Chapter 5

Hierarchical Interpretation of Fractal Image Coding and Its Applications

Z. Baharav, D. Malah, and E. Karnin

Many interesting features associated with iterated function systems and standard fractals seem to fade away as the fractal coding of images evolves. For example, the property of self-similarity at different resolutions, inherent to fractals, does not show up in a simple form in the image coding problem. In this chapter we will demonstrate a hierarchical model for the fractal encoding of images, and we will use it to show that the properties of self-similarity at different resolutions, and the related notion of fractal dimension, exist in fractal coding of images. Moreover, applications of these properties will be given.

The chapter is organized as follows: Section 5.1 reviews the formulation of partitioned iterated function systems (PIFS) coding. Notations and illustrative examples are also given. Section 5.2 presents the main result. It contains a theorem, relating the different resolutions of a signal to its PIFS code, and gives a hierarchical interpretation of this theorem. Section 5.3 develops a matrix form of the PIFS code; a formulation which is needed for the applications sections that follow. In Sections 5.4–5.6 various applications are described: Section 5.4 describes a fast decoding method. Section 5.5 defines the important notion of the PIFS embedded function, and applies it to achieve super-resolution. Section 5.6 revisits the definition of the embedded function, this time to examine different sampling methods. Conclusions are presented in Section 5.7. Finally, the proofs of the three main theorems appear in Addenda A, B, and C.

Y. Fisher (ed.), *Fractal Image Compression*

5.1 Formulation of PIFS Coding/Decoding

In this section we discuss and show an example of a different formulation of PIFS coding and decoding. Also, we will refer to 1-dimensional signals, and we will call them either vectors or blocks. Extensions to two-dimensions are immediate in most cases. The following notation will be used: vectors are in boldface letters (like $\mathbf{a}$) and matrices are in bold uppercase (like $\mathbf{A}$).

Encoding

The task of finding the PIFS code of a vector $\boldsymbol{\mu}_o$ is the task of finding a contractive transformation W, such that its fixed point is as close as possible to $\boldsymbol{\mu}_o$. Formally, this task can be described as follows.

Consider the complete metric space $(\mathbb{R}^N, d^\infty)$, where:

1. $\mathbb{R}^N$ denotes the N-dimensional Cartesian product of the real numbers. Each point in $\mathbb{R}^N$ is a column-vector of size N of real numbers. Thus

$$\mathbf{x} \in \mathbb{R}^N \qquad \text{implies} \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} . \tag{5.1}$$

2. d^∞ is the metric defined by:

$$\mathbf{x}, \mathbf{y} \in \mathbb{R}^N \qquad d^\infty(\mathbf{x}, \mathbf{y}) = \max_{i=1,\ldots,N} |x_i - y_i| \ . \tag{5.2}$$

The vector to be encoded is $\boldsymbol{\mu}_o \in \mathbb{R}^N$. We seek a transformation W, such that the following three requirements are fulfilled:

1. W maps the space into itself:

$$\begin{array}{rccc} W & : & \mathbb{R}^N & \rightarrow & \mathbb{R}^N \\ & & \mathbf{v} & \mapsto & \mathbf{u} = W(\mathbf{v}) \ . \end{array} \tag{5.3}$$

2. W is a contractive transformation:

$$\exists\, s \in [0, 1) \ \mid \ \forall\, \mathbf{x}, \mathbf{y} \in \mathbb{R}^N \qquad d^\infty(W(\mathbf{x}), W(\mathbf{y})) \leq s\, d^\infty(\mathbf{x}, \mathbf{y}) \, . \tag{5.4}$$

These first two requirements define the set of allowed transformations, $W \in \mathcal{W}$.

3. Being a contraction in a complete metric space, W has a unique fixed point $\mathbf{f}_W \in \mathbb{R}^N$ such that $\mathbf{f}_W = W(\mathbf{f}_W)$ (Contraction Mapping Theorem 2.3). The third requirement is that W minimizes the distance between its fixed point $\mathbf{f}_W$ and $\boldsymbol{\mu}_o$:

$$d^\infty(\boldsymbol{\mu}_o, \mathbf{f}_W) = \min_{V \in \mathcal{W}} d^\infty(\boldsymbol{\mu}_o, \mathbf{f}_V). \tag{5.5}$$

That is,

$$W = \arg \min_{V \in \mathcal{W}} d^\infty(\boldsymbol{\mu}_o, \mathbf{f}_V). \tag{5.6}$$

Finding such a transformation W can be a very complex problem, since it involves a minimization over many transformations. An approach to making this problem solvable is to restrict the number of allowed transformations ([45], [44]). The W found this way may be suboptimal, but this is a compromise that one has to make in order to solve the minimization problem.

Thus, we will restrict the allowed transformations W to be systems of M_R functions w_i. Each w_i is further restricted to be of the form:

$$\begin{array}{rccl} w_i & : & \mathbb{R}^D & \rightarrow & \mathbb{R}^B \\ & & \mathbf{d}_{m_i} & \mapsto & \mathbf{r}_i = w_i(\mathbf{d}_{m_i}) = a_i\varphi(\mathbf{d}_{m_i}) + b_i\mathbf{1}_B, \end{array} \tag{5.7}$$

where:

$\mathbf{d}$ - is a block of D consecutive elements extracted from $\mathbf{v}$. It is called the *domain block*. $\mathbf{d}_{m_i}$ is thus the m_i-th domain block in an enumerated list of all such blocks in $\mathbf{v}$. The use of the subscript m_i stresses the fact that the domain block $\mathbf{d}_{m_i}$ is mapped to $\mathbf{r}_i$. For now, no specific mechanism for extracting the blocks will be discussed.

$\mathbf{r}$ - is a block of $B < D$ consecutive elements of $\mathbf{v}$. It is called the *range block*, and $\mathbf{r}_i$ is thus the i-th range block. $\mathbf{r}_i$ belongs to the image of W.

φ - is a *spatial contraction* function which maps blocks of size D to blocks of size B .

a_i - is a scalar *scaling* factor,

$$a_i \in \mathbb{R}, \qquad |a_i| < 1 . \tag{5.8}$$

b_i - is a scalar *offset* value, $b_i \in \mathbb{R}$.

$\mathbf{1}_B$ - is a vector of size B of all 1's.

The three parameters (a_i, b_i, m_i) are called the *transformation parameters*.

