Complexity Based Design for Iterative Joint Equalization and Decoding

Sriram Vishwanath[†] Mohammad Mansour[*] and Ahmad Bahai[*]
[†]Stanford University, Stanford, CA 94305.
[*]National Semiconductor Corp., Semiconductor Drive, Santa Clara, CA USA
E-mail :*sriram@wsl.stanford.edu, mmansour@uiuc.edu, ahmad.bahai@nsc.com*

Abstract—**In this paper we motivate the need for complexity based design for performing joint iterative equalization and decoding. This joint iterative process, which requires the exchange of soft information, incurs a huge complexity increase over hard-decision based algorithms. We introduce complexity as a design parameter and provide two different methodologies. The first approach is a combination of SOVA and DFSE, and is called soft-output DFSE (SO-DFSE). The second approach, called soft-decision DFSE (SD-DFSE) generalizes the notion of reliabilies to soft-decisions through the use of appropriately chosen functions. By varying the design parameters in both approaches, the module can range from being as simple as a soft output DFE to being as complex as a SOVA or APP. We conclude by presenting performance curves of iterative algorithms that utilize these modules.**

I. Introduction

Joint Iterative Equalization and Decoding, a concept first presented as turbo equalization in [1], models the channel as a time varying convolutional code and performs iterative decoding by passing reliability information between two soft-output modules (which we shall call "modules" or "algorithms" interchangeably in the rest of this paper)(Figure 2), one functioning as an equalization and the other as a decoding module. The soft-output modules used by most researchers to this end are either Soft Output Viterbi Algorithm (SOVA) modules [2] or *a-posteriori* probability (APP) modules based on the BCJR algorithm [1]. The complexity and performance of these two modules are compared in [3]. This technique of performing iterative decoding using SOVA modules or BCJR based APP modules shows excellent performance improvements over non-iterative hard decision schemes, but it poses challenges in implementation since it leads to a huge increase in computational complexity and storage requirements[1] over hard decision algorithms. Table I (from [4]) provides a concrete measure of this complexity increase. Note that the complexity of SOVA is roughly twice that of the Viterbi algorithm [5]. Thus, If the number of iterations performed is M, then Table I and [5] indicate that present day iterative algorithms require a complexity increase of at least $2M$. This sharp increase in power consumption and storage requirements at the receiver makes their usage difficult on the downlink in present day wireless systems.

Method	Delay	Products	Sums	Memory
Viterbi	D	$2N.N_I$	$N.N_I$	$D.N$
APP	K	$4NN_I$	$2N.N_I$	$> K.N_I$

TABLE I
Complexity comparison of Viterbi and BCJR based APP, where D minimum delay , N Number of states, K Block length and N_I is the cardinality of the input alphabet.

Although low complexity soft-output algorithms are relatively few, low complexity solutions to the conventional hard decision Viterbi algorithm, called the decision feedback sequence estimation (DFSE) algorithms have been proposed in [6], [7]. In DFSE, a part of the channel impulse response is taken as the trellis definition, while the remaining ISI is removed by a separate decision feedback for each state. Depending on the length of the part of the channel response that is taken as the trellis definition, the DFSE can range from being the same as a zero forcing decision feedback equalizer (DFE) to the same as the Viterbi Algorithm operating on the whole trellis.

A similar algorithm has also been proposed for decoding of binary convolutional codes on AWGN channels [8]. Previously, decision feedback structures (like DFSEs) have been used primarily in the area of equalization. The use of DFSE for decoding of convolutional codes introduces a new concept - that convolutional and trellis encoders can be looked at as "controlled ISI" introduced by the transmitter. Thus "equalizers" can be built for coded systems that act as decoders. In this paper, we shall use this new concept while constructing our new low complexity decoders.

