Recurrence for Pandimensional Space-Filling Functions

Aubrey Jaffer[1]

`agj@alum.mit.edu`

Abstract

A space-filling function is a bijection from the unit line segment to the unit square, cube, or hypercube. The function from the unit line segment is continuous. The inverse function, while well-defined, is not continuous. Space-filling curves, the finite approximations to space-filling functions, have found application in global optimization, database indexing, and dimension reduction among others. For these applications the desired transforms are mapping a scalar to multidimensional coordinates and mapping multidimensional coordinates to a scalar.

Presented are recurrences which produce space-filling functions and curves of any rank $d \geq 2$ based on serpentine Hamiltonian paths on $(\mathbf{Z} \bmod s)^d$ where $s \geq 2$. The recurrences for inverse space-filling functions are also presented. Both Peano and Hilbert curves and functions and their generalizations to higher dimensions are produced by these recurrences. The computations of these space-filling functions and their inverse functions are absolutely convergent geometric series.

The space-filling functions are constructed as limits of integer recurrences and equivalently as non-terminating real recurrences. Scaling relations are given which enable the space-filling functions and curves and their inverses to extend beyond the unit area or volume and even to all of d-space.

This unification of pandimensional space-filling curves facilitates quantitative comparison of curves generated from different Hamiltonian paths. The isotropy and performance in dimension reduction of a variety of space-filling curves are analyzed.

For dimension reduction it is found that Hilbert curves perform somewhat better than Peano curves and their isotropic variants.

Keywords

space-filling functions; space-filling curves; dimension reduction

Table of Contents

---

[1] Present Address: Digilant, 2 Oliver Street, Suite #901, Boston, MA 02109.

1 Introduction

A space-filling function $F(y) : [0, 1) \to [0, 1)^d$ is a continuous bijection (but not a homeomorphism) from the unit line segment to the unit square, cube, or hypercube. Demonstrating the existence of a one-to-one mapping between points in the line and the plane was the original motivation for space-filling functions. It being possible to construct $F(y)$ as a continuous function is remarkable and results in there being practical applications for space-filling functions.

There being uncountably many space-filling functions, actual computation requires a succinct formulation. Restriction to self-similar functions is a step towards this goal. The use of a finite geometric kernel (the cell) allows separation of the recurrence from the kernel, which enables quantitative comparisons of the performance of different cells (with the recurrence) for uses such as dimension reduction.

The recurrence $Q(u)$ is defined in terms of an integer input and vector of integers output. As the recurrence is scaled it approaches $F(y)$. At finite scaling factors the image of the recurrence is a uniform grid of points in the d-dimensional cube. The space-filling curve $C(y)$ interpolates unit line segments between $Q(\lfloor y \rfloor)$ and $Q(\lceil y \rceil)$, yielding the familiar winding line representations.

1.1 Relation to Previous Work

While there are many papers and books giving algorithms for space-filling curves in 2 dimensions, there are only a few addressing space-filling curves of any rank ($d \geq 2$).

Butz [1], [2], [3] gives (different) algorithms for multidimensional Hilbert and Peano curves. Lawder [4] makes some improvements to Butz's Hilbert algorithm.

Alber and Niedermeier [5] investigate Hilbert curves generated by a recurrence and Manhattan Hamiltonian path cells they term generators. They suggest that their method might be extensible to side-lengths greater than 2, but their approach is not taken here. The result of a de-novo analysis, the recurrences and serpentine generator presented here work for all ranks and side-lengths greater than 1.

Jin and Mellor-Crummey [6] use recursion with a table of enumerated states to compute the space-filling inverse function. In order to create the Hilbert cell table, they recommend using the Butz algorithm. The serpentine generator presented here creates working cells for all ranks and side-lengths greater than 1.

The precession of axes (developed here), which improves isotropy for diagonally opposite corner cells, works with cells having s^d nodes, where $d \geq 2$ is the rank and $s \geq 2$ is the side length. In order to implement the precession of axes, Jin and Mellor-Crummey cells would require s^{d^2} nodes.

Moon et al [7] find that Hilbert curves are asymptotically isotropic, as is also determined by the analysis here. Presented here are pandimensional variants of Peano curves which are asymptotically isotropic (traditional Peano curves are not).

All of the algorithms cited so far require that the integer precision for conversion calculations be fixed. In contrast, the $Q(u)$ and $q(V)$ functions developed here automatically scale the precision by powers of s^d, yielding the same continuous curves as would be produced by fixed precision conversions.

Sagan [8] presents several methods for mapping from the unit real interval to unit squares or cubes, but not for mapping from the unit square or cube to the unit line. His *arithmetization* process transforms coordinates using *similarity transforms* which are constructed through examination of the curve. In contrast, the serpentine algorithm generates alignment transforms for space-filling curves of any rank $d \geq 2$ and side-length $s \geq 2$.

Sagan's figures 3.71 and 3.72, *Peano curves of the switch-back type* (which he attributes to Walter Wunderlich) match the Peano and nearly isotropic Peano curves of Figure 11 here, but not the isotropic variant. His figure 3.73, Wunderlich's *Peano curve of the meander type* corresponds to the $s = 3, d = 2$ curve of Figure 4 of this paper.

1.2 Symbols

Scalar variables and functions are named by lowercase letters. Vector-valued variables and functions are named by uppercase letters. A (zero-based) subscript selects one coordinate of a vector-valued variable or function.

Variables:

$d \geq 2$ integer rank of the image;

$s \geq 2$ integer cell width;

$0 \leq t < s^d$ integer Hamiltonian index;

$0 \leq i < d$ integer index of cell advance dimension (constant for each Hamiltonian-path);

$0 \leq j < d$ integer index of dimension;

$0 \leq n$ integer;

u integer;

Functions:

$H : (\mathbf{Z} \bmod s^d) \to (\mathbf{Z} \bmod s)^d$ Hamiltonian-path;

$h : (\mathbf{Z} \bmod s)^d \to (\mathbf{Z} \bmod s^d)$ Hamiltonian-path inverse;

$k : (\mathbf{Z} \bmod s^d) \to (\mathbf{Z} \bmod d)$ index of sub-cell advance dimension;

$\sigma : (\mathbf{Z} \bmod s^d) \to (\mathbf{Z} \bmod d)$ dimension rotation count;

$N : (\mathbf{Z} \bmod s^d) \to (\mathbf{Z} \bmod 2)^d$ sub-cell entry orientation;

$X : (\mathbf{Z} \bmod s^d) \to (\mathbf{Z} \bmod 2)^d$ sub-cell exit orientation;

$A(V, t, w) \to (\mathbf{Z} \bmod 2)^d$ sub-cell alignment;

$A^{-1}(V, t, w) \to (\mathbf{Z} \bmod 2)^d$ sub-cell alignment inverse;

$Q : \mathbf{N} \to \mathbf{N}^d$ integer sequence;

$q : \mathbf{N}^d \to \mathbf{N}$ integer sequence inverse;

$C : [0, \infty) \to [0, \infty)^d$ curve;

$F : [0, 1) \to [0, 1)^d$ space-filling function;

$f : [0, 1)^d \to [0, 1)$ space-filling function inverse;

$E : [0, 1) \to [0, 1)^d$ space-filling function;

$e : [0, 1)^d \to [0, 1)$ space-filling function inverse.

Both A and A^{-1} are vector-valued. The term "rank" is the number of dimensions d.

2 Scaling

It being more convenient to work with integers, the customary practice is to define $F(y)$ as the limit of a scaled self-similar integer injection:

$$\left\lfloor Q\left(us^d\right)/s \right\rfloor = Q(u) \tag{1}$$

$$F(y) = \lim_{n\to\infty} Q\left(\left\lfloor ys^{dn}\right\rfloor\right)/s^n, \qquad 0 \le y < 1 \tag{2}$$

Considering u to be composed of digits in base-s^d and the coordinates returned as digits in base s, equation (1) constrains $Q(u)$ so that the higher order digits of the coordinates it returns are independent of lower-order digits in u.