By loosely using the union notation to describe concatenation of blocks, we can write :

$$\begin{aligned} \mathbf{u} &= W(\mathbf{v}) \\ W(\mathbf{v}) &= \bigcup_{i=1}^{M_R} w_i(\mathbf{d}_{m_i}), \qquad \mathbf{d}_{m_i} \in \mathbf{v}. \end{aligned} \tag{5.9}$$

The length of $\mathbf{u}$, which is the result of concatenating M_R range blocks of size B each, is therefore:

$$N = M_R \cdot B. \tag{5.10}$$

Moreover, the concatenation of range blocks can also be written (using $\mathbf{r}_i(j)$ to mean the j-th coordinate of $\mathbf{r}_i$) as:

$$\mathbf{u}((i-1)\cdot B + j) = \mathbf{r}_i(j); \quad i = 1, \ldots, M_R, \qquad j = 1, \ldots, B. \tag{5.11}$$

Now the mechanism of computing $\mathbf{u} = W(\mathbf{v})$, when all the parameters describing W are known, can be described as shown in Table 5.1:

Table 5.1: Algorithm for Computing the Transformation $\mathbf{u} = W(\mathbf{v})$.

1. For $i = 1$ to M_R:
   (a) Extract the $\mathbf{d}_{m_i}$ block from the vector $\mathbf{v}$.
   (b) Compute
   $$\mathbf{r}_i = w_i(\mathbf{d}_{m_i}) = a_i\varphi(\mathbf{d}_{m_i}) + b_i\mathbf{1}_B\,. \tag{5.12}$$
2. *Concatenate* the range blocks thus obtained, $\mathbf{r}_i, i = 1, \ldots, M_R$, in the natural order, to get the new vector $\mathbf{u}$. The length of the vector $\mathbf{u}$ is $N = M_R \cdot B$.

The transformation W described above is called a *blockwise* transformation, the reason being evident from the computational algorithm.

So far the discussion of the w_i's was quite general. In order to make the discussion both more practical and lucid at this stage, we will make further restrictions and assumptions about the different parameters, shown in Table 5.2.

The description of the PIFS code of a vector, namely, the parameters that define W, can now be summarized. The PIFS code is shown in Table 5.3.

All other relevant parameters needed for decoding, such as $D = 2B$, $D_h = B$, $N = M_R B$, and others, are derived from the PIFS code using the previous assumptions.

The process of encoding, namely, the process of finding the M_R triplets of transformation parameters (a_i, b_i, m_i), is shown in Table 5.4. As stated, we seek to minimize $d^\infty(\boldsymbol{\mu}_o, \mathbf{f})$, where $\boldsymbol{\mu}_o$ is the original vector, and $\mathbf{f}$ is the fixed point of the sought transformation.

Since, by the Collage Theorem,

$$d^\infty(\boldsymbol{\mu}_o, \mathbf{f}) \le \frac{1}{1-s} d^\infty(\boldsymbol{\mu}_o, W(\boldsymbol{\mu}_o)), \tag{5.13}$$

one actually tries to minimize the upper bound on the right of Equation (5.13), by minimizing $d^\infty(\boldsymbol{\mu}_o, W(\boldsymbol{\mu}_o))$ instead of $d^\infty(\boldsymbol{\mu}_o, \mathbf{f})$. (Note that since s is the contraction factor of W, the factor $\frac{1}{1-s}$ depends on W. This factor, however, is not taken into account.) Though this method will not necessarily lead to a minimum of $d^\infty(\boldsymbol{\mu}_o, \mathbf{f})$, it is at present the most practical way of doing the coding. See [64]§4.4 for a statistical motivation for the minimization goal. Since W is a blockwise transformation, this minimization can be done in stages, as described below.

The minimization process to be described consists of finding a W that satisfies $\boldsymbol{\mu}_o \cong W(\boldsymbol{\mu}_o)$. Thus, $\boldsymbol{\mu}_o$ is approximately the fixed point of W. Since W uniquely defines its fixed point, storing W (by storing the parameters that define it) defines a lossy code for $\boldsymbol{\mu}_o$. Note that in this case both the operated-on vector and its image by the transformation W are assumed to be $\boldsymbol{\mu}_o$. Therefore, both $\mathbf{d}_{m_i} \in \boldsymbol{\mu}_o$ and $\mathbf{r}_i \in \boldsymbol{\mu}_o$.

This formulation is now demonstrated with a numerical example. Figures 5.1(a)-(b) present a vector $\boldsymbol{\mu}_o$ and its PIFS code. The PIFS is given in a table form. By performing the transformation described by the PIFS on $\boldsymbol{\mu}_o$, one can verify that in this example the vector $\boldsymbol{\mu}_o$ is a fixed

Table 5.2: Parameters, restrictions, and assumptions.

1. N - The size of the vector $\boldsymbol{\mu}_o$ to be encoded is an integer power of 2.
2. $B = 2^l$ - The size of a range block. B is therefore also some integer power of 2.
3. $D = 2B$ - The size of a domain block is twice the size of a range block.
4. $D_h = B$ - The value of D_h is defined to be the shift between consecutive domain blocks. Thus, the number of domain blocks is $M_D \equiv (\frac{N-D}{D_h} + 1)$, and each domain block is given by

$$\mathbf{d}_{m_i}(j) = \mathbf{v}((m_i - 1)D_h + j), \tag{5.14}$$

$$m_i = 1, 2, \ldots, M_D; \quad j = 1, 2, \ldots, D.$$

   Note that the domain blocks are overlapping, since $D_h < D$.

5. $\varphi(\cdot)$ - The spatial contraction function is defined to be:

$$\varphi(\mathbf{d}_{m_i})(j) \equiv \frac{1}{2}(\mathbf{d}_{m_i}(2j) + \mathbf{d}_{m_i}(2j - 1)), \tag{5.15}$$

   for $j = 1, 2, \ldots, B$. That is, $\varphi(\cdot)$ contracts blocks of size $D = 2B$ into blocks of size B by averaging pairs of adjacent elements in $\mathbf{d}_{m_i}$.

Table 5.3: PIFS code.

1. B - The size of the range blocks.
2. M_R - The number of range blocks.
3. M_R triplets of the transformation-parameters (a_i, b_i, m_i).

Table 5.4: PIFS Coding of μ_o .

1. Store B in the code file.
2. Store M_R in the code file, where $M_R = N/B$, and N is the length of μ_o.
3. Partition μ_o into M_R range blocks, as described in Equation (5.11),

$$\mathbf{r}_i(j) = \mu_o((i-1)\cdot B + j), \tag{5.16}$$

$$i = 1, \ldots, M_R, \qquad j = 1, \ldots, B.$$

4. Extract from μ_o the $M_D = (\frac{N-D}{D_h} + 1)$ domain blocks, according to Equation (5.14)

$$\mathbf{d}_l(j) = \mu_o((l-1)D_h + j), \tag{5.17}$$

$$l = 1, 2, \ldots, M_D, \qquad j = 1, 2, \ldots, D.$$

5. For $i = 1$ to M_R
   (a) Find the best parameters (a_i, b_i, m_i), such that

$$d^{\infty}(\mathbf{r}_i, a_i\varphi(\mathbf{d}_{m_i}) + b_i\mathbf{1}_B) \tag{5.18}$$

   is minimized.
   (b) Store the parameters (a_i, b_i, m_i) in the code file.

