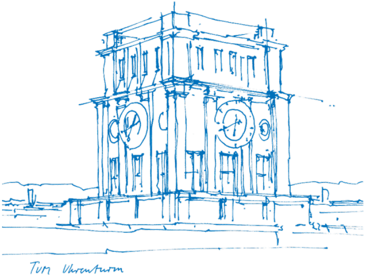
Algorithms for Scientific Computing

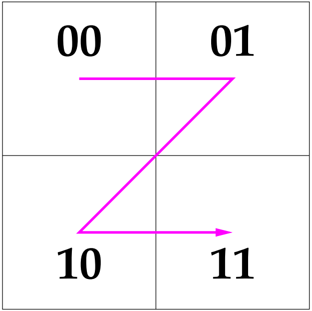
Space-Filling Curves

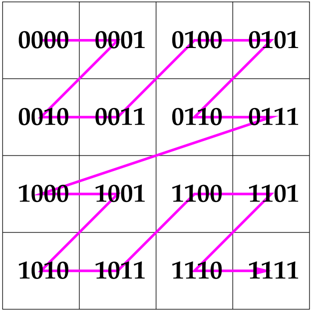
Michael Bader
Technical University of Munich

Summer 2016



Start: Morton Order / Cantor's Mapping





Questions:

- Can this mapping lead to a **contiguous** "curve"?
- i.e.: Can we find a **continuous** mapping?
- and: Can this continuous mapping fill the entire square?

Morton Order and Cantor's Mapping

Georg Cantor (1877):

$$0.01111001\ldots \rightarrow \begin{pmatrix} 0.0110\ldots \\ 0.1101\ldots \end{pmatrix}$$

- **bijective** mapping $[0,1] \rightarrow [0,1]^2$
- proved identical cardinality of $[0,1]$ and $[0,1]^2$
- provoked the question: is there a **continuous** mapping? (i.e. a curve)

History of Space-Filling Curves

1877: Georg Cantor finds a bijective mapping from the unit interval $[0,1]$ into the unit square $[0,1]^2$.

1879: Eugen Netto proves that a **bijective** mapping $f\colon \mathcal{I} \to \mathcal{Q} \subset \mathbb{R}^n$ can not be continuous (i.e., a curve) at the same time (as long as $\mathcal{Q}$ has a smooth boundary).

1886: rigorous definition of **curves** introduced by Camille Jordan

1890: Giuseppe Peano constructs the first space-filling curves.

1890: Hilbert gives a geometric construction of Peano's curve; and introduces a new example – the Hilbert curve

1904: Lebesgue curve

1912: Sierpinski curve

Part I

Space-Filling Curves

What is a Curve?

Definition (Curve)

As a **curve**, we define the image $f_*(\mathcal{I})$ of a *continuous* mapping $f : \mathcal{I} \to \mathbb{R}^n$.
$x = f(t), t \in \mathcal{I}$, is called **parameter representation** of the curve.

With:

- $\mathcal{I} \subset \mathbb{R}$ and $\mathcal{I}$ is compact, usually $\mathcal{I} = [0, 1]$.
- the **image** $f_*(\mathcal{I})$ of the mapping f_* is defined as $f_*(\mathcal{I}) := \{f(t) \in \mathbb{R}^n \mid t \in \mathcal{I}\}$.
- $\mathbb{R}^n$ may be replaced by any Euklidian vector space (norm & scalar product required).

What is a Space-filling Curve?

