22. Konferenz Elektronische Sprachsignalverarbeitung (ESSV), September 2011, Aachen, Germany (TuDPress, ISBN 978-3-942710-37-4)

A Fast Indexing Method for Shells of the Gosset Lattice

Hauke Krüger, Bernd Geiser, Peter Vary
Institute of Communication Systems and Data Processing (ind)
RWTH Aachen University, Germany
krueger@ind.rwth-aachen.de

Haiting Li, Deming Zhang
Huawei Technologies Co., Ltd.
Corporate Research Dept.
Beijing, P.R. of China
lihaiting@huawei.com

Abstract: An iterative algorithm for indexing codevectors that are taken from spherical shells of the Gosset lattice E_8 is described. The proposed approach has a significantly lower computational complexity than the traditional *Schalkwijk* indexing method. Both methods are described in detail and an analysis of the encoding and decoding complexity is presented. The new indexing method is implemented in the "Gosset Low Complexity Vector Quantizer" (GLCVQ) which is used for super-wideband transform audio coding in Amd. 6 to ITU-T Rec. G.729.1.

1 Introduction

In gain-shape vector quantization [1], each input vector $\mathbf{x} \in \mathbb{R}^n$ is decomposed into a gain factor $g \geq 0$ and a shape vector $\mathbf{c} \in \mathbb{R}^n$ which are then quantized independently by means of a scalar and a vector quantizer, respectively. Typically, the Euclidean norm is used and the gain factor g and the shape vector $\mathbf{c}$ are computed as

$$g = |\mathbf{x}| = ||\mathbf{x}||_2 \qquad \text{and} \qquad \mathbf{c} = g^{-1}\mathbf{x}.$$

Hence, all normalized vectors $\mathbf{c}$ are located on the surface of the n-dimensional unit sphere and, consequently, the codevectors of the vector quantizer should cover that surface as uniformly as possible. Such a vector quantizer is referred to as a *spherical vector quantizer* (SVQ). Several approaches have been described and analyzed in the literature, e.g., [2, 3, 4].

A novel and highly efficient realization of a spherical vector quantizer (SVQ) is the "Gosset Low Complexity Vector Quantizer" (GLCVQ) which is used for transform audio coding in Amd. 6 to ITU-T Rec. G.729.1 [5, 6]. The GLCVQ codebook is composed of vectors that are located on spherical shells of the Gosset lattice E_8. The proposed GLCVQ encoding procedure is conducted in two steps. First, in the *quantization step*, the codevector with the minimum distance to the (normalized) input vector $\mathbf{c}$ must be determined. Then, in the *index assignment step*, the determined codevector must be transformed into a binary index i_{vq} which is transmitted to the decoder. Typically, the well-known *Schalkwijk* algorithm [7][1] is employed for lattice point indexing. In this paper, a novel index assignment algorithm is presented which can be realized with significantly lower complexity.

The paper is structured as follows. In Section 2, a general introduction for Gosset lattice based SVQ is given. The quantization step of the GLCVQ algorithm is summarized in Section 3 while the novel fast indexing method is detailed in Section 4. After a complexity and performance evaluation of the GLCVQ algorithm (Section 5), the paper is concluded.

[1]Originally, this algorithm has already been proposed in [8], see also [9].

Table 1 - Exemplary numbers of codevectors, equivalence classes and equiv. root classes for E_8 lattice.

Bit rate $\log_2(N)/n$	Codevectors N	Equiv. classes P (cf. [2])	Equiv. root classes Q
0.988	240	8	2
1.762	17520	42	5
2.257	272160	162	11

2 SVQ Based on the Gosset Lattice

The Gosset lattice is defined in eight dimensions, as the superposition of the checkerboard lattice D_8 and a shifted version thereof,

$$E_8 \doteq D_8 \cup (D_8 + \mathbf{v}), \mathbf{v} = \begin{bmatrix} \frac{1}{2} & \dots & \frac{1}{2} \end{bmatrix}^{\mathrm{T}}. \tag{1}$$