Various papers have addressed the problem of combining low complexity equalizers with decoders. In [9], a combination of reduced-state sequence estimator (RSSE) as the channel equalizer and Viterbi decoder is proposed. In [10] and [11], DFEs are used as channel equalizers and are combined with decoders by using the notion of tentative or soft decisions for feedback. However, these techniques are non-iterative. In [12], turbo equalization using DFEs as equalizers and hard decision Viterbi algorithm as the decoder is proposed. However, iterative decoding using low complexity soft-decision modules *both* as the equalizer and decoder are yet to be studied in detail. In this paper, we present two different approaches to the problem of obtaining modules that generate soft-outputs *and* allow complexity to be a design parameter. In both approaches, our iterative algorithms employ modules that can range from being as high

[1]this increase is when they are implemented digitally

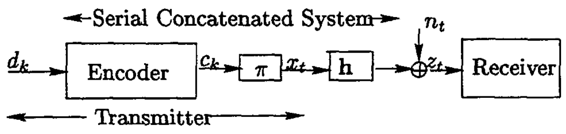

Fig. 1. The System Model

as that of an APP module to as low as the soft-output DFE algorithm. We term these soft DFSE (SDFSE) modules. For our first approach, we show that both the APP [1] and the SOVA [2] algorithms can be suitably modified and combined with the DFSE algorithm in [7]. We call SDFSEs obtained thus as soft-output DFSEs or as SO-DFSEs.

Next, we observe that the notion of soft-output presently used by most researchers is restricted to that of "reliabilities", i.e. probability ratios. in the second approach, we generalize them to "soft-decisions", where the output of these modules is some continuously differentiable function of their inputs. This function can be chosen in different ways [12]. We call these SDFSEs as soft-decision DFSEs or as SD-DFSEs. Lastly, we explore the combination on soft-output and hard-output modules in iterative decoding.

The remainder of this paper is organized as follows. In the next section we present the system model. In Section III, we present an overview of the DFSE and the SOVA algorithms. In Section IV, we present soft-output DFSEs, while in Section V we present soft-decision DFSEs. In Section VI, we present some simulation results and in Section VII we conclude the paper.

II. System Model

We use P_x to denote the *a-priori* probability of a variable x. If x is binary, we use L_x to denote its reliability defined as $\log[P_{(x=+1)}/P_{(x=-1)}]$.

We consider a discrete time transmission model as shown in Figure 1. Since the timing, in terms of number of uses per second of the encoder and channel may differ, we use k to denote the time index for the encoder and t to denote the time index of the channel. The source $\{d_k\}$ are encoded by a convolutional encoder of memory ϕ. The encoded bits $\{c_k\}$ are then interleaved (using an interleaver denoted as π in the Figure 1). These interleaved symbols $\{x_t\}$ are transmitted through an Inter Symbol Interference (ISI) channel with a finite length impulse response $\mathbf{h} = \{h_0, h_1, \dots, h_\tau\}$ with output $\{z_t\}$. We assume d_k, c_k to be vectors with binary elements drawn from $\{-1, +1\}$ and that $x_t \in \{-1, +1\}$. Note that z_t need not be binary in general. Mathematically, we have

$$z_t = \sum_{i=1}^{\tau} x_{t-i} h_i + n_t$$

The channel is assumed to have additive circularly symmetric complex Gaussian noise with variance σ^2. This sampled signal is now processed by an iterative receiver as shown in Figure 2. In Figure 2, the soft-output symbol detector is a soft-output module that is dedicated to "equalizing" the channel, i.e., to reversing the ill-effects of the ISI, while the soft-output channel decoder is dedicated to decoding the convolutional code. After performing many iterations, hard decisions will be made on the soft-output of the channel decoder to obtain the decoded bits.

Throughout this paper, we assume that the receiver has perfect channel state information at all time (perfect RSI), i.e., knows $\mathbf{h}$. Also, we assume that the transmitter has no knowledge of the channel. This completes the system model description. In the next section, we provide the necessary background that helps understand the new algorithms presented later in this paper.

