Sparse Construction of Joint Viterbi Detector Decoder (JVDD) Codes

Ashish James, Kheong Sann Chan, and Sari Shafidah Binte Shafee
Data Storage Institute (DSI), A*STAR (Agency for Science Technology and Research)
Singapore 117608
Email: {jamesa, CHAN_Kheong_Sann, Sari_S}@dsi.a-star.edu.sg

***Abstract*—The Joint Viterbi detector decoder (JVDD) has been proposed as an alternative to the iterative detector, performing both detection and decoding in two stages on a trellis. The first stage estimates and retains a set of survivors, while the second stage performs a parity check on these to compute the minimum metric legal codeword (MMLC). With this structure, near optimal maximum-likelihood decoding (MLD) performance can be achieved but at the cost of complexity especially at long codeword lengths (CWL). JVDD codes have been introduced with the explicit target of reducing this complexity. Further, lower rate codes with more parity checks leads to reduced number of survivors in the JVDD trellis resulting in lower complexity. However, it has been observed that JVDD code performance degrades at low-rates while operating in the low SNR region through an error-floor. This aspect is analyzed in this paper and this performance can be attributed to the JVDD code structure where a constant number of ones are placed to the left of the main diagonal. This results in a dense region of ones at the top left corner where the entire rows are filled with ones. In this paper, a modification to the JVDD code construction is proposed by adjusting the number of ones placed in those rows where the row weight of JVDD codes is less than the available row width. It has been found that such sparse construction results in reducing JVDD complexity, as well as eliminates the error-floor problem. This has been verified through extensive simulation studies.**

Keywords*–*JVDD; Iterative detector; ISI Channel; Sparse construction of JVDD Codes.

I. Introduction

Iterative detectors and decoders have been a subject of intense research due to their outstanding error correcting capabilities with performance very close to the Shannon limit. Although iterative decoding can achieve channel capacity as the block size goes to infinity, there is still a gap with the optimal maximum-likelihood (ML) decoder for any code-structure [1]. ML decoders have been analyzed over different communication channels - additive white Gaussian noise (AWGN) [2] [3], binary symmetric channel (BSC) [3], binary erasure channel (BEC) [4] among others. These have also been employed for two broad classes of codes namely block codes and convolutional codes. With convolutional codes, efficient trellis based algorithm known as Viterbi algorithm (VA) can be employed to perform ML decoding and return the most probable transmitted codeword [5]. However, optimal ML decoding of linear block codes has been proven to be an NP-hard problem [6], whose complexity grows exponentially as the code length increases. There have been many research efforts in this direction to develop optimal or suboptimal decoding algorithms with moderate complexity [7]–[11].

In [11], they introduce the joint Viterbi detector decoder (JVDD) as an alternative optimal ML detection and decoding scheme that attempts to return the minimum metric legal codeword (MMLC) . It operates on a trellis and has a two-stage decoding structure - metric thresholding and parity checking. The first stage executes the normal VA by computing metrics for every possible path to a node. However, the JVDD retains the minimum metric survivor along with a certain number of competing paths in the trellis constrained by a threshold parameter. Thus only survivors with metrics within the threshold of the minimum metric for a particular node are retained. This would typically mean setting a larger threshold to minimize the probability of discarding the MMLC. However, this leads to a larger number of survivors resulting in an increased complexity. The second stage, aims at reducing the complexity by performing parity checking on each incoming survivor path and discarding those which fail the syndrome check, i.e., $c\mathbf{H}^T = 0$, where c is the codeword and $\mathbf{H}$ is the parity check matrix. This parity checking section provides a system tradeoff design between complexity and code-rate. Thus JVDD performance complexity can be reduced by increasing the number of parity checks per bit, i.e., by operating at low code-rates. However, it is found that JVDD codes exhibits an error-floor at low code-rates while operating in the low SNR region. This is the focus of the present paper. The error floor is characterized by a more gradual decrease in error rate as code-rate decreases and can be attributed to the JVDD code structure.

JVDD codes are currently designed with a parity check matrix that has constant number of ones in a row (row weight) placed according to a Gaussian distribution to the left of the diagonal. However, this construction results in a dense region of ones towards the top left corner where the entire row gets filled with ones where the length of the row (row width) is lower than the row weight as observed in Figure 1 (In this figure, the 1's in the parity check matrix are represented as black dots while white spaces represent the 0's). This paper analyses the impact of this dense region through a modification of this construction by adjusting the row weight in this region of the parity check matrix. The results indicate that the error-floor can be mitigated by eliminating this dense region of ones.