Definition (Space-filling Curve)

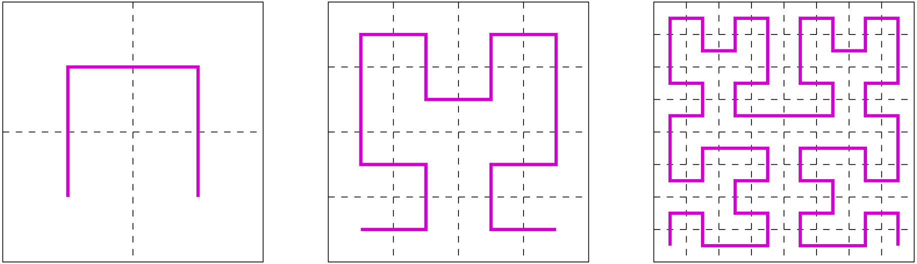
Given a mapping $f\colon \mathcal{I} \to \mathbb{R}^n$, then the corresponding curve $f_*(\mathcal{I})$ is called a **space-filling curve**, if the Jordan content (area, volume, ...) of $f_*(\mathcal{I})$ is larger than 0.

Comments:

- assume $f\colon \mathcal{I} \to \mathcal{Q} \subset \mathbb{R}^n$ to be **surjective** (i.e., every element in $\mathcal{Q}$ occurs as a value of f);

  then, $f_*(\mathcal{I})$ is a space-filling curve, if the area (volume) of $\mathcal{Q}$ is positive.
- if the domain $\mathcal{Q}$ has a smooth boundary, then there can be **no bijective mapping** $f\colon \mathcal{I} \to \mathcal{Q} \subset \mathbb{R}^n$, such that $f_*(\mathcal{I})$ is a space-filling curve (theorem: E. Netto, 1879).

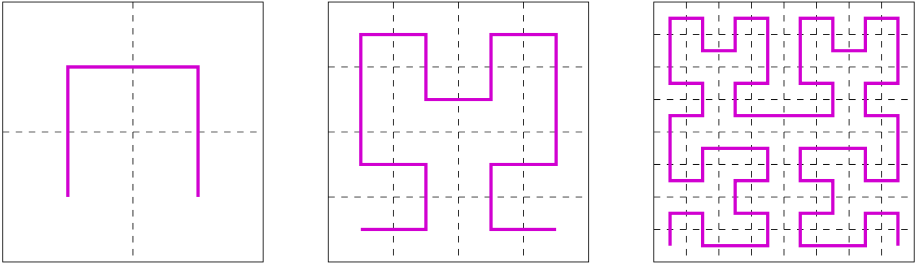
Remember: Construction of the Hilbert Order



Incremental construction of the Hilbert order:

- start with the basic pattern on 4 subsquares
- combine four numbering patterns to obtain a twice-as-large pattern
- proceed with further iterations

Remember: Construction of the Hilbert Order



Recursive construction of the Hilbert order:

- start with the basic pattern on 4 subsquares
- for an existing grid and Hilbert order:
  split each cell into 4 congruent subsquares
- order 4 subsquares following the rotated basic pattern,
  such that a contiguous order is obtained

Definition of the Hilbert Curve's Mapping

Definition: (Hilbert curve)

- each parameter $t \in \mathcal{I} := [0, 1]$ is contained in a sequence of intervals

$$\mathcal{I} \supset [a_1, b_1] \supset \ldots \supset [a_n, b_n] \supset \ldots,$$

  where each interval results from a division-by-four of the previous interval.
- each such sequence of intervals can be uniquely mapped to a corresponding sequence of 2D intervals (subsquares)
  $\rightarrow$ "uniquely mapped" based on grammar for Hilbert order
- the 2D sequence of intervals converges to a unique point q in $q \in \mathcal{Q} := [0, 1] \times [0, 1]$ – q is defined as $h(t)$.

Theorem

$h : \mathcal{I} \rightarrow \mathcal{Q}$ defines a space-filling curve, the ***Hilbert curve****.*

Proof: h defines a Space-filling Curve

We need to prove:

- h is a mapping, i.e. each $t \in \mathcal{I}$ has a **unique** function value $h(t) \to$ OK, if $h(t)$ is independent of the choice of the sequence of intervals (see next chapter)
- $h \colon \mathcal{I} \to \mathcal{Q}$ is **surjective**:
  - for each point $q \in \mathcal{Q}$, we can construct an appropriate sequence of 2D-intervals
  - the 2D sequence corresponds in a unique way to a sequence of intervals in $\mathcal{I}$ – this sequence defines an original value of q
    $\Rightarrow$ every $q \in \mathcal{Q}$ occurs as an image point.
- h is **continuous**