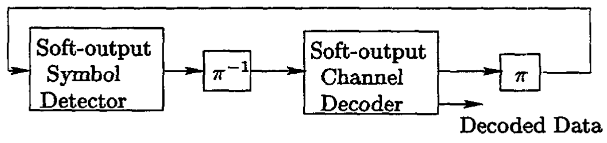

Fig. 2. Joint Iterative Equalizer and Decoder

III. Background : DFSE, SOVA and APP Algorithms

As we are introducing algorithms that have applications as either equalization or decoding modules, we shall use $\{y_i\}$ to denote the input and $\{u_i\}$ to denote the output of some module trying to reverse the effects of an encoder/channel of memory μ. Thus, for a decoding module, $i = k$, y_k is a noisy version of c_k, u_k is the estimate of d_k, and $\mu = \phi$ while for the equalization module $i = t$, $y_t = z_t$, u_t is an estimate of x_t and $\mu = \tau$.

A. DFSE

First, we provide a brief description of the DFSE algorithm in [7]. The DFSE permits us to choose the desired complexity by choosing the trellis size[2] to be 2^γ, where $0 \leq \gamma \leq \mu$ is a design parameter. Once γ is chosen, we use $u_{i-1}, \dots u_{i-\gamma}$ as the trellis state, and the DFSE recursion proceeds as follows:

The DFSE Algorithm [7]

1. At time instant $i-1$, associate each state in the trellis diagram with the best path leading to that state, the metric of the best path and an estimate of $u_{i-\gamma-1}, \dots, u_{i-\mu}$.
2. At time i, for each previous-state next-state pair, we compute the branch metric as $\sum_j |y_i - \hat{y}_i|$, where $\hat{y}_i$ is either the output of the encoder with $u_i, \dots, u_{i-\mu}$ as the input or equals $\sum_{j=1}^{\mu} u_{i-j} h_j$ as the case may be. For this, we derive $u_i, \dots, u_{i-\gamma}$ from the state-pair and $u_{i-\gamma-1}, \dots, u_{i-\mu}$ from the estimates associated with the previous-state.

[2]The complexity of any equalization algorithm is proportional to the trellis size

3. Using the branch metric we compute the path metric for all paths leading to a state s and select the path with the minimum path metric (exactly the same as add compare select in Viterbi algorithm).
4. Based on the path chosen, choose an estimate $u_{i-\gamma}, \ldots, u_{i-\mu+1}$ to be associated with the state s. □

B. SOVA

Next, we provide an outline of SOVA. Since this algorithm is very well known, and [13] an excellent reference for it, we shall not describe its steps here, but refer to steps in the algorithm in [13] when the need arises. Assuming the simple case where trellis has two paths ending at each node, and that the Viterbi algorithm (VA) has a decision delay of δ, we consider the metrics M_1 and $M_2, M_1 \leq M_2$ of the two paths that meet at any state s_i at time i. This metric is computed using the channel information, the channel state diagram, and *a-priori* reliabilities on the input $\{u_i\}$. Denoting $(1+e^{M_2-M_1})^{-1}$ by p, we update the reliabilities $L_{u_{i,j}}$ of u_i at those places j where the two paths differ as $L_{u_{i,j}} \leftarrow L_{u_{i,j}}(1-p)+(1-L_{u_{i,j}})p$. Thus, the SOVA algorithm generates reliability values on the input symbols as its output.

