Precoding Matrix Design in Linear Video Coding

S. Zheng, M. Cagnazzo

LTCI, Télécom ParisTech,
Univ Paris-Saclay, 75013, Paris, France.

Abstract

Linear video coding (LVC) is a promising alternative to classical video coding when video has to be transmitted to wireless receivers experiencing different and time-varying channel conditions. This paper addresses the LVC channel precoding and decoding matrix design when the transmission channel consists of several sub-channels, each with its own power constraint. Such constraints may be found, e.g., in multi-antenna, DSL, or powerline transmission systems. In a previous paper, it has been shown that this matrix design problem may be addressed by an adaptation to LVC of a multi-level water-filling solution proposed for MIMO channels. Here, two suboptimal low-complexity multi-level water-filling techniques are proposed, with different trade-offs between complexity and efficiency. Extensive simulations show that the suboptimal solutions perform very close to the optimal one, with a sensibly reduced complexity.

1. Introduction

Linear video coding (LVC) schemes such as SoftCast [1] and its variants [2–15] have emerged as a promising alternative to classical video coding [16–18] when video has to be transmitted to wireless receivers experiencing different and time-varying channel conditions. In LVC, the video content is encoded with linear-only operators (such as a full-frame DCT and linear channel precoding). Using a linear MMSE estimator at receiver side, users experience reconstructed video quality commensurate with their channel quality [1]. Since the first paper on SoftCast [1], several developments have been considered. The chunks shape and size have been optimized in [7]. The coding gain of the pixel-domain transform is analyzed in [8, 9]. Hybrid digital-analog SoftCast-based architectures have been proposed in [2–6]. The characteristics of the transmission channel have also been better taken into account. First papers considered a wideband AWGN channel [10]. Fading channels and MIMO channels [11–15] have then been considered, mainly under a total transmission power constraint.

This paper addresses the LVC channel precoding and decoding matrix design when the transmission channel consists of several sub-channels, each with its own power constraint. Such constraints may be found, e.g., in multi-antenna, DSL, or powerline transmission systems [19]. In a previous paper [20], it has been shown that this matrix design problem may be addressed by an adaptation to LVC of a multi-level water-filling solution proposed for MIMO channels [21, 22]. Here, two suboptimal low-complexity multi-level water-filling techniques are proposed, with different trade-offs between complexity and efficiency.

The rest of the paper is organized as follows. The precoding and decoding matrix problem is introduced in Section 2. The two low-complexity suboptimal algorithms Power Allocation with Local Power Adjustment (PAISP) and Power Allocation with Local Power Adjustment (PALPA) are presented in Section 3. They are compared in Section 4. Section 5 concludes the paper.

2. Problem Formulation

In SoftCast, video frames are grouped into GoPs which are transmitted using a full-frame 3D-DCT. The resulting DCT coefficients of similar variance are grouped into nc chunks of size ni × nc. For each GoP, a sequence of ni × nc vectors of dimension nc is formed by selecting one coefficient per chunk for each vector, see Figure 1. These chunk vectors are assumed to be realizations of ni × nc independent and identically distributed zero-mean Gaussian random vectors ti, i = 1,...,ni. such chunks are transmitted as metadata.

Consider a wireless channel consisting of Nsc parallel AWGN subchannels with noise covariance matrix N = diag(σ2 1 ,...,σ2 Nsc ). and individual power constraints p,j, j = 1,...,Nsc. To transmit the chunk vectors ti over such channel, one has to find the optimal precoding matrix G and decoding matrix H to minimize the MSE at receiver, while satisfying the per-subchannel power constraints, which can be represented as

\[ \forall j \in \{1,\ldots,Nsc\}, \left( GAG^T \right)_{i,j} \leq p_j . \]

In what follows, the index i of ti is omitted, since all vectors ti have similar distribution and undergo the same processing. The vector t is multiplied by a precoding transform matrix G ∈ R Nsc×ni. The received vector is y = Gt + v, where v is a vector of channel noise with E(v) = 0 and E(vv^T) = N. To recover t, y is multiplied by a decoding matrix H ∈ R ni×Nsc to get ť = Hy.

Assuming that t and v are independent, the decoding matrix minimizing the mean square reconstruction error (MSE) is

\[ H = \Lambda G^T \left( GAG^T + \Lambda \right)^{-1} \]
and the resulting MSE is
\[ \varepsilon = \text{tr} \left( \Lambda - \Lambda G^T (GAG^T + N)^{-1} G \Lambda \right), \] (3)
see [20]. The precoding matrix design problem consists then in minimizing (3) with the constraints (1).