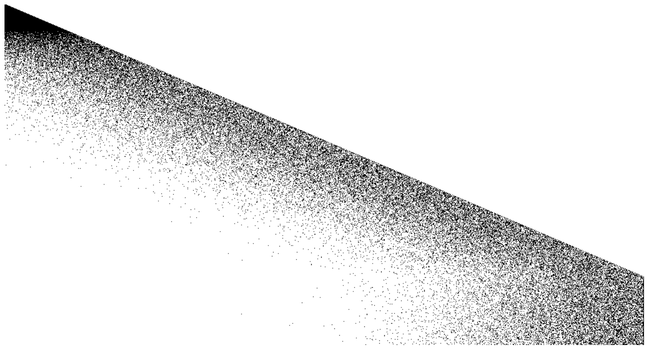


Figure 1. Pictorial depiction of the parity check matrix $\mathbf{H}$ of JVDD codes for codeword length of 1024 and 0.5 rate.

The rest of the paper is organized as follows. Section II describes the JVDD algorithm along with sparse construction of JVDD codes. Simulation results are presented in Section III, followed by concluding remarks in Section IV.

II. Sparse Construction of JVDD Codes

In this paper, binary linear block coding for BPSK signalling over an inter-symbol interference (ISI) channel is considered. Then, the received signal y_k at time k is given as

$$y_k = \sum_{l=0}^{L} f_l x_{k-l} + w_k \tag{1}$$

where f_l is the channel impulse response of order L, x_k is the transmitted encoded bit sequence and w_k corresponds to AWGN with zero mean and variance N_0.

A. JVDD Codes

The receiver is implemented based on the JVDD algorithm, which operates on a trellis where the paths through the trellis correspond to the codewords c that satisfy the parity check condition: $c\mathbf{H}^T = 0$. However, the received sequence y_k may not correspond to a codeword due to the channel and the JVDD finds the path through the trellis which is the closest to the received sequence y_k that is also a legal codeword. This corresponds to the maximum likelihood criteria represented as

$$\max_{c \in C} \sum_{k=0}^{N-1} \ln \Pr(y_k|c_k) = \min_{c \in C} \sum_{k=0}^{N-1} \gamma(y_k|c_k) \tag{2}$$

where $\gamma(y_k|c_k)$ is the branch metric. The branch metrics corresponds to the weights of the trellis transitions and the calculation for each possible transition from state i to l at time step k is given as

$$\gamma_{k,k+1}^{i,l} = c_k y_k = \begin{cases} y_k & \text{if } i \neq l \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

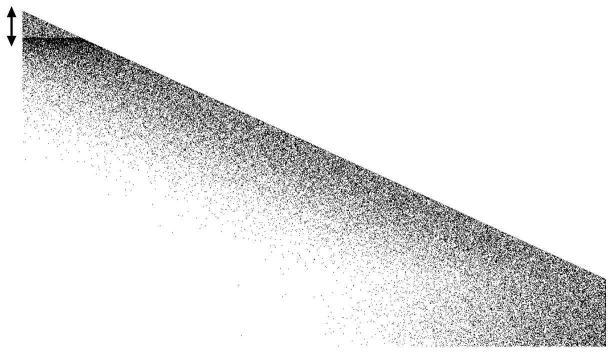
For each state s_{k+1} at time $k+1$, the state metric $\alpha_{l,k+1} = \min(\alpha_{l,k}, \alpha_{i,k} + \gamma_{k,k+1}^{i,l})$ are calculated and correspond to the survivor paths. In the usual Viterbi algorithm, the metrics for each incoming survivor to a node are computed and the larger-metric survivors are discarded. However, this might result in discarding the MMLC and deteriorate the result of subsequent detections. To resolve this problem, the JVDD adjusts the number of surviving paths through metric thresholding, which computes the path metrics but discards survivors with metrics larger than the *threshold*. Let τ denote the threshold, which determines the number of competing paths that are retained for each state to enhance the information update process. Accordingly, the survivors with state metric $\alpha_{l,k+1} < \tau + \alpha_{min}$ (where α_{min} is the minimum metric) are retained. This would typically mean setting a larger threshold for minimizing the probability of discarding the MMLC. However, this leads to a larger number of survivors resulting in an increased complexity.