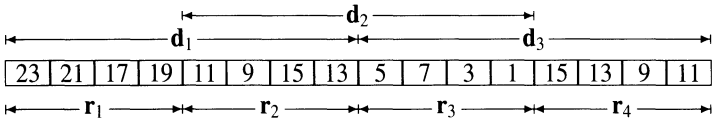
point of the transformation (namely, $\mu_o = W(\mu_o)$), and thus the coding in this case is lossless. Indeed, as shown in Figure 5.1(c), the result of computing the first code-line, namely, w_1, on μ_o, produces exactly $\mathbf{r}_1$.

Decoding

The process of decoding is straightforward, since it involves the finding of a fixed point of a contractive transformation W. This can be done by repeatedly iterating W on *any* initial vector until a desired proximity to the fixed point is reached [4].

In Table 5.5, the decoding of the PIFS code of Figure 5.1b is demonstrated, starting from an initial vector of all 0's.

Looking at the example in Figure 5.1, an important fact about the decoding (as well as the PIFS definition) can be observed. In the example, the PIFS code, with the prescribed $B = B_1 = 4$, resulted in a transformation $W^1 : \mathbb{R}^{16} \mapsto \mathbb{R}^{16}$ with fixed point $\mathbf{f}^1 \in \mathbb{R}^{16}$. Suppose, however, that the value of B is changed to some other value than the one prescribed in the PIFS



(a)

Range-block index i	Scale a_i	Domain-block index m_i	Offset b_i
1	0.5	1	12
2	0.5	3	8
3	0.5	2	0
4	0.5	1	4

$B = 4$, $M_R = 4$

(b)

$$\begin{aligned}\mathbf{r}_1 &= 0.5 \cdot \varphi(\mathbf{d}_1) + 12 = \\ &= 0.5 \cdot \varphi([23, 21, 17, 19, 11, 9, 15, 13]) + 12 \\ &= 0.5 \cdot [22, 18, 10, 14] + 12 \\ &= [23, 21, 17, 19].\end{aligned}$$

(c)

Figure 5.1: (a) An original vector μ_o. (b) The PIFS code of μ_o. (c) An example of computing $\mathbf{r}_1$ using the first code line in (b) and $\mathbf{d}_1 \in \mu_o$ given in (a).

code, e.g., $B = \frac{1}{2}B_1 = 2$. We have thus created a new transformation, denoted here by $W^{\frac{1}{2}}$, which is clearly a contractive transformation in $\mathbb{R}^8$. The decoding process of $W^{\frac{1}{2}}$ will therefore yield a fixed-point vector $\mathbf{f}^{\frac{1}{2}}$ of length $N = N_{\frac{1}{2}} = 8$, which is half the length of $\mathbf{f}^1$. Thus, we conclude that the PIFS can be decoded in different spaces, yielding a different fixed point in each space. Through the remainder of the chapter, we will keep using the notation W^q (and $\mathbf{f}^q$) to denote the transformation (and fixed point) which results from the PIFS code when using $B = qB_1$; where $B = B_1$ is the size of the original PIFS code range block corresponding to W^1 and $\mathbf{f}^1$. In Figures 5.2(a)-(c), three fixed points of the same PIFS code (using different B's) are described, each one being in a different space – $\mathbb{R}^{16}$, $\mathbb{R}^8$, and $\mathbb{R}^4$, respectively.

The exact relation between these different fixed points, and its interpretation, is actually the main subject of this chapter and is examined in detail next.

Table 5.5: Decoding by iterations.

iter	vector															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	12	12	12	12	8	8	8	8	0	0	0	0	4	4	4	4
2	18	18	16	16	8	8	10	10	4	4	0	0	10	10	8	8
3	21	20	16	17	10	8	13	12	4	5	2	0	13	12	8	9

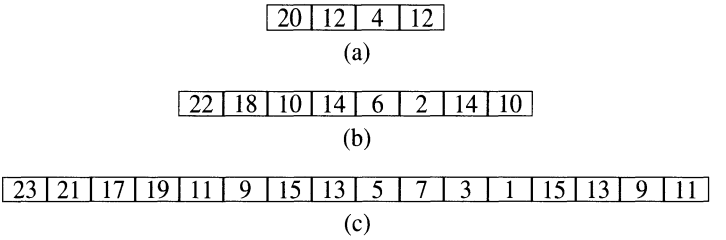


Figure 5.2: Decoding of the PIFS code of Figure 5.1 with (a) $B = 1$, (b) $B = 2$, and (c) $B = 4$.

5.2 Hierarchical Interpretation

As described in the previous section, each value of B leads to a different transformation with a different fixed point. The following theorem describes the relation between two different fixed points that arise when B is halved.

Theorem 5.1 (Zoom) *Given a PIFS code that leads to W^1 with $B = B_1$, and to $W^{\frac{1}{2}}$ with $B = B_1/2$:*

1. **Zoom-out:** *Let $\mathbf{f}^1$ be the fixed point of W^1. Then the fixed point $\mathbf{f}^{\frac{1}{2}}$ of $W^{\frac{1}{2}}$ is given by:*

$$\mathbf{f}^{\frac{1}{2}}(j) = \frac{1}{2}\left\{\mathbf{f}^1(2j) + \mathbf{f}^1(2j-1)\right\}, \qquad j = 1, \ldots, \frac{N_1}{2}, \tag{5.19}$$

   where $N_1 \equiv M_R \cdot B_1$.

2. **Zoom-in:** *Let $\mathbf{f}^{\frac{1}{2}}$ be the fixed point of $W^{\frac{1}{2}}$. Then the fixed point $\mathbf{f}^1$ of W^1 is given by:*

$$\mathbf{f}^1((i-1)B_1 + j) = a_i \mathbf{f}^{\frac{1}{2}}((m_i - 1)D_h{}^{\frac{1}{2}} + j) + b_i, \tag{5.20}$$

$$i = 1, \ldots, M_R, \qquad j = 1, \ldots, B_1,$$

   where $D_h{}^{\frac{1}{2}} \equiv \frac{D_h{}^1}{2} = \frac{B_1}{2}$.

The proof of the theorem is given in Addendum A at the end of the chapter.

An interpretation of the theorem is as follows: In order to compute each element of $\mathbf{f}^{\frac{1}{2}}$ from a given $\mathbf{f}^1$, one has to take the average of two adjacent elements in $\mathbf{f}^1$, as described in Equation (5.19). On the other hand, in order to compute $\mathbf{f}^1$ from a given $\mathbf{f}^{\frac{1}{2}}$, one follows Equation (5.20), which is similar to computing W^1 itself (compare with Equation (5.12)).