The checkerboard lattice is defined for arbitrary dimensions n as

$$D_n \doteq \{\mathbf{x} = \begin{bmatrix} x_0 & \dots & x_{n-1} \end{bmatrix}^{\mathrm{T}} \in \mathbb{Z}^n : (\sum_{i=0}^{n-1} x_i) \bmod 2 \equiv 0\}. \tag{2}$$

Lattice vectors with a constant distance to the origin define a *shell of a lattice*. The spherical vector codebook of the SVQ to be investigated in the following is composed of all N vectors which fulfill the Gosset lattice condition (1) and at the same time are located on a shell with a specific radius, normalized to have unit absolute value. Targeting a nearest-neighbor quantization with low complexity and memory, due to the invariance of (1) against permutation of the vector coordinates, the N codevectors populating the SVQ codebook can be represented by *permutation codes* as shown in [2].

Each of the permutation codes is defined by one out of P *classleader vectors* $\tilde{\mathbf{x}}_p \in \mathbb{R}^n$ which is composed of $L \leq n$ different real valued amplitudes μ_l distributed over the n vector coordinates in decreasing order $\mu_0 > \mu_1 > \cdots > \mu_{L-1}$, i.e.,

$$\begin{aligned} \tilde{\mathbf{x}}_p &= \begin{bmatrix} \tilde{x}_{p,0} & \tilde{x}_{p,1} & \dots \tilde{x}_{p,n-1} \end{bmatrix}^{\mathrm{T}} \\ &= \begin{bmatrix} \overset{\leftarrow w_0 \rightarrow}{\mu_0 \quad \mu_0} & \overset{\leftarrow w_1 \rightarrow}{\mu_1 \quad \mu_1} \dots & \overset{\leftarrow w_{L-1} \rightarrow}{\mu_{L-1} \quad \mu_{L-1}} \end{bmatrix}^{\mathrm{T}}. \end{aligned} \tag{3}$$

Each of the real values μ_l can occur w_l times within the vector. A *permutation* of the vector $\tilde{\mathbf{x}}_p$ is defined as another vector $\tilde{\mathbf{x}}$ that is composed of the same real values μ_l but in a different order. An *equivalence class* is defined as the set of codevectors which can be produced by arbitrary permutations of a single classleader vector. Finally, the SVQ codebook is defined as the aggregation of the codevectors of the equivalence classes related to all P classleader vectors, normalized to have unit absolute value.

Due to the permutation code representation of the codebook, an efficient nearest-neighbor quantization routine can be employed as proposed in [10] where only the classleader vectors must be evaluated rather than all vectors in the codebook. Examples of the number of spherical codevectors and corresponding classleader vectors for codebook designs at different effective bit rates per vector coordinate are listed in Tab. 1.

3 Gosset Low Complexity VQ — Quantization Step

The computational complexity of the SVQ approach as described in the previous section is still quite high, in particular at higher bit rates, because a relatively large number of classleader

Table 2 - Example type A+B classleader root vectors and the corresp. sets of classleader vectors for E_8.

Classleader root vector ($\tilde{\mathbf{x}}_q$)	Associated classleader vectors ($\tilde{\mathbf{x}}_p$)
$\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}^{\mathrm{T}}$ (Type A)	$\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}^{\mathrm{T}}, \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 0 & 0 & -1\end{bmatrix}^{\mathrm{T}}$
$\begin{bmatrix}\frac{3}{4} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2}\end{bmatrix}^{\mathrm{T}}$ Type B (odd parity)	$\begin{bmatrix}\frac{3}{4} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2}\end{bmatrix}^{\mathrm{T}}, \begin{bmatrix}\frac{3}{4} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2}\end{bmatrix}^{\mathrm{T}}$ $\begin{bmatrix}\frac{3}{4} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2}\end{bmatrix}^{\mathrm{T}}, \begin{bmatrix}\frac{3}{4} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2}\end{bmatrix}^{\mathrm{T}}$ $\begin{bmatrix}\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{3}{4}\end{bmatrix}^{\mathrm{T}}, \begin{bmatrix}\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{3}{4}\end{bmatrix}^{\mathrm{T}}$ $\begin{bmatrix}\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{3}{4}\end{bmatrix}^{\mathrm{T}}, \begin{bmatrix}\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{3}{4}\end{bmatrix}^{\mathrm{T}}$

vectors must be evaluated in order to find the optimal codevector for a given input vector $\mathbf{x}$. To reduce the complexity, the GLCVQ approach defines $Q < P$ *classleader root vectors* $\tilde{\mathbf{x}}_q$ so that only Q instead of P vectors must be accounted for in the quantization routine. Some examples for typical relations of P and Q are listed in Tab. 1.