In pursuit of self-similarity, the equation (2) can be strengthened by linking F and Q for any (integer) $n > 0$, $s \ge 2$, (rank) $d \ge 2$ and $0 \le u < s^{dn}$.

$$Q(u) = \left\lfloor s^n F\left(u/s^{dn}\right)\right\rfloor \tag{3}$$

Equation (3) holds for unit cells where the travel direction of the first node (sub-cell) matches the net travel direction of the cell; for example, Peano, but not Hilbert cells. The more general case which will be used for the rest of the analysis is:

$$Q(u) = \left\lfloor Q\left(us^{d^2}\right)/s^d \right\rfloor \tag{4}$$

$$F(y) = \lim_{n\to\infty} Q\left(\left\lfloor ys^{d^2 n}\right\rfloor\right)/s^{dn}, \qquad 0 \le y < 1 \tag{5}$$

$$Q(u) = \left\lfloor s^{dn} F\left(u/s^{d^2 n}\right)\right\rfloor, \qquad 0 \le u < s^{d^2 n} \tag{6}$$

This groups d digits of base s^d into one digit of base s^{d^2}. Although for some unit cells fewer digits could be grouped, grouping d digits allows all orientations of the unit cell to be cycled through, improving the isotropy of the resulting curve.

By equation (5):

$$F(0) = (0,\dots,0) \tag{7}$$

By equations (7) and (6):

$$Q(0) = (0,\dots,0) \tag{8}$$

A novel approach (for pandimensional space-filling functions) presented here is the non-terminating recurrence $E(y)$, for which more elegant scaling holds:

$$E\left(\frac{y}{s^{d^2}}\right) = \frac{E(y)}{s^d} \qquad 0 \le y < 1 \tag{9}$$

$$E(y) = F(y) \qquad 0 \le y < 1 \tag{10}$$

3 Unit Cells

The unit cell is defined on a cubic grid of s^d points as a Manhattan Hamiltonian-path $H(t)$ of coordinate vectors in $(\mathbf{Z} \bmod s)^d$. This path visits each grid point exactly once using only unit-length axis-parallel segments.

All unit cells have corner points; when $s = 2$ all of the points are corners. This analysis will concern cells whose entry and exit points are corners. Without loss of generality, the entry point for every cell will have coordinates $(0, \ldots, 0)$.

Of particular interest are what here are termed serpentine paths, which create unit cells for any rank $d \geq 2$ and side-length $s \geq 2$. Serpentine paths have full-length runs parallel to one axis and unit-length runs in the other directions. For serpentine cells, the exit and entry points are either adjacent corners (Figure 3) or furthest diagonally opposite corners (Figure 2). This investigation is concerned with adjacent (even or odd s) and opposite (odd s) cell corners. Adjacent corner paths with odd s exist (Figure 4), but are not serpentine.

```
(define (make-serpentine-path rank side)
  (define (mspr path rnk)
    (define (msps seq cn)
      (if (negative? cn)
          seq
          (msps (append (map (lambda (coords) (cons cn coords))
                             (if (odd? cn) (reverse path) path))
                        seq)
                (- cn 1))))
    (if (>= rnk rank)
        path
        (mspr (msps '() (- side 1))
              (+ rnk 1))))
  (mspr '(()) 0))
```

Figure 1

The Scheme procedure in Figure 1 returns a list of the (list of integer) coordinates of a serpentine path having the given rank and side-length. In internal procedure `mspr`, `path` is the list of coordinate lists for a cell of rank `rnk` − 1. The internal procedure `msps` appends `side` copies (alternating reversal) of `path` with copy-number `cn` adjoined to the front of each coordinate list.

3.1 Diagonal Travel

For the diagonal case, a serpentine pattern can fill a unit cell when s is odd (Figure 2).

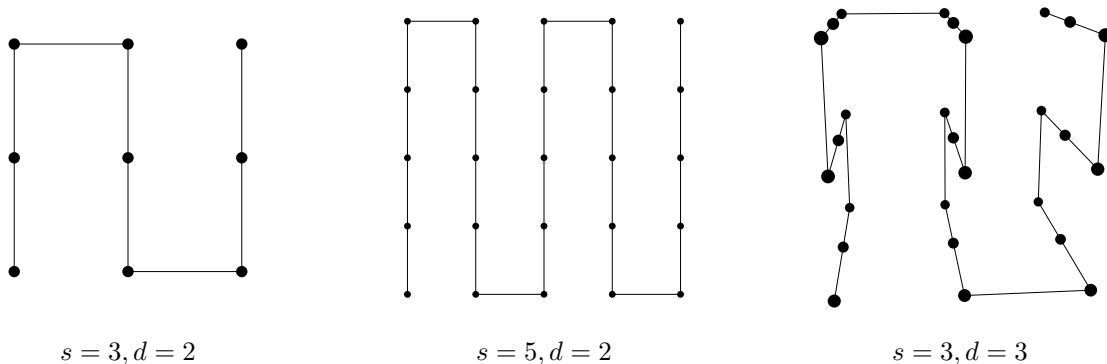$s = 3, d = 2$ $s = 5, d = 2$ $s = 3, d = 3$

Figure 2

For the case of s is even, Dr. Glenn Iba points out that if the points are alternately labeled black and white, then the starting and ending points must be opposite colors, which happens only when the rank d is odd. Even with odd d, the path reaches the outside corner of each corner sub-cell, from which it can't reach any other. In order to reach each node in the cell, each straight run must have odd length. Thus for use with these recurrences, diagonally opposite corner cells can be constructed only when $s > 2$ is odd.

5

3.2 Adjacent Travel

For adjacent entry and exit corners when s is even, a serpentine pattern fills the square (Figure 3). Stacking an even number of those squares and connecting the exits and entries results in adjacent entry and exit corners. This generalizes to all $d \geq 2$.

$s = 2, d = 2$ $s = 4, d = 2$ $s = 2, d = 3$
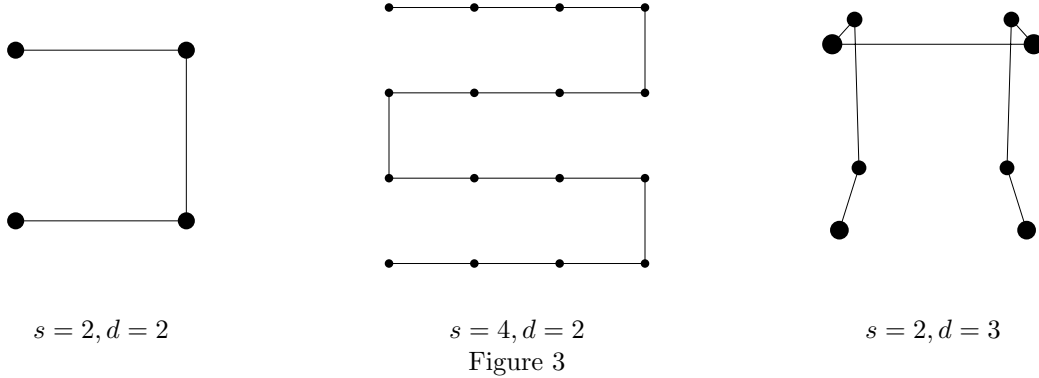
Figure 3

When s is odd, a concentric pattern like that shown in Figure 4 fills the square and has adjacent entry and exit corners. Stacking an odd number of these squares and connecting them vertically results in entry and exit points which are not adjacent. But stacking $d - 1$ cubes of rank $d - 1$ which are rotated $90°$ to each other results in a path between diagonally opposite corners of the d-cube. Stacking an additional cube of rank $d - 1$ with a diagonal path brings the entry and exit points to be above one another, resulting in the entry and exit points being vertically adjacent. Height can be increased by adding pairs of oppositely aligned diagonal cubes of rank $d - 1$.