C. APP

The APP algorithm is very well known, and also, [4] is an excellent reference on it. Hence, we shall only outline the algorithm. We compute state dependent parameters $\alpha_i(s)$ and $\beta_i(s)$ using forward and backward recursions:

$$\alpha_i(s) = \sum_{e:s_e^f=s} \alpha_{i-1}(s_e^s)P(u_i(e))P(y_i(e))$$

$$\beta_i(s) = \sum_{e:s_e^f=s} \beta_{i+1}(s_e^s) + P(u_i(e))P(y_i(e))$$

where P denotes the input *a-priori* probabilities, s_e^s the starting state and s_e^f the ending state of edge e. Next, we find the extrinsic information by

$$P^e_{u_{i,j}=x} = \frac{\sum_{e:u_{i,j}=x}\{\alpha_{i-1}(s_e^s)P(u_i)P(y_i)\beta_k(s_e^f)\}}{P(u_{i,j}=x)}$$

$$P^e_{y_{i,j}=x} = \frac{\sum_{e:y_{i,j}=x}\{\alpha_{i-1}(s_e^s)P(u_i)P(y_i)\beta_i(s_e^f)\}}{P(y_{i,j}=x)}$$

Note that we can use probabilities and reliabilities for binary variables interchangeably using the transformations presented in [13]. In the next section, we present the soft-output DFSE algorithm.

IV. The Soft-Output DFSE Algorithm

As mentioned in Section I, the Soft-output DFSE module is designed to be a low complexity version of SOVA and BCJR based APP modules, and like them, generates reliabilities as its output. First, we present the SOVA based SO-DFSE algorithm.

A. SOVA based SO-DFSE Algorithm

1. The desired complexity level for the algorithm is fixed by choosing the trellis size 2^γ. Between time instants $i-1$ and i, construct a trellis section using $u_{i-1}, \ldots u_{i-\gamma}$ for the trellis definition. This trellis section is referred to as the *reduced trellis.*
2. At time instant $i-1$, associate with each state in the reduced trellis with the best path leading to that state, the metric of the best path and an estimate of $u_{i-\gamma-1}, \ldots, u_{i-\mu}$ (same step as DFSE).
3. At time i, we obtain the metric associated with each previous state next state pair in the reduced trellis as $y_i\hat{y}_i - L_{u_i}/2$ plus the path metric of the previous state at time $i-1$. As before, $\hat{y}_i$ is obtained by combining the state information $u_i, \ldots u_{i-\gamma}$ and the estimates $u_{i-\gamma-1}, \ldots, u_{i-\mu}$ associated with the state. Note that the metric used in this step is identical to that in SOVA ([13]), while the process employed to obtain $\hat{y}_i$ is identical to that used in DFSE.
4. Perform an add-compare select at each state in the reduced trellis at time i. This step is common to the Viterbi, SOVA, DFSE and the SO-DFSE algorithms.
5. Perform either the trace-back or the stack register updation techniques used in SOVA [13] to update the values of L_{u_k}, which are the desired soft-outputs of this module.□

In order to perform iterative equalization and decoding, we must obtain extrinsic information from the $\{L_{u_k}\}$ in order to be interleaved and input to the next module as *a-priori* information. This procedure of computing the extrinsic information $\{L^e_{u_k}\}$ is identical to SOVA and hence is not explained further. Next, we explain how this SO-DFSE algorithm can be used to function as either an equalization or as a decoding module.

B. SO-DFSE as an Equalizer Module

For the SO-DFSE equalizer, the desired inputs to the module are the received symbols z_t, the channel state $\mathbf{h}$ and the *a-priori* reliability information on the interleaved coded bits $L^e_{x_t}$, while the output is the updated reliability information on the interleaved coded bits L_{x_t}.

C. SO-DFSE as a Decoder Module

For the SO-DFSE decoder module, the desired inputs are noisy coded bits (denoted r_k, the encoder state diagram, and the *a-priori* information $L^e_{d_k}$, while the output is the updated reliability information on the input L_{d_k}. Note that there is an incompatibility between the inputs and the outputs of the equalizer and decoder modules. To make them compatible, we need to perform some post-processing on the output of the equalizer, and to modify the decoder module.