Usually, the Gosset lattice is defined for eight dimensions. Due to its construction rule which is based on the general D_n lattice, formally, the proposed GLCVQ concept can be generalized to arbitrary dimensions. High quantization performance, however, can only be achieved for dimensions which are multiples of eight [11].

3.1 Definition of Classleader Root Vectors

A *classleader root vector* $\tilde{\mathbf{x}}_q$ is defined in analogy to (3) but contains only *positive* real valued amplitudes $\mu_l \geq 0$. Given a classleader root vector, sets of classleader vectors can be constructed by combining the classleader root vector coordinates with a specific distribution of positive and negative signs. However, in order to fulfill the lattice constraint (1), a specific *sign parity condition* must be considered for two classes of classleader root vectors, described in the following. In analogy to Sec. 2, all codevectors which can be produced based on a specific classleader root vector form an *equivalence root class*.

Type A classleader root vectors are the basis to produce codevectors which fulfill the constraint as defined in the *first* part of (1) (the definition of the D_n lattice). In particular, any valid vector $\mathbf{x_A} = \begin{bmatrix} x_{A,0} & \cdots & x_{A,n-1} \end{bmatrix}^{\mathrm{T}}$ which fulfills the lattice constraint (1) can be transformed into another valid vector $\mathbf{x_A}'$ by inverting the sign of *one* (arbitrary) vector coordinate i_{A_0} as

$$(\sum_{i=0}^{n-1} x'_{A,i}) \bmod 2 = \left((\sum_{i=0}^{n-1} x_{A,i}) - 2 \cdot x_{A,i_{A_0}} \right) \bmod 2 \equiv 0 \tag{4}$$

whereby $x_{A,i_{A_0}} \in \mathbb{Z}$. Valid codevectors are hence produced from type A classleader root vectors by setting arbitrary combinations of positive and negative signs at all vector coordinates, followed by a permutation of the vector coordinates and normalization.

Type B classleader root vectors are the basis to produce codevectors which fulfill the constraint as defined in the *second* part of (1) (the definition of the *shifted* D_n lattice). Here, a vector $\mathbf{x_B}'$, produced by inverting the sign of *one* arbitrary vector coordinate i_{B_0} of a valid vector $\mathbf{x_B} = \begin{bmatrix} x_{B,0} & \cdots & x_{B,n-1} \end{bmatrix}$, would *not* comply with the definition of (3) anymore. Instead, the sign inversion of *two* different vector coordinates i_{B_0} and i_{B_1} is guaranteed to result in another valid vector $\mathbf{x_B}''$ because

$$\begin{aligned}(\sum_{i=0}^{n-1} x''_{B,i}) \bmod 2 &= \left((\sum_{i=0}^{n-1} x_{B,i}) - 2x_{B,i_{B_0}} - 2x_{B,i_{B_1}} \right) \bmod 2 \\ &= (0-1-1) \bmod 2 \equiv 0. \qquad (5)\end{aligned}$$

As a conclusion, valid codevectors are produced from type B classleader root vectors by setting such combinations of positive and negative signs that fulfill a particular *sign parity constraint*, followed by a coordinate permutation and normalization. The sign parity constraint can either be *even* or *odd* according to the definition

$$\text{parity}(\mathbf{x}) = \left(\sum_{i=0}^{n-1} \frac{1-\text{sign}(x_i)}{2} \right) \bmod 2 = \begin{cases} 1 & \text{odd} \\ 0 & \text{even} \end{cases} \qquad (6)$$

where $\text{sign}(x_i) = 1$ if $x_i \geq 0$ and -1 otherwise. Tab. 2 demonstrates how groups of classleader vectors $\tilde{\mathbf{x}}_p$ can be expressed by means of a type A classleader root vector as well as a Type B classleader root vector with odd sign parity constraint.

