Abstract—Recent progress in wireless receiver design has been towards iterative processing, where channel estimation and decoding is considered a joint optimization problem. Sparse channel estimation is another recent advancement, which exploits the inherent structure of those wireless channels that are composed of a small number of multipath components. In this work we design iterative receivers which incorporate sparse channel estimation. State-of-the-art sparse channel estimators simplify the estimation problem to be a finite basis selection problem by restricting the multipath delays to the discrete domain (i.e. to a grid). Our main contribution is a receiver without such a restriction; the delays are estimated directly as continuous values. As a result, our receiver does not suffer from the leakage effect which destroys sparsity when the delays are restricted to the discrete domain. We discuss certain connections between continuous and discrete domain sparse estimation methods. Our receivers outperform state-of-the-art sparse channel estimation iterative receivers in terms of bit error rate.

Index Terms—Iterative receivers, message-passing algorithms, sparse channel estimation, sparse Bayesian learning, off-the-grid.

I. INTRODUCTION

One of the major challenges in designing receivers for wireless systems is mitigation of multipath effects through channel estimation and equalization. To facilitate channel estimation, current systems embed pilot signals into the transmitted signal. An example is orthogonal frequency-division multiplexing (OFDM) systems where a number of subcarriers are assigned to transmit pilot symbols. The number of pilots is chosen to optimize throughput as a trade-off between the amount of bandwidth and power sacrificed as pilots and fidelity of the channel estimate. In this work we seek to improve upon this trade-off through a unified receiver design which incorporates two main ideas:  

1. joint channel estimation and decoding and  
2. sparse channel estimation.

A. Joint Channel Estimation and Decoding

Classical receiver designs employ a functional splitting of the reception into independent subtasks, as illustrated in Fig. 1. Such a structure is suboptimal, since the information learned from the received signal in any of the subtasks is only utilized in subsequent subtasks. To remedy this sub optimality, a feedback loop can be introduced between the functional blocks in the receiver. This is known as the turbo principle [1] due to the resemblance to iterative decoding of turbo codes.

Application of the turbo principle has lead to many iterative receiver designs, e.g. [2]–[6]. Common to these works is that each of the subtasks are designed independently. Typically each subtask optimizes a likelihood (ML), a-posteriori probability (MAP) or mean squared error (MMSE) objective function, from which soft-out information is heuristically found. The work [7] introduced receiver design from the perspective of inference in a factor graph. This allows receivers to be designed such that the subtasks work together to optimize a joint objective function, which for example could be the MAP estimate of the information bits. We refer to such receivers as joint channel estimation and decoding (JCED) receivers. Due to tractability and computational constraints, approximate inference methods must be employed for JCED receiver design. Examples of such approximated designs are using expectation propagation in [8], belief propagation (BP) with approximated messages in [9], a merged BP and mean-field (MF) framework in [10], [11], relaxed BP in [12] and generalized approximate message-passing (GAMP) in [13].

B. Sparse Channel Estimation

The (wireless) channel impulse response (CIR) is often described with a tapped delay line model:

\[
g(\tau) = \sum_{l=1}^{L} \alpha_l \delta(\tau - \tau_l),
\]

where \(L\) is the number of multipath components. The \(l\)th channel tap coefficient (weight) is denoted as \(\alpha_l \in \mathbb{C}\) with corresponding tap (relative) delay \(\tau_l \in \mathbb{R}\). In propagation scenarios where the number of multipath components \(L\) is relatively small, the model (1) is parsimonious and it is advantageous to perform channel estimation by estimating the parameters of this model, i.e. estimating \(L\), \(\alpha_l\) and \(\tau_l\) for \(l = 1, \ldots, L\). This idea is referred to as sparse channel estimation [14]–[17].

Sparse channel estimation is closely related to the concept of compressed sensing [18], where knowledge of sparsity is
exploited to allow reconstruction of a signal from a small number of measurements. In sparse channel estimation, we exploit knowledge of sparsity to allow CIR estimation from observations at only a few pilot symbols.

Most of the current literature on both compressed sensing and sparse channel estimation employs a grid-based approximation of the model (1), where the tap delays \( \tau_l, l = 1, \ldots, L \), are restricted to a grid of possible values. Often, a sample-spaced grid is assumed, i.e. the spacing between the grid points is given by the inverse of the system bandwidth. The grid-based approximation results in a leakage effect [16, 19], which causes the CIR to only be approximately sparse in the grid-based representation [20]. From a compressed sensing point of view the effect of the grid-based approximation can be understood as a basis mismatch [21]. A recent development in compressed sensing is a move away from the grid-based approximation, see for example the works [22]–[26].

C. Prior Art and Contributions

In this work, we build on recent advances within both JCED receivers and sparse channel estimation, to derive receivers which merge these two techniques. In particular, for approximate inference we use the merged belief propagation and mean-field (BP-MF) framework [27] which allows the receiver to be derived within a completely rigorous framework, as opposed to a design resorting to a heuristically chosen connection between receiver subtasks.

Several prior works incorporate sparse channel estimation in an iterative receiver design. In [20] a joint sparse channel estimation and detection scheme is proposed. Channel decoding is not considered in the joint processing and an expectation-maximization (EM) algorithm is used for channel inference. The delay values are restricted to the sample-spaced grid, even though a finer grid could in principle be used in the receiver. This work is very similar in approach to our work, since the channel estimation part of [20] can be derived in a mean-field framework (as shown in [28]), similar to the channel estimation part of our algorithm. However, we extend the channel estimation to a more general prior model and we do not assume all tap delays to be restricted to a grid.

The works [12], [13] consider JCED receiver design for OFDM systems via GAMP and relaxed BP. Due to restrictions imposed by the used inference algorithms, the tap delays are restricted to the sample-spaced grid (a finer grid cannot be used, since the dictionary must have orthogonal columns). In [12] the investigated CIRs are generated to only have delay values on the grid, thus avoiding any leakage effects by introducing an unrealistic channel model. In [13] continuous-valued delays are assumed and it is shown that the CIR is not sparse on the sample-spaced grid. A channel tap coefficient prior which exploits the fact that most taps have small coefficients is introduced, to exploit some structure in the CIR.

In this paper, we present three different variants of the BP-MF based JCED receiver: a) a receiver which estimates CIRs with continuous-valued tap delays; b) a variant of this algorithm with reduced computational requirements; and c) a variant which restricts the tap delays to a (arbitrarily fine) grid. We also discuss that under certain circumstances, continuous delay tap estimation can be considered an instance of grid-based delay tap estimation.

Our receivers do not require statistical information about the channel (except for a single parameter, which controls the CIR sparsity level). This is in contrast to for example MMSE channel estimators (which require second-order statistics of the channel transfer function) [5], [29] and the GAMP receiver [12] (which relies on the second-order statistics of the CIR at all tap delays on the sample-spaced grid).

D. Notation and Contents

In the paper we denote column vectors as \( \mathbf{a} \) and matrices as \( \mathbf{A} \). Conjugate (Hermitian) transpose is denoted as \( \mathbf{a}^\mathsf{H} \) and non-conjugate transposition as \( \mathbf{a}^\mathsf{T} \). The scalar \( a_i \) or \( [\mathbf{a}]_i \) gives the \( i \)th entry of vector \( \mathbf{a} \), while \( \mathbf{a}_S \) gives a vector containing the entries in \( \mathbf{a} \) at the indices in the integer set \( S \). The notation \( [\mathbf{A}]_{i,k} \) gives the \( (i,k) \)th element of matrix \( \mathbf{A} \). We denote the vector \( \mathbf{a} \) with the \( i \)th entry removed as \( \mathbf{a}_i \) and use a similar notation for matrices with columns and/or rows removed (e.g. \( [\mathbf{A}]_{i,k} \) for the \( i \)th row with \( k \)th entry removed). The notation diag(\( \mathbf{a} \)) denotes a matrix with the entries of \( \mathbf{a} \) on the diagonal and zeros elsewhere. The indicator function \( 1_{[\cdot]} \) gives 1 when the condition in the brackets is fulfilled and 0 otherwise. The probability density functions of the (vector) complex normal and gamma distributions are defined as follows:

\[
\text{CN}(\mathbf{x}; \mu, \Sigma) \triangleq \pi^{\dim(\mathbf{x})/2} |\Sigma|^{-1} \exp\left(-\frac{1}{2} (\mathbf{x} - \mu)^\mathsf{H} \Sigma^{-1} (\mathbf{x} - \mu)\right)
\]

\[
\text{Ga}(\mathbf{x}; a, b) \triangleq \frac{b^a}{\Gamma(a)} x^{a-1} \exp(-bx),
\]

where \( \Gamma(\cdot) \) is the gamma function.