C.1 Post-Processing the Output of the SO-DFSE equalizer module

We use the extrinsic information to generate r_k We model r_k as $c_k + n_c[k]$, where $n_c[k]$ is Gaussian and has the same noise variance σ^2 as the additive Gaussian noise in the system model (Section II). This is a valid assumption, since it is equivalent to assuming that the equalizer module has perfectly equalized the channel. Since the extrinsic information is designed to be the *a-priori* information for the next iteration, it can be written as

$$L^e_{c_k} = \log \frac{P(r_k = +1)}{P(r_k = -1)} \qquad (1)$$

This, with the Gaussian assumption gives us

$$r_k = \sigma^2 L^e_{c_k]}/2 \qquad (2)$$

C.2 Obtaining L_{c_k} instead of L_{d_k}

The output desired from the SO-DFSE decoding algorithm is reliability information on the coded symbols, L_{c_k}, while the SO-DFSE algorithm outputs L_{d_k}. We generate L_{c_k} by using the updation $L_{c_{k,j}} \leftarrow L_{c_{k,j}}(1-p)+(1-L_{c_{k,j}})p$ wherever the two paths differ in c_k. We derive extrinsic information $L^e_{c[k]}$ from $L_{c[k]}$, interleave it and provide it to the equalizer as *a-priori* information. This completes our discussion of the SO-DFSE algorithm.

Note that the incompatibility problems arise because the SOVA and the SO-DFSE modules as presented in [13] and in Algorithm IV-A) respectively, are designed to work as such for parallel concatenated systems.

D. APP based SO-DFSE Algorithm

1. Pick a reduced trellis of size 2^γ.
2. At time $i-1$, associate with each state s $\alpha_{i-1}(s)$, $\beta_{i-1}(s)$ (Same as APP). In addition, associate with each state s estimates of $u_{i-\gamma-1}, \dots, y_{i-\mu}$.
3. Generate $\alpha_i(s')$, $\beta_i(s')$ and the extrinsic probabilities as done in the APP algorithm (given in Section III-C.
4. Use the MAP rule to obtain an estimate of u_i.

The APP based SO-DFSE algorithm has the same output as the APP alogirithm, and hence can be used for equalization and decoding modules in the same manner as illustrated in [1].

V. The Soft-Decision DFSE Algorithm

In most of the references cited above, the term soft-output has been used to refer to a probabilistic value. The SOVA algorithm, for instance, generates such a reliability value. However, soft information can in general be some real (unquantized) number that characterizes the input, which we term the "soft-decision". In the special case of DFEs, a notion of soft-output other than that of "reliability" has already been developed in [12]. This notion was developed by replacing the the traditionally used hard limiter in the DFE by another non-linear device that has real values as outputs. This device is picked based on either the MMSE criterion or a minimum BER criterion.

The same idea can be generalized to generate soft-decisions for the DFSE algorithm, and in fact for the Viterbi algorithm. We use the same notation here as developed in Section III.

A. The SD-DFSE Algorithm

1. Choose a desired complexity level by choosing the trellis size $2^\gamma(\leq 2^\mu$ and a decision delay $\delta >> \mu$.
2. At time instant i, associate $u_{i-\gamma-1}, \dots, u_{i-\mu}$ with each state in the reduced trellis. (Same as DFSE)
3. Perform an add-compare-select operation for each state at time $i+1$ using $f(y_i - \hat{y}_{k-\delta})$ as the metric, where f is any continuous function. If f is the L^2 norm, then this metric is identical to the metric in the DFSE algorithm.
4. Retrace your algorithm to time $i-\delta$. Assuming only one path survives, use $h_1(u_{i-\delta} + h_2(y_i - \hat{y}_{k-\delta})$ as the soft decision on $u(i{-}delta)$, where h_1 and h_2 are suitably chosen functions. □

Note that a DFE is when $\delta = 0$, h_1 is the signum function and h_2 is the identity function, while the DFSE is when h_1 is the identity function and h_2 the null function.

