Abstract

Space filling curves are widely used in Computer Science. In particular Hilbert curves and their generalisations to higher dimension are used as an indexing method because of their nice locality properties. This article generalises this concept to the systematic construction of \( p \)-adic versions of Hilbert curves based on affine transformations of the \( p \)-adic Gray code, and develops an efficient scaled indexing method for data taken from high-dimensional spaces based on these new curves, which with increasing dimension is shown to be less space consuming than the optimal standard static Hilbert curve index. A measure is derived which allows to assess the local sparsity of a data set, and is tested on some data.

1 Introduction

Motivated by the fact proven by Cantor that the unit interval has the same cardinality as any Euclidean \( n \)-space [4], Peano discovered in 1890 a continuous map from the unit interval onto the unit square, i.e. a first example of a space-filling curve [18]. Its construction uses an approximation by iteratively subdividing the square into \( 3^k \) subsquares and aligning these. A year later, Hilbert modified this 3-adic approach in order to obtain a binary construction of approximations to such a curve [12]. These are so-called space-filling curves and can be generalised to higher dimension, i.e. a continuous map onto an \( n \)-dimensional hypercube.

The binary reflected Gray code allows for a generalisation of the Hilbert curve to higher dimension. Such higher dimensional Hilbert curves are used as an index for point clouds in \( \mathbb{R}^n \). They have the property that nearby points on the curve are also nearby in \( \mathbb{R}^n \). Another property of Hilbert curves is their fractal structure: The \( k+1 \)-st iteration of the Hilbert curve is the union of copies of the \( k \)-th iteration. In [11], it is asked how many different generalisations of the Hilbert curve are there to dimension \( n \geq 3 \). A Gray-code index based on the Gray code for point clouds whose coordinates have varying precision is introduced in [10].

This article has a similar aim as [10] in that the scalability of the Hilbert curve should be exploited to produce a scalable index whose local precision adapts to the local density of the point cloud, and to be able to do this for any \( p \)-adic approximation to a Hilbert-like space-filling curve. For this, the reflected \( p \)-adic Gray code and affine transformations will be used in order to construct a multitude of \( p \)-adic Gray-Hilbert curves. The emphasis on this article is to
develop the theory underlying the $p$-adic Gray-Hilbert curve and to substantiate this theory by some experiental results with small data sets. A bigger data application is undertaken in [15] for a large rain forest tree data set.

In the following section, we fix some notation. Section 3 recalls properties of the $p$-adic reflected Gray code. Section 4 deals with the binary reflected code separately. Section 5 studies affine transformations of Gray codes. These are used in Section 6 to construct affine Gray-Hilbert codes. Section 7 goes to the projective limit, where the affine Gray-Hilbert curve becomes a continuous map between formal power series rings. Section 8 shows how to approximate a point on the 2-adic affine Gray-Hilbert curve, the case $p > 2$ having already been done in Section 6. Section 9 introduces a scaled indexing algorithm by mapping points onto different iterations of the swapping Hilbert curve. An insertion algorithm is introduced as well. Section 11 deals with upper complexity bounds. This is followed by Section 12 containing experiments. A conclusion finishes the core of this article.

2 Notation

Let $p$ be a prime number. The finite field with $p$ elements 0, 1, \ldots, $p - 1$ is denoted as $\mathbb{F}_p$. The $n$-dimensional $\mathbb{F}_p$ vector space $\mathbb{F}_p^n$ is viewed here as the set of maps

$$x = (x_{n-1}, \ldots, x_0): \{0, \ldots, n - 1\} \to \mathbb{F}_p, \ i \mapsto x_i$$

with the usual point-wise addition and scalar multiplication. The elements of $\mathbb{F}_p^n$ are also coefficients of polynomials or formal power series: the set of polynomials is denoted as $\mathbb{F}_p^n[t]$, and the set of formal power series as $\mathbb{F}_p[[t]]$. For $x \in \mathbb{F}_p^n$ we denote the support of $x$ as

$$\text{supp}(x) = \{i \in \{0, \ldots, n - 1\} \mid x_i = 1\}$$

The standard unit vectors are denoted as $e_i$ and have

$$\text{supp}(e_i) = \{i\}$$

for $i \in \{0, \ldots, n - 1\}$. The Hamming distance on $\mathbb{F}_p^n$ is denoted as $d_H$. $S_n$ will denote the group of permutations of the set $\{0, \ldots, n - 1\}$. If $x \in \mathbb{F}_p^n$, then

$$x^\sigma := (x_{\sigma(n-1)}, \ldots, x_{\sigma(0)})$$

for $x = (x_{n-1}, \ldots, x_0) \in \mathbb{F}_p^n$.

3 $p$-Adic reflected Gray code

We take the recursive definition of the reflected $p$-adic Gray code from [19]:

$$G(1, p) = (i),$$

$$G(n, p) = (i \ G(n - 1, p)^{\sigma_i})_i$$
where $i$ runs through the set $\{0, \ldots, p - 1\}$ and $\sigma_i \in S_p$ is the permutation

$$\sigma_i = \begin{cases} \text{id}, & i \equiv 0 \hspace{1em} \text{mod} \hspace{1em} 2 \\ \sigma & i \equiv 1 \hspace{1em} \text{mod} \hspace{1em} 2 \end{cases}$$

where

$$\sigma = \begin{pmatrix} 0 & 1 & \cdots & p - 1 \\ p - 1 & p - 2 & \cdots & 0 \end{pmatrix}$$

In other words, $G(n - 1, p)^{\sigma_i}$ is $G(n - 1, p)$ in reverse order, if $i$ is odd. Notice that $G(n, p)$ is a sequence of elements from $F_p^n$, i.e. a map $F_p^n \to F_p^n$.

Let

$$\text{bit}_n : \mathbb{Z}/p^n\mathbb{Z} \to F_p^n$$

be the map which takes every number to its $p$-adic representation.

Instead of $G(n, p)$, we will write $gc_n$, suppressing the $p$ in the notation.

**Lemma 3.1.** It holds true that

$$\begin{align*}
    gc_n(\text{bit}_n(0)) &= (0 \ldots 0) \\
    gc_n(\text{bit}_n(p^n - 1)) &= (-1 \cdots -1)
\end{align*}$$

**Proof.** We prove the assertions by induction.

Clearly, we have for $n = 1$:

$$\begin{align*}
    gc_1(0) &= (0) \\
    gc_1(p - 1) &= (-1)
\end{align*}$$

Now, assume that the assertions are true for $n > 0$. Then

$$\begin{align*}
    gc_{n+1}(\text{bit}_{n+1}(0)) &= (0 \hspace{1em} gc_n(\text{bit}_n(0))^\sigma) \\
    &= (0 \hspace{1em} 0 \ldots 0) \\
    gc_{n+1}(\text{bit}_{n+1}(p^n - 1)) &= (-1 \hspace{1em} gc_n(\text{bit}_n(p^n - 1))^\sigma) \\
    &= (-1 \hspace{1em} -1 \cdots -1)
\end{align*}$$

This proves the assertions. \(\square\)

We say that elements $v$ and $v^\perp$ of $F_p^n$ are opposite, if

$$v + v^\perp = d$$

with $d = (-1 \cdots -1) \in F_p^n$.

**Lemma 3.2.** It holds true that

$$\begin{align*}
    gc_n(x)^\sigma &= gc_n(x^\perp)
\end{align*}$$

**Proof.** The Gray code in reverse order at $x = \text{bit}_n(i)$ is obtained by taking the element of the Gray code at the point whose index is $p^n - 1 - i$. But the $p$-adic representation of this number is $d - x = x^\perp$ (observe that in this case, there is no “carry” taking place, as the digits of $d$ are all highest possible: $p - 1$). \(\square\)
4 Binary reflected Gray code

Here, we consider the case \( p = 2 \) separately. Let

\[
\text{bit}_n : \mathbb{Z}/2^n \mathbb{Z} \to \mathbb{F}_2^n
\]

be the bijective map which takes every number to its binary representation.

A (cyclic) binary Gray code is a bijective mapping

\[
g : \mathbb{F}_2^n \to \mathbb{F}_2^n
\]

such that for all \( i \in \mathbb{Z}/2^n \mathbb{Z} \) it holds true that

\[
d_H(\alpha(i), \alpha(i + 1)) = 1
\]

for \( \alpha = g \circ \text{bit}_n \) and where \( d_H \) is the Hamming distance on \( \mathbb{F}_2^n \).

The binary reflected Gray code \([7]\) is the mapping

\[
gc_n : \mathbb{F}_2^n \to \mathbb{F}_2^n, \ x \mapsto x + (x \uparrow 1)
\]

where \( x \uparrow k \) means a right shift by \( k \) bits (cf. \([9]\)).

The map \( \gamma_n : \mathbb{Z}/2^n \mathbb{Z} \to \mathbb{F}_2^n \) is defined as \( \gamma_n = gc_n \circ \text{bit}_n \). According to \([9, \text{Thm. 2.1}]\), these definitions are equivalent with the binary reflected Gray code defined there. Notice, that \([9]\) does not distinguish between \( \gamma_n \) and \( gc_n \). For this reason, we also call \( \gamma_n \) a binary Gray code.

The following observation explains the coinciding notation for \( p = 2 \).

**Lemma 4.1.** The binary reflected Gray code coincides with the 2-adic reflected Gray code.

*Proof.* This is proven in \([16]\). \( \square \)

**Lemma 4.2.** The mapping \( gc_n \) is a linear automorphism of the \( \mathbb{F}_2 \)-vector space \( \mathbb{F}_2^n \).

*Proof.* On \( \mathbb{F}_2^n \) we consider the bitwise addition \( + \) of binary numbers. Clearly, \( gc_n \) is a bijection. It follows from the definition that

\[
gc_n(x) + gc_n(y) = x + (x \uparrow 1) + y + (y \uparrow 1)
\]

\[
= (x + y) + [(x + y) \uparrow 1]
\]

\[
= gc_n(x + y)
\]

as, clearly,

\[
(x \uparrow 1) + (y \uparrow 1) = (x + y) \uparrow 1
\]

Hence, \( gc_n \) is also \( \mathbb{F}_2 \)-linear. \( \square \)

**Remark 4.3.** Observe that for \( p > 2 \), the Gray code \( gc_n \) is not linear. E.g. let \( x = (0, \ldots, 0, 1, 0) \), \( y = (0, \ldots, 0, p - 1, 0) \). Then

\[
gc_n(x) = (0, \ldots, 0, 1, p - 1)
\]

\[
gc_n(y) = (0, \ldots, 0, p - 1, 0)
\]

\[
gc_n(x + y) = gc_n(\text{bit}_n(0)) = \text{bit}_n(0)
\]

\[
gc_n(x) + gc_n(y) = (0, \ldots, 0, p - 1)
\]

which contradicts linearity.
5 Transformed Gray code

5.1 Case $p = 2$

The binary reflected Gray code $\gamma_n$ has the starting point $e' = (0, \ldots, 0)$ and the ending point $f' = (1, 0, \ldots, 0) = e_{n-1}$. The direction $d'$ of $\gamma_n$ is given by

$$e' + f' = e_{d'}$$

So, $d' = n - 1$. Any bijection $T$ of $F_n^2$ which makes $\gamma_n$ to a (cyclic) binary Gray code $\gamma_T = T^{-1} \circ \gamma_n$ must satisfy:

$$d_H(\gamma(i), \gamma(i + 1)) = 1$$

for all $i \in \mathbb{Z}/n\mathbb{Z}$, where

$$\gamma_T = T^{-1} \circ \gamma_n$$

In particular, $\gamma_T$ must satisfy

$$\gamma_T(e') + \gamma_T(f') = e_d$$

for some $d \in \{0, \ldots, n - 1\}$. This means that $T$ takes $\gamma_T(e')$ to $e'$ and $e_d$ to $e_{d'}$. Furthermore, for all $i \in \mathbb{Z}/n\mathbb{Z}$ we have

$$\gamma_T(i) + \gamma_T(i + 1) = e_j$$

and also

$$\gamma_n(i) + \gamma_n(i + 1) = e_{\sigma^{-1}(j)}$$

with some $j \in \mathbb{Z}/n\mathbb{Z}$, and a permutation $\sigma = \sigma_d$ of $\{0, \ldots, n - 1\}$ induced by $T$, which takes $n - 1$ to $d$.