The paper is structured as follows: In Section II we specify our observation model and sparsity-inducing prior model. In Section III our approach to approximate Bayesian inference is discussed. The inference algorithm is derived in detail in Section IV. In Section V we employ further approximation to reduce the computational requirements of the inference scheme. Section VI presents a variant of the receiver which employs the grid-based approximation. Finally, our numerical evaluation of the algorithms is presented in Section VII and conclusions are given in Section VIII.

II. Modelling

We consider data transmission in a single-input single-output OFDM system. Since we do not exploit any structure between consecutive OFDM symbols, we model each OFDM symbol to be identical and independent distributed (i.i.d.) samples of a random process. The OFDM system has \( P \) pilot subcarriers and \( D \) data subcarriers, such that the total number of subcarriers is \( N = P + D \). The sets \( \mathcal{P} \) and \( \mathcal{D} \) give the indices of the pilot and data subcarriers, respectively. It follows that \( \mathcal{D} \cup \mathcal{P} = \{1, \ldots, N\} \) and \( \mathcal{D} \cap \mathcal{P} = \emptyset \).

A. OFDM System

The \( K \) (equi-probable) information bits to be transmitted are stacked in vector \( \mathbf{u} \in \{0, 1\}^K \). These bits are coded by a
rate $R$ coder and interleaved to get the $K/R$ coded bits $c = C(u)$. The coding and interleaving function $C : \{0,1\}^K \rightarrow \{0,1\}^{K/R}$ can use any coder, e.g. a turbo [30], low-density parity check (LDPC) [31] or convolutional code. We split the coded bits $c$ into sub vectors $c_i \in \{0,1\}^Q$, $i \in D$, such that $c_i$ contains the $Q$ bits which are mapped to the $i$th subcarrier. The complex symbols $x_i = M(c_i)$, $i \in D$, are obtained via the $2^Q$-ary mapping $M : \{0,1\}^Q \rightarrow \mathbb{A}_d \subset \mathbb{C}$, where $\mathbb{A}_d$ is the range of $M$, i.e. the data (symbol) alphabet. The pilots are selected in the pilot alphabet $\mathbb{A}_p \subset \mathbb{C}$. In OFDM, $\mathbb{A}_d$ is typically a $2^Q$-ary quadrature amplitude modulation (QAM) alphabet and $\mathbb{A}_p$ a quadrature phase shift keying (QPSK) alphabet. The pilot and data symbols are collected in the vector $x$, such that $x_D$ contains the data symbols and $x_P$ contains the pilot symbols.

The channel impulse response is assumed fixed during the OFDM symbol period (block fading channel assumption) and we further assume that the delay spread of the channel is smaller than the cyclic prefix length $T_{CP}$. Neglecting interference and any imperfections other than noise in the transmitter and receiver, we have that the orthogonality of the subcarriers is preserved. The received symbols $y$ are therefore given by

$$y = Xh + w,$$

(2)

where $X = \text{diag}(x)$, the vector $h$ contains the channel frequency response at the subcarrier frequencies and $w$ is a white Gaussian noise vector with component variance $\lambda^{-1}$.

### B. Channel Model

Since the CIR model (1) has delay spread not larger than the cyclic prefix time $T_{CP}$ and we assume the CIR to be constant during one OFDM symbol, we can obtain the channel frequency response $h$ by evaluating the Fourier transform of the CIR at the subcarrier frequencies. Using (1) we arrive at

$$h = \Psi(\tau)\alpha,$$

(3)

where the matrix $\Psi(\tau) \in \mathbb{C}^{N \times L}$ has $(n,l)$th entry $\exp(-j2\pi \Delta f n \tau_l)$ with $\Delta f$ denoting the subcarrier spacing used in the OFDM system. We have stacked the channel tap coefficients and delays into vectors $\alpha = [\alpha_1, \ldots, \alpha_L]^T \in \mathbb{C}^L$ and $\tau = [\tau_1, \ldots, \tau_L]^T \in [0, T_{CP}]^L$.

The validity of the channel model (1) can be discussed; the model is certainly not valid in all propagation environments. An important point here is that we are not expecting the model to fully capture all energy in the CIR. Conversely, we merely state that the CIR is composed of a small number of dominant components which we model and neglect the remaining energy. It is thus expected that sparse channel estimation can improve receiver performance in those propagation environments, where the majority of the energy in the CIR is contained in a small number of dominant components. As demonstrated in Sec. VII, our algorithms perform well even if the number of multipath components $L$ is large ($L \gg N \Delta f T_{CP}$) and the individual components in (1) can no longer be resolved.

Even if the CIR can be modelled as a small number of distinct components (pulses) in the time domain, it is not given that the Dirac delta is a good model for the shape of that pulse. The CIR is observed through a limited number of observations in the frequency domain, i.e. through a low-pass filter. The Dirac delta is only required to model the true shape of the pulse within the frequency-band dictated by that low-pass filter (i.e. the system bandwidth).

In our treatment, we have not included the effects of any anti-aliasing filters (also described as transmit and receive filters in some works). This is because zero-padding of the Fourier transform in a typical OFDM transceiver works together with the anti-aliasing filter to create a combined response which is practically ideal (as a sinc filter). We note that if this combined response is non-ideal, it can trivially be included into the dictionary matrix $\Psi(\tau)$ to allow a sparse representation of the CIR [32]. The iterative receivers with sparse channel estimation in [13] and [20] include root raised cosine anti-aliasing filters into their signal model. In fact, these filters reduce the leakage effect which prohibits a sparse representation to be found when the tap delays are restricted to a sample-spaced grid. It is not detailed in these works how their signal model relates to typical OFDM systems, which do not use root raised cosine filters for anti-aliasing.

### C. CIR Prior Model for Estimation

We now move away from modelling in its traditional meaning. Instead of seeking a model which describes a system for a given purpose, we wish to construct a prior density of the coefficients $\alpha$ which promotes sparsity on the channel impulse response. The sparsity-inducing prior is such that a large prior probability is assigned to the event $\alpha_l = 0$. As we will see, this structure allows us to derive an algorithm which inherently estimates the number of channel taps $L$.

The sparsity inducing prior we will use, was originally used for compressed sensing reconstruction via sparse Bayesian learning [23], [24], [28], [33]–[35]. The channel tap coefficients are modelled as i.i.d. random variables via a two-layer hierarchical model. The first layer is a complex normal density

$$p(\alpha_l | \gamma_l) = \text{CN}(\alpha_l; 0, \gamma_l^{-1}).$$

(4)

The hyper-parameter $\gamma_l$ is assigned a gamma hyper-prior

$$p(\gamma_l) = \text{Ga}(\gamma_l; \varepsilon, \eta),$$

(5)

where $\varepsilon$ and $\eta$ are parameters of the model.

Even though we are using tools from approximate Bayesian inference to derive our algorithms in the following, it is important to note that our assigned prior on $\alpha_l$ is fundamentally different from what is used in a traditional Bayesian sense. The hierarchical prior on $\alpha_l$ is not a believed prior density on the coefficient value; it is instead selected because we know from empirical evidence that it promotes sparse solutions while also giving a tractable model. As such, the hierarchical prior is a tool employed to perform model order selection. For a discussion as to why our prior specification leads to sparse solutions, see [36].

We note that the delays in $\tau$ can take any value in $[0, T_{CP}]$. Since no further information is available we can model the

1. The CIR representation on the sample-spaced grid, is obtained by convolving the sparse CIR with the anti-aliasing filter response. The raised cosine filter response has much smaller sidelobes than that of the sinc filter, thereby approximately preserving sparsity in the representation.
delays as i.i.d. uniform random variables, i.e.
\[ p(\tau_i) = \text{unif}(\tau_i; 0, T_{CP}). \] (6)

Finally, the noise precision prior is modelled as a gamma pdf, as this is the conjugate prior for the precision of Gaussian observations:
\[ p(\lambda) = \text{Ga}(\lambda; a, b). \] (7)

### III. Inference Method

In this work, we wish to estimate the data bits as the maximum a-posterior probability (MAP) value
\[ \hat{u}_k = \arg \max_{u_k \in \{0,1\}} p(u_k | y), \quad k \in K, \] (8)

where \( K = \{1, \ldots, K\} \) is the information bit index set. The pdf \( p(u_k | y) \propto p(u_k, y) \) can ideally be found by marginalizing all variables but \( u_k \) in the joint pdf\(^2\)
\[ p(y, \alpha, \gamma, \tau, x_D, c, u) = p(y|x_D, \alpha, \tau, \lambda)p(\lambda) \] (9)
\[ \prod_{l \in L} p(\alpha_l | y_l)p(\gamma_l)p(\tau_l) \prod_{i \in D} p(x_i|c_i)p(c|u) \prod_{k \in K} p(u_k), \]

where \( L = \{1, \ldots, L\} \) is the channel tap index set. However, calculating the marginals of \( u_k, k \in K \), is intractable and we resort to approximate Bayesian inference.