3.2 Nearest-Neighbor Quantization

The nearest-neighbor quantization routine of the GLCVQ approach takes advantage of the representation of all valid codevectors by means of type A and B classleader root vectors, cf. [12]. The algorithm is briefly summarized here.

Given a normalized input vector $\mathbf{c} \in \mathbb{R}^n$, first the magnitude vector $\mathbf{c}_{\text{mag}} \doteq \mathbf{P_c} \cdot \begin{bmatrix} |c_0| & \cdots & |c_{n-1}| \end{bmatrix}^{\text{T}}$ is constructed, where $\mathbf{P_c}$ is the permutation matrix to obtain a sorted vector with *decreasing* magnitudes. Since Type B classleader root vectors might require an additional sign inversion, an auxiliary vector $\mathbf{s}_q$ is defined as

$$\mathbf{s}_q = \begin{cases} \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}^{\text{T}} - 2 \cdot \mathbf{e}_{j_q} & \text{if } \tilde{\mathbf{x}}_q \text{ is of Type B and} \\ & \text{parity}(\mathbf{c}) \neq \text{parity}(\tilde{\mathbf{x}}_q) \\ \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}^{\text{T}} & \text{else} \end{cases} \qquad (7)$$

with the j_q-th unit vector $\mathbf{e}_{j_q}$. To minimize the impact of sign inversion, j_q must be chosen as the last non-zero vector coordinate of $\tilde{\mathbf{x}}_q$. Then, the optimum classleader root vector is found by computing

$$q_{\text{opt}} = \arg \max_{q \in \{0,\dots,Q-1\}} \mathbf{c}_{\text{mag}}^{\text{T}} \cdot (\tilde{\mathbf{x}}_q \circ \mathbf{s}_q) \qquad (8)$$

with the component-wise multiplication operator $\circ$. Finally, the quantized vector $\tilde{\mathbf{c}} \in \mathbb{R}^n$ is given in its unnormalized form as $\tilde{\mathbf{x}} = \mathbf{P}_{\mathbf{c}}^{\text{T}} \cdot \left(\tilde{\mathbf{x}}_{q_{\text{opt}}} \circ \mathbf{s}_{q_{\text{opt}}} \circ \mathbf{c}_{\text{sig}} \right)$ with the (permuted) sign vector $\mathbf{c}_{\text{sig}} = \mathbf{P_c} \cdot \begin{bmatrix} \text{sign}(c_0) & \cdots & \text{sign}(c_{n-1}) \end{bmatrix}^{\text{T}}$.

4 Gosset Low Complexity VQ — Index Assignment Step

Let $\tilde{\mathbf{x}}$ be the optimal codevector determined in the nearest-neighbor quantization that is a permuted version of a classleader vector $\tilde{\mathbf{x}}_p$ according to the definition from (3). The index p as well as the permutation of the coordinates shall be transformed into a unique codevector index i_{GLCVQ}.

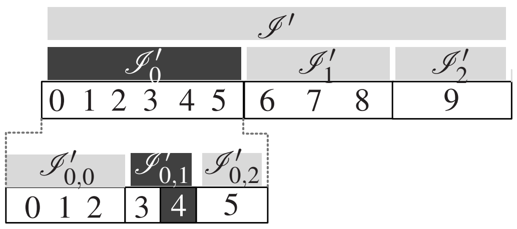


Figure 1 - Index range $\mathscr{I}'$ and decomposition into subranges.