We will not investigate all possible Hamming distance preserving bijections, but are interested in affine transformations, because they contain a translation which takes $\gamma_T(e')$ to $e'$.

Let $e \in F_n^2$. Then the translation with $e$ is denoted as

$$A_e : F_n^2 \rightarrow F_n^2, \quad x \mapsto x + e$$

**Theorem 5.1.** If $T$ is an affine transformation such that $\gamma_T$ is a Gray code, then it can be written a translation followed by an index permutation:

$$T = \sigma_d \circ A_e$$

with prescribed $e \in F_n^2$ for some permutation $\sigma_d \in S_n$.

Here, we keep $d = \sigma_d^{-1}(n - 1)$ in the notation of $\sigma_d$.

**Proof.** If $T$ is an affine transformation of $F_n^2$, then it can be written in the form

$$T : x \mapsto \Lambda x + e^\sigma$$

with some $\Lambda \in \text{GL}_n(F_2)$ and some $e^\sigma \in F_n^2$. We have already seen that $\Lambda$ takes each $e_i$ to some $e_{\sigma(i)}$, and $e_{d}$ to $e_{n-1}$. Hence,

$$x \mapsto x^\sigma + e^\sigma = (x + e)^\sigma$$

This proves the assertion. 

$\square$
We call \( gc_T \) an affine Gray code if
\[
gc_T = T^{-1} \circ gc_n
\]
where \( T \) is an affine transformation.

**Corollary 5.2.** There are precisely \((n - 1)!\) affine Gray codes for \( F_p^n \) with prescribed starting point \( e \) and direction \( d \).

**Proof.** This is an immediate consequence of Theorem 5.1.

5.2 Case \( p > 2 \)

We define a corner of \( F_p^n \) to be a vector \( v \in F_p^n \) whose components \( v_i \) are in \( \{0, -1\} \). The set of all corners of \( F_p^n \) will be denoted as \( C \).

Assume \( p \) odd. Then let for \( e \in C \):
\[
\mathcal{T}_e = \{ \text{affine transformations } T: F_p^n \to F_p^n \text{ s.t.} \\
T(e) = 0, \ T(e^\perp) = d, \ T(C) = C, \text{ and} \\
x - y = e_i \Rightarrow T(x) - T(y) = \pm e_{\tau(i)} \text{ for all} \\
i \in \{0, \ldots, n - 1\} \text{ and some permutation } \tau. \}
\]

**Theorem 5.3.** Let \( T: F_p^n \to F_p^n \) be an affine transformation. It holds true that \( T \in \mathcal{T}_e \) if and only if \( T \) is given as follows:
\[
T: x \mapsto A(x - e)
\]
with \( A \in \text{GL}_n(F_p) \) such that
\[
Ae_i = \begin{cases} 
- e_{\tau(i)}, & i \in \text{supp}(e) \\
+ e_{\tau(i)}, & i \in \text{supp}(e^\perp) 
\end{cases}
\]
for some permutation \( \tau \) of the set \( \{0, \ldots, n - 1\} \).

We call \( A \) the linear part of \( T \), and \( \tau \) the permutation associated with \( T \).

**Proof.** The affine transformation can be written as
\[
T: x \mapsto Ax + c
\]
with \( c \in F_p^n \).

\( \Leftarrow. \) If \( T \) is as asserted, then \( c = -Ae \) and
\[
T(e) = Ae + c = 0
\]
Likewise,
\[
T(e^\perp) = Ae^\perp - Ae
= \sum_{i \in \text{supp}(e^\perp)} - e_{\tau(i)} + \sum_{i \in \text{supp}(e)} - e_{\tau(i)}
= d
\]
Let \( c' \in \mathcal{C} \), and set \( C' = \text{supp}(c') \). Then
\[
T(c') = Ac' + c = \sum_{i \in \text{supp}(e) \cap C'} e_{\tau(i)} - \sum_{i \in \text{supp}(e) \cap C'} e_{\tau(i)} - Ae = \sum_{i \in \text{supp}(e) \cap C'} e_{\tau(i)} - \sum_{i \in \text{supp}(e) \cap C'} e_{\tau(i)}
\]

This is a corner. Now, assume that \( x - y = e_i \). Then
\[
Tx - Ty = Ax - Ay = Ae_i = \pm e_{\tau(i)}
\]

Hence, we have shown that \( T \in \mathcal{T}_e \).

\[ \Rightarrow \] Let \( T \in \mathcal{T}_e \). As \( 0 \in \mathcal{C} \), it follows that
\[
c = T(0) \in \mathcal{C}
\]

Assume now that \( x - y = e_i \) for \( x, y \in \mathbb{F}_p^n \). Then
\[
Ae_i = Ax - Ay = Tx - Ty = \pm e_{\tau(i)}
\]

for some permutation \( \tau \in S_n \). As \( T(e) = 0 \), we have \( Ae = -c \). Hence,
\[
-c = Ae = \sum_{i \in \text{supp}(e)} \alpha_i e_{\tau(i)}
\]

with \( \alpha_i \in \{ \pm 1 \} \) for \( i \in \text{supp}(e) \). As \( c \in \mathcal{C} \), it follows that \( \alpha_i = 1 \) for \( i \in \text{supp}(e) \).

Hence,
\[
Ae_i = -e_{\tau(i)}
\]

if \( i \in \text{supp}(e) \). As \( T(e^\perp) = d \), it follows that
\[
Ae^\perp = d - c = c^\perp \in \mathcal{C}
\]

Hence,
\[
c^\perp = Ae^\perp = \sum_{i \in \text{supp}(e^\perp)} \alpha_i e_{\tau(i)}
\]

with \( \alpha_i \in \{ \pm 1 \} \) for \( i \in \text{supp}(e^\perp) \). It follows that \( \alpha_i = -1 \) for all \( i \in \text{supp}(e^\perp) \).

Hence,
\[
Ae_i = e_{\tau(i)}
\]

for all \( i \in \text{supp}(e^\perp) \). This proves the assertion about \( T \). \( \square \)

**Corollary 5.4.** It holds true that \( |\mathcal{T}_e| = n! \).

*Proof.* \( T \in \mathcal{T}_e \) is completely determined by the associated permutation \( \tau \) as in Theorem 5.3. The map
\[
\mathcal{T}_e \to S_n, \ T \mapsto \tau
\]

is clearly bijective. \( \square \)

**Corollary 5.5.** Let \( A \) be the linear part of \( T \in \mathcal{T}_e \). Then
\[
A(e^\perp - e) = d
\]
Proof. From Theorem 5.3 we have

\[
A(e^\perp - e) = \sum_{i \in \text{supp}(e^\perp)} -Ae_i + \sum_{i \in \text{supp}(e)} Ae_i
\]

\[
= \sum_{i=0}^{n-1} -e_{\tau(i)} = d
\]
as asserted.

Corollary 5.6. The transformation \(T \in \mathcal{T}_\varepsilon\) maps opposite points to opposite points.

Proof. Let \(x \in \mathbb{F}_p^n\). Then

\[
T(x) + T(x^\perp) = A(x - e) + A(x^\perp - e)
\]

\[
= A(d - 2e) = A(e^\perp - e) = d
\]

where the last equality holds true by Corollary 5.5. Hence, \(T(x)\) and \(T(x^\perp)\) are opposite points.

Corollary 5.7. Let \(T \in \mathcal{T}_\varepsilon\) with \(A \in \text{GL}_n(\mathbb{F}_p)\) its linear part and \(\tau \in S_n\) its associated permutation as in Theorem 5.3. Then

\[
Ae = -e^\tau
\]

\[
Ae^\perp = (e^\tau)^\perp
\]

Proof. This follows from the definition of \(A\), as \((e^\perp)^\tau = (e^\tau)^\perp\).

Corollary 5.8. If \(T \in \mathcal{T}_\varepsilon\), then \(T^{-1} \in \mathcal{T}_{\varepsilon^\tau}\), where \(\tau \in S_n\) is the permutation associated with \(T\).

Proof. Let \(A \in \text{GL}_n(\mathbb{F}_p)\) be the linear part of \(T\). Then \(T^{-1}\) is given as

\[
T^{-1}(x) = A^{-1}x + e = A^{-1}(x + Ae) = A^{-1}(x - e^\tau)
\]

according to Corollary 5.7. From this it follows that

\[
T^{-1}(e^\tau) = 0
\]

and

\[
T^{-1}\left((e^\tau)^\perp\right) = A^{-1}\left((e^\perp)^\tau - e^\tau\right)
\]

\[
= A^{-1}(Ae^\perp + Ae)
\]

\[
= e^\perp + e = d
\]

Clearly, \(T^{-1}\) induces a bijection on \(\mathcal{C}\). Further, \(A^{-1}\) is given by

\[
Ae_i = \begin{cases} 
-e_{\tau^{-1}(i)}, & i \in \text{supp}(e^\tau) \\
e_{\tau^{-1}(i)}, & i \in \text{supp}\left((e^\tau)^\perp\right)
\end{cases}
\]

This means that if \(x - y = e_i\), then

\[
T^{-1}(x) - T^{-1}(y) = A^{-1}(x - y) = A^{-1}e_i = \pm e_{\tau^{-1}(i)}
\]

From these considerations, it follows that \(T^{-1} \in \mathcal{T}_{\varepsilon^\tau}\).
Corollary 5.9. Let $T \in \mathcal{T}_e$. Then

$$T(x)^\perp = T(x^\perp)$$

for all $x \in \mathbb{F}_p^n$.

Proof. It holds true that

$$T(x)^\perp = d - T(x) = T(e^\perp) - T(x)$$
$$= A(e^\perp - e) - A(x - e)$$
$$= A(e^\perp - x) = A(x^\perp - e)$$
$$= T(x^\perp)$$

as asserted.