The complexity of the JVDD by retaining survivors in the metric thresholding section can be reduced through the parity checking stage that follows. Parity checking occurs on specific nodes corresponding to the last one of a given row of the parity check matrix. At these particular nodes, all bits required to perform the check are detected and the syndrome can be computed. In this stage, the JVDD performs parity checks on each incoming survivor path to the parity check node and discards paths which fail the syndrome check, i.e., $ch_i^T = 0$ where c is the detected codeword and h_i^T is the transpose of the ith row of the parity check matrix $\mathbf{H}$. The parity check matrix $\mathbf{H}$, is an $M \times N$ matrix where M is the number of parity checks and N is the number of coded bits, with each row corresponding to one of the M parity checks. In this context, JVDD codes were designed to evenly space the parity checking functionality throughout the trellis resulting in fewer number of survivors especially as the codeword length (CWL) increases. This is achieved by designing the parity check matrix with a constant number of ones in a row (row weight) placed according to a Gaussian distribution to the left of the main diagonal. However, this construction results in a dense region of ones towards the top left corner where the entire row gets filled with ones as the length of the row (row width) is lower than the row weight as observed in Figure 1. This paper analyses the impact of this dense region through a modification of this construction as described in the following section.

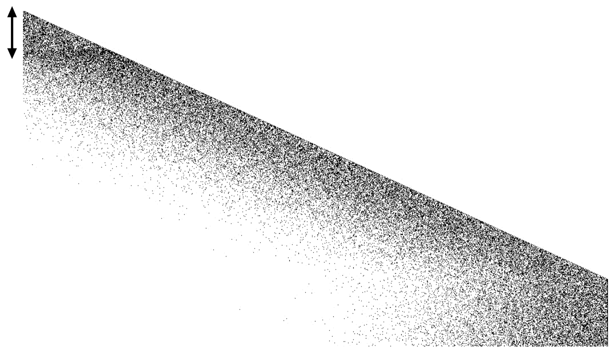
B. Sparse JVDD Codes

Since the JVDD codes are generated using a parity check matrix with a fixed row weight, traditional JVDD code construction results in a dense region of ones when the row width is less than the row weight. This aspect is addressed in this paper through the sparse JVDD code construction by adjusting the row weight in this region of the parity check matrix $\mathbf{H}$. In sparse JVDD code construction, instead of having a constant row weight, the number of 1's placed in $\mathbf{H}$ is varied according to the width of the row. This paper considers two levels of depth for sparseness - region where the row width is less than or equal to the row weight and region where the row width is equal to 1.5 times the row weight. The number of 1's in these rows is placed randomly based on Gaussian distribution to the left of main diagonal and is a factor of the row width. This results in a less dense $\mathbf{H}$ matrix where the rows till the depth of sparseness is filled with only a specified percentage of 1's based on the row width and is referred to hereby as the level of sparseness. Figure 2 is a pictorial representation of a parity check matrix generated through such a construction with 50% sparseness, i.e., filling 50% of the row width with 1's.

The effect of such a sparse construction on the column weight of parity check matrix is shown in Figure 3. It is seen



(a) Depth - Row width $\leq$ Row weight



(b) Depth - Row width $\leq 1.5\times$Row weight

Figure 2. Pictorial depiction of the parity check matrix $\mathbf{H}$ of Sparse JVDD codes for codeword length of 1024 and 0.5 rate at different depth of sparseness. 1's are represented as black dots while white spaces represent 0 in these figures.

that sparse construction results in lowering the column weight for the depth of sparseness.