$s = 3, d = 2$ $s = 5, d = 2$ $s = 3, d = 3$

Figure 4
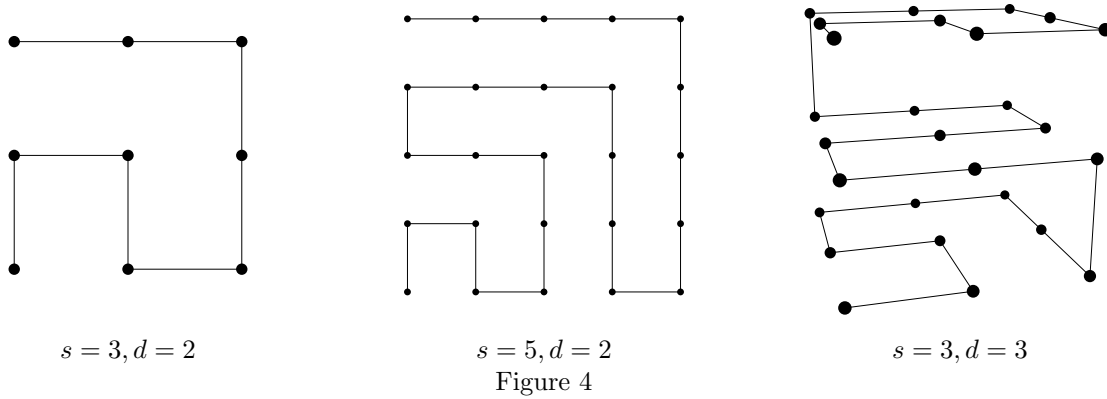
So adjacent-corner cells can be constructed when $s \geq 2$ is even and when s is odd and $s \geq d \geq 2$.

4 Sub-Cell Orientation

The recurrence to be developed replaces each node of the parent cell with a (possibly reflected, possibly rotated) child cell whose edge length is the edge length of its parent cell divided by s. Each edge of the parent cell is replaced with a short edge (parent edge length divided by s) parallel to the edge being replaced. The child cell has s^d points connected by $s^d - 1$ edges.

The challenge is to systematically select the orientation of the child cell so that its entry and exit points connect to its immediate neighbor cells with single short axis-parallel segments.

Given the $H(t)$ sequence of vectors, define vector sequences $N(t)$ and $X(t)$ to describe the entry and exit orientations. Because the entry and exit points are always on corners of the sub-cell, the coordinates in each vector need only take two values. Thus vectors $N(t)$ and $X(t)$ can be elements of $(\mathbf{Z} \bmod 2)^d$.

Because $H(0) = (0, \dots, 0)$, it follows that $N(0) = (0, \dots, 0)$. Similarly $(s-1) \cdot X(s^d - 1) = H(s^d - 1)$. The relation between the entrance of a cell and the exit of its predecessor is (with coordinate subscript $0 \le j < d$):

$$N_j(t) = \begin{cases} 0, & \text{if } t = 0; \\ X_j(t-1) + H_j(t) - H_j(t-1) \bmod 2, & \text{otherwise.} \end{cases} \tag{11}$$

For diagonal-travel cells $X(s^d - 1) = (1, \dots, 1)$. For adjacent-travel cells $X(s^d - 1)$ differs from $(0, \dots, 0)$ by having $s - 1$ in only one coordinate.

For diagonal-travel H-paths, $N(t)$ and $X(t)$ must have opposite parity in all coordinates:

$$X_j(t) = 1 - N_j(t) \tag{12}$$

For adjacent-travel H-paths the mapping from $N(t)$ to $X(t)$ is more complicated.

Consider the first sub-cell, which replaces the $H(0)$ node. In order to reach the next sub-cell, the first sub-cell must traverse a distance of s along one of its axes. The only way to do this is for $N(1) - X(0)$ (traversing $s - 1$) to be in the same direction as $H(1) - H(0)$, which will be non-zero in only one coordinate. Similarly, the cell which replaces the $H(s^d - 1)$ node must align $X(s^d - 1) - N(s^d - 1)$ with $H(s^d - 1) - H(s^d - 2)$.
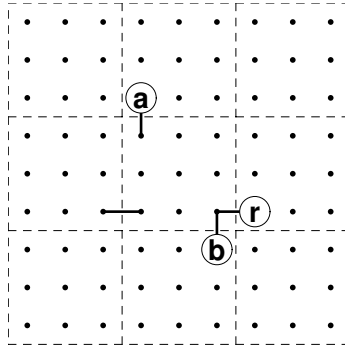

Figure 5

Now consider a $d = 2$ cell at the center of Figure 5 which is neither first nor last, indexed by $0 < t < s^d - 1$. $N(t)$ is a corner of the sub-cell. Without loss of generality assume that $N(t)$ is $(0,0)$ and that the previous cell is to the left of the current cell. After visiting all the points in the current cell there are 3 cells that the path could connect to: right($\mathbf{r}$), above($\mathbf{a}$), and below($\mathbf{b}$).

(**r**) If $H(t+1) - H(t) = (1,0)$, then $X(t) = (1,0)$ and net travel is to the right.

(**a**) If $H(t+1) - H(t) = (0,1)$, then $X(t) = (0,1)$ and connects to the bottom left corner of the cell above.

(**b**) If $H(t+1) - H(t) = (0,-1)$, then $X(t) = (1,0)$ and connects to the top right corner of the cell below.

In cases (**r**) and (**a**), $X(t) - N(t)$ must be in the same direction as $H(t+1) - H(t)$ in order to reach the next cell. In case (**b**) $X(t)$ is already adjacent to the cell below, so $X(t) - N(t)$ can't be in the direction of $H(t+1) - H(t)$. When $d > 2$ there are more possible directions for $X(t) - N(t)$ which may or may not be allowed depending on where $H(t)$ is in the parent cell. The direction which is always allowed is $H(t) - H(t-1)$, so choose it.[2]

---

[2] The note at the end of the next section discusses an exception.

4.1 Adjacent-Corner Cells

Generalizing this to $d \geq 2$ and all orientations, $H(t+1) - H(t)$ will be non-zero in only one coordinate; for $0 < t < s^d - 1$, let $0 \leq k(t) < d$ be the index of the non-zero coordinate in $H(t+1) - H(t)$.

The first and last sub-cells must terminate at adjacent corners of the cell. Because $H(0) = (0, \ldots, 0)$, it follows that $N(0) = (0, \ldots, 0)$ and $X(0) = H(1) - H(0) = H(1)$.

$$X_j(s^d - 1) = \frac{H_j(s^d - 1) - H_j(0)}{s - 1} = \frac{H_j(s^d - 1)}{s - 1} \tag{13}$$

The rest of the cases concern $0 < t < s^d - 1$.

If $N_{k(t)}(t) = 0$ and $H_{k(t)}(t+1) - H_{k(t)}(t) = 1$ or $N_{k(t)}(t) = 1$ and $H_{k(t)}(t+1) - H_{k(t)}(t) = -1$, then the net travel through the cell $X(t) - N(t)$ will be in the same direction as $H(t+1) - H(t)$:

$$X_j(t) = \begin{cases} 1 - N_j(t), & \text{if } j = k(t); \\ N_j(t), & \text{otherwise.} \end{cases} \tag{14}$$

If that is not the case, then travel through the cell will be in the direction of $H(t) - H(t-1)$:

$$X_j(t) = \begin{cases} 1 - N_j(t), & \text{if } j = k(t-1); \\ N_j(t), & \text{otherwise.} \end{cases} \tag{15}$$

The cases of $X_j(0)$ and $X_j(s^d - 1)$ can be absorbed by formulas (14) and (15) by defining $k(0)$ to have the same value as $k(1)$, and $k(s^d - 1)$ to have the same value as $k(s^d - 2)$.

Note: While computer tests show that these rules work with all serpentine cells of up to $(s^d \leq) 10^6$ points, they do not work with every adjacent-corner Manhattan Hamiltonian-path on $(\mathbf{Z} \bmod s)^d$ with odd s. The right side of Figure 6 shows a non-serpentine cell for which equations (14) and (15) don't work. It is unknown whether there exists rules which will work with that cell.