### 3. PRECODING MATRIX DESIGN

We assume, without loss of generality, that the chunk indexing is such that \( \lambda_1 \geq \cdots \geq \lambda_{n_{SC}} \) and that the subchannels are indexed by decreasing SNR constraints:
\[ \frac{\sigma^2_1}{p_1} \geq \frac{\sigma^2_2}{p_2} \geq \cdots \geq \frac{\sigma^2_{n_{SC}}}{p_{n_{SC}}}. \]

One introduces the vector of channel SNR constraints as \( s = (p_1/\sigma^2_1, \ldots, p_{n_{SC}}/\sigma^2_{n_{SC}})^T \).

The optimal precoding matrix design has been considered in [21, 23] and adapted to LVC in [20]. First, using water-filling, one computes an optimal diagonal precoding matrix \( \tilde{G} \) that satisfies the total equivalent channel power constraint \( p_{eq} = \sum_{i=1}^{n_{SC}} p_i/\sigma^2_i \). Then one searches an orthogonal matrix \( Z \) such that \( Z \tilde{G} \) satisfies the per-subchannel SNR constraints. Sufficient conditions on the vector of eigenvalues \( \tilde{m} = (\tilde{m}_1, \ldots, \tilde{m}_{n_{SC}})^T \) of \( GAG^T \) are provided in [24, 9.B.2] to guarantee the existence of such matrix \( Z \). The conditions are expressed in Theorem 1.

**Theorem 1.** [24, 9.B.2] If the entries of \( s \) and \( \tilde{m} \), arranged in non-increasing order \( \tilde{m}_1 \geq \cdots \geq \tilde{m}_{n_{SC}}, s_1 \geq \cdots \geq s_{n_{SC}} \), satisfy
\[
\sum_{i=1}^{k} s_i \leq \sum_{i=1}^{k} \tilde{m}_i \tag{4}
\]
for all \( k = 1, 2, \ldots, n_{SC} - 1 \) and
\[
\sum_{i=1}^{n_{SC}} s_i \geq \sum_{i=1}^{n_{SC}} \tilde{m}_i \tag{5}
\]
then there exists a Hermitian matrix with diagonal \( s \) and vector of eigenvalues \( \tilde{m} \).

If the sufficient conditions of Theorem 1 are satisfied ((5) is always satisfied when \( G \) is evaluated under total SNR constraint), several techniques are available to obtain \( Z \) solving a Structured Hermitian Inverse Eigenvalue [25–27]. Then the optimal precoding matrix is \( G = N^{\frac{1}{2}} Z \tilde{G} \). If the sufficient conditions are not satisfied, the optimal Multi-level Water-filling method proposed in [21, Section VI] has been used in [20]. The vector of variances \( \lambda \) and the vector of SNR constraints \( s \) are split into subvectors on which the conditions of Theorem 1 are tested again. If they are not satisfied, the subvectors are split again recursively. At the end, if they are satisfied, the precoding matrix has the following block diagonal structure
\[
G = N^{\frac{1}{2}} \text{BlockDiag} \left( G_1', \ldots, G_m', \ldots, G_{n_{SV}}' \right), \tag{6}
\]
where \( n_{SV} \) is the number of subvectors, and \( G_i' = Z_{(i)} \tilde{G}_{(i)} \) is the precoding matrix for each subvector where \( i \) is the index of Theorem 1 is satisfied.

In the optimal method, the split position in test corresponds to the largest index \( k \) that violates the conditions of Theorem 1 in the last recursion. The complexity of each recursion (mainly due to the solution of the water-filling problem) is proportional to the length of each subvector being tested. To find all subvectors in the worst case, \( \lambda \) and \( s \) are split into \( n_{SC} \) components, the complexity is \( O \left( n_{SC}^3 \right) \) see [21, Appendix D].