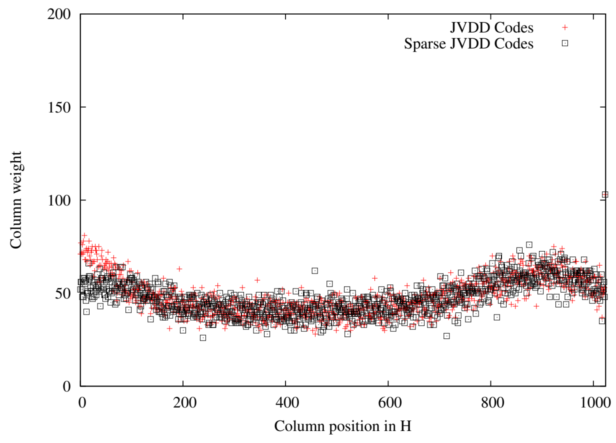


Figure 3. Variation of column weight of parity check matrix **H** of Sparse and traditional JVDD codes for codeword length of 1024.

The class of JVDD codes used in this paper is the variable-gradient Gaussian distribution linear diagonal (VGGDLD) codes introduced in [12], where the gradient of the diagonal is varied through two independent shift parameters dx and dy. As observed in [12], increasing the parameter dx leads to increased number of survivors in the trellis as the parity checking gets shifted in the trellis, thus the optimum value is set to $dx = 0$. However, the parameter dy has to be designed optimally, where too small a value leads to under-protected bits at the end of the codeword, and too large a value reduces the number of parity checks which increases the complexity of the trellis. The impact on performance and complexity of JVDD by implementing sparse construction on the VGGDLD codes is studied in this paper.

III. Simulation Results

In this section, performance results of sparse JVDD codes are analyzed. For simulations, the codeword length (CWL) is fixed at 1024, i.e., the number of columns N in the parity check matrix **H** is 1024. The number of rows M in the parity check matrix **H** is varied from 512 to 102 which correspond to code-rates 0.5 to 0.9. The best VGGDLD codes with typical values of dx=0% of N and dy=20% of M are employed for the simulations. Sparse construction is implemented on such codes and the level of sparseness is varied to determine the optimal sparseness for such codes. JVDD performance is also compared with iterative detector employing random codes at the same code-rates which are used as a benchmark. The simulation parameters are specified in Table I.

TABLE I. Simulation Parameters

Codeword Length (CWL)	1024
Code-rate (R)	[0.90, 0.85, 0.80, 0.70, 0.60, 0.50]
SNR (dB)	[6, 7]
[dx (% of N), dy (% of M)]	[0, 20]
Row weight	100
Iterative detector	Random code (column weight - 4)
JVDD Max. No. Survivors	30000
Channel (f)	$\frac{1}{\sqrt{6}}$[1 2 1]
Level of sparseness	[1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 99]

A. Complexity of JVDD with sparse codes

Initially, the complexity of the JVDD algorithm employing sparse codes is analyzed, where the complexity is measured as the average number of survivors in the JVDD trellis. This set of simulations are performed for a sparseness depth where the row weight is less than or equal to the row width. Figure 4 depicts the variation of complexity with threshold for a code-rate of 0.5 at an SNR of 6 dB. Typically JVDD performance enhances with threshold but at the cost of increasing complexity [11]. The level of sparseness is also varied and it is observed that the complexity increases when the JVDD codes are too sparse (1%) or highly dense (99%). This basically means an increase in the number of survivors when JVDD is processing that region. Thereby, it is desirable to produce codes that minimize JVDD complexity, as analyzed in the following section.

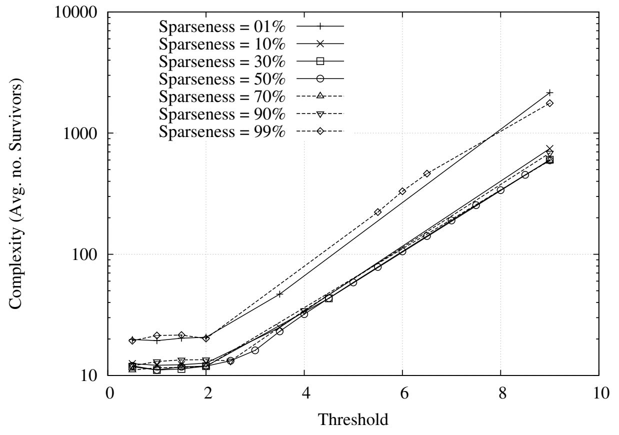


Figure 4. Variation of JVDD complexity with threshold and level of sparseness at SNR of 6 dB and 0.5 code-rate

In order to realize the optimal level of sparseness, the complexity of JVDD is analyzed against sparseness and frame error rate (FER) as depicted in Figure 5 and 6, respectively. Similar to the previous observation, too sparse and highly dense codes increases the complexity of the JVDD. However, the complexity is found to minimize and saturate when sparseness is in the range 30 - 70% as observed in Figure 5. Further, in Figure 6 the optimal level of sparseness to achieve the lowest FER is found to be 50%. Thereby from Figures 4, 5 and 6, typical level of sparseness considered for further analysis is 50%.