Continuity of the Hilbert Curve

A function $f\colon \mathcal{I} \to \mathbb{R}^n$ is uniformly **continuous**, if

for each $\epsilon > 0$

a $\delta > 0$ exists, such that

for all $t_1, t_2 \in \mathcal{I}$ with $|t_1 - t_2| < \delta$, the following inequality holds:

$\|f(t_1) - f(t_2)\|_2 < \epsilon$

Strategy for the proof:
For any given parameters t_1, t_2, we compute an estimate for the distance $\|h(t_1) - h(t_2)\|_2$ (functional dependence on $|t_1 - t_2|$).
$\Rightarrow$ for any given ϵ, we can then compute a suitable δ

Continuity of the Hilbert Curve (2)

- given: $t_1, t_2 \in \mathcal{I}$; choose an n, such that $|t_1 - t_2| < 4^{-n}$
- in the n-th iteration of the interval sequence, all interval are of length 4^{-n} $\Rightarrow [t_1, t_2]$ overlaps at most two neighbouring(!) intervals.
- due to construction of the Hilbert curve, the values $h(t_1)$ and $h(t_2)$ will be in neighbouring subsquares with face length 2^{-n}.
- the two neighbouring subsquares build a rectangle with a diagonal of length $2^{-n} \cdot \sqrt{5}$;
  therefore: $\|h(t_1) - h(t_2)\|_2 \leq 2^{-n}\sqrt{5}$

For a given $\epsilon > 0$, we choose an n, such that $2^{-n}\sqrt{5} < \epsilon$.
Using that n, we choose $\delta := 4^{-n}$; then, for all t_1, t_2 with $|t_1 - t_2| < \delta$, we get:
$\|h(t_1) - h(t_2)\|_2 \leq 2^{-n}\sqrt{5} < \epsilon$. Which proves the continuity!

Part II

Arithmetisation of Space-Filling Curves

Space-filling Orders – Required Algorithms

Traversal of h-indexed objects:

- given a set of objects with "positions" $p_i \in \mathcal{Q}$
- traverse all objects, such that $\bar{h}^{-1}(p_{i_0}) < \bar{h}^{-1}(p_{i_1}) < \ldots$
- solved by **grammar representation**

Compute mapping:

- for a given index $t \in \mathcal{I}$, compute the image $h(t)$

Compute the index of a given point:

- given $p \in \mathcal{Q}$, find a parameter t, such that $h(t) = p$
- problem: inverse of h is not unique (h not bijective!)
- define a "technically unique" inverse mapping $\bar{h}^{-1}$

Mapping and index computation required for random access to a data structure!

Arithmetic Formulation of the Hilbert Curve

Idea:

- interval sequence within the parameter interval $\mathcal{I}$ corresponds to a **quaternary representation**; e.g.:

$$\left[\tfrac{1}{4}, \tfrac{2}{4}\right] = [0_4.1, 0_4.2]\,, \quad \left[\tfrac{3}{4}, 1\right] = [0_4.3, 1_4.0]$$

- **self-similarity**: every subsquare of the target domain contains a scaled, translated, and rotated/reflected Hilbert curve.

$\Rightarrow$ **Construction** of the arithmetic representation:

- find quaternary representation of the parameter
- use quaternary coefficients to determine the required sequence of operations

Arithmetic Formulation of the Hilbert Curve (2)

Recursive approach:

$$h(0_4.q_1q_2q_3q_4\ldots) = H_{q_1} \circ h(0_4.q_2q_3q_4\ldots)$$

- $\tilde{t} = 0_4.q_2q_3q_4\ldots$ is the relative parameter in the subinterval $[0_4.q_1, 0_4.(q_1+1)]$
- $h(\tilde{t}) = h(0_4.q_2q_3q_4\ldots)$ is the relative position of the curve point in the subsquare
- H_{q_1} transforms $h(\tilde{t})$ to its correct position in the unit square:
  - rotation
  - translation
- expanding the recursion equation leads to:

$$h(0_4.q_1q_2q_3q_4\ldots) = H_{q_1} \circ H_{q_2} \circ H_{q_3} \circ H_{q_4} \circ \cdots$$