B. Special Case: Soft Decision DFE decoders

Soft decision DFEs have been previously constructed as equalizers [12]. Here, we construct soft decision DFEs as decoders by using the analytic expression for encoders introduced in [14]. The output of an encoder can be written in terms of the input as some non-linear function $c_k = h_1(d_k, d_{k-1}, \dots, d_{k-phi})$. In many cases, this enables us to rewrite d_k as $h_2(c_k, d_{k-1}, \dots, d_{k-\phi})$. Thus, if we possess a noisy estimate of c_k, say r_k, then we can estimate d_k as $h_2(r_k, d_{k-1}, \dots, d_{k-\phi})$. This method is further illustrated by the example in the next section.

VI. Simulation Results

As an example of the SO-DFSE algorithm, we consider a system where the transmitter has a rate 1/2 encoder with octal representation (23,35) and a 64×64 block interleaver. The channel has a constant impulse response given by $\mathbf{h} = [0.671, 0.5, 0.387, 0.316, 0.224]$ (identical to the example in [1]) which is known to the receiver. We compare the performance results obtained by using APP modules at both the equalizer and decoder with the performance of a 4-state SOVA based SO-DFSE equalizer and a 4-state APP based SO-DFSE decoder module. We compare their performance with that obtained using hard decision non-iterative equalization and decoding. Figure 3 shows our results.

Next we provide an example for the second approach, i.e. where we utilize soft-decisions instead of reliabilities. We consider a transmitter to be a TCM encoder of memory length 2 with analytic representation given by $f(b_1, b_2, b_3) = b_2 - 2b_1b_3$, an interleaver of size 96, and binary modulation. We consider a constant channel, where the channel combined with the receiver feed-forward filter has coefficients $\mathbf{h} = [1\ 0.05\ 0.6\ \ -0.27\ 0.03\ 0.12\ 0.22]$. We also assume perfect side information at the receiver. We wish to design an iterative processor at this point that has two SDDFE modules as its components. SDDFEs are of two types, the first type use soft-decisions in their feedback, and the second type use hard decisions. We consider three different combinations of these for our simulations:

1. The equalizer-DFE uses hard decision feedback and passes hard decisions to the decoder-DFE, while the encoder uses hard decision feedback but passes soft information to the equalizer.
2. The equalizer and decoder use hard decision feedback, but exchange soft information.
3. The equalizer-DFE uses soft decision feedback and exchanges soft information with the decoder, and the decoder uses hard decision feedback.

The performance of these three schemes is plotted in Figure 4. Their performance is compared with that of a DFE equalizer Viterbi decoder combination. In this figure, we observe that the soft DFE case outperforms all the other cases, and that the gap between the soft-DFE case and the DFE-Viterbi case reduces with increasing SNR.

VII. Conclusions

In this paper, we point out that although the presently known iterative decoding algorithms perform much better than non-iterative ones, they are much more complex. We obtain two different classes of soft-output algorithms, where both classes allow a designer to pick the desired complexity level, at the cost of performance. The first class is termed the SO-DFSE algorithms, whose performance is shown to be in between that of APP/SOVA based algorithms and the non-iterative ones. The second class is termed the SD-DFSE algorithms. These are obtained by generalizing the notion of soft-output from reliabilities to "soft-decisions". The performance of a special case of these, the soft-decision DFE is also presented in this paper.

References

[1] C. Douillard, M. Jezequel, C. Berrou, A. Picart, P. Didier, and A. Glavieux, "Iterative cancellation of intersymbol interference: Turbo equalization," *European Transactions on Telecommunications*, pp. 507–511, September 1995.

[2] J. Hagenauer and P. Hoeher, "A viterbi algorithm with soft-decision outputs and its applications," in *Proceedings of IEEE GLOBECOM*, (Dallas, TX, USA), pp. 1680–1686, Nov. 1989.