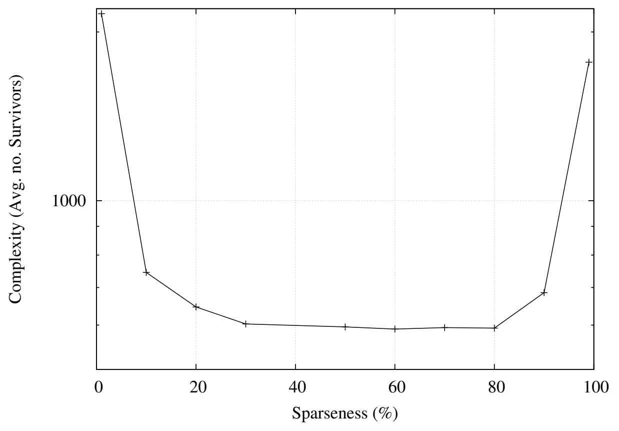


Figure 5. Variation of JVDD complexity with sparseness at SNR of 6 dB and 0.5 code-rate

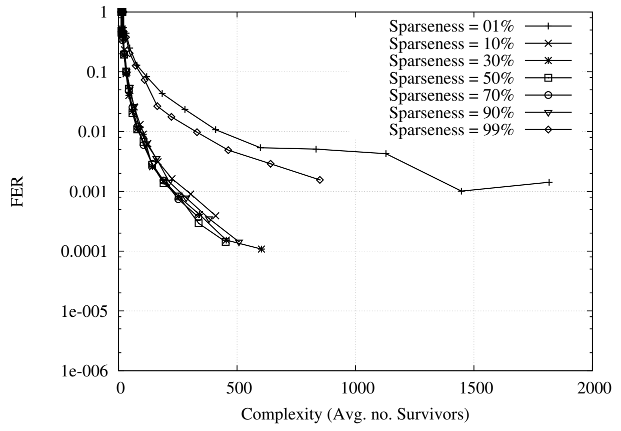


Figure 6. FER comparison with JVDD complexity at different level of sparseness at SNR of 6 dB and 0.5 code-rate

Figure 7 depicts the JVDD complexity at different depths of sparseness and varying code-rates at an SNR of 6 dB. The depth of sparseness refers to the number of rows where sparseness is introduced into the JVDD codes. Two levels are compared where sparseness is introduced in the region where row weight is less than or equal to - the row width and 1.5 times the row width. It is observed that the complexity remains almost the same irrespective of the depth of sparseness at different code-rates. Further it is shown that JVDD complexity increases with code-rate. This can be attributed to the fact that at lower code-rates, there are more parity checks per information bit which means that parity checking will occur more frequently in the JVDD trellis thereby keeping the number of survivors more manageable.