$s = 3, d = 3$ adjacent-corners

$s = 3, d = 3$ adjacent-corners
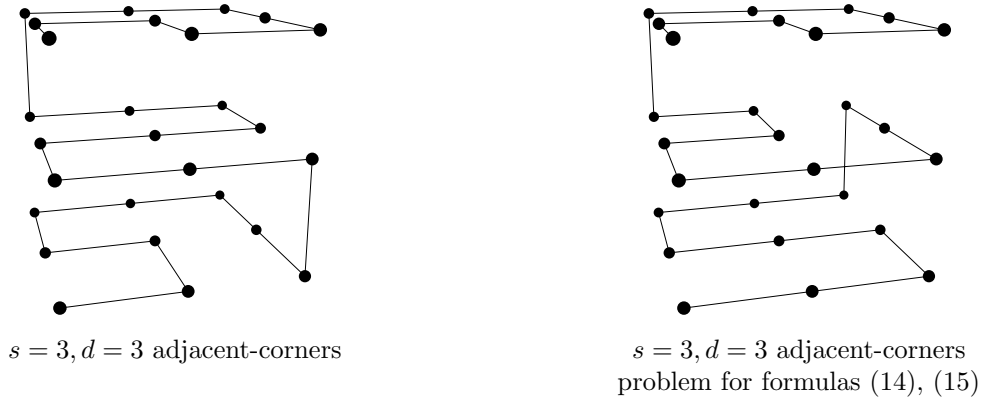problem for formulas (14), (15)

Figure 6

5 Aligning Sub-Cells

The integer recurrence will use $s^n H(t)$ as the coordinates of the parent layer; each sub-cell (having coordinates $s^{n-1}H(t')$) will be reflected and rotated to match orientation with $N(t)$ and $X(t)$.

For diagonal travel unit-cells, all coordinates change between $N(t)$ and $X(t)$; so once the sub-cell is aligned with $N(t)$, it will also be aligned with $X(t)$. The sub-cell can be aligned with $N(t)$ by simply complementing each coordinate in the sub-cell for which the corresponding coordinate in $N(t)$ is nonzero:

$$A_j(V, t, w) = \begin{cases} V_j, & \text{if } N_j(t) = 0; \\ w - V_j, & \text{otherwise.} \end{cases} \tag{16}$$

In adjacent travel cells, $H(s^d - 1)$ has only one nonzero coordinate. Because the recurrence is self-similar, for each t, $N(t) - X(t)$ also has only one nonzero coordinate. When $N(t) - X(t)$ and $H(s^d - 1)$ are nonzero in the same coordinate position, then $A_j(V, N, s)$ defined in equation (16) will align the sub-cell with $N(t)$ and $X(t)$.

If $N(t) - X(t)$ and $H(s^d - 1)$ are nonzero in different coordinate positions $0 \leq k(t) < d$ and $0 \leq i < d$ respectively, then the ith coordinate of the sub-cell must be moved to the $k(t)$th coordinate position. This can be done by applying a cyclic permutation to the coordinates V:

$$A_j(V, t, w) = \begin{cases} V_{j+i-k(t) \bmod d}, & \text{if } N_j(t) = 0; \\ w - V_{j+i-k(t) \bmod d}, & \text{otherwise.} \end{cases} \tag{17}$$

Formula (17) works also when $k(t) = i$. Note that $\sigma(t) = i - k(t) \bmod d$ can be precomputed for each node in the cell's Hamiltonian-path (see equation (35)).

6 The Recurrence

R is the recurrence for integer coordinates. u is the non-negative integer scalar input; a vector of non-negative integers is returned. The highest order digits of u are processed first. At each stage R orients the coordinates at the next smaller scale w/s so that the start and end of its Hamiltonian-path matches the $N(\lfloor u/m \rfloor)$ and $X(\lfloor u/m \rfloor)$ of the current scale.

$$l(u) = \lceil \log_{s^{d^2}}(1 + u) \rceil$$

$$R(u, m, w) = \begin{cases} wH(\lfloor u/m \rfloor) + A\left(R\left(u \bmod m, \lfloor m/s^d \rfloor, \lfloor w/s \rfloor\right), \lfloor u/m \rfloor, w - 1\right) & \text{if } w > 0; \\ (0, 0, \ldots) & \text{otherwise.} \end{cases} \tag{18}$$

$$Q(u) = R\left(u, s^{d^2 l(u) - d}, s^{d \cdot l(u) - 1}\right)$$

$s^{d^2 l(u)}$ is the smallest non-negative integer power of s^{d^2} greater than u. $s^{d^2 l(u) - d}$ is one s^d digit shorter than s^{d^2} so that R starts with the most significant digit instead of a gratuitous digit of zero. The base for the logarithm is s^{d^2} in order to always work on groups of d (base s^d) digits. This is needed for Hamiltonian-paths where $X(0) \neq H(s^d - 1)/(s - 1)$.

$C(y)$ is the curve which bridges the gaps between successive (integer) points with unit line segments:

$$C(y) = (1 + \lfloor y \rfloor - y) Q(\lfloor y \rfloor) + (y - \lfloor y \rfloor) Q(\lfloor y \rfloor + 1) \tag{19}$$
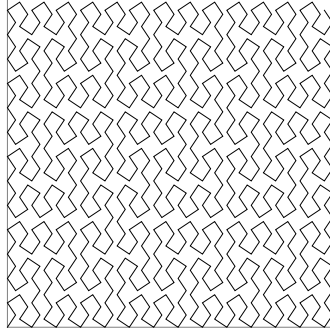
$Q(u)$ can be used as a space-filling function by scaling by $n > 0$ groups of digits:

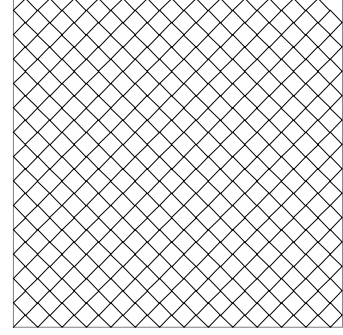$$F(y) = \lim_{n \to \infty} Q\left(\lfloor y s^{d^2 n} \rfloor\right) / s^{dn} \tag{5}$$

Figures 7 and 8 show $F(y)$ evaluated at 729 and 1024 points respectively for 3 values for n. The reason that the $n = 3$ curves don't resemble the $n = 1$ curves is because many of the inflection points of the $n = 3$ curve are not being traced.
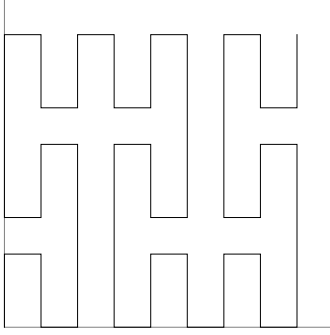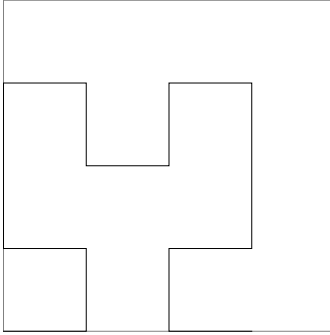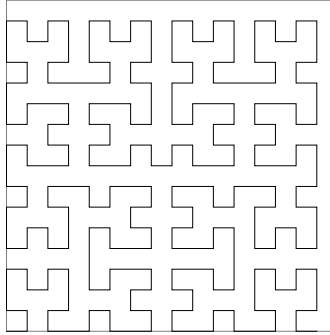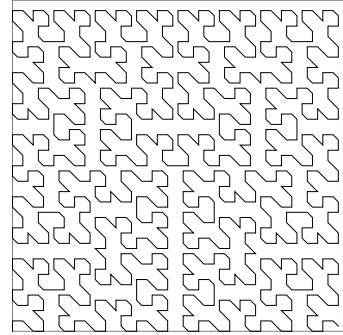
9

Peano $n = 1$ Peano $n = 2$ Peano $n = 3$

Figure 7

Hilbert $n = 1$ Hilbert $n = 2$ Hilbert $n = 3$

Figure 8

7 The Inverse Recurrence

Like $Q(u)$, its scalar-valued inverse $q(V)$ processes the highest order digits of its argument (in this case the coordinates V) first. $s^{d \cdot l(V)}$ is the lowest integer power of s^d greater than the largest coordinate in V. $s^{d \cdot l(u)-1}$ is one s digit shorter than s^d so that r starts with the most significant digit instead of a gratuitous digit of zero. $h\left(\lfloor V/w \rfloor\right)$ is the index in H of the highest base-s digit of the coordinates V.