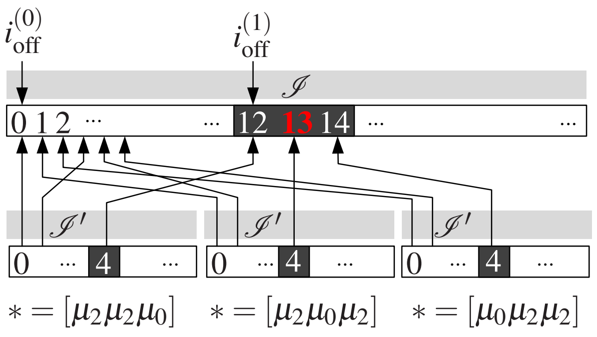


Figure 2 - Construction of index range $\mathscr{I}$ from three versions of the index range $\mathscr{I}'$ for $i^{(0)}_{\text{off}} = 0$ (following (13)).

The transformation of a permutation of a classleader vector into a unique index is traditionally achieved by the well-known *Schalkwijk* indexing method [7]. In the following, novel indexing method that is significantly more efficient for the GLCVQ codebook will be described. An exact description of the respective encoding and decoding process is given in Amd. 6 to [5]. Here, due to the lack of space, only the underlying principles shall be illustrated based on a concrete example for $n = 5$.

Let the selected classleader vector $\tilde{\mathbf{x}}_p \in \mathbb{R}^5$ be composed of w_l distinct amplitudes μ_l with $\mathbf{w} = \begin{bmatrix}1 & 2 & 2\end{bmatrix}^{\mathrm{T}}$ and $\mu_0 > \mu_1 > \mu_2$. The corresponding overall number of possible permutations of coordinates can be computed via the *multinomial coefficient* (for $L = 3$):

$$N_{\tilde{\mathbf{x}}} = \frac{(\sum_{l=0}^{L-1} w_l)!}{\prod_{l=0}^{L-1} w_l!} = 30. \tag{9}$$

Furthermore, let an exemplary optimal codevector be

$$\tilde{\mathbf{x}} \doteq \begin{bmatrix}\mu_2 & \mu_1 & \mu_0 & \mu_1 & \mu_2\end{bmatrix}^{\mathrm{T}}. \tag{10}$$

Since $\tilde{\mathbf{x}}$ is a permuted version of the classleader vector, the amplitudes are in general unsorted. An index i_{P} to represent the permutation of the vector coordinates of the optimal codevector shall be computed in the following. The codevector index must be in the range $0 \le i_{\mathrm{P}} < N_{\tilde{\mathbf{x}}}$ which shall be considered as the *codevector index range* $\mathscr{I}$. In the proposed approach to transform the permutation into an index, the codevector index range shall be successively subdivided into subranges to finally yield i_{P}.

Prerequisites: The computation of the index i_{P} will be carried out in iterations. Prior to the start of the actual computation, an index offset is introduced which is set to $i^{(0)}{}_{\text{off}} = 0$ for the first iteration. In later iterations, this offset will be set according to previous iterations of the algorithm. Moreover, the first iteration starts with the vector $\tilde{\mathbf{x}}^{(0)} = \tilde{\mathbf{x}}$. Its weights w_l and amplitudes μ_l are reordered such that the weights are in decreasing order, i.e., for the present example: $\bar{\mathbf{w}}^{(0)} \doteq \begin{bmatrix}w_1 & w_2 & w_0\end{bmatrix}^{\mathrm{T}} = \begin{bmatrix}2 & 2 & 1\end{bmatrix}^{\mathrm{T}}$ and $\bar{\mu}^{(0)} \doteq \begin{bmatrix}\mu_1 & \mu_2 & \mu_0\end{bmatrix}^{\mathrm{T}}$.

Iteration: To begin with the indexing, the multi-amplitude vector $\tilde{\mathbf{x}}^{(0)}$ is transformed into an intermediate vector

$$\tilde{\mathbf{x}}' \doteq \begin{bmatrix}* & \mu_1 & * & \mu_1 & *\end{bmatrix}^{\mathrm{T}} \tag{11}$$

which is only composed of the *first* amplitude value in $\bar{\mu}^{(0)}$ (here: μ_1) and of a wildcard $*$ which represents all amplitudes *except* μ_1. It is important to note that the *first* entry of $\bar{\mu}^{(0)}$ is chosen since it represents the most frequent amplitude value. The weights of the transformed vector are $w'_1 = \bar{w}^{(0)}{}_0 = w_1 = 2$ and $w'_* = \bar{w}^{(0)}{}_1 + \bar{w}^{(0)}{}_2 = 3$. Now the number of different permutations of the vector $\tilde{\mathbf{x}}'$ is given by the *binomial coefficient*

$$N_{\tilde{\mathbf{x}}'} = \binom{w'_1 + w'_*}{w'_*} = \binom{5}{3} \doteq C_5^3 = 10. \tag{12}$$