Theorem 5.1 establishes a relation between the pair $\mathbf{f}^1$ and $\mathbf{f}^{\frac{1}{2}}$. The same relation is carried over to the pair $\mathbf{f}^{\frac{1}{2}}$ and $\mathbf{f}^{\frac{1}{4}}$,

$$\mathbf{f}^{\frac{1}{4}}(j) = \frac{1}{2}\left\{\mathbf{f}^{\frac{1}{2}}(2j) + \mathbf{f}^{\frac{1}{2}}(2j-1)\right\}, \tag{5.21}$$

$$\mathbf{f}^{\frac{1}{2}}((i-1)B_{\frac{1}{2}} + j) = a_i \mathbf{f}^{\frac{1}{4}}((m_i - 1)D_h{}^{\frac{1}{4}} + j) + b_i\ . \tag{5.22}$$

The relation also holds for the pair $\mathbf{f}^{\frac{1}{4}}$ and $\mathbf{f}^{\frac{1}{8}}$, and so on. The collection of fixed points can thus be described in terms of a hierarchical structure, as was shown in Figure 5.2. This structure is a *pyramid of the PIFS fixed points*, with $\mathbf{f}^{1/2^p}$ comprising the p-th level of the pyramid. Thus, $\mathbf{f}^{1/2^p}$ has $N_1/2^p$ elements, and $\mathbf{f}^1$ corresponds to $p = 0$ and is of length N_1. The level with the coarsest resolution (Figure 5.2c) is called the *top level.*

The relations between two adjacent levels in the pyramid can be summarized as follows:

1. In order to go *up* the pyramid, from level p to level $p + 1$, the operation is as follows (zoom-out):

$$\mathbf{f}^{1/2^{p+1}}(j) = \frac{1}{2}\left\{\mathbf{f}^{1/2^p}(2j) + \mathbf{f}^{1/2^p}(2j-1)\right\}, \tag{5.23}$$

$$j = 1, \ldots, N_1/2^{p+1},$$

and is identical to Equation (5.19) for $p = 0$.

2. In order to go *down* the pyramid, from level $p + 1$ to level p, the operation is as follows (zoom-in):

$$\mathbf{f}^{1/2^p}((i-1)B_{1/2^p} + j) = a_i \mathbf{f}^{1/2^{p+1}}((m_i - 1){D_h}^{1/2^{p+1}} + j) + b_i, \tag{5.24}$$

$$\mathrm{i} = 1, \ldots, M_R, \qquad j = 1, \ldots, B_{1/2^p},$$

where ${D_h}^{1/2^{p+1}} \equiv {D_h}^1/2^{p+1} = B_1/2^{p+1}$. Here, Equation (5.20) is obtained for $p = 0$.

An intuitive understanding of the process can be gained by noting that the domain blocks of $\mathbf{f}^1$, after contraction, are actually blocks which are contained in $\mathbf{f}^{\frac{1}{2}}$. Formally, this can be shown by writing down the expression for the l-th element in the contraction of the m_i-th domain block of $\mathbf{f}^1$ (Equations (5.14)-(5.15)):

$$\varphi({\mathbf{d}_{m_i}}^1)(l) = \frac{1}{2}\left\{\mathbf{f}^1((m_i - 1)D_{h1} + 2l - 1) + \mathbf{f}^1((m_i - 1)D_{h1} + 2l)\right\} \tag{5.25}$$

$$l \in 1, \ldots, B_1.$$

According to Equation (5.19), the right-hand side is *exactly* $\mathbf{f}^{\frac{1}{2}}((m_i - 1)D_{h1}/2 + l)$, and if we further denote, as before, $D_h{}^{\frac{1}{2}} \equiv D_{h1}/2$ and $\mathbf{d}_{m_i}{}^{\frac{1}{2}}$ as the m_i-th domain block of $\mathbf{f}^{\frac{1}{2}}$, we conclude that:

$$\varphi({\mathbf{d}_{m_i}}^1)(l) = \mathbf{d}_{m_i}{}^{\frac{1}{2}}(l), \qquad l \in 1, \ldots, B_1\,. \tag{5.26}$$

The question arises: what is the smallest size of the top level such that the above relations between two adjacent levels still hold? This is answered by the following corollary.

Corollary 5.1 *Let* $\mathbf{f}^1 \in \mathbb{R}^N$ *be a fixed point of a given PIFS, and let* $B = B_1 = 2^l$, $D = D_1 = 2B_1$, *and* $D_h = B_1$. *Then, the number of levels in the pyramid of PIFS fixed points is*

$$\log_2(B_1) + 1 = l + 1, \tag{5.27}$$

leading to a top-level size of $N/2^l (= M_R)$.

Proof: Ascending one level in the hierarchy means halving the size of the range block B. Since this size must be at least 1 in order that the PIFS can be applied, the corollary follows. ∎

Table 5.6 summarizes the notation.

In some cases, the top level is directly derivable from the code. For example, if the coding procedure is modified so that domains are orthogonalized with respect to DC (monotone) blocks, as in Chapter 8, then the top level is the DC value associated with each range block. Further relation between this method and the method of Chapter 8 can be found in [65].

Table 5.6: Pyramid of fixed points: Notation summary.

Level-num. p	Range-block size B	Num. of elements N	fixed point $\mathbf{f}$
0	B_1	$N_1 = M_R \cdot B_1$	$\mathbf{f}^1$
1	$B_{1/2} = B_1/2$	$N_{1/2} = N_1/2$	$\mathbf{f}^{1/2}$
⋮	⋮	⋮	⋮
$\log_2 B_1 = l$ (top level)	$B_{1/B_1} = 1$	$N_{1/B_1} = M_R$	$\mathbf{f}^{1/B_1}$

5.3 Matrix Description of the PIFS Transformation

In this section, a matrix description for the PIFS transformation W described above is given. We assume that a PIFS code is given. This code includes all the necessary information for computing W, as described in Section 5.1.