#### A. Factor Graph Representation

As our approximate inference scheme, we employ the BP-MF framework [27], which can be derived as message-passing in a factor graph. The product (9) is represented by the factor graph in Fig. 2, with factors defined as follows:
\[ f_{\alpha_l}(\alpha, \gamma_l) \triangleq p(\alpha_l | \gamma_l) = \text{CN}(\alpha_l; 0, \gamma_l^{-1}), \quad l \in L, \]
\[ f_{\gamma_l}(\gamma_l) \triangleq p(\gamma_l) = \text{Ga}(\gamma_l; \varepsilon, \eta), \quad l \in L, \]
\[ f_{\tau_l}(\tau_l) \triangleq p(\tau_l) = \text{unif}(\tau_l; 0, T_{CP}), \quad l \in L, \]
\[ f_{\lambda}(\lambda) \triangleq p(\lambda) = \text{Ga}(\lambda; a, b), \]
\[ f_{j_p}(\alpha, \tau, \lambda) \triangleq p(y_j | \alpha, \tau, \lambda) \]
\[ = \text{CN}(y_j; x_j \{\Psi(\tau) \alpha\}_j, \lambda^{-1}), \quad j \in P, \]
\[ f_{D_i}(x_i, \alpha, \tau, \lambda) \triangleq p(y_i | x_i, \alpha, \tau, \lambda) \]
\[ = \text{CN}(y_i; x_i \{\Psi(\tau) \alpha\}_i, \lambda^{-1}), \quad i \in D, \]
\[ f_{M_i}(x_i, c_i) \triangleq p(x_i | c_i) = 1_{[x_i = M(c_i)]}, \quad i \in D, \]
\[ f_{c}(c, u) \triangleq p(c | u) = 1_{[c = c(u)]}, \]
\[ f_{u_k}(u_k) \triangleq p(u_k) = 0.5 \mathbb{1}_{[u_k \in \{0, 1\}]}, \quad k \in K. \]

Taking the product of all the factors listed above gives the joint pdf (9). We note that our estimation scheme also estimates the number of multipath components \( L \), even though this variable is not explicitly represented in the factor graph.

#### B. Decoupled Channel Estimation and Decoding

The decoding of many popular codes can be described as loopy belief propagation (BP) [37] in a factor graph [38–40]. In our factor graph representation (Fig. 2), this would correspond to implementing BP in the subgraph which contain the variable nodes \{\( c_l \), \ldots, \( c_q \)\} \( i \in D \), \( \{u_k\} k \in K \) and all neighbouring factor nodes.

As an example, we note that for convolutional codes this approach leads to the BCJR algorithm [41]. Performing BP in this subgraph leads to tractable algorithms, because all variables are discrete-valued. It is intractable to apply BP to the complete factor graph in Fig. 2 because intractable integrals arise in deriving the BP update formulas.

The subgraph containing the factors \( \{f_{\alpha_k}\} k \in P, \quad \{f_{\gamma_k}, f_{\tau_k}\} k \in L \) and neighbouring variables, represents a model which is based on Gaussian conditional pdfs with conjugate priors. In such models variational inference utilizing the mean-field approximation [42] is known to give tractable algorithms. In this framework, the product of factors to be marginalized is approximated as a product of independent factors:
\[ \prod_{l \in L} f_{\gamma_l}(\gamma_l)f_{\tau_l}(\tau_l) \prod_{j \in P} f_{j_p}(\alpha, \tau, \lambda)f_{\lambda}(\lambda) \approx q(\alpha) \prod_{l \in L} q(\gamma_l)q(\tau_l)q(\lambda). \] (10)

We abuse notation and let \( q(\cdot) \) denote the approximating belief for each of the different variables. In the following we refer to a MF algorithm, as one which employ an approximation as (10) and proceeds by sequentially estimating the approximating factors \( q(\cdot) \) by minimizing the Kullback-Liebler divergence of the true product (left-hand side of (10)) from the approximated product (right-hand side of (10)). In general such an approach leads to the algorithm in [42]. Examples of algorithms for parameter estimation in the model (10), can be found in the literature based on both MF (see [23], [24]) and the so-called Type-II evidence procedure (see [22]).

#### C. Joint Channel Estimation and Decoding

We now wish to devise an algorithm which jointly performs channel estimation, demodulation and decoding via inference in the factor graph in Fig. 2. For decoding we wish to use BP as it leads to well-known algorithms which are known to be tractable and show good performance for many popular codes. For channel estimation we wish to use MF, as it leads to well-performing and tractable algorithms (as showcased by [23]). To merge these two approaches, we follow the framework in [27], which derives such a merged algorithm by writing a joint cost function based on the region-based free energy approximation given in [43]. In this framework, the factor graph is split into a MF subgraph and a BP subgraph and it is well-defined how messages are passed between the two parts of the factor graph. In Fig. 2 we have given the splitting into the two subgraphs by a dotted line, such that the splitting is done at the variable nodes \( \{x_i\} i \in D \).

For tractability of our algorithm we restrict the estimates of variables \( \{\tau_l\} \in L \) to be point estimates, i.e. we restrict their beliefs to be Dirac deltas
\[ q(\tau_l) = \delta(\tau_l - \hat{\tau}_l). \] (11)
The point estimates \( \{ \hat{\tau}_l \}_{l \in \mathcal{L}} \) are found by maximizing the unrestricted belief. In a pure MF context such a restriction corresponds to EM estimation of \( \{ \tau_l \}_{l \in \mathcal{L}} \) with all other variables treated as latent variables [27].

In the following two sections, we derive two iterative receiver algorithms based on the above ideas. The first one is derived from the factor graph in Fig. 2, while we in the second one split the variable node \( \alpha \) into \( \mathcal{L} \) nodes \( \{ \alpha_l \}_{l \in \mathcal{L}} \). We are in other words assuming the following form of \( q(\alpha) \):

\[
q(\alpha) = \prod_{l \in \mathcal{L}} q(\alpha_l). \tag{12}
\]

The MF subgraph of the factor graph with this modification is shown in Fig. 3. We refer to this simplification as the configuration with disjoint tap coefficients. As we will see, this factorization leads to update expressions which have a very intuitive interpretation and to a reduction of the computational requirements of the resulting algorithm.

IV. SPARSE BP-MF RECEIVER ALGORITHM

We now proceed by applying the MF-BP algorithm given by Eq. (21)–(22) in [27] within the factor graph of Fig. 2. In [11] a similar BP-MF receiver is derived which does not exploit channel sparsity. In the following we use the notation \( \langle \cdot \rangle_\alpha \) to denote expectation with respect to the belief density \( q(\alpha) \). We follow the convention of [27] in naming the messages.