A temporary index $i_{\mathrm{P,loc}}$ is then introduced to represent the permutation of coordinates μ_1 and $*$ within the vector $\tilde{\mathbf{x}}'$ (11). Correspondingly, another *index range* $\mathscr{I}'$ is defined as $0 \leq i_{\mathrm{P,loc}} < N_{\tilde{\mathbf{x}}'}$ as shown in Fig. 1. $\mathscr{I}'$ can be subdivided into three subranges that correspond to three possible *vector start sequences* (up to the first wildcard):

- Subrange $\mathscr{I}'_0$: Sequence $\begin{bmatrix} * & \cdots \end{bmatrix}^{\mathrm{T}}$ ($C_4^2 = 6$ vectors).
- Subrange $\mathscr{I}'_1$: Sequence $\begin{bmatrix} \mu_1 & * & \cdots \end{bmatrix}^{\mathrm{T}}$ ($C_3^2 = 3$ vectors).
- Subrange $\mathscr{I}'_2$: Sequence $\begin{bmatrix} \mu_1 & \mu_1 & * & \cdots \end{bmatrix}^{\mathrm{T}}$ (1 vector).

In all three cases, $\cdots$ is used as a placeholder for all permutations of μ_1 and $*$ distributed over the remaining vector coordinates. The corresponding number of permutations is given for each subrange in parentheses, e.g., the placeholder $\cdots$ represents six possible permutations of μ_1 and $*$ in the last four vector coordinates of $\tilde{\mathbf{x}}'$ for $\mathscr{I}'_0$. In the present example, by comparison to the original vector (11), the index subrange $\mathscr{I}'_0$ is identified to match the vector $\tilde{\mathbf{x}}'$.

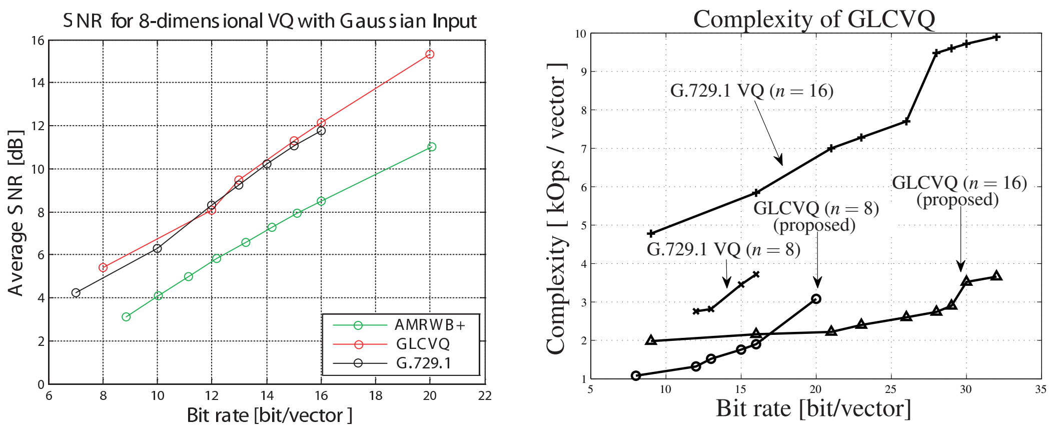
In the next step, $\mathscr{I}'_0$ must be further subdivided into subranges by considering the four remaining coordinates $\begin{bmatrix} \mu_1 & * & \mu_1 & * \end{bmatrix}^{\mathrm{T}}$ of $\tilde{\mathbf{x}}'$ in analogy to the decomposition of $\mathscr{I}'$ as

- Subrange $\mathscr{I}'_{0,0}$: $\begin{bmatrix} * & \cdots \end{bmatrix}^{\mathrm{T}}$ ($C_3^1 = 3$ vectors).
- Subrange $\mathscr{I}'_{0,1}$: $\begin{bmatrix} \mu_1 & * & \cdots \end{bmatrix}^{\mathrm{T}}$ ($C_2^1 = 2$ vectors).
- Subrange $\mathscr{I}'_{0,2}$: $\begin{bmatrix} \mu_1 & \mu_1 & * & \cdots \end{bmatrix}^{\mathrm{T}}$ ($C_1^1 = 1$ vector).