$$V \bmod w \equiv (V_0 \bmod w, \ldots, V_{d-1} \bmod w)$$

$$\left\lfloor \frac{V}{w} \right\rfloor \equiv \left(\left\lfloor \frac{V_0}{w} \right\rfloor, \ldots, \left\lfloor \frac{V_{d-1}}{w} \right\rfloor \right)$$

$$l(V) = \left\lceil \log_{s^d}(1 + \max(V_0, \ldots, V_{d-1})) \right\rceil$$

$$r(u, V, w) = \begin{cases} r\left(us^d + h\left(\lfloor \frac{V}{w} \rfloor\right), A^{-1}\left(V \bmod w, h\left(\lfloor \frac{V}{w} \rfloor\right), w - 1\right), \lfloor \frac{w}{s} \rfloor\right) & \text{if } w > 0; \\ u & \text{otherwise.} \end{cases} \qquad (20)$$

$$q(V) = r\left(0, V, s^{d \cdot l(V)-1}\right)$$

$$f(Y) = \lim_{n \to \infty} q\left(\lfloor Y s^{dn} \rfloor\right) / s^{d^2 n}$$

While A as a function of V is its own inverse, A' and A'' (introduced in the "Isotropy" section) will not be their own inverses when $d > 2$.

$$A_j'^{-1}(V, t, w) = A_{j-i \bmod d}(V, t, w) \qquad (21)$$

$$A_j''^{-1}(V, t, w) = A_{j-t-1 \bmod d}(V, t, w) \qquad (22)$$

8 Non-Terminating Recurrence

Notice that in the alignment function $A(V, t, w)$ nothing constrains the argument V to be a vector of integers. It also works as a vector of real numbers. By scaling the input by s^d and the output by s at each call, a non-terminating recurrence (with $0 \le y < 1$) can be written:

$$E(y) = \frac{H\left(\lfloor s^d y \rfloor\right) + A\left(E\left(s^d y - \lfloor s^d y \rfloor\right), \lfloor s^d y \rfloor, 1\right)}{s} \tag{23}$$

Because the denominators are increasing powers of s, $E(y)$ is an absolutely convergent geometric series.

H, s, and d are all characteristics of the cell path P. Writing A^*, H^*, and E^* as functions of P, the similarity to a fixed-point combinator can be seen:

$$E^*(P, y) = \frac{H^*\left(P, \lfloor s^d y \rfloor\right) + A^*\left(P, E^*\left(P, s^d y - \lfloor s^d y \rfloor\right), \lfloor s^d y \rfloor, 1\right)}{s} \tag{24}$$
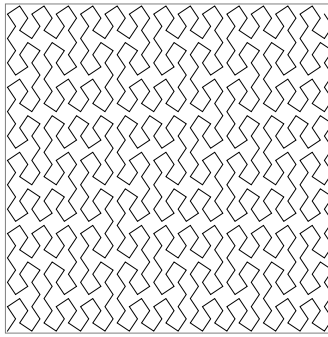
Because it doesn't terminate, equation (23) can't be computed directly. By limiting its depth to nd, the curves produced by equation (25) can be explored:

$$E''(y, c) = \frac{1}{s} \begin{cases} H\left(\lfloor s^d y \rfloor\right) + (1/2, \ldots, 1/2), & \text{if } c \le 1; \\ H\left(\lfloor s^d y \rfloor\right) + A\left(E''\left(s^d y - \lfloor s^d y \rfloor, c-1\right), \lfloor s^d y \rfloor, 1\right), & \text{otherwise.} \end{cases} \tag{25}$$
$$E''(y) = E''(y, nd)$$
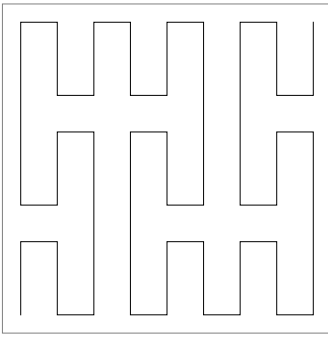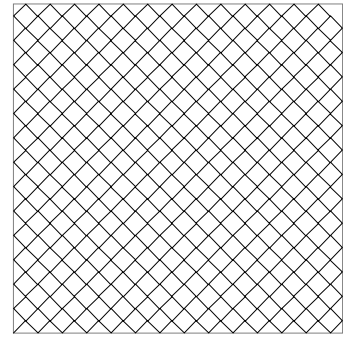
Peano $n = 1$　　　　Peano $n = 2$　　　　Peano $n = 3$

Figure 9
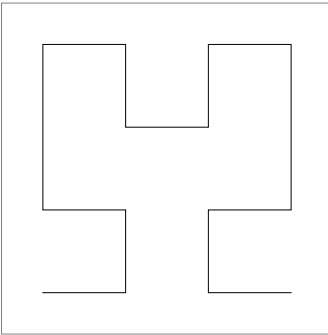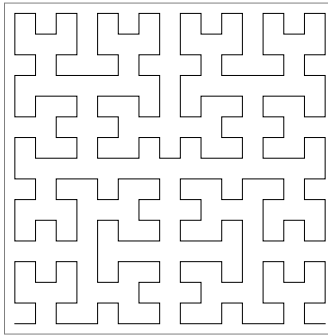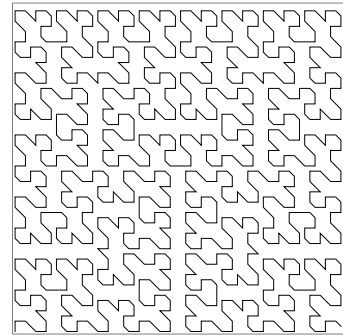
Hilbert $n = 1$　　　　Hilbert $n = 2$　　　　Hilbert $n = 3$

Figure 10

As with Figures 7 and 8, Figures 9 and 10 are traced at 729 and 1024 points respectively. Note that E'' terminates at the middle of each s^{-nd} sub-cell; its plots are centered at $(1/2, \ldots, 1/2)$ where Figures 7 and 8 are pinned to the origin.

The inverse non-terminating recurrence is:

$$e(V) = \frac{h\left(\lfloor sV \rfloor\right) + e\left(A^{-1}\left(sV - \lfloor sV \rfloor, h\left(\lfloor sV \rfloor\right), 1\right)\right)}{s^d} \tag{26}$$

9 Isotropy

A complete pattern contains s^{dn} points connected in sequence by $s^{dn} - 1$ edges. There are s^d sub-cells, each of which has $s^{d(n-1)} - 1$ edges. The total for the sub-cells is $s^{dn} - s^d$ edges; but the edges of the top-level cell connect the sub-cells, resulting in $s^{dn} - s^d + s^d - 1 = s^{dn} - 1$ edges.

Each edge connecting grid points in a complete pattern is parallel to one of the axes. Isotropy is the condition that equal lengths of edges are parallel to each axis.

The Peano curve on the left in Figure 11 has 60 vertical and 20 horizontal segments, much more vertical than horizontal travel. Each of its sub-cells could be oriented horizontally or vertically, but all are vertical. There are only straight vertical runs of lengths 2 and 5, and all horizontal runs are length 1.