A. Belief Updates for Channel Estimation

We start by finding belief update expressions for all variable nodes in the MF subgraph. To illustrate the procedure we write the calculations which go into finding the belief of \( \alpha \). First, the messages passed to the node \( \alpha \) are found as

\[
m_{f_{\alpha_l} \rightarrow \alpha}(\alpha) \propto \exp \left\{ \langle \log f_{\alpha_l}(\alpha_l, \gamma_l) \rangle_{\gamma_l} \right\}
\]

\[
\propto \exp \left\{ -|\alpha_l|^2 \langle \gamma_l \rangle_{\gamma_l} \right\}
\]

\[
m_{f_{D_l} \rightarrow \alpha}(\alpha) \propto \exp \left\{ \langle \log f_{D_l}(x_l, \alpha_l, \tau, \lambda) \rangle_{\tau, \lambda} \right\}
\]

\[
\propto \exp \left\{ -\langle \lambda \rangle \langle y_l - x_l \rangle_{\Psi(\tau)\alpha_l}^2 \right\}
\]

\[
m_{f_{P_l} \rightarrow \alpha}(\alpha) \propto \exp \left\{ -\langle \lambda \rangle \langle y_l - x_l \rangle_{\Psi(\tau)\alpha_l}^2 \right\},
\]

which holds for all \( l \in \mathcal{L}, i \in \mathcal{D} \) and \( j \in \mathcal{P} \). Taking the product of all messages going to the node \( \alpha \) gives its belief

\[
q(\alpha) = \prod_{l \in \mathcal{L}} m_{f_{\alpha_l} \rightarrow \alpha}(\alpha) \prod_{i \in \mathcal{D}} m_{f_{D_l} \rightarrow \alpha}(\alpha) \prod_{j \in \mathcal{P}} m_{f_{P_l} \rightarrow \alpha}(\alpha)
\]

\[
q(\alpha) = \mathcal{CN}(\alpha; \hat{\mu}, \hat{\Sigma}), \tag{13}
\]

with

\[
\hat{\Sigma} = \left( \hat{\lambda} \Psi^H(\hat{\tau}) \langle X^H X \rangle_{\tau} \Psi(\hat{\tau}) + \text{diag}(\hat{\gamma}) \right)^{-1}
\]

\[
\hat{\mu} = \hat{\lambda} \hat{\Sigma} \Psi^H(\hat{\tau}) \langle X \rangle_{\tau} y.
\]

We have used the fact that the belief of \( \gamma_l \) is a Dirac delta to write \( \langle \Psi(\tau) \rangle_{\tau} = \Psi(\hat{\tau}) \) in accordance with (11). We have further defined \( \hat{\lambda} \triangleq \langle \lambda \rangle_{\lambda} \) and \( \hat{\gamma}_l \triangleq \langle \gamma_l \rangle_{\gamma_l}, l \in \mathcal{L} \).

To obtain the beliefs of \( \gamma_l, l \in \mathcal{L} \), and \( \lambda \) we carry out similar steps and get

\[
q(\gamma_l) = \mathcal{Ga}(\gamma_l; \hat{\epsilon} + 1, \eta + \langle |\alpha_l|^2 \rangle_{\alpha_l}),
\]

\[
q(\lambda) = \mathcal{Ga}(\lambda; a + N, b + \langle |y - X\Psi(\hat{\tau})\alpha_l|^2 \rangle_{\alpha_l, \tau, \lambda}).
\]

It turns out that the beliefs of all other variables only depend on the mean values

\[
\hat{\gamma}_l = \frac{\hat{\epsilon} + 1}{\eta + |\hat{\mu}_l|^2 + \langle \Sigma \rangle_{\lambda, l}}
\]

\[
\hat{\lambda} = \frac{a + N}{b + u - 2 \text{Re} \{ y^H \langle X \rangle_{\tau} \Psi(\hat{\tau}) \hat{\mu} \}},
\]

with \( u = ||y||^2 + \text{tr} \{ \langle X^H X \rangle_{\tau} \Psi(\hat{\tau}) \hat{\mu} \hat{\mu}^H + \Sigma \Psi^H(\hat{\tau}) \} \).

We continue with the estimation of the delays \( \{ \tau_l \}_{l \in \mathcal{L}} \). In correspondence with (11), these are given as the point estimates

\[
\hat{\tau}_l \triangleq \arg \max_{\tau_l} \log \tilde{q}(\tau_l).
\]

The function \( \tilde{q}(\tau_l) \) is the unrestricted belief of \( \tau_l \), which is
found by taking the product of all messages going into variable node $\tau_i$. Doing so we find

$$\log \hat{q}(\tau_i) \propto \log \langle |y - X\Psi(\tau)\alpha||^2 \rangle_{\alpha, x_\tau, \tau_i} + \text{const.}$$

Re $\{\Psi(x)\tau_i\}$ + const. \quad 0 \leq \tau_i \leq T_{CP} \quad (20)$$

where

$$t = \langle \alpha_i^\tau \rangle_{\alpha} \langle X^H x_{\tau} \rangle - \langle X^H X \rangle_{x_{\tau}} \Psi(\tau) \langle \alpha \rangle.$$ 

Note that $\langle \alpha_i^\tau \rangle_{\alpha} = \mu_i^\tau$ and $\langle \alpha_i^\tau \rangle_{\alpha} = \mu_i^\tau + \langle \Sigma \rangle_{\tau_i}$. The vector $\psi(\tau)$ gives the column of $\Psi(\tau)$ corresponding to $\tau_i$, i.e. the $j$th column. Due to the prior $p(\tau_i)$, the function $\hat{q}(\tau)$ diverges to $-\infty$ for $\tau_i < 0$ or $\tau_i > T_{CP}$.

There is no analytical expression for the maximizer of $\hat{q}(\tau)$. We instead follow the approach of [22] and employ Newton’s method to update $\hat{q}_i$ as

$$\hat{q}_i^{[t+1]} = \hat{q}_i^{[t]} - \nabla \log \hat{q}(\tau_i) / \nabla^2 \log \hat{q}(\tau_i)^2,$$

where $\nabla \log \hat{q}(\tau)$ and $\nabla^2 \log \hat{q}(\tau)$ are the first and second order derivatives of $\log \hat{q}(\tau)$, respectively. Superscript $[t]$ denotes iteration number.

B. Message-Passing for Decoding

In the previous subsection we derived the belief functions $q(\cdot)$ of the variables whose factor neighbours are in the MF subgraph only. In the BP subgraph (i.e. detection, demapping, decoding and deinterleaving,) we instead focus on calculating the messages which are passed in the factor graph. All messages passed to the right of factors $f_{M_{i}}\in D$ are discrete-valued and it is therefore tractable to calculate these messages directly by the sum-product algorithm. This has been studied thoroughly in the literature [38], [39], for many different coding schemes. Due to space constraints, we will not discuss this part of the algorithm in detail.

The only messages in the BP subgraph, which cannot be calculated by direct evaluation of the sum-product algorithm, are

$$n_{x_i \rightarrow f_{M_i}}(x_i) = m_{f_{M_i} \rightarrow x_i}(x_i)$$

$$= \nabla \log \hat{q}(\tau_i) / \nabla^2 \log \hat{q}(\tau_i)^2,$$

where $\nabla \log \hat{q}(\tau)$ and $\nabla^2 \log \hat{q}(\tau)$ are the first and second order derivatives of $\log \hat{q}(\tau)$, respectively. Superscript $[t]$ denotes iteration number.

When BP messages have been passed in the BP subgraph, the beliefs of the data symbols $\{x_i\}_{i \in D}$ are calculated from

$$q(x_i) \propto m_{f_{M_i} \rightarrow x_i}(x_i)m_{f_{M_j} \rightarrow x_i}(x_i).$$

The beliefs $q(x_i)$ are discrete-valued because the messages $m_{f_{M_i} \rightarrow x_i}(x_i)$ are only non-zero for $x_i \in \mathbb{A}_d$, i.e. at the points of the data symbol alphabet. By straightforward evaluation of finite sums involving $q(x_i)$ we can obtain $\langle x_{\tau} \rangle$ which are used in the belief updates in the MF subgraph.

C. Improving Convergence Speed

We now follow the approach proposed in [35] to calculate in closed form the beliefs $q(x_i)$ and $q(\alpha)$ which results after infinite iteration between the updates of these two beliefs. This significantly improves convergence speed and, as we will see, allows us to construct an incremental algorithm which computes a solution by iteratively adding components into the model.

We start by using the block matrix inversion theorem [44] on $\Sigma$ to rewrite $\langle |\alpha|^2 \rangle_{\alpha} = \langle |\Sigma \rangle_{\tau_i} + |\mu|^2$ as follows (see [35] for details):

$$\langle |\alpha|^2 \rangle_{\alpha} = (s + \hat{\gamma}_i)^{-1} + |q|^2(s + \hat{\gamma}_i) - 2$$

We now consider the value $\hat{\gamma}_i^{[\infty]}$ which results from applying the recursive iteration $\hat{\gamma}_i^{[\infty]} = \hat{\gamma}_i^{[0]}$ ad-infinitum. To simplify things, we restrict our analysis to the case $\epsilon > 0$ and $\eta = 0$. By solving the fixed-point equation $\hat{\gamma}_i^{[\infty]} = \hat{\gamma}_i^{[\infty]}$ we find that several fixed points exist. To analyse the stability of the fixed points we follow an analysis similar to that in [35]. It turns out that the recursive iteration converges or diverges according to

$$\hat{\gamma}_i^{[\infty]} = F(\hat{\gamma}_i^{[\infty]}) \triangleq \begin{cases} \left( \frac{s + \hat{\gamma}_i^{[0]}}{\eta} \right)^{2} & \text{if } \hat{\gamma}_i^{[0]} > 0 + 2 \sqrt[s + \hat{\gamma}_i^{[0]}]{s + \hat{\gamma}_i^{[0]}}, \\ \infty & \text{otherwise}, \end{cases}$$

where $\Delta \triangleq (2 \epsilon + 1)s + |q|^2 > 2 \epsilon + 2 \sqrt[s + \hat{\gamma}_i^{[0]}]{s + \hat{\gamma}_i^{[0]}}, \hat{\gamma}_i^{[0]}$ gives the initial value of $\hat{\gamma}_i$. When $\hat{\gamma}_i^{[\infty]}$ diverges to $\infty$ for some index $i$, it corresponds to removing the $i$th component from the model since the corresponding $\{\mu_i\}$ converges to 0.