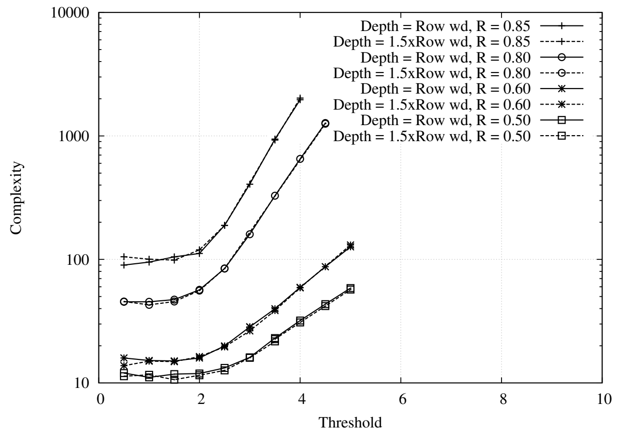
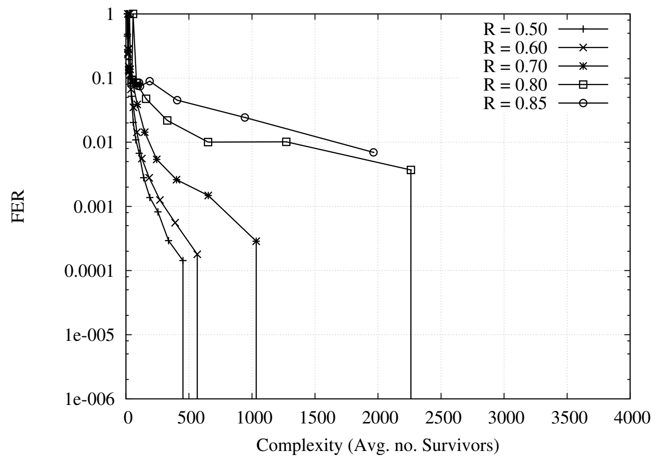
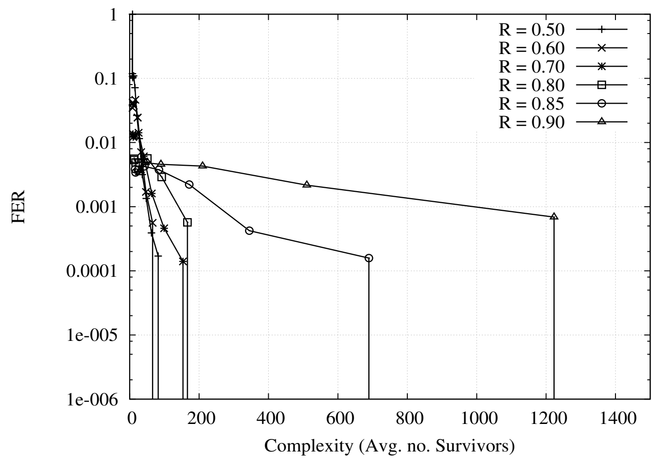


Figure 7. Variation of JVDD complexity with threshold at different depths of sparseness and code-rates at SNR 6 dB

Figure 8 shows the variation of FER with complexity of JVDD at different rates. The operating SNRs considered for comparison is 6 and 7 dB. It is observed that the complexity increases with code-rate and decreases with increasing SNR. Lowering the code-rate is expected to reduce errors in any channel at the cost of increased coding overhead through the increased number of parity check bits. For the JVDD, this also results in reducing the number of survivors as parity checking kills survivors that violate the syndrome. Further, increasing the SNR will result in reduced distortions from the channel and will lower the chance of deviating from the correct path in the JVDD trellis. This results in fewer survivors and reduced complexity. These trends indicate that operating at low code-rates would benefit JVDD due to reduced complexity but the original JVDD code performance is found to deteriorate in these conditions due to the dense region of ones in the upper left corner of the **H** matrix as observed in the following section.



(a) SNR = 6 dB



(b) SNR = 7 dB

Figure 8. FER comparison with JVDD complexity at different rates and SNRs.

B. FER Performance of Sparse JVDD Codes for Different Rates

Sparse JVDD code performance is compared with traditional JVDD codes and iterative detector at various code rates in Figure 9. It is known that JVDD code performance improves with increasing complexity and the performance comparison needs to be performed by normalizing the complexity. In this work, the complexity is normalized to 1000 and 2000 survivors, respectively. It is observed that the original JVDD code performance (with the dense region of ones) deteriorates at low code-rates through the appearance of an error floor which is characterized by a gradual decrease in error rate as code-rate decreases. This performance degradation has been mitigated through the sparse construction developed in this paper. This confirms that the performance loss of traditional JVDD codes can be attributed to the code construction which results in a dense region of ones towards the top left corner where the entire row gets filled with ones as the length of the row (row width) is much lower than the row weight as observed in Figure 1.