With serpentine diagonal cells, the major axis travel is $(s-1)s^{d-1}$, the next most traveled axis travel is $(s-1)s^{d-2}$, etc. With cells at all levels aligned, the counts will be proportional, and indeed 60 and 20 are proportional to 6 and 2.
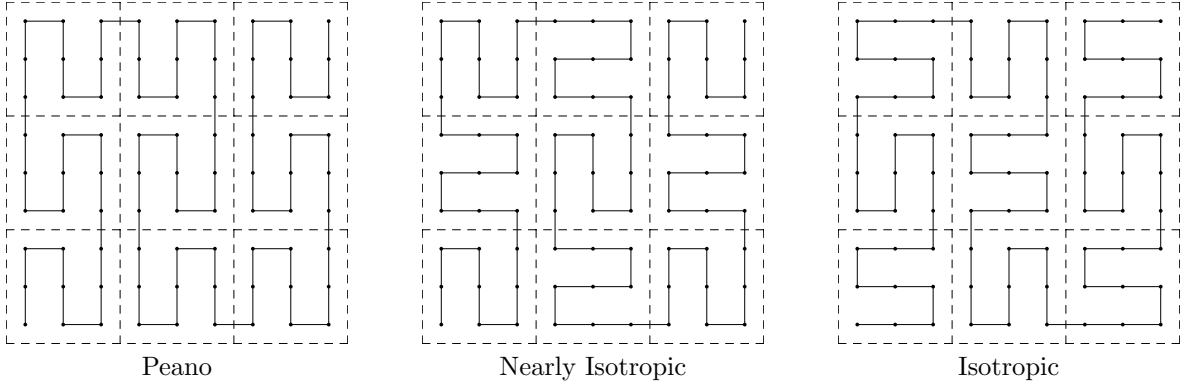

| Peano | Nearly Isotropic | Isotropic |

Figure 11

If the orientation of the cells is rotated from sub-cell to sub-cell, then a s^{d^2} point pattern can be constructed which is closer to isotropic. This can be done by applying a cyclic permutation to the coordinates V in the alignment function:

$$A'_j(V, t, w) = \begin{cases} V_{j+t \bmod d}, & \text{if } N_j(t) = 0; \\ w - V_{j+t \bmod d}, & \text{otherwise.} \end{cases} \tag{27}$$

The center panel of Figure 11 shows the result, having 4 horizontally oriented sub-cells and 5 vertically oriented sub-cells in the enclosing vertical cell for a total of 44 vertical and 36 horizontal segments. There are straight runs of length 1, 2, and 3 in each direction.

If the permutation count is offset by 1 so that the enclosing cell has a different orientation from the first sub-cell, then balance is achieved for this $s = 3, d = 2$ case:

$$A''_j(V, t, w) = \begin{cases} V_{j+t+1 \bmod d}, & \text{if } N_j(t) = 0; \\ w - V_{j+t+1 \bmod d}, & \text{otherwise.} \end{cases} \tag{28}$$

With the opposite sub-cell orientations, but with the same enclosing cell orientation, the curve on the right side of Figure 11 has 40 horizontal and 40 vertical segments.

A serpentine d-level (s^{d^2} point) pattern with A'' precession will be isotropic when its s^d sub-cells and 1 parent cell have orientations evenly distributed among the dimensions. A necessary (but not sufficient) condition for isotropy is: $s^d + 1 \equiv 0 \bmod d$. This will not happen when $d = s = 3$. But the precessing $d = s = 3$ case is asymptotically isotropic:

nodes	$j = 0$		$j = 1$		$j = 2$	
27	2	7.4%	6	22.2%	18	66.7%
729	236	32.4%	240	32.9%	252	34.6%
19683	6554	33.3%	6558	33.3%	6570	33.4%

The A' precession gives the same results (just permuted):

12

nodes	$j = 0$		$j = 1$		$j = 2$	
27	18	66.7%	2	7.4%	6	22.2%
729	240	32.9%	252	34.6%	236	32.4%
19683	6554	33.3%	6558	33.3%	6570	33.4%

The orientation of a serpentine cell is the direction of its straight runs of length $s - 1$ (all perpendicular runs will have length 1). In order for there to be a straight run longer than s, two cells and the edge that connects them must all be aligned. Precessing the sub-cell orientations prevents long runs between adjacent sub-cells at the lowest level. What about connections between sub-cells of adjacent parent cells? Because the cell orientations precess, the shift-count offset between the exit and entry cells is $1 - (s^d - 1) \bmod d = -s^d \bmod d$. The cells align only when $s^d \equiv 0 \bmod d$. While long straight runs are prevented when $s = 3$ and $d = 2$, they will occur when $s = d = 3$.

The isotropy of $s = d = 3$ can be somewhat improved by reflecting one of its planes along its diagonal, as shown in Figure 12:

$s = 3, d = 3$ diagonal-corners

Figure 12

This reduces the segment-count disparity between directions, which results in the complete patterns being closer to evenly divided.

nodes	$j = 0$		$j = 1$		$j = 2$	
27	2	7.4%	10	37.0%	14	51.9%
729	236	32.4%	244	33.5%	248	34.0%
19683	6554	33.3%	6562	33.3%	6566	33.4%

9.1 Adjacent-Corner Isotropy

Even though it is asymmetrical, the $s = 3, d = 2$ cell on the left side of Figure 4 is balanced. All complete $s^{d^2 n}$ patterns constructed with it will be balanced:

nodes	$j = 0$		$j = 1$	
9	4	44.4%	4	44.4%
81	40	49.4%	40	49.4%
729	364	49.9%	364	49.9%
6561	3280	50.0%	3280	50.0%

With an even s, the number of edges in a complete pattern, $s^{d^2} - 1$, will be odd. The tally of edges by direction for the 2-dimensional Hilbert curve is:

nodes	$j = 0$		$j = 1$	
4	2	50.0%	1	25.0%
16	7	43.8%	8	50.0%
64	32	50.0%	31	48.4%
256	127	49.6%	128	50.0%

But if one more edge is included in the tally, this particular pattern achieves perfect balance:

nodes	$j = 0$		$j = 1$	
4	2	50.0%	2	50.0%
16	8	50.0%	8	50.0%
64	32	50.0%	32	50.0%
256	128	50.0%	128	50.0%

The 3-dimensional and 4-dimensional Hilbert cells aren't balanced, but their patterns are asymptotically balanced:

nodes	$j = 0$		$j = 1$		$j = 2$	
8	4	50.0%	1	12.5%	2	25.0%
64	18	28.1%	22	34.4%	23	35.9%
512	171	33.4%	174	34.0%	166	32.4%
4096	1374	33.5%	1355	33.1%	1366	33.3%
32768	10902	33.3%	10926	33.3%	10939	33.4%

nodes	$j = 0$		$j = 1$		$j = 2$		$j = 3$	
16	8	50.0%	1	6.2%	2	12.5%	4	25.0%
256	44	17.2%	58	22.7%	71	27.7%	82	32.0%
4096	1002	24.5%	1104	27.0%	1098	26.8%	891	21.8%
65536	17011	26.0%	16562	25.3%	15544	23.7%	16418	25.1%

For adjacent-corner cells with $d = 2$ the orientation of each sub-cell is constrained by the fact that only one of axes has nonzero net-travel (there being only two permutations of 2 axes). For $d > 2$ there is wobble possible by using permutations other than powers of the one cyclic permutation. This will not make the cell isotropic but could make the $s^{d^2 n}$ patterns become more isotropic as n increases.

14
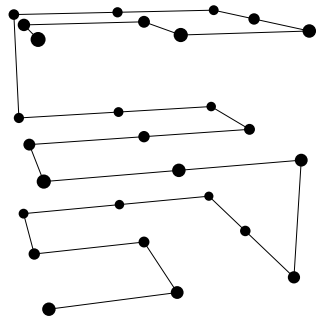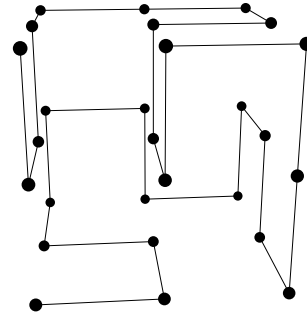
$s = 3, d = 3$ adjacent-corners $s = 3, d = 3$ adjacent-corners

Figure 13

Because the start and end segments are not part of straight runs, the maximum run length for curves formed from the cell on the left side of Figure 13 (the same as the cell from Figure 4) is 3. Despite that, small patterns from that cell are quite anisotropic:

nodes	$j = 0$		$j = 1$		$j = 2$	
27	10	37.0%	14	51.9%	2	7.4%
729	288	39.5%	256	35.1%	184	25.2%
19683	7158	36.4%	6290	32.0%	6234	31.7%