In the update (29) the role of $\epsilon$ becomes apparent. Larger values of $\epsilon$ make the conditions in the logical expression in (29) more strict and larger values of $\epsilon$ are therefore more sparsity-inducing. We can thus control the sparsity of the CIR estimate by selecting $\epsilon$. We note that the corresponding result for $\epsilon = \eta = 0$ is given in [28], [35] and that our choice of

Note that in [35] the problem is parameterized in $\omega_i = \eta_i^{-1}$ and $\omega_i^2 = |\eta|^2\eta_i^{-2}$ and a purely real signal model is considered. It is straightforward to generalize the analysis to the complex signal model that we consider here. Also note that [35] considers the case $\epsilon = \eta = 0$, while we instead analyse the case $\epsilon > 0, \eta = 0$. 

\[\hat{\gamma}_i^{[\infty]} = \hat{\gamma}_i^{[0]} \]
Algorithm 1: Sparse BP-MF receiver.

Input: Observations $y$, pilot indices $P$ and pilot symbols $x_P$.

Output: Belief functions of data bits $\{q(u_i)\}_{i \in K}$.

Parameters: Prior parameters $\varepsilon$, $\alpha$ and $b$ ($\eta = 0$ assumed).

1. $\hat{\tau} \leftarrow$ Vector with values from equispaced grid on $[-1/2, T_{\text{up}}]$.
2. $\tau, \gamma \leftarrow$ Empty vectors.
3. Initialize $\lambda$ and the messages $n_{x_i \rightarrow f_{\text{update}}}(x_i)$ for all $i \in D$.
4. while Outer stopping criterion not met do
   5. Compute one iteration of all messages in the BP subgraph.
   6. Update the beliefs $q(x_i)$ from (23).
   7. while Inner stopping criterion not met do
      8. $\Sigma, \hat{\mu} \leftarrow$ Update from (14), (15).
      9. Find one candidate component via grid search:
         $r \leftarrow (X^H x, y - (X^H X) x_P) \psi(\hat{\mu})\Sigma \tau_{\text{new}} \leftarrow \arg \max_{\tau \in \mathbb{R}} |\psi(\hat{\mu})\tau|^2$.
      10. Append $\tau_{\text{new}}$ and $\gamma_{\text{new}}$ to the vectors $(\hat{\tau}, \hat{\gamma})$.
      11. $l \leftarrow |\hat{\tau}|_0$ (i.e. $l$ is index of $\tau_{\text{new}}$ and $\gamma_{\text{new}}$).
      12. for $n = 1, \ldots, 25$ or until convergence of $\hat{\tau}$ do
         13. $\hat{\tau} \leftarrow$ Update from (14), (15).
         14. $\hat{\gamma} \leftarrow$ Update from (21).
      15. $\hat{\gamma}_l \leftarrow$ Update from (29).
      16. if $\hat{\gamma}_l = \infty$ then
          17. Remove $l$th entry from the vectors $(\hat{\tau}, \hat{\gamma})$.
      18. end
      19. end
      20. Update all components currently included in model:
         for $l = 1, \ldots, |\hat{\tau}|_0$ do
            21. for $n = 1, \ldots, 25$ or until convergence of $\hat{\tau}_l$ do
               22. $\hat{\tau}_l \leftarrow$ Update from (14), (15).
               23. $\hat{\gamma}_l \leftarrow$ Update from (21).
            24. $\hat{\gamma}_l \leftarrow$ Update from (29).
            25. if $\hat{\gamma}_l = \infty$ then
                26. Remove $l$th entry from the vectors $(\hat{\tau}, \hat{\gamma})$.
            27. end
         28. $\hat{\tau}_l \leftarrow$ Update from (14), (15).
      29. $\hat{\gamma} \leftarrow$ Update from (18).
   30. end
   31. Update the messages $n_{x_i \rightarrow f_{\text{update}}}(x_i)$ from (22).
end

$\varepsilon > 0$ gives more sparsity-inducing algorithms.

D. An Incremental Algorithm

Algorithm 1 combines the derived belief update expressions into an iterative algorithm which performs joint sparse channel estimation and decoding. The algorithm is split into two parts: channel estimation (lines 7 - 35) and decoding (line 5). The outer loop alternates between these two steps until the information bit estimates have not changed in 10 iterations or a maximum of 50 iterations is reached.

The scheduling of the channel estimation is inspired by [22]. The basic idea is to construct a sparse representation of the wireless channel in the form of (3), by sequential refinement of the components in the constructed channel model. One component is determined by the set of parameters $\{\alpha_l, \gamma_l, \tau_l\}$ for a particular index $l$. The channel estimation procedure starts at (line 9) by finding a new candidate component via a grid search on possible values of $\hat{\tau}_l$. When subsequently a value of $\hat{\gamma}_l$ is calculated for the candidate component (line 18), the update formula (29) gives a criterion which determines if the new component should be included in the model or not. The choice of $\tau_{\text{new}}$ in line 11 is a heuristic guess of an initial tap delay value for the candidate component. This is, however, exactly the update which arises when assuming disjoint tap coefficients (see Sec. V-B), giving this choice some justification.

In the subsequent step (starting at line 22), all components in the model are sequentially refined. Again, the update (29) gives a criterion which allows the removal of components from the model. The channel estimation thus iteratively adds, updates and possibly removes components until convergence. As inner stopping criterion we use $|\lambda^t - \lambda^{t-1}| < \lambda^{t-1}/10^4$, where $t$ gives inner iteration number. The number of inner iterations is further limited to 50.

We initialize the noise precision as $\hat{\lambda} = 2N/||y||^2$. To initialize the messages $n_{x_i \rightarrow f_{\text{update}}}(x_i)$ we first observe that the non-sparse BP-MF receiver in [11], [45] includes a corresponding set of messages. We run one iteration of the channel estimator in this non-sparse BP-MF receiver and use the value of $n_{x_i \rightarrow f_{\text{update}}}(x_i)$ which arise here as the initialization of our algorithm. The non-sparse BP-MF algorithm requires a prior density on the channel frequency response $h$, which is selected to be a zero-mean Gaussian with covariance matrix calculated assuming a flat power delay profile, i.e. the “robust” covariance matrix of [29]. This initialization procedure obtains estimates of $x_D$ via a soft-output MMSE channel estimator which only uses the pilot symbols.

E. Computational Complexity

We note that the computationally heavy part of Algorithm 1 is the grid search in line 11 and the calculation of $\Sigma$ and $\Sigma_l$. We note that the grid search can be performed efficiently with a zero-padded fast Fourier transform at complexity $O(N \log(N))$, when the grid used is assumed of size $O(N)$. Calculating the update of $\Sigma$ and $\Sigma_l$ from (14) and (27) has complexity $O(N L^2)$, where $L$ is the number of components currently included in the model. Given that these matrices are calculated $O(L)$ times in the inner iteration, the overall computational complexity of each inner loop iteration is $O(N \log(N) + N L^2)$.

In the appendix update formulas are given which allows for $\Sigma$ and $\Sigma_l$ to be calculated in an incremental way. Doing so reduces the computational complexity of each inner loop iteration to $O(N \log(N) + N L^2)$. We note that for the update in line 8, the beliefs of $\lambda$ and $x_D$ have changed and the update formulas can therefore not be used. Since this update is only calculated once per inner loop iteration, this limitation does not affect the computational complexity.