[3] G. Bauch and V. Franz, "A comparison od soft-in/soft-out algorithms for "turbo-detection"," *Proceedings of the International Conference on Telecommunications (ICT '98)*, pp. 259–263, June 1998.

[4] S. Benedetto and D. Divsalar, *Short Course on Turbo Codes: Course notes.* UCLA Extension, USA, 1999.

[5] C. Berrou, P. Adde, E. Angui, and S. Faudeil, "A low complexity soft-output Viterbi decoder architecture," *Proceedings of the ICC'93*, pp. 737–740, 1993.

[6] A. Duel and C. Heegard, "Delayed decision-feedback sequence estimation," *IEEE Transactions on Communications*, pp. 428–436, May 1989.

[7] M. V. Eyuboglu and S. U. H. Qureshi, "Reduced-state sequence estimation with set partition and decision feedback," *IEEE Transactions on Communications*, vol. 36, pp. 13–20, January 1988.

[8] J. B. Anderson and E. Offer, "Reduced-state sequence detection with convolutional codes," *IEEE Transactions on Information Theory*, pp. 965–972, May 1994.

[9] M. V. Eyuboglu and S. U. H. Qureshi, "Reduced-state sequence estimation for coded modulation on intersymbol interference channels," *IEEE Journal on Selected Areas in Communications*, pp. 989–995, August 1989.

[10] S. H. Muller, W. H. Gerstacker, and J. B. Huber, "Reduced-state soft-output trellis-equalization incorporating soft-feedback," in *Proceedings of IEEE GLOBECOM*, pp. 95–100, 1996.

[11] S. Ariyavisitakul and Y. Li, "Joint coding and decision feedback equalization for broadband wireless channels," in *Proceedings of IEEE Vehicular Technology Conference*, 1998.

[12] J. Balakrishnan, "Mitigation of error propagation in decision feedback equalization," Masters thesis, Cornell University, August 1999.

[13] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. IT-42, pp. 429–445, March 1996.

[14] P. J. M. Ezio Biglieri, Dariush Divsalar and M. K. Simon, *Introduction to Trellis-Coded Modulation with Applications.* Macmillan Publishing Company, 1991.

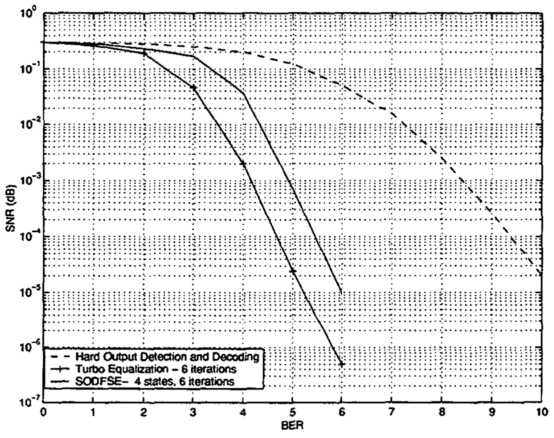

Fig. 3. Performance comparison between hard-decision, BCJR based APP and the SO-DFSE algorithms

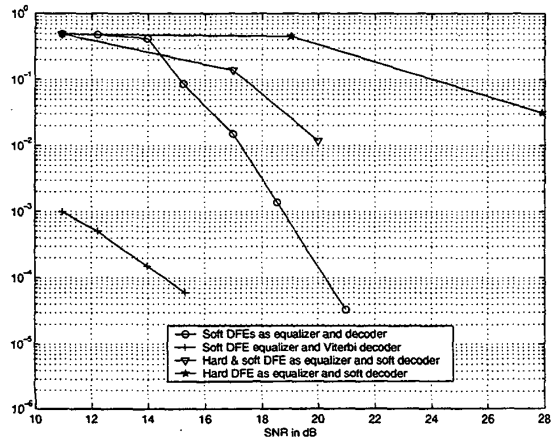

Fig. 4. Performance comparison between different classes of SD-DFSE algorithms