Algorithms of Scientific Computing

Space-Filling Curves

Michael Bader
Summer Term 2015
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 $f : \mathcal{I} \to Q \subset \mathbb{R}^n$ cannot be continuous (i.e., a curve) at the same time (as long as $Q$ 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 Euclidean vector space (norm & scalar product required).
What is a Space-filling Curve?

Definition (Space-filling Curve)

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

Comments:

- assume \( f : \mathcal{I} \rightarrow Q \subset \mathbb{R}^n \) to be surjective (i.e., every element in \( Q \) occurs as a value of \( f \));
  then, \( f^*(\mathcal{I}) \) is a space-filling curve, if the area (volume) of \( Q \) is positive.

- if the domain \( Q \) has a smooth boundary, then there can be no bijective mapping \( f : \mathcal{I} \rightarrow Q \subset \mathbb{R}^n \), such that \( f^*(\mathcal{I}) \) is a space-filling curve (theorem: E. Netto, 1879).
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
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) → “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

$h : \mathcal{I} \to \mathcal{Q}$ 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 $h(t)$
  $\rightarrow$ OK, if $h(t)$ is independent of the choice of the sequence of intervals (see next chapter)

- $h: \mathcal{I} \rightarrow \mathbb{Q}$ is surjective:
  - for each point $q \in \mathbb{Q}$, 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 $q \in \mathbb{Q}$ occurs as an image point.

- $h$ is continuous
A function $f: \mathcal{I} \rightarrow \mathbb{R}^n$ is uniformly \textbf{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:

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

\textbf{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 $\epsilon$, we can then compute a suitable $\delta$
Continuity of the Hilbert Curve (2)

- given: \( t_1, t_2 \in I \); choose an \( n \), such that \( |t_1 - t_2| < 4^{-n} \)
- in the \( n \)-th iteration of the interval sequence, all intervals 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
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.:

\[
\left[ \frac{1}{4}, \frac{2}{4} \right] = [0.4, 0.2], \quad \left[ \frac{3}{4}, 1 \right] = [0.3, 1.0]
\]

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

⇒ 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:

\[
h(0.4.q_1 q_2 q_3 q_4 \ldots) = H_{q_1} \circ h(0.4.q_2 q_3 q_4 \ldots)
\]

- \(\tilde{t} = 0.4.q_2 q_3 q_4 \ldots\) is the relative parameter in the subinterval 
  \([0.4.q_1, 0.4.(q_1 + 1)]\)
- \(h(\tilde{t}) = h(0.4.q_2 q_3 q_4 \ldots)\) 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:

\[
h(0.4.q_1 q_2 q_3 q_4 \ldots) = H_{q_1} \circ H_{q_2} \circ H_{q_3} \circ H_{q_4} \circ \ldots
\]
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:

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

$$H_2 := \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} \frac{1}{2} x + \frac{1}{2} \\ \frac{1}{2} y + \frac{1}{2} \end{pmatrix} \quad H_3 := \begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} -\frac{1}{2} y + 1 \\ -\frac{1}{2} x + \frac{1}{2} \end{pmatrix}$$
Matrix Form of the Operators $H_0, \ldots, H_3$

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

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

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

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

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

Governating 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:

$$h(0.4.q_1 \ldots q_n) = h(0.4.q_1 \ldots q_n000 \ldots),$$

Outline of the proof:

1. compute the limit $\lim_{n \to \infty} H^n_0$, or $\lim_{n \to \infty} H^n_0 \begin{pmatrix} x \\ y \end{pmatrix}$;

   Result: $\lim_{n \to \infty} H^n_0 \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

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:

\[
    h(0.4.q_1 \ldots q_n) = h(0.4.q_1 \ldots q_{n-1}(q_n - 1)333 \ldots), \quad q_n \neq 0
\]

(if \( q_n = 0 \), then consider \( 0.4.q_1 \ldots q_n = 0.4.q_1 \ldots q_{n-1} \))

Outline of the proof:

1. compute the limits \( \lim_{n \to \infty} H^n_0 \) and \( \lim_{n \to \infty} H^n_3 \).

2. for \( q_n = 1, 2, 3 \), show that

\[
    H_{q_n} \circ \lim_{n \to \infty} H^n_0 = H_{q_{n-1}} \circ \lim_{n \to \infty} H^n_3
\]
Algorithm to Compute the Hilbert Mapping

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

**Most important subtasks:**

1. compute quaternary digits – use multiply by 4:

   $4 \cdot 0.4q_1q_2q_3q_4\ldots = (q_1q_2q_3q_4\ldots)_4$

   and cut off the integer part

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

   $$h(0.4q_1q_2q_3q_4\ldots) = H_{q_1} \circ H_{q_2} \circ H_{q_3} \circ H_{q_4} \circ \ldots \left( \begin{array}{c} 0 \\ 0 \end{array} \right)$$

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)

1: if eps > 1 then
2:    return (0, 0)
3: else
4:    q ← floor(4 * t)
5:    r ← 4 * t – q
6:    (x, y) ← hilbert(r, 2 * eps)
7:    switch q do
8:      case q = 0: return (y/2, x/2)
9:      case q = 1: return (x/2, y/2 + 0.5)
10:     case q = 2: return (x/2 + 0.5, y/2 + 0.5)
11:     case q = 3: return (–y/2 + 1.0, –x/2 + 0.5)
12:    end
13:   end if
Computing the Inverse Mapping

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

**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 $\tilde{h}^{-1}$

**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 $H_0, \ldots, H_3$

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

By similar computations:

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

$$
H_2^{-1} := \begin{pmatrix}
  x \\
  y
\end{pmatrix}
\rightarrow
\begin{pmatrix}
  2x - 1 \\
  2y - 1
\end{pmatrix}
H_3^{-1} := \begin{pmatrix}
  x \\
  y
\end{pmatrix}
\rightarrow
\begin{pmatrix}
  -2y + 1 \\
  -2x + 2
\end{pmatrix}
$$
Algorithm to Compute the Inverse Mapping

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

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

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

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

(stopping criterion still to be added)
Implementation of the Inverse Hilbert Mapping

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

1: if eps > 1 then return 0
2: if x ≤ 0.5 then
3: if y ≤ 0.5 then
4: return (0 + *hilbertInverse*(2 * y, 2 * x, 4 * eps)/4)
5: else
6: return (1 + *hilbertInverse*(2 * x, 2 * y − 1, 4 * eps)/4)
7: end if
8: else
9: if y ≤ 0.5 then
10: return (3 + *hilbertInverse*(1 − 2 * y, 2 − 2 * x, 4 * eps)/4)
11: else
12: return (2 + *hilbertInverse*(2 * x − 1, 2 * y − 1, 4 * eps)/4)
13: end if
14: end if