V. ASSUMING DISJOINT TAP COEFFICIENT BELIEFS

To reduce the computational requirements of the scheme, we now assume that the tap coefficient belief $q(\alpha)$ factorizes into a product of disjoint factors as given by (12) and depicted in Fig. 3. The derivation is very similar to that in Section IV and in the following we only discuss the aspects which have changed from the previous derivation. As we will see, this also leads to an algorithm which allows for each update expression to be interpreted in an intuitive way.
A. Beliefs of Tap Coefficients

The only belief update which changes from the derivation in Section IV is that of the weighing coefficients \( \alpha_l \). In the model with disjoint tap coefficient beliefs, the belief of each \( \alpha_l \) is found to be

\[
q(\alpha_l) = \mathcal{N}(\alpha_l; \tilde{\mu}_l, \tilde{\sigma}^2_l),
\]

with

\[
\tilde{\sigma}^2_l = (\tilde{s}_l + \tilde{\gamma}_l)^{-1}
\]

\[
\tilde{\mu}_l = \tilde{\sigma}^2_l \tilde{q}_l
\]

\[
\tilde{s}_l = \lambda \psi^H(\tilde{\gamma}_l) \left\langle X^H X \right\rangle_{x_d} \psi(\tilde{\gamma}_l)
\]

\[
\tilde{q}_l = \lambda \psi^H(\tilde{\gamma}_l) r_l
\]

\[
r = \left\langle X^H X \right\rangle_{x_d} y - \left\langle X^H X \right\rangle_{x_d} \Psi(\tilde{\gamma}_l) \tilde{\mu}_l.
\]

Here the vector \( \tilde{\mu}_l \) is the stacking of the values \( \tilde{\mu}_k, k \neq l \) with \( \tilde{\mu}_l \) calculated from (32).

We note that from (31) and (32) we have

\[
\langle |\alpha_l|^2 \rangle_{\alpha_l} = \tilde{\sigma}^2_l + |\tilde{\mu}_l|^2 = (\tilde{s}_l + \tilde{\gamma}_l)^{-1} + |\tilde{q}_l|^2 (\tilde{s}_l + \tilde{\gamma}_l)^{-2},
\]

which is clearly equal to (24) with \( s_l \) and \( q_l \) replaced by \( \tilde{s}_l \) and \( \tilde{q}_l \). Reusing the analysis in Section IV, we conclude that the fixed point resulting from iteration between \( q(\gamma_l) \) and \( q(\alpha_l) \) ad-infinitum is given by (29) with \( s_l \) and \( q_l \) replaced by \( \tilde{s}_l \) and \( \tilde{q}_l \).

B. Joint Update of Tap Delay and Coefficient Beliefs

To improve the convergence speed of the algorithm, we now proceed to minimize the Kullback-Leibler divergence jointly with respect to the beliefs \( q(\alpha_l) \) and \( q(\gamma_l) = \delta(\gamma_l - \tilde{\gamma}_l) \) (as opposed to separate minimization which is described in Sec. V-A). Using the fact that \( q(\gamma_l) \) describes a point-estimate, we show that the solution reads

\[
\hat{\gamma}_l = \arg \max_{\gamma_l} \int Q(\alpha_l, \gamma_l) \, d\alpha_l
\]

\[
q(\alpha_l) = \frac{Q(\alpha_l, \hat{\gamma}_l)}{\int Q(\alpha_l, \gamma_l) \, d\gamma_l},
\]

where \( Q(\alpha_l, \gamma_l) \) is the optimal (in terms of minimal Kullback-Leibler divergence) joint belief of \( \alpha_l \) and \( \gamma_l \) when no factorization (i.e. mean-field approximation) is imposed over these two variables. We can employ a factor graph where \( \alpha_l \) and \( \gamma_l \) are merged into one node to arrive at

\[
Q(\alpha_l, \gamma_l) \propto \mathcal{N}(\alpha_l; \hat{\mu}_l, \hat{\sigma}^2_l) \exp \left( \frac{|\tilde{q}_l|^2}{\tilde{s}_l + \tilde{\gamma}_l} \right),
\]

where \( \hat{\sigma}^2_l, \hat{\mu}_l, \tilde{s}_l \) and \( \tilde{q}_l \) are given by (31)-(34) with all instances of \( \tilde{\gamma}_l \) replaced by \( \gamma_l \).

Using the above we get

\[
\hat{\gamma}_l = \arg \max_{\gamma_l} |\psi^H(\gamma_l)r_l|^2,
\]

while \( q(\alpha_l) \) remains as in (30). We note that the same result can be derived by calculating in closed form the beliefs which arise from iterating the updates of \( q(\alpha_l) \) and \( q(\gamma_l) \) ad-infinitum. We again employ Newtons method to solve (38).

C. Iterating all Tap Coefficients Ad-Infinum

We continue in a similar way and calculate in closed-form from the beliefs \( q(\alpha_l), l \in L \), which result from iterating the updates of these beliefs ad-infinitum. First note that the variance of a tap coefficient \( \hat{\sigma}_l^2 \) does not depend on the beliefs of the remaining tap coefficients \( q(\alpha_k), k \neq l \). The mean of the \( l \)th tap coefficient, on the other hand, depends on the remaining mean values as

\[
\hat{\mu}_l = \frac{\hat{\sigma}_l^2 \left( \hat{\lambda} \psi^H(\hat{\gamma}_l) \langle X^H X \rangle_{x_d} y - \sum_{k \neq l} \hat{\lambda} \psi^H(\hat{\gamma}_l) \langle X^H X \rangle_{x_d} \psi(\hat{\gamma}_k) \hat{\mu}_k \right)}{|Q_{\gamma_l}|^2}.
\]

This equation resembles the Gauss-Seidel [44] iteration for solving the system of linear equations

\[
Q \hat{\mu} = p
\]

with

\[
p = \hat{\lambda} \Psi^H(\hat{\gamma}) \langle X^H X \rangle_{x_d} y
\]

\[
Q = \hat{\lambda} \Psi^H(\hat{\gamma}) \langle X^H X \rangle_{x_d} \Psi(\hat{\gamma}) + \text{diag}(\hat{\gamma}).
\]

It follows that iterating the updates of \( \hat{\mu}_l, l \in L \), converges to the solution \( \hat{\mu} \) found by solving (39). Notice that this update of \( \hat{\mu} \) is equal to that in (15).

D. An Incremental Algorithm

Algorithm 2 describes an incremental algorithm which performs channel estimation and decoding with disjoint tap coefficient beliefs. The basic structure of the algorithm is the same as that of Alg. 1.

The computational complexity of Alg. 2 is dominated by the calculation of (39). Calculation of \( Q \) has complexity \( O(NL^2) \), while solving the system of linear equations (39) has complexity \( O(L^3) \). The total computational complexity for one inner loop iteration is thus \( O(N \log N + N^2L^2) \), assuming \( L \leq N \). Even though the inner loop of Alg 1 has the same complexity, the overall computational requirements of Alg. 2 is less than that of Alg. 1 due to the simpler structure of the latter. In particular, the introduction of the update (38) avoids the need of an explicit iteration between \( q(\alpha_l) \) and \( q(\gamma_l) \) as seen in lines 14 and 24 of Alg. 1.

VI. RESTRICTING THE TAP DELAY VALUES TO A GRID

As a reference and to investigate how a grid-based model approximation influences receiver behaviour, we now present an adaption of the sparse BP-MF receiver which restricts the tap delays to a grid. We here adapt the algorithm in Sec. IV, but note that a similar adaptation could be made to the algorithm in Sec. V.

A. Grid-based Version of Our Sparse BP-MF Receiver

The channel estimation procedure of Alg. 1 finds new components to add into the model via a search on a grid. It can therefore be turned into a grid-based procedure by skipping the loops which update \( \tilde{\gamma}_l \) in lines 14 and 24. Surprisingly it turns out that such an approach does not work well on its own. We therefore follow an approach similar in spirit to that presented in [28], [34] and modify the way new candidate
components are chosen (line 9). This approach is based on the (log) marginal posterior $p(\tau, \gamma | y)$. Following the steps in [34] we get

$$l(\tau_l, \gamma_l) = \log \int p(y | \alpha, \lambda, \tau, \lambda) p(\alpha | \gamma) d\alpha p(\tau) p(\gamma)$$

$$= - \log \left[ 1 + \frac{s_l}{\gamma_l} + \frac{|q_l|^2}{\gamma_l + s_l} + (\varepsilon - 1) \log \gamma_l - \eta \gamma_l + \text{const.} \right]$$

(40)

where $s_l$ and $q_l$ are given by (25) and (26) with $\hat{\tau}_l$ replaced by $\tau_l$. All $\gamma_l$ and $\tau_k$ for $k \neq l$ are implicitly evaluated at their estimates $\hat{\gamma}_l$ and $\hat{\tau}_l$ in (40).