The right side of Figure 13 shows a cell with improved isotropy. The top and bottom halves have 5 vertical segments each; the left and right halves have 4 lateral segments each; and the front and back halves have 4 front-to-back segments each. There are only 2 straight runs:

nodes	$j = 0$		$j = 1$		$j = 2$	
27	8	29.6%	8	29.6%	10	37.0%
729	248	34.0%	240	32.9%	240	32.9%
19683	6552	33.3%	6578	33.4%	6552	33.3%

15

10 Filling All of Space

So far the self-similarity of space-filling curves has been realized by identifying the origin of the Hamiltonian-path with the origin of the path of the first sub-cell, which results in ranges with non-negative coordinates. For serpentine diagonal cells (with odd s) it is possible to identify the midpoint of the Hamiltonian-path with the midpoint of the center sub-cell, extending the range to all integer coordinates. While the treatment above could be redeveloped using a symmetric modulo operator, it is simpler to use scaled offsets of the recurrence (18) inputs and outputs.

$$l(u) = \left\lceil \log_{s^{d^2}} (2 \cdot (1 + |u|)) \right\rceil$$
$$Q'(u) = Q\left(u + \left\lfloor s^{d^2 l(u)}/2 \right\rfloor\right) - \left(\left\lfloor s^{d \cdot l(u)}/2 \right\rfloor, \left\lfloor s^{d \cdot l(u)}/2 \right\rfloor, \ldots\right) \tag{29}$$

$s^{d^2 l(u)}$ finds the smallest integer power of s^{d^2} half of which is large enough to offset $-|u|$ to be positive. Half of that is added to u and the corresponding offset is subtracted from each coordinate returned. Because the base-s digits of $s^n/2$ are $\lfloor s/2 \rfloor, \lfloor s/2 \rfloor, \ldots$, the offsets bias $u = 0$ to the center sub-cell at each scale.

With such auto-scaling u can take any integer value and $Q'(u)$ can reach any point with integer-coordinates. How do these curves spread to the whole plane? Figure 14 shows $Q'(u)$ for u over the range of $-3280 = -\lfloor 9^4/2 \rfloor$ to 0. The Peano and its nearly-isotropic variant wrap around the origin in a spiral; the positive u values do so also. The isotropic variant wraps back and forth around the origin.
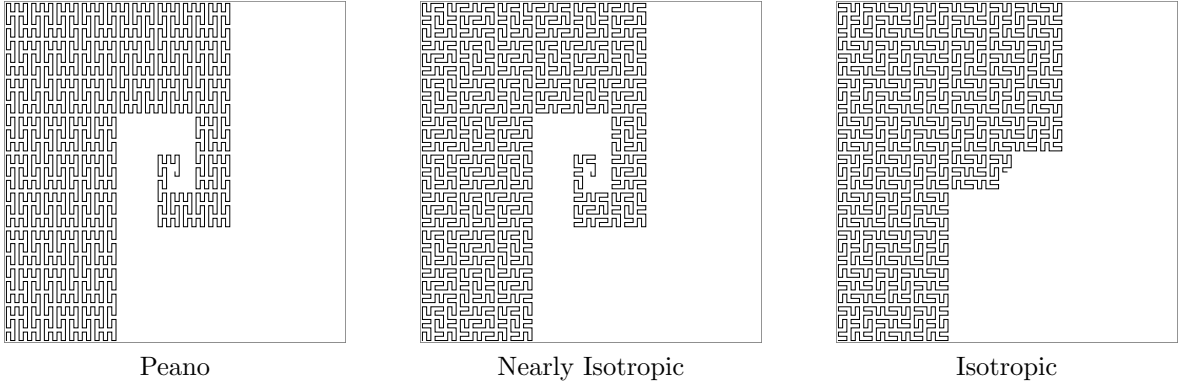

| Peano | Nearly Isotropic | Isotropic |

Figure 14

The inverse function scales according to the largest coordinate in input vector V.

$$l'(V) = \left\lceil \log_{s^d} (2 \cdot (1 + \max(|V_0|, \ldots, |V_{d-1}|))) \right\rceil$$
$$q'(V) = q\left(V + \left(\left\lfloor s^{d \cdot l'(V)}/2 \right\rfloor, \left\lfloor s^{d \cdot l'(V)}/2 \right\rfloor, \ldots\right)\right) - \left\lfloor s^{d^2 l'(V)}/2 \right\rfloor \tag{30}$$

Equation (30) q' is particularly useful for reducing the dimension of data-sets whose bounds are not known in advance.

Because the most significant digit of F and f (and E and e) is always the most significant fractional digit, centering the unit real space-filling function is easy:

$$F'(y) = F\left(y + \frac{1}{2}\right) - \left(\frac{1}{2}, \frac{1}{2}, \ldots\right) \tag{31}$$

$$f'(Y) = f\left(Y + \left(\frac{1}{2}, \frac{1}{2}, \ldots\right)\right) - \frac{1}{2} \tag{32}$$

$$E'(y) = E\left(y + \frac{1}{2}\right) - \left(\frac{1}{2}, \frac{1}{2}, \ldots\right) \tag{33}$$

$$e'(Y) = e\left(Y + \left(\frac{1}{2}, \frac{1}{2}, \ldots\right)\right) - \frac{1}{2} \tag{34}$$

16

11 Dimension Reduction Performance

For dimension reduction it is desirable that points that are close in d-dimensional space map to scalar values that are close and that points that are distant map to scalar values that are distant.
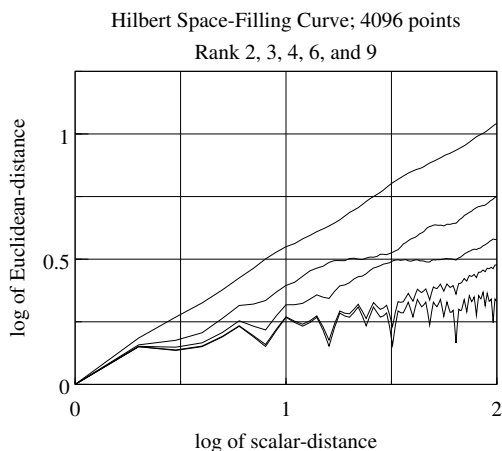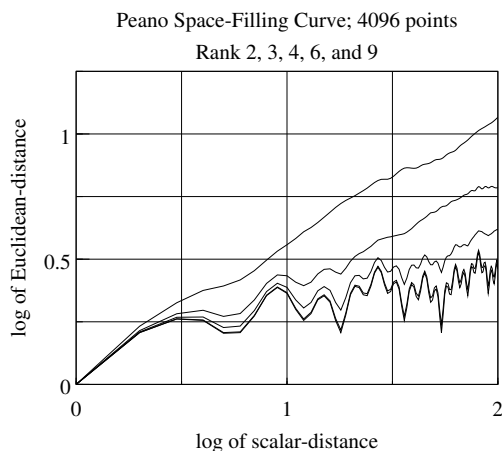
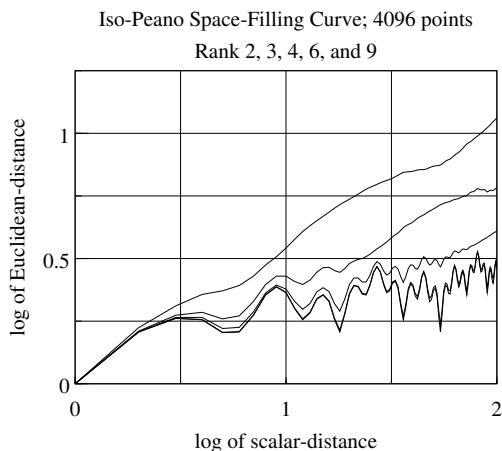Hilbert Space-Filling Curve; 4096 points