In what follows, assume that $e \in \mathbb{C}$ and $T_e \in \mathcal{T}_e$ are fixed.

Let

$$gc_e^n := T_e^{-1} \circ gc_n : \mathbb{F}_p^n \to \mathbb{F}_p^n$$

Lemma 5.10. It holds true that

$$gc_e^n(0) = e, \quad gc_e^n(d) = e^\perp$$

for $e \in \mathbb{F}_p^n$.

Proof. We have

$$gc_e^n(0) = T_e^{-1}(gc_n(0)) = T_e^{-1}(0) = e$$
$$gc_e^n(d) = T_e^{-1}(gc_n(d)) = T_e^{-1}(d) = e^\perp$$

where the last equalities hold true by definition of $T_e$.

6 Affine Gray-Hilbert curves

In [9] there is a construction of a Hilbert curve in arbitrary dimension $n$ by applying the Gray code to each subhypercube after a certain affine transformation. Here, instead of this particular one, we allow any affine transformation to obtain a transformed Gray code as in Section 5.

The zero-th iteration of this Hilbert curve is the $p$-adic reflected Gray code for $\mathbb{F}_p^n$. The $k+1$-st iteration is obtained from the $k$-th iteration by subdividing each hypercube into $p^n$ subhypercubes and gluing together transformed Gray code curves for each of those hypercubes. These Gray code pieces are called local pieces and are determined by the local pieces from the $k$-th iteration.

Definition 6.1. A curve constructed as above is called an affine Gray-Hilbert curve.

Theorem 6.2. The number of $k$-th iterations of an affine Gray-Hilbert curve equals

$$\begin{cases}
(n-1)! \cdot \alpha_2, & p = 2 \\
n! \cdot \alpha_p, & p > 2
\end{cases}$$
with
\[ \alpha_p = \sum_{i=0}^{k} (p^n)^k = \frac{p^{n(k+1)} - 1}{p^n - 1} \]
for all \( p \).

**Proof.** This follows from Corollary 5.2 for \( p = 2 \), or Corollary 5.4 for \( p > 2 \), by induction.

There is a natural aggregation map between the \( k+1 \)-st iteration and the \( k \)-th iteration of an affine Gray-Hilbert curve which takes each hypercube to the superhypercube containing it.

In the following two subsections, we are going to explicitly construct affine Gray-Hilbert curves.

### 6.1 Case \( p = 2 \)

Consider the \( i \)-th point of the \( k \)-th iteration of an affine Gray-Hilbert curve for \( [0,1]^n \). In the \( k+1 \)-st iteration, this point is replaced by a transformed Gray code \( \gamma_{(e,d)}^{(e,d)} \), depending on \( e \in \mathbb{F}_2^n \) and \( d \in \mathbb{Z}/n\mathbb{Z} \). We may assume that this point lies in the image under the aggregation map of an untransformed Gray code curve \( \gamma_n \). Here, the first point of \( \gamma_{(e,d)}^{(e,d)} \) is \( e \) and the direction is \( d \). Following Section 5, we have
\[ \gamma_{(e,d)}^{(e,d)} = T^{-1}_{(e,d)} \circ \gamma_n \]
where we choose the particular affine transformation
\[ T_{(e,d)} = \sigma_d \circ A_e \]
with \( \sigma_d \) a permutation which takes \( n-1 \) to \( d \).

We also define
\[ \text{gc}_{(e,d)}^{(e,d)} := T^{-1}_{(e,d)} \circ \text{gc}_n \]

Next, we get from the definition of the binary reflected Gray code that
\[ \gamma_n(i) + \gamma_n(i+1) = e_{\tau_n(i)} \]
with some \( \tau_n(i) \in \mathbb{Z}/n\mathbb{Z} \) for all \( i \in \mathbb{Z}/2^n\mathbb{Z} \). According to [9, Lemma 2.3], it holds true that
\[ \tau_n(i) = \text{# of trailing 1 bits of bit}_n(i) \mod n \]

This leads to:

**Corollary 6.3.** It holds true that
\[ \gamma_{(e,d)}^{(e,d)}(i) + \gamma_{(e,d)}^{(e,d)}(i+1) = e_{\tau_n(i)}^{\sigma_d} \]

**Proof.** It holds true that
\[ \gamma_{(e,d)}^{(e,d)}(i) + \gamma_{(e,d)}^{(e,d)}(i+1) = e_{\tau_n(i)}^{\sigma_d} + e_{\gamma_n(i+1)}^{\sigma_d} + e \]
This proves the assertion. \( \square \)
We define:

\[
\tau_{n}^{(e,d)} : \mathbb{Z}/2^n \mathbb{Z} \to \mathbb{Z}/n \mathbb{Z},
\]

\[
i \mapsto \log_2 \left( \left( e \sigma_{d} \tau_{n}^{i} \right) \mod n \right)
\]

Notice that \( \tau_{n}^{(e,d)} \) does not depend on \( e \). We keep \( e \) in the notation anyway, as it is part of a transformation of parts of Hilbert curves depending on \( e \) and \( d \).

We will need the following map:

\[
\text{bit}_n^k : \mathbb{Z}/(2^n)^k \mathbb{Z} \to \mathbb{F}_2^n[t]/t^k \mathbb{F}_2^n[t]
\]

\[
x = \sum_{\nu=0}^{k-1} \alpha_{\nu} \cdot (2^n)^{\nu} \mapsto \sum_{\nu=0}^{k-1} \text{bit}_n(\alpha_{\nu}) \cdot t^{k-1-\nu}
\]

where \( x \) is given by its \( 2^n \)-adic expansion.

Observe that in the following map:

\[
c_n^k : \mathbb{F}_2^n[t]/t^k \mathbb{F}_2^n[t] \to [0, 1]^n
\]

\[
\sum_{\nu=0}^{k-1} \alpha_{\nu} t^{\nu} \mapsto \sum_{\nu=1}^{k} \alpha_{\nu} \left( \frac{1}{2} \right)^{\nu}
\]

that \( c_n^k(f) \) is the lower left corner of a sub-hypercube of \([0, 1]^n\) of sidelength \( \left( \frac{1}{2} \right)^k \).

There is the reduction map

\[
\rho_k : \mathbb{F}_2^n[t]/t^{k+1} \mathbb{F}_2^n[t] \to \mathbb{F}_2^n[t]/t^k \mathbb{F}_2^n[t]
\]

\[
f \mapsto f \mod t^k
\]

Observe that in the commutative diagram

\[
\begin{array}{ccc}
\mathbb{F}_2^n[t]/t^{k+1} \mathbb{F}_2^n[t] & \xrightarrow{c_n^{k+1}} & \text{im}(c_n^{k+1}) \\
\rho_k \downarrow & & \downarrow \mu_k \\
\mathbb{F}_2^n[t]/t^k \mathbb{F}_2^n[t] & \xrightarrow{c_n^k} & \text{im}(c_n^k)
\end{array}
\]

(2)

the pre-image of a point under \( \mu_k \) induces a subdivision of the corresponding hypercube into \( 2^n \) hypercubes.

The following commutative diagram with bijective horizontal maps

\[
\begin{array}{ccc}
\mathbb{Z}/(2^n)^{k+1} \mathbb{Z} & \xrightarrow{\text{bit}_n^{k+1}} & \mathbb{F}_2^n[t]/t^{k+1} \mathbb{F}_2^n[t] \\
\pi_k \downarrow & & \downarrow \rho_k \\
\mathbb{Z}/(2^n)^k \mathbb{Z} & \xrightarrow{\text{bit}_n^k} & \mathbb{F}_2^n[t]/t^k \mathbb{F}_2^n[t]
\end{array}
\]

(3)

defines the vertical map \( \pi_k \).
Lemma 6.4. It holds true that the representatives in \( \{0, \ldots, 2^{n(k+1)} - 1\} \) of \( \pi_k^{-1}(m) \) for any \( m \in \mathbb{Z}/(2^n)^k \mathbb{Z} \) form a consecutive set of numbers.

Proof. See how (3) looks on elements:

\[
\sum_{\nu=0}^{k} \alpha_\nu (2^n)^\nu \quad \rightarrow \quad \sum_{\nu=0}^{k} \alpha_\nu t^{k-\nu}
\]

As

\[
\sum_{\nu=1}^{k} \alpha_\nu t^{k-\nu} = \sum_{\nu=0}^{k-1} \alpha_{\nu+1} t^{k-1-\nu}
\]

it follows that

\[
m = \sum_{\nu=0}^{k-1} \alpha_{\nu+1} (2^n)^\nu
\]

and all possible pre-images are of the form

\[
\alpha_0 + 2^n \cdot m
\]

with \( \alpha_0 \in \{0, \ldots, 2^n - 1\} \). This forms a set of consecutive numbers. \( \square \)

Let \( b + F_2^n t^k \) be the pre-image of \( b \in F_2^n[t]/t^k F_2^n[t] \) under the reduction map \( \rho_k \). Then we define a bijection

\[
D_*^k: b + F_2^n t^k \rightarrow F_2^n, \quad b + \alpha t^k \mapsto \alpha
\]

and obtain a map \( h_*^k \) via the commutative diagram:

\[
\begin{array}{ccc}
 b + F_2^n t^k & \xrightarrow{h_*^k} & F_2^n \\
 \downarrow{D_*^k} & & \downarrow{D_*^k} \\
 F_2^n & \xrightarrow{g_{c_k(b), d_k(b)}} & F_2^n \\
\end{array}
\]

with \( c_k(b) \in F_2^n \) and \( d_k(b) \in \mathbb{Z}/n\mathbb{Z} \).

We define a distance \( d \) on \( F_2^n/t^k \mathbb{F} \) as

\[
d \left( \sum_{\nu=0}^{k-1} \alpha_\nu t^\nu, \sum_{\nu=0}^{k-1} \beta_\nu t^\nu \right) = \sum_{\nu=0}^{k-1} d_H(\alpha_\nu, \beta_\nu)
\]
Theorem 6.5. For all natural \( k \in \mathbb{N} \) there exist functions

\[ e_k : \mathbb{F}_2^t[t]/t^k\mathbb{F}_2^t[t] \to \mathbb{F}_2^t \]

\[ d_k : \mathbb{F}_2^t[t]/t^k\mathbb{F}_2^t[t] \to \mathbb{Z}/n\mathbb{Z} \]

such that the maps \( h_k^b \) are the restrictions of maps \( h^{k+1} \) fitting into the commutative diagram

\[
\begin{array}{ccc}
\mathbb{F}_2^t[t]/t^{k+1}\mathbb{F}_2^t[t] & \xrightarrow{h^{k+1}} & \mathbb{F}_2^t[t]/t^{k+1}\mathbb{F}_2^t[t] \\
\downarrow{\rho_k} & & \downarrow{\rho_k} \\
\mathbb{F}_2^t[t]/t^k\mathbb{F}_2^t[t] & \xrightarrow{h^k} & \mathbb{F}_2^t[t]/t^k\mathbb{F}_2^t[t]
\end{array}
\]

and such that with

\[
p_\ell = h^{k+1}(\text{bit}_n^{k+1}(2^n \cdot \ell + (2^n - 1)))
\]

\[
q_\ell = h^{k+1}(\text{bit}_n^{k+1}(2^n \cdot (\ell + 1)))
\]

it holds true that

\[ d(\rho_k(p_\ell), \rho_k(q_\ell)) = \delta_{k,0} \]

for all \( \ell \in \{0, \ldots, 2^{nk} - 1\} \).