The transformation $\mathbf{u} = W(\mathbf{v})$ is composed of $\{w_i\}_{i=1}^{M_R}$, where each w_i operates on a domain block. The j-th element of the i-th range block of $\mathbf{u}$ is, according to Equation (5.11):

$$\mathbf{r}_i(j) = \mathbf{u}((i-1)\cdot B + j)\,; \quad i = 1, \ldots, M_R\,; \quad j = 1, \ldots, B. \tag{5.28}$$

This element is, according to Equation (5.12), the result of transforming the appropriate domain block, as given by the PIFS code:

$$\mathbf{r}_i(j) = a_i \cdot \varphi(\mathbf{d}_{m_i})(j) + b_i\,. \tag{5.29}$$

Substituting the definition of φ given in Equation (5.15) and using Equation (5.14) gives

$$\mathbf{r}_i(j) = \frac{a_i}{2}\left(\mathbf{v}((m_i-1)D_h + 2j) + \mathbf{v}((m_i-1)D_h + 2j - 1)\right) + b_i\,. \tag{5.30}$$

Writing the last equations in a matrix form for all the elements in $\mathbf{u}$ yields:

$$\mathbf{u} = \mathbf{F}\mathbf{v} + \mathbf{b}\,, \tag{5.31}$$

where $\mathbf{F}$ and $\mathbf{b}$ are described below:

- **b** - An $N \times 1$ offset vector. It is a block vector, composed of (N/B) basic blocks of length B,

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_{N/B} \end{bmatrix}, \tag{5.32}$$

where,

$$\mathbf{b}_i = \begin{bmatrix} b_i \\ \vdots \\ b_i \end{bmatrix}, \qquad i = 1, 2, \ldots, \frac{N}{B} \tag{5.33}$$

is a $B \times 1$ vector.

- **F** - An $N \times N$ transfer matrix, given by :

$$\mathbf{F} = \mathbf{A} \cdot \frac{1}{2} \cdot \mathbf{D} . \tag{5.34}$$

In this decomposition of **F** we have:

- **D** - A domain-location matrix. The product $\frac{1}{2}\mathbf{D} \cdot \mathbf{v}$ plays the role of $\varphi(\mathbf{d}_{m_i})$, and therefore has a dual role:

  1. Extract $\mathbf{d}_{m_i}$ from **v**.
  2. Perform the spatial contraction operation on $\mathbf{d}_{m_i}$.

  D is a block matrix with a structure that is best demonstrated by the example below. In this example $\mathbf{d}_3$ is transformed to $\mathbf{r}_1$, $\mathbf{d}_1$ to $\mathbf{r}_2$, and $\mathbf{d}_4$ to $\mathbf{r}_3$

$$\mathbf{D} = \begin{bmatrix} \mathbf{0}_B & \mathbf{0}_B & \boxed{\mathbf{D}_{B\times D}} & \mathbf{0}_B & \mathbf{0}_B & \mathbf{0}_B & \mathbf{0}_B & \cdots \\ \boxed{\mathbf{D}_{B\times D}} & \mathbf{0}_B & \mathbf{0}_B & \mathbf{0}_B & \mathbf{0}_B & \mathbf{0}_B & \mathbf{0}_B & \cdots \\ \mathbf{0}_B & \mathbf{0}_B & \mathbf{0}_B & \boxed{\mathbf{D}_{B\times D}} & \mathbf{0}_B & \mathbf{0}_B & \mathbf{0}_B & \cdots \\ \vdots & & \vdots & & \vdots & & \vdots & \ddots \end{bmatrix}_{N\times N} \tag{5.35}$$

  The matrix $\mathbf{0}_B$ denotes an all-zero matrix of size $B \times B$. The location of the $B \times D$ blocks $\mathbf{D}_{B\times D}$ is determined by the indices m_i, whereas the spatial contraction is performed by its elements, as explained below:

  1. If domain block m_i is transformed to range block i, then a block matrix $\mathbf{D}_{B\times D}$ would be positioned with top left element at

$$((i-1)B + 1, (m_i - 1)D_h + 1)). \tag{5.36}$$

2. The spatial contraction mapping is performed by $\mathbf{D}_{B\times D}$, which has the following structure:

$$\mathbf{D}_{B\times D} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & & \cdots \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & & \cdots \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & & \cdots \\ \vdots & & & & & & \ddots & & \vdots \\ 0 & 0 & & & & & 0 & 1 & 1 \end{bmatrix}_{B\times D} \tag{5.37}$$

Namely, $\frac{1}{2}\mathbf{D}_{B\times D}$ maps blocks of size D to blocks of size B by *averaging* every two adjacent elements.

- $\mathbf{A}$ - A scaling matrix. It is a block-diagonal matrix:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{0}_B & \mathbf{0}_B & \cdots \\ \mathbf{0}_B & \mathbf{A}_2 & \mathbf{0}_B & \cdots \\ \mathbf{0}_B & \mathbf{0}_B & \mathbf{A}_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}_{N\times N} \tag{5.38}$$

where

$$\mathbf{A}_i = a_i \cdot \mathbf{I}_{B\times B}, \qquad i = 1, 2, \ldots, \frac{N}{B}. \tag{5.39}$$

The matrix $\mathbf{I}_{B\times B}$ denotes the $B \times B$ identity matrix. If, for a certain code, $a_i = a$ for every i, then $\mathbf{A} = a\mathbf{I}_{N\times N}$.

The size of each of the matrices $\mathbf{F}, \mathbf{A}, \mathbf{D}$, as well as the size of the vector $\mathbf{b}$, are seen to depend directly on B. Indeed, as we have already seen before, changing the value of B results in a different W and therefore leads also to different matrices, though their *basic structure* remains the same. For example, the matrix form of the PIFS code described in Figure 5.1(b) with $B = 2$ is described in Figure 5.3.

5.4 Fast Decoding

A fast decoding method, which we call *hierarchical decoding*, follows directly from the hierarchical interpretation of the PIFS code. In this method, one begins by computing the *top level*. This can be done either by computing it directly, when possible (for example, with the method of Chapter 8), or by iterating the PIFS with $B = 1$, that is, by applying W to an initial vector of size M_R until a fixed point is reached (or closely approximated). Then, one follows the deterministic algorithm Equation (5.24) to advance to a higher resolution. The process of advancing to a higher resolution is repeated until the desired vector size is achieved. This method is compared below with the conventional *iterative decoding* method [48], where the iterations are done on a *full-scale* image. The computational savings obtained in using the hierarchical decoding method stems mainly from the fact that the iterations are done *only* in order to find the top level, which is a small-size vector.

$$\mathbf{A} = \tfrac{1}{2} \cdot \mathbf{I}_{8\times 8}, \qquad \mathbf{b} = [12, 12, 8, 8, 0, 0, 4, 4]^T$$

$$\mathbf{D} = \begin{bmatrix}
\boxed{\begin{matrix}1&1\\0&0\end{matrix}}\ \boxed{\begin{matrix}0&0\\1&1\end{matrix}} & \begin{matrix}0&0\\0&0\end{matrix} & \begin{matrix}0&0\\0&0\end{matrix} & \mathbf{d}_1 \rightarrow \mathbf{r}_1 \\
\begin{matrix}0&0\\0&0\end{matrix} & \boxed{\begin{matrix}1&1\\0&0\end{matrix}}\ \boxed{\begin{matrix}0&0\\1&1\end{matrix}} & \begin{matrix}0&0\\0&0\end{matrix} & \mathbf{d}_3 \rightarrow \mathbf{r}_2 \\
\begin{matrix}0&0\\0&0\end{matrix} & \begin{matrix}0&0\\0&0\end{matrix} & \boxed{\begin{matrix}1&1\\0&0\end{matrix}}\ \boxed{\begin{matrix}0&0\\1&1\end{matrix}} & \mathbf{d}_2 \rightarrow \mathbf{r}_3 \\
\boxed{\begin{matrix}1&1\\0&0\end{matrix}}\ \boxed{\begin{matrix}0&0\\1&1\end{matrix}} & \begin{matrix}0&0\\0&0\end{matrix} & \begin{matrix}0&0\\0&0\end{matrix} & \mathbf{d}_1 \rightarrow \mathbf{r}_4
\end{bmatrix}$$

Figure 5.3: The matrices **A** and **D**, and the vector **b** describing the PIFS code of Figure 5.1, for $B = 2$.