Rank 2, 3, 4, 6, and 9

Figure 15

Peano Space-Filling Curve; 4096 points

Rank 2, 3, 4, 6, and 9

Figure 16

Iso-Peano Space-Filling Curve; 4096 points

Rank 2, 3, 4, 6, and 9

Figure 18

Near-Iso-Peano Space-Filling Curve; 4096 points

Rank 2, 3, 4, 6, and 9
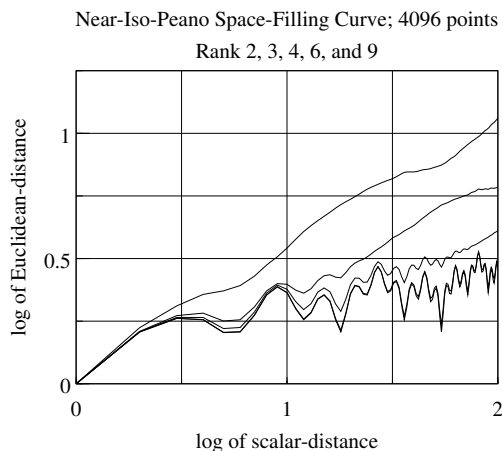
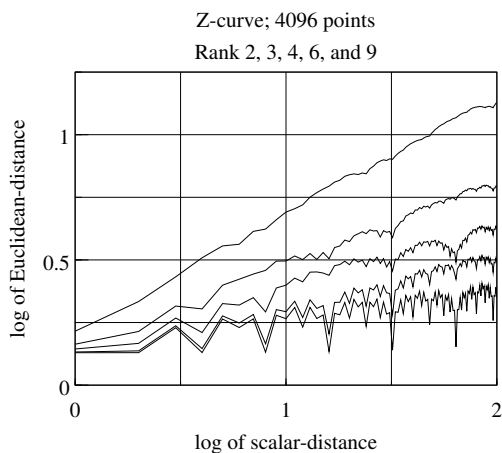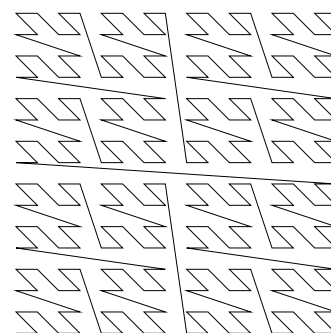Figure 17

Z-curve; 4096 points

Rank 2, 3, 4, 6, and 9

Figure 19

Z-curve

Figure 20

Figures 15 through 19 are log-log plots of the average euclidean distance $||Q(v_1) - Q(v_2)||$ versus scalar lengths $|v1 - v2|$ from 1 to 100 for Hilbert ($s = 2$), Peano-like ($s = 3$), and Z curves of ranks 2, 3, 4, 6, and 9.

The top trace of each plot is for rank $d = 2$; each successively lower curve is of higher rank.

Figure 20 shows the Z-curve (which Sagan attributes to Lebesgue in [8]), one of the simplest space-filling functions, although it isn't continuous. It forms each vector coordinate from every d-th bit of the scalar. It is self-avoiding, but contains some long edges.

The Z-curve has the widest spread and is particularly poor in 2-dimensions. Worse still is that most unit length scalar edges map to coordinates which are more than one unit separated. Thus the average displacement for unit edges averages above 1.6 for 2-dimensions and above 1.35 for the other ranks.

Ranks 6 and 9 appear coincident in the graphs of Peano-like curves. Except for the Z-curve, the rank-2 curves show euclidean distance being slightly longer than the square-root of scalar distance. Curves for the higher ranks are not as close to the dth root.

The great similarity between Figures 16, 17, and 18 indicate that these measurements are largely unaffected by isotropy and run-length distribution.

12 Praxis

The cell properties $H(t)$, d, i, s, $N(t)$, $X(t)$, $H^{-1}(V)$, and $k(t)$ can be precomputed and stored within the data-structure for the cell. If the shift-count $\sigma(t)$ is also precomputed:

$$\sigma(t) = \begin{cases} i - k(t), & \text{if } H(t) \text{ is an adjacent-corner cell;} \\ 0, & \text{if } H(t) \text{ is a Peano cell;} \\ t, & \text{if } H(t) \text{ is a near-isotropic opposite-corner cell } (A'); \\ t + 1, & \text{if } H(t) \text{ is an isotropic opposite-corner cell } (A''). \end{cases} \tag{35}$$

then all the alignment functions can be computed using the same formula:

$$A_j(V, t, w) = \begin{cases} V_{j + \sigma(t) \bmod d}, & \text{if } N_j(t) = 0; \\ w - V_{j + \sigma(t) \bmod d}, & \text{otherwise.} \end{cases} \tag{36}$$

Similarly, the inverse alignment functions can computed using a single formula:

$$A_j^{-1}(V, t, w) = \begin{cases} V_{j - \sigma(t) \bmod d}, & \text{if } N_j(t) = 0; \\ w - V_{j - \sigma(t) \bmod d}, & \text{otherwise.} \end{cases} \tag{37}$$

Thus the recurrences treat diagonal-corner and adjacent-corner patterns uniformly.

Equations (18), (20), (29), and (30) have terms involving ceilings of logarithms. While mathematically correct, practical implementations will compute these quantities using integers as suggested by their descriptions in the text. Floored division ($\lfloor u/m \rfloor$) is simply integer division.

When $F(y)$ or $E'(y)$ is computed with inexact numbers, small increments ($\approx 1 \times 10^{-9}$) must be added to floor function inputs so that rounding and truncation don't foil the calculation.

R5RS Scheme (Kelsey et al [9]) code implementing cell construction and the Q, q, Q', and q' integer recurrences is found in the SLIB Scheme Library's `space-filling` package:

http://cvs.savannah.gnu.org/viewvc/slib/slib/rmdsff.scm?view=markup. Documentation is at http://people.csail.mit.edu/jaffer/slib/Multidimensional-Space_002dFilling-Curves.html

13 Conclusion

Algorithms implementing the recurrences presented here compute the space-filling coordinates or the scalar value (from space-filling coordinates) with geometric convergence. These algorithms work with serpentine cells of any rank $d \geq 2$ and side length $s \geq 2$ and many non-serpentine cells.

The serpentine cell algorithm produces cells for Hilbert and Peano space-filling curves and functions, their pandimensional analogs and inverses, as well as novel pandimensional space-filling curves with improved properties.

By offsetting and scaling the scalar and coordinates, the functions and curves can be extended beyond the unit (hyper)cube, even to all of d-space for serpentine diagonal-travel cells (odd $s \geq 3$).

The performance of space-filling curves for dimension-reduction was measured for multidimensional Hilbert and Peano curves; Hilbert curves performed somewhat better than Peano curves and their isotropic variants.

References

[1] Arthur R. Butz. Space filling curves and mathematical programming. *Information and Control*, 12(4):314 – 330, 1968.

[2] Arthur R. Butz. Convergence with Hilbert's space filling curve. *Journal of Computer and System Sciences*, 3(2):128 – 146, 1969.

[3] A.R. Butz. Alternative Algorithm for Hilbert's Space-Filling Curve. *IEEE Transactions on Computers*, 20(4):424–426, 1971.

[4] J.K. Lawder. Calculation of Mappings Between One and n-dimensional Values Using the Hilbert Space-filling Curve. Research Report JL1/00, School of Computer Science and Information Systems, Birkbeck College, University of London, 2000.

[5] Jochen Alber and Rolf Niedermeier. On Multi-dimensional Hilbert Indexings. In Wen-Lian Hsu and Ming-Yang Kao, editors, *COCOON*, volume 1449 of *Lecture Notes in Computer Science*, pages 329–338. Springer, 1998.

[6] Guohua Jin and John Mellor-Crummey. SFCGen: A Framework for Efficient Generation of Multi-dimensional Space-filling Curves by Recursion. *ACM Trans. Math. Softw.*, 31(1):120–148, March 2005.

[7] Bongki Moon, H.V. Jagadish, Christos Faloutsos, and Joel H. Saltz. Analysis of the Clustering Properties of Hilbert Space-filling Curve. *IEEE Trans. on Knowledge and Data Engineering (IEEE-TKDE)*, 13(1):124–141, 2001.

[8] Hans Sagan. *Space-Filling Curves*. Springer-Verlag New York, 1994.

[9] W. Kelsey, R. Clinger and J. (Eds) Rees. Revised[5] Report on the Algorithmic Language Scheme. *ACM SIGPLAN Notices*, 33(9):26–76, 1998.