We call \( h_k^b \) a local piece of \( h^{k+1} \), and \( h^{k+1} \) is called a refinement of \( h^k \).

Proof. We prove the assertion by induction: namely that for all \( N \in \mathbb{N} \) the assertion holds true for all \( k \leq N \).

\( N = 0 \). In this case, we have to check only the one diagram

\[
\begin{array}{ccc}
\mathbb{F}_2^t & \xrightarrow{h^0} & \mathbb{F}_2^t \\
\downarrow{\rho_0} & & \downarrow{\rho_0} \\
0 & \xrightarrow{h^0} & 0
\end{array}
\]

where \( h^0 \) is the zero map. According to (1), this works with \( h^1 = \text{gc}_n = \text{gc}_n^{(0,0)} \)

being the Gray code, i.e. \( e_0 \) and \( d_0 \) map everything to zero. The statement about the distance is clearly true.

Assume now that the assertion holds true for all \( k \leq N - 1 \) with \( N \geq 1 \). This means that we have maps \( e_{N-1} \) and \( d_{N-1} \) such that the maps \( h_b^{N-1} \) for \( b \in \mathbb{F}_2^t[t]/t^{N-1}\mathbb{F}_2^t[t] \) are the restrictions of a map

\[ h^N : \mathbb{F}_2^t[t]/t^N\mathbb{F}_2^t[t] \to \mathbb{F}_2^t[t]/t^N\mathbb{F}_2^t[t] \]

satisfying the distance condition. Let \( h_b^{N-1} \) be a local piece of \( h^N \). Without loss of generality, we may assume that \( h_b^{N-1} = \text{gc}_n \). Then we need to construct a Hilbert curve \( \chi_b \) such that

\[
\begin{array}{ccc}
\mathbb{F}_2^t[t]/t^2\mathbb{F}_2^t[t] & \xrightarrow{\chi_b} & \mathbb{F}_2^t[t]/t^2\mathbb{F}_2^t[t] \\
\downarrow{\rho} & & \downarrow{\rho} \\
\mathbb{F}_2^t & \xrightarrow{\text{gc}_n} & \mathbb{F}_2^t
\end{array}
\]
where \( \rho \) is the canonical projection. Let \( a \in F_2^n \). We need to show that \( h_a \) in the following diagram is a local piece of \( \chi_b \):

\[
\begin{array}{c}
\rho^{-1}(a) = a + F_2^n t \\
\downarrow \lambda_a \\
F_2^n \\
\downarrow g(c_{e_a,d_a}) \\
F_2^n \\
\end{array}
\]

Let \( i = \gamma_n^{-1}(a) \), and set \( e_a = e(i) \) and \( d_a = d(i) \) as in [9], where it is shown that the family of \( h_a \) form a second iteration Hilbert curve \( \chi_b \). We now need to show that each \( \chi_b \) is a local piece of a Hilbert curve \( h_{N+1} \). Strictly speaking, [9] shows only that the second iteration is a Hilbert curve.

\( h_N \) being a Hilbert curve means that (via \( e_{2n} \)) the local pieces \( h_{N-1}^b \) traverse hypercubes \( C \) which themselves are elements of a Hilbert curve. Let \( b, b' \) be such that \( (C, C') \) is an ordered pair of neighbouring hypercubes. Thus, the last hypercube traversed by \( h_{N-1}^b \) contains a vertex \( v \) of \( C \), and the first hypercube traversed by \( h_{N-1}^{b'} \) contains a vertex \( v' \) of \( C' \). As \( h_N \) is a Hilbert curve, it follows that \( v = v' \).

Now, refinements \( \chi_b \) and \( \chi_{b'} \) are also Hilbert curves. Hence, the last hypercube traversed by \( \chi_b \) contains a vertex \( w \) of \( C \), and the first hypercube traversed by \( \chi_{b'} \) contains a vertex \( w' \) of \( C' \). As those Hilbert curves are refinements of \( h_{N-1}^b \) and \( h_{N-1}^{b'} \), respectively, it follows that \( w = w' = v \).

Hence, \( v = v' \). This proves that the distance between those two small hypercubes also equals one. Hence, \( h_{N+1} \) is a Hilbert curve.

6.2 Case \( p > 2 \)

If we view \( x \in F_p^n \) as a word over the alphabet \( \{0, \ldots, p-1\} \), then we can define the longest common prefix length as the number \( \ell(x,y) \) of common initial consecutive letters of \( x \) and \( y \). Then we can define

\[
\text{trail}_{-1}(x) := \ell(x,d)
\]

for \( x \in F_p^n \), and \( d = (-1 \cdots -1) \).

**Lemma 6.6.** Let \( k = \text{trail}_{-1}(x) \) for \( x = \text{bit}_n(i) \) with \( i \in \mathbb{Z}/p^n\mathbb{Z} \). Then

\[
\text{bit}_n(i+1) = (y_\nu)
\]

with

\[
y_\nu = \begin{cases} 
0, & \nu < k \\
x_\nu + 1, & \nu = k \\
x_\nu, & \nu > k 
\end{cases}
\]

and \( \nu \in \{0, \ldots, n-1\} \).
Proof. For $\nu < k$, all $x_\nu$ get raised by one from $p - 1$ to 0, and then there is a last “carry”, which raises $x_k$ by one. As $x_k \neq p - 1$, it follows that there is no “carry” beyond this place, and so $y_\nu = x_\nu$ for $\nu > k$.

Let

$$\Delta(i) := \text{gc}_n(\text{bit}_n(i + 1)) - \text{gc}_n(\text{bit}_n(i))$$

**Lemma 6.7.** It holds true that

$$\Delta(i) = \pm e_{g(i)}$$

is plus or minus a standard unit vector with

$$g(i) = \text{trail}_{-1}(\text{bit}_n(i))$$

for all $i \in \mathbb{Z}/p^n\mathbb{Z} \setminus \{-1\}$.

**Proof.** The first statement is proven in [3].

Let $x = \text{bit}_n(i)$, $y = \text{bit}_n(i + 1)$, $x' = \text{gc}_n(x)$, $y' = \text{gc}_n(y)$, and $k = \text{trail}_{-1}(x)$. From Lemma 6.6, it follows that the last $n - k - 1$ digits of $x'$ and $y'$ coincide. As $x_k \neq y_k$ it follows that

$$x'_k = \tau(x_k) \neq \tau(y_k) = y'_k$$

with either $\tau = \text{id}$ or $\tau = \sigma$ for the permutation $\sigma$ in the definition of the Gray code. This shows that

$$g(i) = k$$

as asserted.

View the $p$-adic Gray code $G(n, p)$ as an ordering of points on a hypercube $I^n$. Assume that this hypercube is further regularly subdivided into $p^n$ hypercubes, each of which is ordered by a transformed $p$-adic Gray code $\text{gc}_e$ for some corner $e \in \mathbb{F}_p$ as follows: Let $\epsilon(i)$ and $\phi(i)$ be the entry and exit points of the $i$-th subhypercube in the transformed Gray code ordering of that individual subhypercube. The $i$-th and the $i + 1$-th subhypercubes are neighbors along the $g(i)$-th coordinate, we require

$$\epsilon(i + 1) - \phi(i) = e_{g(i)}$$

for $i \in \mathbb{Z}/p^n\mathbb{Z} \setminus \{-1\}$. We call this construction the first iteration of a $p$-adic Gray-Hilbert curve.

**Lemma 6.8.** Entry and exit points of a given subhypercube are corners and are opposite to each other:

$$\epsilon(i) \in \{0, -1\}^n, \quad \phi(i) = \epsilon(i)^\perp$$

for $i \in \mathbb{Z}/p^n\mathbb{Z} \setminus \{-1\}$.
Proof. Now, $\epsilon(i) = gc_n^e(0)$ for some corner $e \in F_p^n$. It follows that
$$
\epsilon(i) = gc_n^e(0) = T_e^{-1}(gc_n(0)) = T_e^{-1}(0) = e
$$
Thus, $\epsilon(i)$ is a corner. Further,
$$
\phi(i) = gc_n^e(d) = T_e^{-1}(gc_n(d)) = T_e^{-1}(d) = e^\perp = \epsilon(i)^\perp
$$
as asserted.

Hence, we have
$$
\epsilon(i + 1) - \epsilon(i)^\perp = e_{\gamma(i)}
$$
for $i \in \mathbb{Z}/p^n \mathbb{Z} \setminus \{-1\}$.

Further, $\epsilon(0)$ equals the entry point of the parent hypercube, and $\phi(p^n - 1)$ equals the exit point of the parent hypercube.

Lemma 6.9. The entry points of the first iteration of a $p$-adic Gray-Hilbert curve satisfy:
$$
\epsilon(i) = \sum_{j=0}^{i-1} (-1)^{i-1-j} (e_{\gamma(j)} + d)
$$
for $i \in \{0, \ldots, p^n - 1\}$.

Proof. This follows by induction from (5).

Let
$$
\epsilon^e := T_e^{-1} \circ \epsilon: \mathbb{Z}/p^n \mathbb{Z} \to F_p^n
$$
for $e \in F_p^n$. Let $i \in \mathbb{Z}/(p^n)^{k+1} \mathbb{Z}$, and let
$$
P_i(t) = \sum_{k=0}^{k} {i_k t^k} \in \mathbb{Z}/p^n \mathbb{Z}[t]
$$
be such that $P_i(p^n)$ is the $p^n$-adic expansion of $i$. $P_i(t)$ is called the $p^n$-adic polynomial for $i$.

Definition 6.10. A reflected $p$-adic Gray-Hilbert curve is a map
$$
b_n: F_p^n[[t]] \to F_p^n[[t]], \sum_{k \in \mathbb{N}} x_k t^k \mapsto \sum_{k \in \mathbb{N}} gc_n^{e_k}(x_k) t^k
$$
where $e_k = e_k(x_k) \in F_p^n$ are corners such that
$$
e_{k+1} - e_k^\perp = \pm e_{\tau_k}
$$
for some $\tau_k \in S_n$ and all $k \in \mathbb{N}$.