Next, in Section 3.1, it is shown that by inferring the split positions, the computation cost may be significantly reduced with respect to optimal algorithm. A second suboptimal power allocation scheme is presented in Section 3.2. In the following we assume that three algorithms are available. \textit{OptTotalPower} computes the optimal precoding matrix \( \tilde{G} \) and power allocation \( \tilde{m} \) under total SNR constraint. \textit{CheckSuffCond} verifies whether the sufficient conditions (4) are satisfied. If this is not the case, it returns the largest index \( k \) such that \( \sum_{i=1}^{k} s_i > \sum_{i=1}^{k} \tilde{m}_i \). \textit{SHIE} (Structured Hermitian Inverse Eigenvalue) computes the orthogonal transform matrix \( Z \) (See Theorem 1). Moreover, by a proper chunk size selection, one assumes that \( n_{SC} = n_{SC} \).

#### 3.1. Power Allocation with Inferred Split Position (PAISP)

PAISP takes initially \( \lambda = (\lambda_1 \ldots \lambda_{n_{SC}}) \) and \( s = (s_1 \ldots s_{n_{SC}}) \) of length \( \mu = n_{SC} \) as inputs. At first the largest index \( \tau \) that violates Conditions (4) is evaluated. Then PAISP searches the optimal split position at the interval \( [a, b] = [1, \tau] \) by dichotomy. First, the midpoint \( c = \frac{a+b}{2} \) is considered. If Conditions (4) are satisfied for \( (\lambda_1 \ldots \lambda_{c}) \) and \( (s_1 \ldots s_{c}) \), then PAISP updates \( a = c \); Else the largest index \( \tau \) that violates Conditions (4) is evaluated and PAISP updates \( b = \tau \). These iterations are repeated until the difference between two successive midpoints is not larger than 1, see Algorithm I.

To evaluate the complexity of PAISP to find all subvectors. The worst case is when \( \tau = \mu - 1 \) at each recursion of PAISP and when \( \lambda \) is split into \( n_{SC} \) components at the end. In such case, there are \( n_{SC} \) recursions and the total complexity is proportional to \( \sum_{\mu=1}^{n_{SC}} \mu \) and hence is \( O \left( n_{SC}^2 \right) \).
3.2. Power Allocation with Local Power Adjustment

The Power Allocation with Local Power Adjustment (PALPA) is a recursive algorithm taking initially $\lambda = (\lambda_1, \ldots, \lambda_{n_{SC}})$ and $s = (s_1, \ldots, s_{n_{SC}})$ of length $\mu = n_{SC}$ as inputs. PALPA evaluates first the preceding matrix with a total SNR constraint. The resulting allocated power vector has entries $\tilde{m}_i^{(0)}, i = 1, \ldots, n_{SC}$. If the conditions of Theorem 1 are satisfied, the preceding matrix is then built. Otherwise, let $\tau$ be the largest index for which Condition (4) is violated. Since $\sum_{i=1}^{n_{SC}} \tilde{m}_i^{(0)} = \sum_{i=1}^{n_{SC}} s_i$, one may introduce $\Delta^{(0)} = \sum_{i=1}^{\tau} s_i - \sum_{i=1}^{\tau} \tilde{m}_i^{(0)} = \sum_{i=\tau+1}^{n_{SC}} \tilde{m}_i^{(0)} - \sum_{i=\tau+1}^{n_{SC}} s_i$, i.e., too much power has been allocated to the last $n_{SC} - \tau$ components of $\lambda$.

The main idea of PALPA is to correct the values of $\tilde{m}_i^{(0)}, i = \tau + 1, \ldots, n_{SC}$. One evaluates first $\bar{\ell} = \max_{\tau \leq \ell' \leq n_{SC}} \ell'$ such that

$$\bar{\ell} = \sum_{i=\tau+1}^{\bar{\ell}} \tilde{m}_i^{(0)} - \sum_{i=\tau+1}^{\bar{\ell}} s_i \geq 0.$$  

Then for $i = \tau + 1, \ldots, n_{SC}$, the updated allocated powers are

$$\tilde{m}_i^{(1)} = \begin{cases} \tilde{m}_i^{(0)} - \frac{\sum_{i=\tau+1}^{n_{SC}} s_i}{\bar{\ell} - \tau} \tilde{m}_i^{(0)} & \text{if } i \leq \bar{\ell} \\ 0 & \text{otherwise.} \end{cases}$$  

(7)

This correction corresponds to an increase of the water level, see Figure 2. It ensures that the source components with large variance are still allocated a larger power [20, 23, 28].