Again, the number of allowed permutations of amplitudes of the placeholder $\cdots$ is given in parentheses. In the example, the subrange $\mathscr{I}'_{0,1}$ matches the last four vector coordinates of $\tilde{\mathbf{x}}'$, and the placeholder $\cdots$ represents the last two coordinates of $\tilde{\mathbf{x}}'$, i.e., $\begin{bmatrix} \mu_1 & * \end{bmatrix}^{\mathrm{T}}$. In the last step, the subrange $\mathscr{I}'_{0,1}$ is further subdivided into two subranges, each populated by one index to finally yield the temporary index $i_{\mathrm{P,loc}} = 4$ in the example in Fig. 1.

So far, the index $i_{\mathrm{P,loc}}$ was found with respect to vector $\tilde{\mathbf{x}}'$ in (11) in which the wildcard $*$ represents either μ_0 or μ_2. When actually filling in the wildcard $*$ in (11), $C_3^2 = 3$ different permutations of μ_0 and μ_2 are allowed. Therefore, the index range $\mathscr{I}$ for the vector $\tilde{\mathbf{x}}$ is mapped to a combination of $C_3^2 = 3$ versions of the index range $\mathscr{I}'$ for the vector $\tilde{\mathbf{x}}'$ as shown in Fig. 2. Also, the index offset $i^{(0)}{}_{\mathrm{off}}$ must be considered in this mapping which, however, is zero in the first iteration. Due to the ambiguity in the mapping of the index $i_{\mathrm{P,loc}} = 4$ from index range $\mathscr{I}'$ to index range $\mathscr{I}$, $i_{\mathrm{P,loc}} = 4$ is mapped to the sequence $\{12, 13, 14\}$ of candidate indices in Fig. 2. As a consequence, the iteration procedure described before is repeated involving a modified vector $\tilde{\mathbf{x}}^{(1)} = \begin{bmatrix} \mu_2 & \mu_0 & \mu_2 \end{bmatrix}^{\mathrm{T}}$ as well as a modified weight vector $\bar{\mathbf{w}}^{(1)}$ and a modified amplitude vector $\bar{\mu}^{(\mathbf{1})}$ which are derived from the respective versions of the previous iteration by removing all coordinates with amplitude μ_1. In this next iteration, a new index offset

$$i_{\mathrm{off}}^{(1)} = i_{\mathrm{off}}^{(0)} + C_3^2 \cdot i_{\mathrm{P,loc}} = 12 \tag{13}$$



(a) (b)

Figure 3 - (a): SNR performance of 8-dim. GLCVQ compared with lattice VQ of ITU-T Rec. G.729.1 [5, 6] and 3GPP AMR-WB+ (b): Comparison of computational complexity measured in 1000 weighted operations per quantized vector (encoder and decoder).

must be considered to contribute for the index range determined in the first iteration. By repeating the described iteration procedure, one distinct amplitude value μ_l after the other is removed from the vector $\tilde{\mathbf{x}}$ to finally yield the index $i_\mathrm{P} = 13$.

Since the GLCVQ codebook is populated by permutations of P classleader vectors, in the final step, an index offset must be added to i_P to account for the classleader vector index p:

$$i_\mathrm{GLCVQ} = i_\mathrm{P} + \mathrm{offset}(p). \tag{14}$$

In order to retain a high coding efficiency, the index offsets for all classleader vectors are stored in lookup tables.