Theorem 6.11. Consider for each $i \in \mathbb{Z}/(p^n)^{k+1}$ the sequence
$$
\epsilon_0 = 0, \quad \epsilon_{\lambda+1} = e^{e_\lambda}(i_\lambda)
$$

where \( i_\lambda \in \mathbb{Z}/p^n\mathbb{Z} \) is a coefficient of the \( p^n \)-adic polynomial

\[
P_i(t) = \sum_{\lambda \in \mathbb{N}} \lambda t^\lambda
\]

for \( i \). Then it holds true that

\[
h_n(x) = \sum_{\lambda \in \mathbb{N}} \text{gc}_{n, x}(x_\lambda) t^\lambda
\]

with \( x = \sum_{\lambda \in \mathbb{N}} x_\lambda t^\lambda \) and

\[x_\lambda = \text{bit}(i_\lambda)\]

yields a well-defined \( p \)-adic Gray-Hilbert curve

\[
h_n : \mathbb{F}_p^n[[t]] \to \mathbb{F}_p^n[[t]], \ x \mapsto h_n(x)
\]

for any \( p > 2 \) prime and \( n > 0 \).

We call

\[
h_n^k : \mathbb{F}_p^n[t]/t^{k+1}\mathbb{F}_p^n[t] \to \mathbb{F}_p^n[t]/t^{k+1}\mathbb{F}_p^n[t], \ x \mapsto h_n^k(x) = \sum_{\lambda=0}^{k} \text{gc}_{n, x}(x_\lambda) t^\lambda
\]

the \( k \)-th iteration of \( h_n \).

Proof. We need to show that the Hamming distance between the exit point of one curve piece and the entry point of the next curve piece equals one for any iteration \( k \) of the \( p \)-adic Gray-Hilbert curve. Let \( h_n^k \) be the \( k \)-th iteration of \( h \).

Then two consecutive Gray-code pieces belonging to neighboring hypercubes at level \( k \) come from the same hypercube at level \( k - 1 \). Hence, we are interested in what happens for points \( x = \text{bit}_n(i) \), \( x' = \text{bit}_n(i') \) with \( i'_k = i_k - 1 \) and \( i'_\lambda = i_\lambda \) for \( \lambda < k \). This means that \( \epsilon_k \) coincides for both points. The difference between the exit point at \( i_k - 1 \) and entry point at \( i_k \) is thus

\[
\epsilon^{i_k}(i_k - 1) - \epsilon^{i_k}(i_k) = T_{\epsilon_k}^{-1}(\epsilon(i_k)) - T_{\epsilon_k}^{-1}(\epsilon(i_k - 1))
\]

\[
= A_k^{-1}(\epsilon(i_k)) - A_k^{-1}(\epsilon(i_k - 1)) = \pm e_{\tau_k^{-1}}(\epsilon(i_k))
\]

where \( A_k^{-1} \) is the linear part of and \( \tau_k \) the permutation associated with \( T_{\epsilon_k}^{-1} \).

7 Taking the projective limits

The diagram \( \Box \) is commutative for all \( p \):
\[
\begin{array}{c}
\mathbb{Z}/(p^n)^{k+1}\mathbb{Z} \xrightarrow{\text{bit}_n^{k+1}} F_p^n[t]/t^{k+1}F_p^n[t] \\
\downarrow \pi_k \\
\mathbb{Z}/(p^n)^k\mathbb{Z} \xrightarrow{\text{bit}_n^k} F_p^n[t]/t^kF_p^n[t] \xrightarrow{\rho_k} 
\end{array}
\]

where \(\rho_k\) is the canonical projection.

The horizontal bijections \(\text{bit}_n^k\) induce a homeomorphism between the projective limits:

\[
\mathbb{Z}^{\pi_{p^n}} := \lim_{\leftarrow} \mathbb{Z}/(p^n)^k\mathbb{Z} \xrightarrow{\text{bit}_n^k} F_p^n[t]
\]

Here, \(\lim_{\leftarrow}\) is the projective limit with respect to the inverse system \(\{\pi_k \mid k \in \mathbb{N}\}\) (cf. [14, §2.5] for the definition of projective limit).

**Theorem 7.1.** The Hilbert curve constructions lead to a continuous surjective map \(\mathcal{H}_n\) which fits into the following commutative diagram:

\[
\begin{array}{c}
\mathbb{Z}^{\pi_{p^n}} \xrightarrow{\text{bit}_n^k} F_p^n[t] \xrightarrow{h_n} F_p^n[t] \\
\downarrow c_n \downarrow
\end{array}
\]

The map \(c_n\) is induced by

\[
c_n^k : F_p^n[t]/t^kF_p^n[t] \to [0, 1]^n
\]

and is continuous, the maps \(h_n\) and \(\text{bit}_n^k\) are homeomorphisms.

**Proof.** The commutative diagram of Theorem 6.5 carries over to the case \(p > 2\) also. The horizontal bijections in that diagram induce a homeomorphism

\[
h_n : F_p^n[t] \to F_p^n[t]
\]

The commutative diagram (2) also carries over to the case \(p > 2\). The horizontal maps there induce a map

\[
c_n : F_p[t] \to [0, 1]
\]

which is surjective, and also continuous, as it is (1-)Lipschitz:

\[
|c_n(f) - c_n(g)| \leq |f - g|_p
\]

where

\[
|x|_p = \begin{cases} p^{-\nu}, & x = t^\nu \cdot u, \ u = \sum_{\mu=0}^{\infty} \alpha_\mu t^\mu, \ \alpha_0 = 1 \\ 0, & x = 0 \end{cases}
\]

is the \(p\)-adic norm. Thus all assertions are proven. \(\square\)
Notice that $c_n$ is not injective. Namely,
\[
    c_n \left( \frac{(p-1)t}{1-t} \right) = \sum_{\nu=1}^{\infty} (p-1)p^{-\nu-1} = \frac{1}{p} = c_n(1)
\]

We call $H_n$ an $n$-dimensional $p$-adic reflected Hilbert curve.

Notice also that $|\cdot|_p$ is a non-Archimedean (also called ultrametric) norm on $F_p^n$: it satisfies the strict triangle inequality
\[
    |x + y|_p \leq \max \left\{ |x|_p, |y|_p \right\}
\]

8 Approximating points on 2-adic affine Gray-Hilbert curves

In this section, we will show that, analogous to Theorem 6.11, the local pieces of 2-adic affine Gray-Hilbert curves are 2-adic approximations of that curve.

Let $p = 2$ and

\[
    \epsilon^{(e,d)} : \mathbb{Z}/2^n \mathbb{Z} \to \mathbb{F}_2^n
\]

\[
    i \mapsto \begin{cases} 
        \gamma^{(e,d)}(0), & i = 0 \\
        \gamma^{(e,d)}(i - 1), & i \equiv 1 \mod 2 \\
        \gamma^{(e,d)}(i - 2), & i \equiv 0 \mod 2
    \end{cases}
\]

and

\[
    \delta^{(e,d)} : \mathbb{Z}/2^n \mathbb{Z} \to \mathbb{Z}/n \mathbb{Z}
\]

\[
    i \mapsto \begin{cases} 
        \tau^{(e,d)}(0), & i = 0 \\
        \tau^{(e,d)}(i), & i \equiv 1 \mod 2 \\
        \tau^{(e,d)}(i - 1), & i \equiv 0 \mod 2
    \end{cases}
\]

with $\tau_n^{(e,d)}$ as in (1).

**Lemma 8.1.** The quantities $\epsilon^{(e,d)}$ and $\delta^{(e,d)}$ define the sequences of intra-sub-hypercube directions and of entry points for $g_c^{(e,d)}$.

**Proof.** This is an immediate consequence of [9, Thm. 2.9, Thm. 2.10]. □

For $k \geq 0$ on smallest local pieces:

\[
    \gamma_n^{(e_k,d_k)}(i_k) + \mathbb{F}_2^n t \xrightarrow{D_r} \gamma_n^{(e_k,d_k)}(i_k) + \mathbb{F}_2^n t
\]

\[
    \mathbb{F}_2^n \xrightarrow{g_c^{(e_{k+1},d_{k+1})}} \mathbb{F}_2^n
\]

with

\[
    \epsilon_k^{(e_k,d_k)} = \epsilon^{(e_k,d_k)}(i_k), \quad \delta_k^{(e_k,d_k)} = \delta^{(e_k,d_k)}(i_k)
\]

\[
    \epsilon_0 = 0, \quad \delta_0 = 0
\]

19
Or on $t$-adically expanded local pieces:

$$
\begin{array}{c}
\begin{array}{c}
\xymatrix{
E_n^{k+1}(\mathbb{F}_2) \ar[r]^-{D_t^k} & \mathbb{F}_2^{k+1} \\
\mathbb{F}_2^{k+1} \ar[r]^-{D_t^k} & \mathbb{F}_2^{k+1}
}
\end{array}
\end{array}
$$

with

$$b_k = \sum_{\kappa=0}^{k} \gamma_n(\epsilon_\kappa, \delta_\kappa)(i_\kappa) t^\kappa$$

We see that

$$D_t^k = D_t \circ \cdots \circ D_t$$

$k$ times

with $D_t = D_1^1$.

**Lemma 8.2.** Let $I(t) \in \mathbb{Z}/2^n \mathbb{Z}[t]$ be such that $I(2^n)$ is the $2^n$-adic expansion of $i \in \mathbb{Z}/(2^n)^{k+1} \mathbb{Z}$. Then

$$\pi_k(i) = \Delta_t I(2^n)$$

where

$$\Delta_t : \mathbb{Z}/2^n \mathbb{Z}[t] \to \mathbb{Z}/2^n \mathbb{Z}[t]$$

$$\sum_{\kappa=0}^{k} a_\kappa t^\kappa \mapsto \sum_{\kappa=1}^{k} a_\kappa t^{\kappa-1}$$

**Proof.** One readily checks this. \qed

With the $2^n$-adic expansion

$$i = \sum_{\kappa=0}^{k} i_\kappa (2^n)^{k-\kappa}$$

we have

$$e_k(b) = \epsilon_k, \quad d_k(b) = \delta_k$$

for

$$b = b(i) = \sum_{\kappa=0}^{k} \gamma_n(\epsilon_\kappa, \delta_\kappa)(i_\kappa) t^\kappa \in \mathbb{F}_2^n[t]/t^{k+1} \mathbb{F}_2^n[t]$$

**Theorem 8.3.** It holds true that

$$h_n(x) \equiv h^{k+1}(x_k) = \sum_{\kappa=0}^{k} g_n(\epsilon_\kappa, \delta_\kappa)(a_\kappa) t^\kappa \mod t^{k+1}$$

for $x = \sum_{\kappa=0}^{\infty} a_\kappa t^\kappa \in \mathbb{F}_2^n[[t]]$ and $x_k = \sum_{\kappa=0}^{k} a_\kappa t^\kappa$, and with $\epsilon_\kappa$ and $\delta_\kappa$ as in [7], where $i_\kappa = \text{bit}_n^{-1}(a_\kappa)$.  