Computational Cost

A detailed discussion of the computational cost, for both the 1-dimensional (1-D) and 2-dimensional (2-D) cases, is given in [2]. Here we concentrate on the simple 1-D case, assuming a floating point processor (which means that addition and multiplication operations take the same time to perform).

The following notation and definitions are used:

t_{op} - the computation time of a summation or a multiplication operation.

t_{tot} - the total computation time.

I - the number of iterations.

In arriving at the following results, multiplications by $\frac{1}{2}$ or $\frac{1}{4}$ were not counted (counting them shows even a greater savings in using the hierarchical decoding, because they occur mainly in the iterative method, when applying φ).

With $N = N_1$ denoting the input vector length and $B = B_1$ the range block size, we have:

- *Iterative decoding* - Referring to Equations (5.12) and (5.15), for a single iteration, the computation time is:

$$N_1 \cdot [(1 + 1)\text{sums} + 1\text{mult.s}] = N_1 \cdot 3t_{op} . \tag{5.40}$$

  Thus, the total computation time is:

$$t^i_{tot} = I \cdot N_1 \cdot 3t_{op} . \tag{5.41}$$

- *Hierarchical decoding* - Recall that at the top level there are M_R elements (by Corollary 5.1). Hence, according to the result in Equation (5.41), we know the computation time needed to compute the top level. According to Corollary 5.1, there are $\log_2(B_1)$

levels in the pyramid, excluding the top level, with $N_1/2^p$ elements in the p-th level ($p = \log_2 B_1$ at the top level). Referring to Equation (5.24), we can compute the cost of transforming from the top level to $\mathbf{f}^1$:

$$\sum_{p=1}^{\log_2(B_1)} \left(2^p \frac{N_1}{B_1}\right) \cdot [1\text{sums} + 1\text{mult.s}] = (B_1 - 1)\frac{N_1}{B_1} 2 \cdot 2t_{op} \approx N_1 \cdot 4t_{op} \,. \tag{5.42}$$

The total computation time is therefore

$$t_{tot}^h \approx N_1 \cdot \left(\frac{3I}{B_1} + 4\right) \cdot t_{op} \,. \tag{5.43}$$

Thus, the ratio of computation times is :

$$Q \equiv \frac{t_{tot}^h}{t_{tot}^i} = \frac{(\frac{3I}{B_1} + 4)}{3I} = \left(\frac{1}{B_1} + \frac{4}{3I}\right). \tag{5.44}$$

It is seen that the larger the values of I and B_1, the more advantageous the hierarchical decoding method is. For example, for a typical case in which $I = 8$, $B_1 = 8$, one gets $Q = 0.3$.

In [2] it is shown that for the 2-D case (assuming ${B_1}^2 \gg I$),

$$Q_{2-D} = \frac{8}{15 \cdot I} \,. \tag{5.45}$$

Thus, the savings can reach more than an order of magnitude.

5.5 Super-resolution

The subject of resolution is inherently related to the discretization of a function of a continuous variable. The process of discretization is called sampling. In the following definition we define a specific method of sampling.

Definition 5.1 *Given a function $G(x) \in L^\infty[0, 1]$, define $G_r(i)$ by*

$$G_r(i) \equiv r \int_{\frac{(i-1)}{r}}^{\frac{i}{r}} G(x)\, dx, \qquad i = 1, \ldots, r \,, \tag{5.46}$$

which is the function $G(x)$ at resolution r. *We say that $G_{r_1}(i)$ is* finer (*i.e., having higher resolution*) *than $G_{r_2}(i)$ (which is* coarser) *if $r_1 > r_2$.*

The following theorem introduces the new notion of the PIFS embedded function and relates it to the PIFS fixed point.

Theorem 5.2 (PIFS Embedded Function) *Given a PIFS code, there exists a unique function $G(x) \in L^\infty[0, 1]$ such that a vector $\mathbf{v}_N \in \mathbb{R}^N$ is a fixed point of the PIFS iff it is equal to the function $G(x)$ at resolution $r = N, \forall N$, i.e.,*

$$\mathbf{v}_N(j) = G_N(j), \qquad j = 1, 2, \ldots, N. \tag{5.47}$$

The function $G(x)$ is called the PIFS embedded function.

The theorem is proved in Addendum B.

In Figure 5.4, an intuitive demonstration of the PIFS embedded function theorem is given. The PIFS is the one described previously in Figure 5.1. Its fixed points, for $B = 1$, $B = 2$, and $B = 4$ are shown in Figure 5.4 as functions of a continuous variable $x \in [0, 1]$. For example, the fixed point using $B = 1$ is the vector $[20, 12, 4, 12]$, which has $N = 4$ elements. Thus, the fixed point is drawn as a piecewise constant function:

$$f(x) = \begin{cases} 20 & x \in [0, \frac{1}{4}) \\ 12 & x \in [\frac{1}{4}, \frac{1}{2}) \\ \vdots & \vdots \end{cases} \tag{5.48}$$

The embedded function $G(x)$ is also shown. One easily sees that the fixed points "approach" the PIFS embedded function. Moreover, it is seen that the value of each function, in each of its intervals, equals the mean of the PIFS embedded function over the appropriate interval as described by Equations (5.46) and (5.47).