Arithmetic Formulation of the Hilbert Curve (3)

If t is given in quaternary digits, i.e. $t = 0_4.q_1 q_2 q_3 q_4 \ldots$,
then $h(t)$ may be represented as

$$h(0_4.q_1 q_2 q_3 q_4 \ldots) = H_{q_1} \circ H_{q_2} \circ H_{q_3} \circ H_{q_4} \circ \cdots \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

using the following operators:

$$H_0 := \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} \frac{1}{2} y \\ \frac{1}{2} x \end{pmatrix} \qquad H_1 := \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} \frac{1}{2} x \\ \frac{1}{2} y + \frac{1}{2} \end{pmatrix}$$

$$H_2 := \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} \frac{1}{2} x + \frac{1}{2} \\ \frac{1}{2} y + \frac{1}{2} \end{pmatrix} \qquad H_3 := \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} -\frac{1}{2} y + 1 \\ -\frac{1}{2} x + \frac{1}{2} \end{pmatrix}$$

Matrix Form of the Operators $H_0, \ldots, H_3$

In matrix notation, the operators $H_0, \ldots, H_3$ are:

$$H_0 := \begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \qquad H_1 := \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{2} \end{pmatrix}$$

$$H_2 := \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} \qquad H_3 := \begin{pmatrix} 0 & -\frac{1}{2} \\ -\frac{1}{2} & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 1 \\ \frac{1}{2} \end{pmatrix}$$

Governing operations:

- scale with factor $\frac{1}{2}$
- translate start of the curve, e.g. $+\begin{pmatrix} 0 \\ \frac{1}{2} \end{pmatrix}$
- reflect at x and y axis (for H_3)

A First Comment Concerning Uniqueness

Question:

Are the values $h(t)$ independent of the choice of quaternary representation of t concerning trailing zeros:

$$h(0_4.q_1 \dots q_n) = h(0_4.q_1 \dots q_n 000 \dots),$$

Outline of the proof:

1. compute the limit $\lim_{n\to\infty} H_0^n$, or $\lim_{n\to\infty} H_0^n \begin{pmatrix} x \\ y \end{pmatrix}$;
   Result: $\lim_{n\to\infty} H_0^n \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$
2. show: $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ is a fixpoint of H_0, i. e. $H_0 \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.

$\Rightarrow$ independence of trailing zeros, as H_{q_n} is applied to the fixpoint!

A Second Comment Concerning Uniqueness

Question:

Are the values $h(t)$ independent of the choice of quaternary representation of t, as in:

$$h(0_4.q_1 \dots q_n) = h(0_4.q_1 \dots q_{n-1}(q_n - 1)333\dots), \qquad q_n \neq 0$$

(if $q_n = 0$, then consider $0_4.q_1 \dots q_n = 0_4.q_1 \dots q_{n-1}$)

Outline of the proof:

1. compute the limits $\lim_{n\to\infty} H_0^n$ and $\lim_{n\to\infty} H_3^n$.
2. for $q_n = 1, 2, 3$, show that

$$H_{q_n} \circ \lim_{n\to\infty} H_0^n = H_{q_n - 1} \circ \lim_{n\to\infty} H_3^n$$

Algorithm to Compute the Hilbert Mapping

Task: given a parameter t, find $h(t) = (x, y) \in \mathcal{Q}$

Most important subtasks:

1. compute quaternary digits – use multiply by 4:

$$4 \cdot 0_4.q_1 q_2 q_3 q_4 \ldots = (q_1.q_2 q_3 q_4 \ldots)_4$$

   and cut off the integer part

2. apply operators H_q in the correct sequence – use recursion:

$$h(0_4.q_1 q_2 q_3 q_4 \ldots) = H_{q_1} \circ H_{q_2} \circ H_{q_3} \circ H_{q_4} \circ \cdots \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