20
Proof. Observe that, by construction,
\[ h^{k+1}(x_k) = h^{k+1}(\text{bit}^{k+1}(i)) = b(i) \]
where \( i = \text{bit}^{-1}(x_k) \) and \( b(i) \) as in \([10]\) for the \( 2^n \)-adic expansion of \( i \) given in \([3]\). According to Lemma \([8]\) \([9]\) defines the correct intra sub-hypercube directions and entry points. Clearly, we have
\[ h_n(x) \equiv h^{k+1}(x_k) \mod t^{k+1} \]
This proves the assertion. \(\Box\)

9 Indexing a sample of hypercube points using \(p\)-adic Hilbert curves

9.1 The \(p\)-adic Gray-Hilbert tree

Let \( \mathcal{T}_n \) be the finite rooted tree in which each non-leaf vertex has precisely \( p \) children, and with \( p^n \) leaves. A level is the length of a path from root to a vertex of \( \mathcal{T}_n \). The \( p\)-adic Gray-Hilbert tree \( \mathcal{G}_n \) is obtained by gluing a copy of \( \mathcal{T}_n \) to each leaf by identifying that leaf with the root of \( \mathcal{T}_n \). This copy will be called a local piece. Each level of a local piece is labelled by the coordinate given by the local ordering of coordinates which is determined by the transformation used for building up the Gray-Hilbert curve at this particular hypercube subdivision. The \( k\)-th iteration of \( \mathcal{G}_n \) is the finite subtree having the same root as \( \mathcal{G}_n \) and \( p^nk \) leaves. A path in the \( p\)-adic Gray-Hilbert tree from root to a given vertex corresponds to a sequence of subdivisions of certain coordinates which yields a subcuboid of \( I^n \) whose edges are one of two kinds: either short or long.

Let \( S \subset I^n \) be a finite point cloud. The Gray-Hilbert tree \( \mathcal{G}_n(S) \) generated by \( S \) is defined as being the smallest subtree of the \( p\)-adic Gray-Hilbert tree \( \mathcal{G}_n \) having the same root, and whose leaves correspond to subcuboids of \( I^n \) each containing precisely one element of \( S \). Later in this section, we will show how to obtain \( \mathcal{G}_n(S) \) efficiently.

9.2 The inverse Hilbert curve map: \( p = 2 \).

According to Theorem \([3]\) the inverse map \((h_n^{k+1})^{-1}\) is given for each coefficient by
\[ (\text{gc}^\mathcal{G}_n^{(\epsilon,\delta)})^{-1} = \text{gc}_n^{-1} \circ T_{(\epsilon,\delta)} \]
and \( \text{gc}_n^{-1} \) is the map
\[ \mathbb{F}_2^n \to \mathbb{F}_2^n, x \mapsto A \cdot x \]
with \( A = (\alpha_{ij}) \) and
\[ \alpha_{ij} = \begin{cases} 1, & i \leq j \\ 0, & \text{otherwise} \end{cases} \]
as can be readily checked.
9.3 The inverse Hilbert curve map: \( p > 2 \)

In the case \( p > 2 \), the inverse map \((h^k)_{n}^{-1}\) is given coefficient-wise by

\[
(g_{c_{n}})^{-1} = g_{c_{n}}^{-1} \circ T_{k}
\]

Let \( x \in \mathbb{F}_{p}^{n} \). We can write

\[
x = (x_{n-1} x^{n-1})
\]

with \( x^{n-1} = (x_{n-2}, \ldots, x_0) \in \mathbb{F}_{p}^{n-1} \).

**Lemma 9.1.** It holds true that

\[
g_{c_{n}}(x) = g_{c_{n}}(x^{\perp})
\]

for \( x \in \mathbb{F}_{p}^{n} \).

**Proof.** Here, we will write \( d_{n} \) for the vector \( d \in \mathbb{F}_{p}^{n} \). We proceed by induction. For \( n = 1 \), the assertion is clear. Assume that the assertion holds true for \( n - 1 \). Now, we have

\[
g_{c_{n}}(x) + g_{c_{n}}(x^{\perp}) = (x_{n-1} g_{c_{n-1}}(x^{n-1})^{\sigma_{x_{n-1}}}) + ((x_{n-1}^{\perp}) g_{c_{n-1}}(x^{n-1})^{\sigma_{x_{n-1}}})
\]

\[
= (p - 1) g_{c_{n-1}}(x^{n-1})^{\sigma_{x_{n-1}}}
\]

\[
+ g_{c_{n-1}}((x_{n-1}^{\perp})^{\sigma_{x_{n-1}}})
\]

\[
= (p - 1) w^{\sigma_{x_{n-1}}}
\]

with

\[
w = g_{c_{n-1}}(x^{n-1}) + g_{c_{n-1}}((x_{n-1}^{\perp})^{n-1})
\]

\[
= g_{c_{n-1}}(x^{n-1}) + g_{c_{n-1}}(x^{n-1})^{\perp}
\]

\[
= d_{n-1}
\]

as clearly, \((x_{n-1}^{\perp})^{n-1} = (x^{n-1})^{\perp}\). This proves the assertion. \(\square\)

**Lemma 9.2.** It holds true that

\[
g_{c_{n}} \circ g_{c_{n}} = id
\]

i.e. \( g_{c_{n}} \) is its own inverse.

**Proof.** We proceed by induction. For \( n = 1 \), the assertion is clear. Assume that the assertion holds true for \( n - 1 \). Now, we have

\[
g_{c_{n}}(g_{c_{n}}(x)) = (x_{n} g_{c_{n-1}}(g_{c_{n}}(x)^{n-1})^{\sigma_{x_{n-1}}})
\]

\[
g_{c_{n}}(x)^{n-1} = g_{c_{n-1}}(x^{n-1})^{\sigma_{x_{n-1}}}
\]
If \( x_{n-1} \equiv 0 \mod 2 \), then
\[
\begin{align*}
gc_{n-1} \left( gc_n(x)^{n-1} \right)^{\sigma x_{n-1}} &= gc_{n-1} \left( gc_{n-1} \left( x^{n-1} \right)^{\sigma} \right) \\
&= x^{n-1}
\end{align*}
\]

If \( x_{n-1} \equiv 1 \mod 2 \), then
\[
\begin{align*}
gc_{n-1} \left( gc_n(x)^{n-1} \right)^{\sigma x_{n-1}} &= gc_{n-1} \left( gc_{n-1} \left( x^{n-1} \right)^{\sigma} \right) \\
&= gc_{n-1} \left( gc_{n-1} \left( x^{n-1} \right)^{\sigma} \right) \\
&= gc_{n-1} \left( gc_{n-1} \left( x^{n-1} \right)^{\sigma} \right) = x^{n-1}
\end{align*}
\]
From this, the assertion follows.

9.4 The indexing method

The indexing method uses the inverse Hilbert curve map \((h^k)^{-1}\).

Let \( S \subset \mathbb{F}_p^n \) be a finite subset.

**Algorithm 9.3** (Indexing). Let \( S_0 = S \) and \( T_0 = \emptyset \).

Let for \( k \geq 1 \)
\[
S_k = S_{k-1} \setminus \left\{ s \in S_{k-1} \mid s \text{ is unique in } S_{k-1} \text{ with } s \equiv s' \mod t^k \text{ for } s' \in S_{k-1} \mod t^k \right\} \\
S_k = \left( S_{k-1} \setminus S^k \right) \mod t^k
\]

The map \((h^k)^{-1}\) induces a linear ordering on \( S_k \).

Map \( S_k \) via \( \rho_{k-1} \) to \( \mathbb{F}_p^n[t]/t^{k-1}\mathbb{F}_p^n[t] \). Then \((h^{k-1})^{-1}\) induces a linear ordering on \( \rho_k(S_k) \cup S_{k-1} \). This induces a linear ordering on \( S_k \cup S_{k-1} \), and thus on
\[
T_k = T_{k-1} \cup S_k
\]

Hence, obtain a partitioned set
\[
T_N = S_1 \cup \ldots \cup S_N
\]
consisting of partial sums of elements of \( S \) such that \( T_N \) and \( S \) have the same cardinality, and a linear ordering on \( T_N \). This induces a linear ordering on \( S \).

The set \( T_N \) coming out of Algorithm [9.3] corresponds to a set of hypercubes of various sizes, each being the largest containing precisely one element of the finite set \( S \). The following algorithm is a simplification of the algorithm in [2], §3.2.2] and finds from this the Gray-Hilbert tree generated by \( S \).
Algorithm 9.4 (Gray-Hilbert tree of a set). Let $S \subset \mathbb{F}_p^n[[t]]$ be a finite set. For a subset $S' \subset S$ let
\[ T' = g_n(S') \]
whose edges are labelled with the digits of the coefficients in the formal power series representing the elements of $S'$. Start with $s_0 \in S$. Set $S_0 = \{ s_0 \}$. Then $T_0 = g_n(S_0)$ is the root and a leaf containing $s_0$.

Assume $S_N \in S \setminus S_{N-1} \neq \emptyset$. In order to create $T_N = g_n(S_N)$ with $S_N = S_{N-1} \cup \{ s_N \}$, let $v_N$ be the nearest vertex of $T_{N-1}$ on the path from root to the end $s_N$ in the infinite $p$-adic Gray-Hilbert tree. This vertex can be found by tracing the coefficients in the formal power series representing $s$ until a coefficient is found which has a digit not being a label of an edge emanating from a vertex in $T_{N-1}$. The next vertex is a leaf containing $s_N$.

Continue as in the previous step until $S_N = S$.

The insert algorithm is as follows:

Algorithm 9.5 (Insert). Given a finite subset $S \subset \mathbb{F}_p^n[[t]]$ and a partition
\[ S = S_1 \cup \ldots \cup S_N \]
as obtained from Algorithm 9.3 and some $f \in \mathbb{F}_p^n[[t]]$. Then find all $s \in S$ such that
\[ |f - s|_p \]
is smallest possible. This means
\[ f \equiv s \mod t^k, \quad f \not\equiv s \mod t^{k+1} \]
and $k$ is largest possible in $\{ 1, \ldots, N \}$. If $s$ is unique, then for all $s' \in S \setminus \{ s \}$:
\[ |s - s'|_p = |s - f + f - s'|_p \overset{(*)}{=} |f - s'|_p = p^{-\ell} \]
with $\ell < k$, where $(*)$ holds true because of the non-archimedean property [5, Ch. 2]. Hence, $s \mod t^\ell \in S_j$ with $j \leq k$. So, update $S_{k+1}$ and $S_j$:
\[ S_{k+1} \leftarrow S_{k+1} \cup \{ f, s \} \mod t^{k+1} \]
\[ S_j \leftarrow S_j \setminus \{ s \mod j \} \]
Now, update the linear ordering with the help of $\rho_k$ as in Algorithm 9.3.

If $s$ is not unique, then for $s' \neq s$ another such nearest neighbour of $f$ it holds true that:
\[ |s - s'|_p = |s - f + f - s'|_p \leq p^{-k} \]
Hence, $s \in S_j$ with $j > k$. So, update $S_{k+1}$:
\[ S_{k+1} \leftarrow S_{k+1} \cup \{ f \mod t^k \} \]
and update the linear ordering again with $\rho_k$ as in Algorithm 9.3.
9.5 Upper complexity bounds

**Complexity of finding a point on** $h^{k+1}$. The time complexity of computing $h^{k+1}(x_k)$ equals $k$ times the time complexity of computing $gc_{c,d}^{e,n}$ for $n$-bit binary numbers. As $gc_{c,d}^{e,n}$ consists of a bounded number of bit-wise additions, shifts and swaps, independent of $n$, its time complexity is in $O(n)$. Hence, the time complexity of $h^{k+1}(x_k)$ is in $O(kn)$.

