications*, vol. 57, no. 12, pp. 3524–3528, Dec. 2009.