**Proposition 2.** The power allocation adjustment performed by PALPA using (7) is such that for $k = \tau + 1, \ldots, n_{SC} - 1$,

$$\sum_{i=\tau+1}^{k} \tilde{m}_i^{(1)} \geq \sum_{i=\tau+1}^{k} s_i \text{ and } \sum_{i=\tau+1}^{n_{SC}} \tilde{m}_i^{(1)} = \sum_{i=\tau+1}^{n_{SC}} s_i.$$  

Then the corresponding part of the preceding matrix can be computed (See Theorem 1).

**Proof.** Using (7), one has $\Delta^{(0)} = \sum_{i=\tau+1}^{n_{SC}} (\tilde{m}_i^{(0)} - \tilde{m}_i^{(1)})$.

On the other hand, since $\tau$ is the largest index for which Condition (4) is violated. One has $\forall k \in \{\tau + 1, \ldots, n_{SC} - 1\}$,

$$\sum_{i=\tau+1}^{k} \tilde{m}_i^{(0)} \geq \sum_{i=\tau+1}^{k} s_i \Rightarrow \sum_{i=\tau+1}^{n_{SC}} \tilde{m}_i^{(0)} \geq \sum_{i=\tau+1}^{n_{SC}} s_i.$$  

(8)

Now, $\forall k \in \{1, \ldots, n_{SC} - 1\}$, $\Delta^{(0)} = \sum_{i=\tau+1}^{n_{SC}} (\tilde{m}_i^{(0)} - \tilde{m}_i^{(1)}) \geq \sum_{i=\tau+1}^{k} (\tilde{m}_i^{(0)} - \tilde{m}_i^{(1)})$. Thus from (8), one gets $\sum_{i=\tau+1}^{k} \tilde{m}_i^{(1)} \geq \sum_{i=\tau+1}^{k} s_i \geq \sum_{i=\tau+1}^{n_{SC}} \tilde{m}_i^{(1)} \geq \sum_{i=\tau+1}^{n_{SC}} s_i$. The proof of $\sum_{i=\tau+1}^{n_{SC}} \tilde{m}_i^{(1)} = \sum_{i=\tau+1}^{n_{SC}} s_i$ follows the same lines.

Then PALPA is called on the subvectors $\lambda = (\lambda_1, \ldots, \lambda_{\tau})$ and $s = (s_1, \ldots, s_{\tau})$ of length $\mu = \tau$, see Algorithm 2.

**To evaluate the complexity to find all subvectors with PALPA.**

The worst case is $\tau = \mu - 1$ at each recursion. Therefore the total complexity is proportional to $\sum_{\mu=1}^{n_{SC}} \mu$ (u is complexity of waterfall in each recursion) and is again $O(n_{SC}^2)$.

**Algorithm 2**

$$G' = \text{PALPA} (\lambda, s)$$

1. $\mu = \text{length}(\lambda) \% \text{ number of components of } \lambda$
2. $[\tilde{G}, \tilde{m}] = \text{OptTotalPower}(\lambda, s)$
3. $(\nu, \tau) = \text{CheckSuffCond} (\tilde{m}, s)$
4. if $\nu$ is true % Conditions (4) and (5) satisfied
5. $Z = \text{SHIE}(\tilde{m}, s)$
6. $G' = Z \tilde{G}$
7. else
8. $\Delta^{(2)} = \sum_{i=\tau+1}^{n_{SC}} \tilde{m}_i^{(0)} - \sum_{i=\tau+1}^{n_{SC}} s_i$
9. $\lambda^{(1)} = (\lambda_1, \ldots, \lambda_{\tau})$
10. $s^{(1)} = (s_1, \ldots, s_{\tau})$
11. $\tilde{m}_i^{(2)}(s^{(1)}) = (\tilde{m}_{\tau+1}^{(0)}, \ldots, \tilde{m}_{\mu}^{(0)})$
12. $s^{(2)} = (s_{\tau+1}, \ldots, s_{\mu})$
13. $G^{(1)} = \text{PALPA}(\lambda^{(1)}, s^{(1)})$
14. $Z^{(2)} = \text{SHIE}(\tilde{m}^{(1)}(\lambda^{(1)}), s^{(2)})$
15. $G' = Z^{(2)} \text{diag} \left( \frac{\sqrt{\tilde{m}_{\tau+1}^{(1)} / \lambda_{\tau+1}}, \ldots, \sqrt{\tilde{m}_{\mu}^{(1)} / \lambda_{\mu}}} \right)$

4. SIMULATION RESULTS

In the following simulations, one assumes that video has to be transmitted over an in-home power line channel [19]. The spacing between subchannels is $f_{SC} = 24.414$ kHz and the maximum number of subchannels that may be used for data transmission is $n_{SC} = 3217$. Not all subchannels are allowed for data transmission. In OFDM-based PLT systems like AV2, typically SNRs per subchannel are available. A realization of the individual subchannel SNRs is represented in Figure 3, which relates to a bad SISO link from ETSI STF 477 database [29].