**Complexity of indexing algorithm.** For the indexing method (Algorithm 9.3), the inverse Gray code $gc_n^{-1}$ needs to be computed. This is $n$ times the complexity of $n$-bit operations, so it is in $O(n^2)$. So, the time complexity of finding the image of a point under $(h^k)^{-1}$ is in $O(kn^2)$. If the points of the finite set $S \subset F_2^n$ are all distinguishable modulo $t^{k+1}$, then in the worst case, $k$ iterations are needed. As all points of $S$ need to be taken into account, it follows that the overall time complexity is in $O(|S|kn^2)$.

**Complexity of Gray-Hilbert tree of a set.** For computing the Gray-Hilbert tree of a finite set $S \subset F_2^n$, all elements of $S$ need to be searched, and in the worst case, $S_n(S)$ is a finite $p$-regular tree (with $S$ leaf nodes). This gives an upper complexity bound of $O(|S|\log |S|)$.

**Complexity of Insert.** Also here, the inverse Gray code needs to be computed, and all elements of $S \subset F_2^n$ need to be considered. This means that the time complexity of Algorithm 9.5 is in $O(|S|kn^2)$, as also here, an image under $(h^k)^{-1}$ is needed.

10 An implementation of a dynamic 2-adic scaled Hilbert indexing method

A binary tree has been chosen as basic data structure for the implementation of a dynamic 2-adic scaled Hilbert index. The Hilbert value is obtained by interpreting the movement to the first sub-node of a node as false/zero and the movement to the second sub-node as true/one. A binary number is built up bit by bit with every step up the tree. This binary number serves us as the Hilbert value of the final leaf node.

Starting from root node with $n$ as the dimension, $n$ steps are needed to fulfil one complete Hilbert curve iteration process. The standard Hilbert curve index deals with a constant number of Hilbert curve iterations $k$. Therefore, every leaf node of the resulting binary tree is on the same level. The tree will contain $\ell = k \cdot n$ levels and $2^\ell$ leaf nodes. The hyper-cuboids of the leaf nodes are equal in size and the tree will have under- and overfilled leaf nodes, if nodes are viewed as ‘buckets’ which may contain up to a certain amount $s$ of data points; if there are more than $s$ points in a ‘bucket’, then the vertex splits, unless the prescribed maximal number of levels is reached. So, if the data is clustered in a certain way, then there will be leaf nodes containing more than $s$ points, while other leaf nodes contain fewer or even none of the points.

The tree covers a maximal $n$-dimensional hyper-cuboid. Each indexed object has a representation as an object in an $n$-dimensional space. In order to insert,
delete or retrieve objects, the tree must be queried from root to leaf node. One step along a branch splits the hyper-cuboid of a node into two parts by cutting orthogonally to a splitting coordinate through the centre of the node's hyper-cuboid. For each step up the tree, it has to be decided which splitting coordinate should be taken and how to map the two hyper-cuboid parts to the two sub-nodes. There are two cases for how to map the bit of the Hilbert value to the hyper-cuboid part it belongs to. The first case maps the lower part of the hyper-cuboid to the first sub-node (Algorithm 10.1 line 8) and the upper part of the hyper-cuboid to the second sub-node (Algorithm 10.1 line 9). The second case does the same vice versa (Algorithm 10.1 lines 5 and 6). The Gray code defines which cases have to be chosen within one Hilbert curve iteration process.

How to split a hyper-cuboid within one Hilbert curve iteration process also depends on the path taken within the previous Hilbert curve iteration process. To connect the ends of each sub-hyper-cuboids with the start of the next neighbouring sub-hyper-cuboid on the next Hilbert curve iteration correctly, it is necessary to know if the start or end should lie in the lower or upper part of the sub-hyper-cuboid regarding each coordinate. It is also important which splitting coordinate comes first in order to rotate the sub-hyper-cuboid correctly. The order of the following coordinates does not matter when building a space filling curve (Corollary 5.2).

The following implementation uses a list of coordinate identifiers and pulls up the first coordinate $d$ to the front of the list for the next Hilbert curve iteration process using the following permutation:

\[
\begin{pmatrix}
  n - 1 & n - 2 & \ldots & d & d - 1 & \ldots & 1 & 0 \\
  d & n - 1 & \ldots & d + 1 & d - 1 & \ldots & 1 & 0
\end{pmatrix}
\]

and call this Gray-Hilbert curve bubble.

We also use a ring of coordinate identifiers and point to the coordinate $d$ to start with at each start of a Hilbert curve iteration process, using the following permutation:

\[
\begin{pmatrix}
  n - 1 & n - 2 & \ldots & n - (d + 1) & \ldots & 0 \\
  d & d - 1 & \ldots & 0 & \ldots & d + 1
\end{pmatrix}
\]

and call this curve ring.

The developed dynamic variant of the Hilbert index deals with a maximal number of objects $s$ contained within each leaf node. It dynamically splits a leaf node if the maximal number is exceeded, or collapses a leaf node if no objects are left within the leaf node. But if the prescribed maximal number of levels is reached, no splitting occurs, and then there may be overfilled leaves containing more than $s$ objects.

This mechanism takes place on any level of the tree. Therefore, each object is not indexed on a certain Hilbert curve iteration level.

**Algorithm 10.1.** Input. A tree $T$ covering an $n$-dimensional hyper-cube $C$ and a point $x \in C$

Output. The leaf node $\nu$ where $x$ lies in sub-hypercube $c$.

Pseudo-code of the routine `FINDNODE(T, C, x)`:

26
01. $\text{nu} := \text{root}(T), \ c := C, \ h := \text{false},$
    $\ S := \text{create a list of coord id's},$
    $\ a := S.\text{start}, \ a' := S.\text{end},$
    $\ a'' := S.\text{end}, \ e := \text{null}$
02. while nu is not a leaf
03.   if $h$
04.     then 
05.         { if $x$ in lower part of $c$
06.             \ $\nu := \nu.\text{right}, \ a' := a,$
07.             \ $\ c := \text{lower part of } c \ \text{resp. } a,$
08.         } else 
09.             { $\nu := \nu.\text{left}, \ a'' := a,$
10.                 \ $\ c := \text{upper part of } c \ \text{resp. } a,$
11.                 \ $\ h := \text{false}$
12.         } else 
13.     { if $x$ in lower part of $c$
14.         \ $\nu := \nu.\text{left}, \ a' := a,$
15.         \ $\ c := \text{lower part of } c \ \text{resp. } a,$
16.     } else 
17.     { $\nu := \nu.\text{right}, \ a'' := a,$
18.         \ $\ c := \text{upper part of } c \ \text{resp. } a,$
19.         \ $\ h := \text{true}$
20.     } if $S.\text{hasNext}$
21.     then $a := S.\text{next}, \ a$ equals $e$
22.     else { if $x$ was lower part of $c$
23.         \ $\nu := \nu.\text{right}, \ a' := a,$
24.         \ $\ c := \text{lower part of } c \ \text{resp. } a,$
25.     } else 
26.         { if $x$ was upper part of $c$
27.             \ $\nu := \nu.\text{left}, \ a'' := a,$
28.             \ $\ c := \text{upper part of } c \ \text{resp. } a,$
29.             \ $\ h := \text{true}$
30.     } else if $h$
31.         then 
32.             { if $x$ was in upper part of $c$
33.                 \ $\nu := \nu.\text{right}, \ a' := a,$
34.                 \ $\ c := \text{upper part of } c \ \text{resp. } a,$
35.             } or 
36.             { if $x$ was in lower part of $c$
37.                 \ $\nu := \nu.\text{left}, \ a'' := a,$
38.                 \ $\ c := \text{lower part of } c \ \text{resp. } a,$
39.             } end if
40.     end if
41.     if $h$
42.         then $h := \text{true}$
43.     else $h := \text{false}$
44. end while
45. return $\nu$
11 Storage efficiency of the $p$-adic scaled Gray-Hilbert tree

It is needless to say that the $p$-adic scaled Gray-Hilbert tree for a data set $S \subseteq \mathbb{R}^n$ is storage efficient, because it is the smallest sub-tree of the Gray-Hilbert tree containing $S$ as its leaf nodes. The aim of this section is to compare the storage properties of the scaled Gray-Hilbert tree with the static $p$-adic Gray-Hilbert index.

Given some multi-dimensional point cloud $S$, the nodes of the Gray-Hilbert tree are viewed as buckets which are either empty, filled, underfilled, or overfilled, respectively, depending on if the corresponding hypercuboid contains no, a specified number, less than, or more than a pre-specified number $s$ of data points from $S$. This number is called bucket capacity, and we assume that $s < |S|$, where $|S|$ denotes the cardinality of $S$.

We define the following quantities

$$\Omega(T, S) = (1 + \omega(T, S)) \times |L(T)|$$

$$\omega(T, S) = \frac{\# \text{ overfilled leaf nodes of } T}{\# \text{ non-empty leaf nodes of } T}$$

where

$$L(T) = \{ \text{leaf nodes of } T \}$$

and $T$ is a finite sub-tree of the Gray-Hilbert tree $\mathcal{G}_n$.

The quantity $\Omega(T, S)$ is a measure incorporating the capacity of $T$ (number of leaf nodes) and a factor $1 + \omega(T, S)$ where $\omega(T, S)$ is the number of leaf node splittings which would occur for $S$, if the depth of the tree $T$ were not bounded. Multiplying with this factor means virtually creating space for data which everywhere would have the same overfilling property as $S$. In a sense, this would mean to transfer the average local density of $S$ within the occupied part of the space to the whole hyper-cube. We call $\Omega(T, S)$ the capacity of $T$ for $S$.

Given two trees $T_1$ and $T_2$ for indexing $S$, the capacity ratio is defined as

$$R(T_1, T_2; S) = \frac{\Omega(T_1, S)}{\Omega(T_2, S)}$$

This will be applied to the scaled and an optimal static index for $S$.

We will write

$$T_{\text{scaled}} = \mathcal{G}_n(S)$$

for the Gray-Hilbert tree generated by $S$. The tree

$$T_{\text{static}}$$

is defined as the finite Gray-Hilbert tree with maximal iteration number

$$k = \left\lfloor \log_p \frac{|S|}{s} \right\rfloor$$
This is motivated by wanting to have
\[ p^{nk} \approx \frac{|S|}{s} \]
as the number of leaf nodes in \( T_{\text{static}} \). This is the smallest sub-tree of the \( p \)-adic Gray-Hilbert tree such that on average no leaf node is overfilled.

In order to compare the scaled and the static Gray-Hilbert curve indices, we consider the ratio
\[
R_p(S) = R(T_{\text{scaled}}, T_{\text{static}}; S) = \frac{\Omega(T_{\text{scaled}}, S)}{\Omega(T_{\text{static}}, S)}
\]
which we call the \( p \)-adic Gray-Hilbert capacity ratio for \( S \). If \( R_p(S) < 1 \), then the scaled index is more storage efficient than the static index, otherwise not.