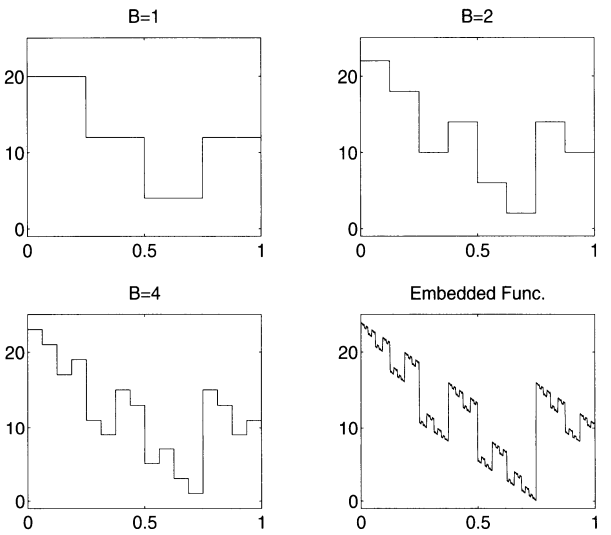


Figure 5.4: Fixed points for $B = 1$, $B = 2$, $B = 4$, and the corresponding PIFS embedded function.

Super-resolution deals with finding a higher resolution of a given discrete signal [73]. For example, suppose that a vector $\mathbf{v}^1$ with elements $\mathbf{v}^1(i) = h_N(i)$, $i = 1, 2, \ldots, N$ is given, where $h_N(i)$ is the unknown signal $h(x) \in L^\infty[0, 1]$ at resolution N. The goal is to find a vector $\mathbf{f}^2$ of length $2N$, which approximates the signal $h(x)$ at resolution $2N$, namely, $h_{2N}(i)$, $i = 1, 2, \ldots, 2N$. The process of transforming from a given resolution to a higher one is also called zoom-in (or simply *zooming*).

The hierarchical representation suggests a simple method for finding higher-resolution representations from a given resolution. Given $\mathbf{v}^1$, we can find its PIFS code. This PIFS code

will be, in general, a lossy one. Thus, its fixed point $\mathbf{f}^1$ will only be an approximation of $\mathbf{v}^1$. The PIFS code enables us to build a hierarchical structure, which we called the pyramid of fixed points. The PIFS code also gives us an algorithm for relating two adjacent levels in the pyramid (zoom theorem; Equations (5.19)-(5.20)). Thus, after finding the PIFS code and $\mathbf{f}^1$, all that is needed in order to get a vector $\mathbf{f}^2$ of length $2N$ is to apply the zoom-in algorithm in Equation (5.24) to $\mathbf{f}^1$. The vector $\mathbf{f}^2$ is an integral part of the pyramid of fixed points and is the fixed point of the PIFS code when using $B = 2B_1$. This vector can be used as an approximation of the higher-resolution representation, namely, an approximation of h_{2N}.

The Fractal Dimension of the PIFS Embedded Function

We have already introduced the PIFS embedded function. It is of interest to note that its *fractal dimension* can be bounded directly from the matrix representation of the PIFS code, as described by the following theorem (see also [5]).

Theorem 5.3 (Dimension Bound of the PIFS Embedded Function) *: Given a PIFS code W, let $\mathbf{F}, \mathbf{A}, \mathbf{D}$, and $\mathbf{b}$ denote its matrix representation, using $B = 1$ (see Section 5.3), such that*

$$W(\mathbf{v}) = \mathbf{F}\mathbf{v} + \mathbf{b} = \left(\mathbf{A} \frac{1}{2} \mathbf{D}\right) \mathbf{v} + \mathbf{b}. \tag{5.49}$$

Let $G(x)$ denote the related PIFS embedded function. The fractal dimension of $G(x)$ is $\mathcal{D}$ where

$$1 \leq \mathcal{D} \leq 1 + \log_2(\lambda). \tag{5.50}$$

and where λ is the largest real eigenvalue of the matrix $(\mathbf{A}_{abs} \cdot \mathbf{D})$, and $\mathbf{A}_{abs}$ denotes a matrix whose elements are the absolute values of the corresponding elements in $\mathbf{A}$.

The proof of the theorem is given in Addendum C.

The importance of the fractal dimension stems from the fact that it tells us about the nature of the super-resolution vectors (see [4] for a detailed discussion concerning fractal interpolation functions). By introducing certain constraints in the coding procedure (for example, on $\frac{D}{B}$ and/or $|a_i|$), one can change the fractal dimension of the resultant PIFS embedded function. Thus, the above-described super-resolution method results in fractal curves (vectors) with a fractal dimension which can be computed from the PIFS code and can be affected by the constraints on transformation parameters.

An application of the super-resolution method is demonstrated in Figure 5.5, in which one sees the "fractal nature" of the interpolation. A vector of size $N_1 = 256$ serves as the original vector, namely, $\mathbf{v}^1$. This vector was used to determine a PIFS code having a fixed point $\mathbf{f}^1$. The fractal dimension of the PIFS embedded function was found (Theorem 5.3) to be 1.16. This PIFS code was then used to find a vector $\mathbf{f}^4$ of length $4 \cdot N_1 = 1024$. The first 32 elements of $\mathbf{f}^4$ are shown ("+" combined with a dash-dot line) on the same graph with the linear-interpolation of the first 8 elements of $\mathbf{v}^1$ ("$\circ$" combined with a dotted line).

The following features of the interpolation are observed:

1. The *mean* of each 4-elements of $\mathbf{f}^4$ is approximately equal to the appropriate element of $\mathbf{v}^1$ (a strict equality would hold if the coding was lossless, i.e., $\mathbf{f}^1 = \mathbf{v}^1$).

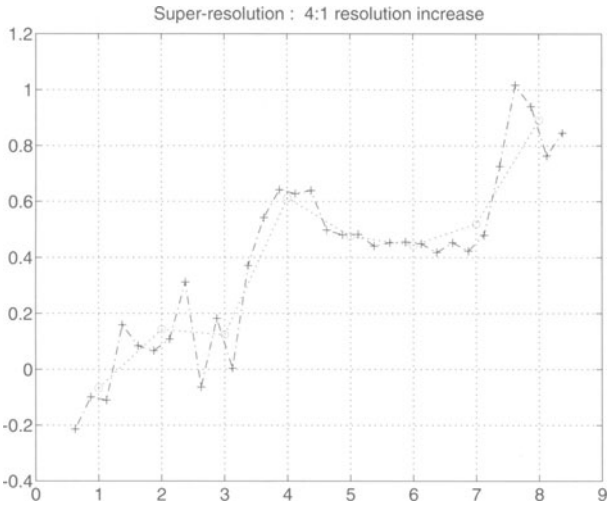


Figure 5.5: Super-resolution via PIFS code ("+" and a dash-dot line) versus linear interpolation ("○" and a dotted line).

2. The interpolation function, introduced by the super-resolution method, *does not* necessarily pass through the given points of $\mathbf{v}^1$. Furthermore, if we try to compute $\mathbf{f}^8$, it will not necessarily pass through the points of $\mathbf{f}^4$.