Since in SoftCast, analog QAM is used with root-raised-cosine Nyquist filters with $\beta = 30\%$ roll-off, one obtains a per-subchannel transmission rate $r_{SC} = \frac{2}{n_{SC}} \times 37.56 \times 10^3$ real-valued symbols per second. Then the number of chunks $v_{SC}$ a subchannel can transmit for the duration of a GoP is evaluated from the video parameters (frame size, frame rate), the chunk size, and the GoP size (8 in our simulations). For the typical values of the parameters considered in these simulations, $v_{SC} > 1$, i.e., several chunks may be transmitted on the same subchannel for the duration of a GoP. To apply the preceding techniques, we partition the $n_{CL}$ chunks in groups of $v_{CH}$ chunks of similar variance. There are thus $n_{CH} = n_{SC} / v_{CH}$ groups of chunks. This requires for each GoP to design $v_{CH}$ smaller preceding matrices of size $n_{SC} \times n_{CH}$ for $n_{CH}$ chunks of same index in each groups of chunks. In the simulations, the best $n_{CH}$ subchannels are always used for the preceding matrix design. If $n_{CH} > n_{SC}$, the lowest variance chunks are discarded [1].

The video sequences of Classes B, C, D, E, and F used by the
Table 1: Simulation results with per-subchannel power constraints

MPEG committee for the standardization of HEVC [30] are considered in simulations (only the luminance component). The chunk size \((n_c \times n_r)\) is chosen as \(40 \times 30\) for Class B, E, F; \(32 \times 30\) for Class C and D. In the simulation, the subchannels are corrupted by independent white Gaussian sequences with unit variance.

PAISP and PALPA are compared to the optimal precoding matrix design method presented in Sect. 3 and to a Simple Chunk Scaling (SCS) approach, where each chunk is simply scaled to match each per-subchannel transmission power constraint. SCS can be considered as an adaptation of the allocation considered in the original SoftCast. The chunk of largest variance is transmitted over the subchannel with the best SNR, the chunk with the second largest variance over the second best subchannel, etc. To fit the per-subchannel power constraints, the \(i\)-th chunk is multiplied by \(g_{SC,i} = \sqrt{p_i/\Lambda_r}, i = 1, \ldots, nsC\). The four power allocation methods are all able to adjust the transmission power of chunks on each subchannel to have subchannel SNR matching those described in Figure 3. The simulation results are shown in terms of average PSNR of the received sequences in Tab. 1. PAISP has almost the same performance as the optimal method. For PALPA, the PSNR gap to optimality is never larger than 0.03 dB. In Fig. 4, the evolution of the PSNR of some videos as a function of the frame index is shown. Since PAISP and PALPA have close PSNR performance, only the results of PAISP are represented. All approaches clearly outperform the SCS.

The simulations are performed using MatlabR2014b on an Intel(R)Xeon(R)CPU ES-1603 v3 @ 2.8GHz. Table 2 provides the speed-up factor for the precoding matrix design of PAISP and PALPA compared to the optimal method. For RaceHorses and BasketballDrill of Class C, SlideShow of Class F, the speed-up of the suboptimal algorithms is close to one, since in most of the GoPs it is not necessary to perform vector splitting. For videos of Class D, there is no split within all GoPs, therefore the three methods again perform similarly. But for the video sequences of class B, class E, and the video BQMall of class C, the speed-up is significant.

### 5. Conclusions

This paper addresses the precoding matrix design problem in the context of LVC, when the video has to be transmitted over a MIMO channel with different per-subchannel power constraints such as PLT channels or multi-antenna systems. Two low-complexity suboptimal precoding matrix design techniques have been presented. These methods have a performance very close to the optimal precoding matrix design algorithm. The complexity of optimal approach may vary with the characteristics of video to encode. The low-complexity approaches are particularly efficient when the optimal precoding matrix design approach is complex. Compared to a naive precoding design, inspired by that of SoftCast, average gains in terms of PSNR range from 2.13dB for Class B videos to 11.55dB for Class F videos.
6. REFERENCES