In order to understand the behaviour of \( R_p(S) \) for increasing dimension, first consider the maximal iteration number \( k \), which can be written as
\[
k = \frac{\log_p |S|}{s} + \epsilon_p
\]
with \( \epsilon_n \in [0, 1) \). We have
\[
\frac{1}{2} p^{-n\epsilon_p} = \frac{|S|/s}{2 p^{nk}} \leq R_p(S) \leq \frac{|S| - s + 1}{p^{nk}} \leq s \cdot \frac{|S|/s}{p^{nk}} = s p^{-n\epsilon_p}
\]
because
\[
|S|/s \leq |L(T_{\text{scaled}})| \leq |S| - s + 1
\]
From this, it follows that
\[
R_p(S) \to 0 \quad (12)
\]
for \( n \to \infty \), as
\[
\lim_{n \to \infty} \epsilon_p = 1
\]
In particular, if the dimension \( n \) is sufficiently high, then \( R_p(S) \leq 1 \). Namely, if
\[
\epsilon_p \geq \frac{\log_p s}{n} \quad (13)
\]
then this is the case. So, if (13) is satisfied, then the scaled \( p \)-adic Gray-Hilbert index index performs better then the best static \( p \)-adic Gray-Hilbert index, the latter being optimal in the sense that on average, no leaf node is overfilled. We summarise this as follows:

**Theorem 11.1.** If \( \frac{\log_p s}{n} < \epsilon_p \) holds true, then \( T_{\text{scaled}} \) is more storage efficient than \( T_{\text{static}} \), and that relative efficiency increases with increasing dimension \( n \). For fixed \( n \), this is the case, if the bucket capacity \( s \) is sufficiently small.
encoding & dim \\
original & 4 \\
complete disjunctive form (cdf) & 400 \\
cdf trimmed & 123 \\
BCH 127 50 27 & 431 \\

Table 1: Different encodings of the iris data set and their dimensions.

The inequalities \((11)\) are equivalent to

\[
0 \leq -\log_p R_p(S) - n\epsilon_p + \log_p s \leq \log_p(2s)
\]

so that in order to have a measure between zero and one, we define the quantity

\[
\rho_p(S, s) = -\frac{\log_p(R_p(S)/s - n\epsilon_p)}{\log_p(2s)} \in [0, 1]
\]

We call this the \(p\)-adic Gray-Hilbert local sparsity measure for \(S\) and \(s\). It holds true that

\[
(2s)^{\rho_p(S, s)} = \frac{|S|}{L(T_{scaled})} (1 + \omega(T_{static}))
\]

Observe that the dependence on \(p\) is through \(T_{static}\) and \(T_{scaled}\), while the dependence on \(s\) is through the definition of a leaf being overfilled in \(\omega(T_{static}, S)\) and \(L(T_{scaled})\). This means that for fixed bucket capacity \(s\), the quantity \(\rho_p(S, s)\) should depend on the distribution of \(S\) only, the question being in which way.

In any case, a value of \(\rho_p(S, s)\) near one means that very many leaf nodes in the \(T_{static}\) are overfilled, possibly because of local non-sparsity, whereas in the case of \(\rho_p(S, s)\) near zero, only few leaf nodes of \(T_{static}\) are overfilled, possibly because of a higher local sparsity of the distribution of \(S\).

Experiments were done in the 2-adic case with various different data sets. Namely, the iris data set \([6]\) in four different encodings, as well as randomly generated data.

The iris data set consists of 150 points. The four different encodings of this data set are listed in Table 1. The complete disjunctive form (cdf) and BCH 127 50 27 contain only 0 and 1 values. The latter is an error-correcting code named after Hocquenghem, Bose and Ray-Chaudhuri \([13, 1]\). This encoding of the iris data set was also used in \([3]\), whereas its complete disjunctive form was studied in \([17]\). The data set named cdf trimmed is obtained by projecting out all coordinates containing only 0 values.

The random point clouds are 100,000 samples taken from \(\mathbb{R}^3\) uniformly and under a standard normal distribution, repeated 100 times.

Table 2 shows the Gray-Hilbert local sparsity measure for the case \(p = 2\) with the different encodings of the iris data set using bubble and ring (as defined in Section [10]). Both methods yield the same values. It was observed that the high-dimensional encodings yield no overfilled leaf nodes in \(T_{static}\). This can be seen as an indicator that in these cases the data could be locally sparse, which agrees well with the ultrametricity property found in \([17]\). The case \(s = 1\) reflects the fact that \(\omega(T_{static}) = 0\) in that case, as then the number of leaf nodes of \(T_{scaled}\) equals the number of points in \(S\).
Table 2: 2-adic Gray-Hilbert local sparsity measures for different encodings of the iris data set using the methods bubble and ring. Both methods yield the same values.

| s  | original bubble | original ring | cdf trimmed | cdf | BCH |
|----|-----------------|---------------|-------------|-----|-----|
| 1  | 0.79            | 0.79          | 0.00        | 0.00| 0.00|
| 2  | 0.61            | 0.63          | 0.22        | 0.22| 0.22|
| 4  | 0.59            | 0.58          | 0.36        | 0.36| 0.36|
| 8  | 0.59            | 0.58          | 0.52        | 0.52| 0.52|
| 16 | 0.68            | 0.68          | 0.48        | 0.48| 0.48|
| 32 | 0.72            | 0.72          | 0.44        | 0.44| 0.44|
| 64 | 0.80            | 0.80          | 0.42        | 0.42| 0.42|

Table 3: The number of leaf nodes in $T_{scaled}$ for different encodings of the iris data set. The methods bubble and ring yield different results only for the original data set.

| s  | original bubble | original ring | cdf trimmed | cdf | BCH |
|----|-----------------|---------------|-------------|-----|-----|
| 1  | 147             | 147           | 147         | 147| 147|
| 2  | 99              | 97            | 108         | 108| 108|
| 4  | 56              | 57            | 70          | 70 | 70  |
| 8  | 32              | 33            | 35          | 35 | 35  |
| 16 | 20              | 20            | 28          | 28 | 28  |
| 32 | 9               | 9             | 24          | 24 | 24  |
| 64 | 3               | 3             | 19          | 19 | 19  |
Table 4: 2-adic Gray-Hilbert local sparsity measures for 100,000 random points in the unit cube from a uniform or standard normal distribution.

Table 3 shows that the number of leaf nodes in the 2-adic scaled tree is higher for the high-dimensional encodings of the iris data set than for the original data, if the bucket capacity $s$ is greater than one. In fact, the trees $T_{scaled}$ are isomorphic for the three high-dimensional encodings.

Table 4 gives the values of $\rho_2(S, s)$ for the random samples from 100,000 points in the unit cube from a uniform or standard normal distribution. For small bucket capacity values $s$, the uniform distribution has a lower value than the normal distribution, but as $s$ increases, it becomes locally less sparse than the normal distribution.

### 12 Conclusion

The $p$-adic generalisation of higher-dimensional Hilbert curves can be accomplished by using the $p$-adic Gray code. Restricting to affine transformations, which are in this case coordinate permutations plus translations, allows the mathematical construction of a plethora of well-defined space-filling curves called $p$-adic affine Gray-Hilbert curves. Each of these can be used for indexing $n$-dimensional point clouds by taking the smallest sub-tree of the tree associated with a $p$-adic Gray-Hilbert curve, whose leaf nodes correspond to hypercuboids containing the point cloud. This is the scaled $p$-adic Gray-Hilbert index. It has efficient access and insert algorithms and can be shown to be more storage efficient than the optimal static version of the $p$-adic Gray-Hilbert index tree, if the dimension is sufficiently high or, equivalently, the bucket capacity is chosen sufficiently low. Optimality is defined here as giving rise to the smallest finite $p$-regular tree containing the data in its leaf nodes in such a way that on average no leaf node is overfilled. The derived $p$-adic local sparsity measure thus contains information about the local distribution of the data set. This has been tested for different encodings of the famous iris data set with the result that the high-dimensional encodings seem to be locally uniform, which agrees with previous findings about their discovered ultrametricity. Potential applications for the index structures constructed here should be seen in the context of massive...
data in any finite dimension.

**Funding**

This work is supported by the Deutsche Forschungsgemeinschaft (DFG) [grant numbers BR 3513/12-1, BR 2128/18-1].

**Acknowledgement**

Norbert Paul is thanked for fruitful discussions.

**References**

[1] R. Bose and D. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1):68–79, 1960.

[2] P.E. Bradley. Mumford dendrograms. *The Computer Journal*, 53(4):393–404, 2010.

[3] P.E. Bradley. Finding ultrametricity in data using topology. *Journal of Classification*, 34:76–84, 2017.

[4] G Cantor. Ein Beitrag zur Mannigfaltigkeitslehre. *Journal für die reine und angewandte Mathematik*, 84(84):242–258, 1878.

[5] T. Diagana and F. Ramaroson. *Non-Archimedean Operator Theory*. SpringerBriefs in Mathematics. Springer, Cham, 2016.

[6] R. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[7] F. Gray. Pulse code communication. US Patent Number 2,632,058, 1953.

[8] D.-J. Guan. Generalized Gray codes with applications. *Proc. Natl. Sci. Counc. ROC(A)*, 22(6):841–848, 1998.

[9] C. Hamilton. Compact Hilbert indices. Technical Report CS-2006-07, Faculty of Computer Science, Halifax, Nova Scotia, Canada, 2006.

[10] C.H. Hamilton and A. Rau-Chaplin. Compact Hilbert indices for multidimensional data. In *Proceedings of the First International Conference on Complex, Intelligent and Software Intensive Systems (CISIS’07)*. IEEE, 2007.

[11] H. Haverkort. How many three-dimensional Hilbert curves are there? *Journal of Computational Geometry*, 8(1):206–281, 2017.

[12] D. Hilbert. Über die stetige Abbildung einer Linie auf ein Flächenstück. *Mathematische Annalen*, 38:459–460, 1891.

[13] A. Hocquenghem. Codes correcteurs derreurs. *Chiffres*, 2:147–156, 1959.
[14] N. Jacobson. *Basic Algebra*, volume 2. W.H. Freeman and Company, 2 edition, 1989.

[15] M.W. Jahn and P.E. Bradley. A scaled space-filling curve index applied to tropical rain forest tree distributions. Work in progress.

[16] D.E. Knuth. A draft of Section 7.2.1.1: Generating all $n$-tuples. In *The Art of Computer Programming*, volume 4A: Enumeration and Backtracking, pre-fascicle 2a. Addison-Wesley, 2004.

[17] F. Murtagh. Identifying and exploiting ultrametricity. In R. Decker and H.-J. Lenz, editors, *Advances in Data Analysis*, pages 263–272, 2007.

[18] G. Peano. Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen*, 36(1):157–160, 1890.

[19] A.J. van Zanten and I.N. Suparta. The separability of standard cyclic $n$-ary Gray codes. *IEEE Transactions on Information Theory*, 49(2), 2003.