For each delay $\hat{\tau}_m$ in the grid $\hat{\tau} = [\hat{\tau}_1, \ldots, \hat{\tau}_M]$ we calculate a candidate $\hat{\gamma}_m$ via (29). The pairs $(\hat{\tau}_m, \hat{\gamma}_m)$ are then considered as point estimates and the pair which maximizes the marginal posterior $l(\hat{\tau}_m, \hat{\gamma}_m)$ is added to the model (if $\hat{\gamma}_m \neq \infty$).

With this approach the channel estimation subtask is equal to the algorithm in [28] with parameters $\varepsilon = 1 - \varepsilon$ and $\eta = 0$, even though we have derived it in a different way. We also note that the above criterion for selecting the best candidate $\tau_n$ can also be used with the off-the-grid algorithm in Sec. IV. We have observed that the latter yields a better reconstruction accuracy for high signal-to-noise ratios (SNR). However, there is no noticeable difference in accuracy for the SNR values considered here (SNR $\leq 20$ dB).

B. Grid-based vs. Off-the-Grid Methods

The off-the-grid channel estimators in Sec. IV and V (and also [22], [23]) explicitly estimate the tap delays via (21) and (38). At first, this seems to be a fundamental difference from the grid-based estimation algorithm presented above (and also in [28], [34], [35]). For the algorithm in Sec V, the value $\tau$ in (38) can be considered a scaled version of the observation residual, i.e. the observation $y$ minus the observations predicted by the current model with the $l$th component removed ($l$ serves a similar role in (20) for the algorithm in Sec IV). The estimation of $\tau_l$ in the off-the-grid methods can therefore be considered a remove-add operation where the $l$th component is removed from the model and added at a new tap delay value. The grid-based sparse estimation algorithms do not allow for such an operation: A component is only removed from the model if the removal operation in itself increases the objective function. The fundamental difference between our grid-based and off-the-grid methods lie in the remove-add operation, and not in the use of a grid itself. We conjecture that the grid-based algorithms can achieve the same performance as the off-the-grid algorithms, if they are extended with the remove-add operation and provided the grid is selected sufficiently fine. In fact, any implementation of the off-the-grid algorithms has an implicit grid dictated by the machine precision of the implementation, thus blurring the distinction between these two approaches.

VII. NUMERICAL EVALUATION

In our numerical evaluation of the receiver algorithms, we use an OFDM system as described in Sec. II with the parameters listed in Table I. We use a rate–1/2 non-systematic
convolutional channel code, decoded by using the loopy BP implementation from the coded modulation library. The pilot signals are chosen at random from a QPSK alphabet.

We use a stochastic channel model with an exponentially decaying power delay profile. The channel responses are generated as follows: First, the number of multipath components $L$ is drawn from a zero-truncated Poisson distribution with mean $\mu_L = 5$. Then the $L$ tap delays in $\tau$ are drawn i.i.d. uniform from $(0, \tau_{\text{max}})$ and the $L$ tap coefficients in $\alpha$ are drawn independently from the conditional density

$$p(\alpha_l|\tau_l) = \mathcal{C}(\alpha_l; 0, u \exp(-\tau_l/v)),$$

where $u$ is chosen such that $\mathbb{E}[h_l] = 1$ and $v = 1.5 \mu s$. The noise precision is calculated based on the realization of the channel frequency response as $\lambda = \text{SNR} : N/||h||_2^2$.

In our numerical investigation, we show coded bit error rate (BER) and mean squared error (MSE) of the channel frequency response vector $h$. These results are averaged over 5000 Monte Carlo experiments, with each experiment containing one OFDM symbols. The OFDM symbols and channel realizations are generated i.i.d. according to the above description.

A. Evaluated Algorithms

We evaluate the algorithms presented in Sec. IV (Sparse BP-MF), Sec. V (Sparse BP-MF Disjoint) and Sec. VI (Grid-based Sparse BP-MF). In the subsequent evaluation, but that reported in Subsection VII-E, our algorithms use $\varepsilon = 1$ and $\eta = 0$. These values are chosen, as we have seen they provide a good trade-off between sparsity and fidelity. The grid-based algorithm of Sec. VI uses a grid of 500 tap-delay values. Our simulations show that no performance improvement results from using a finer grid for the simulation scenarios presented in the sequel. For comparison we also show results for the following reference algorithms:

**Turbo-GAMP:** The algorithm from [13], which employs a sample-spaced grid in the tap delay domain. We use a grid with $L_{\text{pre}} = L_{\text{post}} = 10$ extra taps before and after the cyclic prefix to capture sidelobe energy. The channel model that we use here does not include clustering effects and we therefore use the version of Turbo-GAMP without the hidden Markov chain tap cluster model. For each channel tap a large-tap and small-tap variance is provided along with a large-tap probability (see [13] for more details). These are estimated via the EM algorithm provided in [13] from 10,000 channel realizations.

**Oracle Channel Estimator:** This receiver is provided with the true value of the noise variance $\lambda$ and uses an oracle channel estimator. The oracle channel estimator computes an MMSE estimate of $h$ with the knowledge of the transmitted symbol vector $x$ (i.e. both pilots and data are known), the vector of delays $\tau$ and the probability density function of the channel tap coefficients in $\alpha$. From the estimate of $h$, the messages $n_{x_i} - f_{\mathcal{M}}(x_i)$ are computed for all $i \in \mathcal{D}$ (see (22)), followed by 5 iterations in the BP subgraph of Fig. 2.

B. Varying the Signal-to-Noise Ratio

Fig. 4 shows performance results for varying SNR. We first note that Sparse BP-MF and Sparse BP-MF Disjoint perform equally well in both BER and MSE. The grid-based Sparse BP-MF algorithm shows a slight degradation in terms of MSE as expected. This difference does not have an influence on BER due to the channel code. All Sparse BP-MF algorithms perform remarkably close to the oracle channel estimator in terms of BER, indicating that there is very little margin for improvement of our algorithms.

The two reference algorithms Turbo-GAMP and Freq.-domain BP-MF perform approximately 10 dB worse in terms of MSE. The BER loss compared to the performance of our algorithms corresponds to a $0.5 – 1 \text{ dB}$ difference in SNR. We note that in the high SNR regime, Freq.-domain BP-MF approaches our algorithms in terms of BER (but not of MSE). We conjecture that the comparatively poor performance of Turbo-GAMP is caused by its design restriction of the tap delays to the sample-spaced grid. Note that if the tap delays are generated to be located on such a grid, its performance is very close to that of the oracle channel estimator (not shown here). We have also investigated a version of Turbo-GAMP with the


discretized simulation scenarios. These results are averaged over 5000 Monte Carlo experiments, with each experiment containing one OFDM symbols. The OFDM symbols and channel realizations are generated i.i.d. according to the above description.

**Oracle Channel Estimator:** This receiver is provided with the true value of the noise variance $\lambda$ and uses an oracle channel estimator. The oracle channel estimator computes an MMSE estimate of $h$ with the knowledge of the transmitted symbol vector $x$ (i.e. both pilots and data are known), the vector of delays $\tau$ and the probability density function of the channel tap coefficients in $\alpha$. From the estimate of $h$, the messages $n_{x_i} - f_{\mathcal{M}}(x_i)$ are computed for all $i \in \mathcal{D}$ (see (22)), followed by 5 iterations in the BP subgraph of Fig. 2.

**B. Varying the Signal-to-Noise Ratio**

Fig. 4 shows performance results for varying SNR. We first note that Sparse BP-MF and Sparse BP-MF Disjoint perform equally well in both BER and MSE. The grid-based Sparse BP-MF algorithm shows a slight degradation in terms of MSE as expected. This difference does not have an influence on BER due to the channel code. All Sparse BP-MF algorithms perform remarkably close to the oracle channel estimator in terms of BER, indicating that there is very little margin for improvement of our algorithms.

The two reference algorithms Turbo-GAMP and Freq.-domain BP-MF perform approximately 10 dB worse in terms of MSE. The BER loss compared to the performance of our algorithms corresponds to a $0.5 – 1 \text{ dB}$ difference in SNR. We note that in the high SNR regime, Freq.-domain BP-MF approaches our algorithms in terms of BER (but not of MSE).