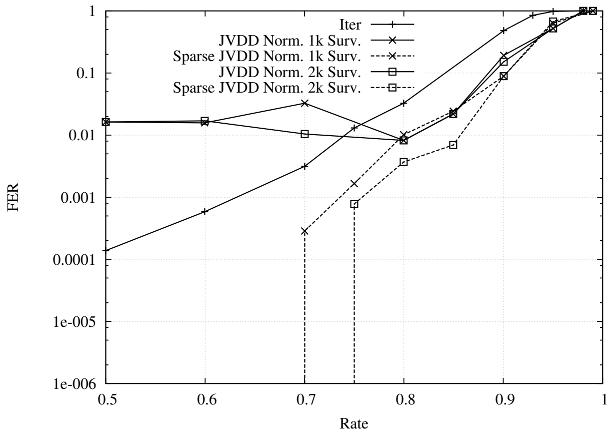


Figure 9. FER performance at different rates at an operating SNR of 6 dB with complexity normalized to 1000 and 2000 survivors for JVDD codes

The sparse construction mitigates the error floor and JVDD is found to outperform the iterative detector at all code-rates. This is due to the fact that iterative detector becomes more efficient for increasing codeword lengths. However, at longer codeword lengths, JVDD becomes more computationally complex. The complexity of JVDD can be controlled through the threshold parameter and increasing the threshold results in retaining more number of survivors in the JVDD trellis. Conversely, this lowers the probability of discarding the MMLC through the metric thresholding section of the JVDD and results in enhanced performance. The performance improvement of JVDD with increasing complexity (average number of survivors) is also depicted in Figure 9.

IV. Conclusions

JVDD algorithm is divided into two stages and the second stage (parity checking) substantially reduces the decoding complexity at low code-rates owing to the increased number of parity checks being performed. However, performance degradation is observed for JVDD at these rates in the low SNR region due an artifact of traditional JVDD code construction. This aspect is analyzed in this paper and attributed to the JVDD code structure where a constant number of ones are placed to the left of the main diagonal. This results in a dense region of ones at the top left corner where the entire rows are filled with ones. A modification to the code construction is proposed in this paper by adjusting the number of ones placed in those rows where the row weight of JVDD codes is less than the available row width. It has been found that too sparse and highly dense region of ones will result in increased JVDD complexity and 50% sparseness will result in optimal JVDD performance. The analysis performed in this paper has been verified through simulation studies.

References

[1] D. Burshtein and G. Miller, “Bounds on the performance of belief propagation decoding,” IEEE Trans. Inf. Theory, vol. 48, no. 1, Jan. 2002, pp. 112–122.

[2] C.-C. Chao, R. McEliece, L. Swanson, and E. R. Rodemich, “Performance of binary block codes at low signal-to-noise ratios,” IEEE Trans. Inf. Theory, vol. 38, no. 6, Nov. 1992, pp. 1677–1687.

[3] S. Haykin, Ed., Communication Systems. 4th Ed., J. Wiley, 2001.

[4] A. Khandekar and R. J. McEliece, “On the complexity of reliable communication on the erasure channel,” in Intl. Symp. Inform. Theory (ISIT), 2001, pp. 7803–7123.

[5] A. J. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” IEEE Trans. Inf. Theory, vol. 13, no. 2, Apr. 1967, pp. 260–269.

[6] S. G. Wilson, Ed., Digital Modulation and Coding. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[7] J. Forney, G., “Generalized minimum distance decoding,” IEEE Trans. Inf. Theory, vol. 12, no. 2, Apr. 1966, pp. 125–131.

[8] D. Chase, “Class of algorithms for decoding block codes with channel measurement information,” IEEE Trans. Inf. Theory, vol. 18, no. 1, Jan. 1972, pp. 170–182.

[9] M. P. C. Fossorier and S. Lin, “Soft-decision decoding of linear block codes based on ordered statistics,” IEEE Trans. Inf. Theory, vol. 41, no. 5, Sep. 1995, pp. 1379–1396.

[10] M. P. C. Fossorier, “Reliability-based soft-decision decoding with iterative information set reduction,” IEEE Trans. Inf. Theory, vol. 48, no. 12, Dec. 2002, pp. 3101–3106.

[11] K. S. Chan, S. S. B. Shafiee, E. M. Rachid, and Y. L. Guan, “Optimal joint viterbi detector decoder (JVDD) over AWGN/ISI channel,” in Intl. Conf. Computing, Networking and Communications (ICNC), Feb. 2014, pp. 282–286.

[12] K. S. Chan and S. S. B. Shafiee, “A study on variable gradient gaussian distribution linear diagonal codes for the JVDD,” in Intl. Conf. Inform. Theory (ICIT), Mar. 2015, pp. 797–801.