3. stop recursion, when a given tolerance is reached
   $\Rightarrow$ track size of interval or set number of digits

Implementation of the Hilbert Mapping

Algorithm 1 *hilbert*(t, eps)

if $eps > 1$ **then**
    return $(0, 0)$
else
    $q \leftarrow floor(4 * t)$
    $r \leftarrow 4 * t - q$
    $(x, y) \leftarrow hilbert(r, 2 * eps)$
    switch q **do**
        case $q = 0$: **return** $(y/2, x/2)$
        case $q = 1$: **return** $(x/2, y/2 + 0.5)$
        case $q = 2$: **return** $(x/2 + 0.5, y/2 + 0.5)$
        case $q = 3$: **return** $(-y/2 + 1.0, -x/2 + 0.5)$
    end
end if

Computing the Inverse Mapping

Task: find a parameter t, such that $h(t) = (x, y)$ for a given $(x, y) \in \mathcal{Q}$

Problem: h not bijective; hence, t is not unique

$\Rightarrow$ a strict inverse mapping h^{-1} does not exist

$\Rightarrow$ instead, compute a "technically unique" inverse $\bar{h}^{-1}$

Recursive Idea:

- determine the subsquare that contains (x, y)
- transform (using the inverse operations of $H_0, \ldots, H_3$) the point (x, y) into the original domain $\rightarrow (\tilde{x}, \tilde{y})$
- recursively compute a parameter $\tilde{t}$ that is mapped to $(\tilde{x}, \tilde{y})$
- depending on the subsquare, compute t from $\tilde{t}$

Inverse Operators of $H_0, \ldots, H_3$

Example $\rightarrow$ compute inverse of operator H_0:

$$\begin{pmatrix} x \\ y \end{pmatrix} = H_0 \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\tilde{y} \\ \frac{1}{2}\tilde{x} \end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = \begin{pmatrix} 2y \\ 2x \end{pmatrix}$$

By similar computations:

$$H_0^{-1} \quad := \quad \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} 2y \\ 2x \end{pmatrix} \qquad H_1^{-1} \quad := \quad \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} 2x \\ 2y-1 \end{pmatrix}$$

$$H_2^{-1} \quad := \quad \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} 2x-1 \\ 2y-1 \end{pmatrix} \qquad H_3^{-1} \quad := \quad \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} -2y+1 \\ -2x+2 \end{pmatrix}$$

Algorithm to Compute the Inverse Mapping

$\bar{h}^{-1}$:=$\texttt{proc}(x, y)$

(1) determine the subsquare $q \in \{0, \ldots, 3\}$ by checking $x <> \frac{1}{2}$ and $y <> \frac{1}{2}$:

1	2
0	3

(treat cases $x, y = \frac{1}{2}$ in a unique way: either $<$ or $>$ $\Rightarrow$ *technically unique inverse*)

(2) set $(\tilde{x}, \tilde{y}) := H_q^{-1}(x, y)$

(3) recursively compute $\tilde{t} := \bar{h}^{-1}(\tilde{x}, \tilde{y})$

(4) return $t := \frac{1}{4}\left(q + \tilde{t}\right)$ as value

(stopping criterion still to be added)

Implementation of the Inverse Hilbert Mapping

Algorithm 2 *hilbertInverse*(x, y, eps)

```
if eps > 1 then return 0
if x <= 0.5 then
    if y <= 0.5 then
        return (0 + hilbertInverse(2 * y, 2 * x, 4 * eps))/4
    else
        return (1 + hilbertInverse(2 * x, 2 * y - 1, 4 * eps))/4
    end if
else
    if y <= 0.5 then
        return (3 + hilbertInverse(1 - 2 * y, 2 - 2 * x, 4 * eps))/4
    else
        return (2 + hilbertInverse(2 * x - 1, 2 * y - 1, 4 * eps))/4
    end if
end if
```