We conjecture that the comparatively poor performance of Turbo-GAMP is caused by its design restriction of the tap delays to the sample-spaced grid. Note that if the tap delays are generated to be located on such a grid, its performance is very close to that of the oracle channel estimator (not shown here). We have also investigated a version of Turbo-GAMP with the

4Available from http://iterativesolutions.com/Matlab.htm

Fig. 4. Simulated BER (left) and channel MSE (right) vs. signal-to-noise ratio. The legend to the right is valid for both plots.
small-tap variance set to 0 and noted that it does not yield better performance than the version of Turbo-GAMP shown here. Grid-based Sparse BP-MF does not suffer significantly from the grid-based approximation, since it employs a much finer grid.

C. Varying the Number of Pilots

In Fig. 5 we vary the number of pilot subcarriers to investigate if the increased channel estimation accuracy offered by our algorithms allow for decreasing the number of pilots without impairing the performance. We have observed that for our algorithms it is beneficial to use a random\(^3\) pilot pattern, which is therefore used for the three variants of Sparse BP-MF. According to our observations all reference algorithms achieve the best performance with an equispaced pilot pattern, which is therefore used in the numerical simulations. The reason why random pilot patterns provide an advantage over equispaced pilot patterns for the sparse channel estimators is the reduced sidelobe size as discussed in [46].

In the figure, we clearly see that the proposed algorithms can do with significantly fewer pilots than the reference schemes. The reference schemes break down at 41 pilots, corresponding to a pilot spacing of 15. We note that this is exactly the point where multipath components cannot be unambiguously identified within the interval \((0, \tau_{\text{max}})\), when an equispaced pilot pattern is used. We reiterate that using a random pilot pattern does not make any of the reference algorithms work better.

D. Convergence Speed

We now proceed to investigate the convergence speed of the algorithms. Fig. 6 show average values of BER and channel reconstruction MSE versus the number of outer loop iterations. The metrics are recorded immediately after the decoding operation. These results are for 10 dB SNR. It is clear that all algorithms converge within 5 iterations. Due to the initialization procedure that we have used, we note that the results for the first iteration correspond to a receiver which only use pilots for channel estimation. Such a receiver clearly gives very low performance. This indicates the significant performance improvements achieved by JCED receivers.

\(^3\)We choose the pilot pattern uniform random from a subset of all possible pilot patterns. The subset is chosen such that pilot patterns with significant sidelobes are excluded.

E. Varying the Number of Multipath Components

We finally investigate how the algorithms perform, when the assumption of a small number of multipath components is not fulfilled. The performance versus the mean number of multipath components \(\mu_L\) is shown in Fig. 7. We have here chosen to only investigate Sparse BP-MF Disjoint and the reference algorithms. To show how the sparsity controlling factor \(\varepsilon\) influences performance, we show results for three such values: \(\varepsilon \in \{0, 0.5, 1\}\). For all three cases we use \(\eta = 0\).

We first note that the most sparsity-inducing prior setting \((\varepsilon = 1)\) works best when the number of multipath components is small \((\mu_L \leq 13)\), while the opposite is true for the least sparsity-inducing setting \((\varepsilon = 0)\). As expected, the setting \(\varepsilon = 0.5\) provides a midpoint between the two other settings.

All the algorithms show decreasing BER performance as the number of multipath components grow. This is because less structure is present for channel estimation and it therefore becomes a harder estimation problem. For Freq.-domain BP-MF the decrease in structure is incarnated as a more frequency-selective channel, which deteriorates the estimator performance. For the sparse channel estimators the increasing number of multipath components means that the channel impulse response cannot be resolved into distinct multipath components, thus deteriorating performance. We note that the BER performance of Sparse BP-MF \((\varepsilon = 0)\) is better than or on par with Freq.-domain BP-MF for all values of \(\mu_L\); i.e. the sparse channel estimator performs as well as an MMSE channel estimator when the individual multipath components cannot be resolved, while offering lower BER when the multipath components can be resolved.

VIII. CONCLUSIONS

In this paper we have derived a number of JCED receivers which employ sparse channel estimation. Unlike other sparse channel estimators, two of our schemes do not restrict the CIR tap delays to a discrete grid. As a result, these receivers can truly exploit sparsity of the channel impulse response, without resorting to approximate sparsity (which in [13] and [20] is achieved by employing anti-aliasing filters with low sidelobes). We have presented a numerical evaluation which compares our algorithms with state-of-the-art methods, i.e. Turbo-GAMP [13] and Freq.-Domain BP-MF [11].

The principal difference between the grid-based Sparse BP-MF algorithm and Turbo-GAMP, is that the latter restricts the tap delays to the sample-spaced grid, while the former uses...
a much finer grid. Comparing performance of these two, it becomes evident that the sample-spaced grid is not sufficiently fine. It can be concluded that if grid-based sparse channel estimation techniques are to be used, it is important to use a grid finer than the sample-spaced grid, i.e. methods which allow correlation between dictionary columns are needed.

Given that a sufficiently fine grid is used, we have argued that the principal difference between our grid-based and off-the-grid algorithms lie in the remove-add operation which is incorporated into the off-the-grid algorithms. Such an operation can easily be incorporated into a grid-based algorithm. The distinction between grid-based and off-the-grid algorithms therefore becomes blurred and considerations of ease of implementation and computational complexity become the deciding factor in choosing between grid-based and off-the-grid algorithms.

The numerical evaluations show several interesting results. When random pilot patterns are used, our receivers allow for a significant reduction in the number of pilot signal, without any decrease in BER performance. We conjecture that even fewer pilots can be used if a pilot pattern designed for sparse channel estimation is employed, e.g. [46]. The pilot signals are most important during the first iteration of our algorithm, to initialize the iterative processing. An interesting research direction is the joint design of pilot signalling and channel coding schemes, which allows initialization of an iterative receiver while using a minimum amount of redundant information. To this end, the idea of using training bits [12], [13] instead of pilot signals may be useful.

The performance of our algorithm degrades stably when the assumption of a sparse CIR is not fulfilled. In the case where the mean number of multipath components $\mu_L$ is very high ($\mu_L \gg N \Delta f T_C$), as evaluated in Fig 7, our algorithms perform as well as a frequency-domain iterative receiver which does not assume channel sparsity. This is a very appealing property as, to obtain good performance, receiver designers need not be worried if the sparsity assumption is fulfilled.

APPENDIX

INCREMENTAL UPDATE EXPRESSIONS

An inverse is required in the update (14). To avoid calculating this inverse, we now find an update expression based on the matrix $\hat{\Sigma}_l$ defined in (27) (a similar approach is presented in [34]). Suppose that we wish to add a component in the $l$th position. To do so, we permute the rows and columns of $\Sigma$ such that the $l$th component is in the last row and column:

$$\Sigma_P = \left[ \begin{array}{c} \Sigma_l^{-1} \\ \hat{s}H \\ \hat{s}H \end{array} \right] = \left[ \begin{array}{c} \hat{s}H \\ \hat{s}H \end{array} \right] \left[ \begin{array}{c} \hat{s}H \\ \hat{s}H \end{array} \right] + \hat{\Sigma}_l \psi(\hat{\tau}) + \hat{\gamma}_l$$

with $s = \hat{\lambda} \hat{\Psi}^H(\hat{\tau}) \langle X^H X \rangle_{\mathbf{X}_D} \psi(\hat{\tau})$. Using block matrix inversion [44] we get

$$\hat{\Sigma}_P = \left[ \begin{array}{c} \mathbf{A} \\ \mathbf{b}H \\ \mathbf{c} \end{array} \right]$$

with definitions $\mathbf{A} = \hat{\Sigma}_l + c \hat{\Sigma}_l s \hat{s}^H \hat{\Sigma}_l$, $\mathbf{b} = -c \hat{\Sigma}_l s$ and $c = \left( \hat{\lambda} \hat{\Psi}^H(\hat{\tau}) \langle X^H X \rangle_{\mathbf{X}_D} \psi(\hat{\tau}) + \hat{\gamma}_l - \hat{s}^H \hat{\Sigma}_l s \right)^{-1}$. We can find the desired matrix $\hat{\Sigma}$ from $\hat{\Sigma}_P$ by reversing the permutation.

In a similar fashion, we can remove the $l$th component from $\hat{\Sigma}$ to get $\hat{\Sigma}_l$ by calculating

$$\hat{\Sigma}_l = \left[ \hat{\Sigma}_l \right]_{l,l} \left[ \hat{\Sigma}_l \right]_{l,l}^H$$

Turbo-GAMP also does not impose sparsity, but rather restricts some tap coefficients to be small. We reiterate that simulations (not presented here) have shown that no performance improvement follows from setting the small-tap variance of Turbo-GAMP to 0.