5 Evaluation

The quantization performance and computational complexity of GLCVQ has been compared with the lattice-based SVQ which is used for transform coding in the TDAC module of ITU-T Rec. G.729.1, see Fig. 3. The GLCVQ achieves a slightly better signal-to-quantization-noise-ratio than the reference VQ module which is in fact close to the theoretical optimum for the considered vector dimensions of $n = 8$ and $n = 16$ [11]. However, a considerable reduction in computational complexity (a factor of $2-3.5$ depending on the bit rate, cf. Fig. 3(b)) is achieved which is due to the efficient representation of the code in terms of classleader root vectors and the particularly efficient indexing procedure for the lattice points. On the other hand, since the resulting codebooks do not represent embedded codes, a little flexibility is sacrificed concerning the available bit rates.

6 Conclusions

We have reviewed the GLCVQ algorithm for spherical vector quantization with codebooks that are based on shells of the Gosset lattice. While maintaining excellent quantization performance, a considerable reduction of the computational complexity could be achieved by grouping the P equivalence classes of the original algorithm [2] into $Q < P$ equivalence root classes. For example, at a rate of approximately 2.257 bit per vector coordinate, only a fraction of $Q/P \approx 6.8\%$ of the candidate vectors have to be evaluated compared to [2].

Particularly, an iterative indexing method for the codevectors has been devised which, when usd in GLCVQ, is significantly less complex than the traditional Schalkwijk indexing. The main algorithmic advantage is that vector coordinates with identical amplitudes are jointly processed within a single algorithmic step, beginning with the most frequently occurring value.

The GLCVQ has been successfully applied for super-wideband speech and audio coding in the candidate codec described in [13]. Recently, it has been included in Amd. 6 to ITU-T Rec. G.729.1.

References

[1] M.J. Sabin and R.M. Gray, "Product Code Vector Quantizers for Waveform and Voice Coding," *IEEE Trans. Acoust., Speech & Sig. Proc.*, vol. 32, no. 3, pp. 474–488, June 1984.

[2] J. P. Adoul, C. Lamblin, and A. LeGuyader, "Baseband speech coding at 2400 bps using spherical vector quantization," in *Proc. of IEEE ICASSP*, San Diego, CA, USA, Mar. 1984.

[3] H. Krüger and P. Vary, "SCELP: Low Delay Audio Coding with Noise Shaping based on Spherical Vector Quantization," in *Proc. of EUSIPCO*, Florence, Italy, September 2006.

[4] B. Matschkal and J. B. Huber, "Spherical logarithmic quantization," *IEEE Trans. Audio, Speech, and Language Processing*, vol. 18, pp. 126–140, Jan. 2010.

[5] ITU-T Rec. G.729.1, "G.729 based embedded variable bit-rate coder: An 8-32 kbit/s scalable wideband coder bitstream interoperable with G.729," 2006.

[6] S. Ragot et al., "ITU-T G.729.1: An 8-32 kbit/s scalable coder interoperable with G.729 for wideband telephony and Voice over IP," in *Proc. of IEEE ICASSP*, Honolulu, Hawai'i, USA, Apr. 2007.

[7] J. P. M. Schalkwijk, "An algorithm for source coding," *IEEE Trans. Information Theory*, vol. 18, no. 3, pp. 395 – 399, May 1972.

[8] T. J. Lynch, "Sequence time coding for data compression," vol. 54, no. 10, pp. 1490–1491, 1966.

[9] L. Davisson, "Comments on 'an algorithm for source coding'," vol. 18, no. 6, 1972.

[10] D. Slepian, "Permutation Modulation," *Proceedings of the IEEE*, vol. 53, no. 3, pp. 228–236, 1965.

[11] H. Krüger, *Low Delay Audio Coding Based on Logarithmic Spherical Vector Quantization*, Ph.D. thesis, RWTH Aachen, 2010.

[12] H. Krüger, B. Geiser, and P. Vary, "Gosset low complexity vector quantization with application to audio coding," in *ITG Fachtagung Sprachkommunikation*, Bochum, Germany, Oct. 2010.

[13] B. Geiser et al., "Candidate proposal for ITU-T super-wideband speech and audio coding," in *Proc. of IEEE ICASSP*, Taipei, Taiwan, Apr. 2009.