3. Linear interpolation tends to smooth the curve, while the PIFS code interpolation preserves the fractal dimension of the curve, which ensures the richness of details even at high resolutions. This feature is most evident when dealing with textures: while linear interpolation typically results in a blurred texture, the PIFS code interpolation preserves the appearance of the texture.

Rational Zoom Factors

After describing the method for achieving super-resolution, the following question arises:

> According to the described method, one can only get resolutions which are N_1 times a power of 2, i.e., $N_1,\ 2N_1,\ 4N_1, \ldots$. Can one also achieve different resolutions as well, such as $\frac{3}{2}N_1$, for example?

This question is addressed below.

In the description of the decoding process of a PIFS code (see Section 5.1), the possibility of replacing the prescribed $B = B_1$ by a new value was suggested. In the construction of the pyramid of fixed points, if super-resolution is desired, the new value of B is taken as $B_2 = 2B_1$. This yields a fixed point of length $N_2 = 2N_1 = 2B_1M_R$. Suppose, however, that we choose to

take $B = B_1 + 1$, which we define as $B_{(B_1+1)/B_1}$. This, in turn, yields a fixed point of length

$$N_{(B_1+1)/B_1} \equiv B_{(B_1+1)/B_1} \cdot M_R = N_1 + M_R \,. \tag{5.51}$$

Similarly, taking $B = B_{(B_1+2)/B_1} = B_1 + 2$ leads to

$$N_{(B_1+2)/B_1} \equiv B_{(B_1+2)/B_1} \cdot M_R = N_1 + 2M_R \,. \tag{5.52}$$

Pursuing the same reasoning, we see that we get a whole range of resolutions, quantized in steps of M_R. These resolution levels yield a pyramidal structure, which we call a *rational pyramid.* Each level of the rational pyramid is a fixed point of the PIFS, and the earlier pyramid of fixed points is contained in this rational pyramid.

For example, Figure 5.6(a)-(c) demonstrates the decoding of the PIFS code given in Figure 5.1, with $B = 4$, $B = 3$, and $B = 2$, respectively. In this case, $M_R = 4$.

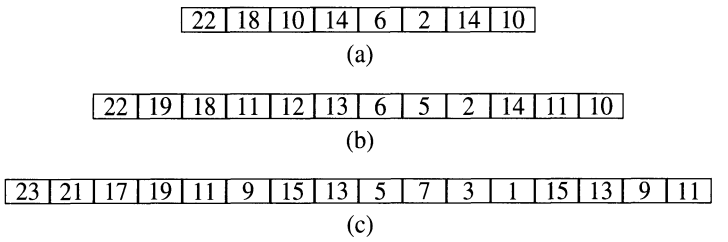


Figure 5.6: Decoding with (a)$B = 2$, (b) $B = 3$ (approximated to the nearest integer value), and (c) $B = 4$.

It is easily shown, by the use of Theorem 5.2, that given the PIFS code and some level of the rational pyramid, one can compute any other level of the rational pyramid, though not necessarily directly. For example, given the PIFS code and $\mathbf{f}^1$ that corresponds to $B = B_1 = 3$, what is the algorithm for computing $\mathbf{f}^{\frac{4}{3}}$ which corresponds to $B = 4$? The method is to first compute $\mathbf{f}^2$ from $\mathbf{f}^1$ and the given PIFS code, according to Equation (5.24). Here, $\mathbf{f}^2$ corresponds to $B = 6$. Then, compute $\mathbf{f}^4$ in the same manner, with $\mathbf{f}^4$ corresponding here to $B = 12$. Each element of $\mathbf{f}^{\frac{4}{3}}$ is now directly computable by averaging of every three consecutive elements from $\mathbf{f}^4$ (this follows directly from Theorem 5.2). The method is summarized as follows (see also Equation (5.46)):

1. Compute $\mathbf{f}^4$ from $\mathbf{f}^1$ and from the PIFS code, by twice applying Equation (5.24) (note that p can be negative).

2. Compute $\mathbf{f}^{\frac{4}{3}}$ by:

$$\mathbf{f}^{\frac{4}{3}}(k) = \frac{1}{3}(\mathbf{f}^4(3k) + \mathbf{f}^4(3k-1) + \mathbf{f}^4(3k-2)), \tag{5.53}$$

$$k = (1, 2, \ldots, M_R \cdot (B_1 + 1)), \text{ where } \quad M_R \cdot (B_1 + 1) = M_R \cdot 4.$$

5.6 Different Sampling Methods

Let $G(x) \in L^\infty[0, 1]$ denote a given embedded function of some PIFS code, and let $g_N(i)$, $i = 1, \ldots, N$, denotes its sampling by integration at resolution N, according to Equation (5.46), i.e.,

$$g_N(i) \equiv N \int_{(i-1)\frac{1}{N}}^{i\frac{1}{N}} G(x)\, dx, \qquad i = 1, \ldots, N. \tag{5.54}$$

As stated in Theorem 5.2, $g_N(i)$ is a fixed point of the corresponding PIFS, with a contraction function φ as defined in Equation (5.15):

$$\varphi(\mathbf{d}_l)(j) = \frac{1}{2}(\mathbf{d}_l(2j) + \mathbf{d}_l(2j-1)). \tag{5.55}$$

Finding the PIFS code for g_N with this φ would result in the correct PIFS (the one with embedded function $G(x)$), and the coding would be lossless. There could be cases where more than one PIFS code has the same fixed-point vector at resolution N, but these codes correspond to different embedded functions. Such cases are not of interest here, so we will not consider such situations in remaining discussion.

Suppose, however, that now $G(x)$ is sampled in a different manner, denoted *point sampling*, to create $g_N^s(i)$;

$$g_N^s(i) \equiv G\left((i-1)\frac{1}{N}\right), \qquad i = 1, \ldots, N. \tag{5.56}$$

All the previous results will be shown to hold here too if we define:

$$\varphi^s(\mathbf{d}_l)(j) \equiv \mathbf{d}_l(2j-1), \tag{5.57}$$

meaning that φ^s is now also a point sampling operation. An earlier work on the subject can be found in [53], where a method to compute discrete values of a fractal function is discussed.

Now, let us restate few of the previous theorems and definitions, for this case of point sampling (the proofs are omitted since they closely follow the previous ones):

Theorem 5.4 (Point-Sampling Zoom) *Given a PIFS code with $\varphi^s(\mathbf{d}_l)(j) \equiv \mathbf{d}_l(2j-1)$, which leads to W^1 with $B = B_1$, and to $W^{\frac{1}{2}}$ with $B = B_1/2$. Let the fixed points of these transformations be $\mathbf{f}^1$ and $\mathbf{f}^{\frac{1}{2}}$, respectively, then:*

1. **Zoom-out:**

$$\mathbf{f}^{\frac{1}{2}}(j) = \mathbf{f}^1(2j-1), \qquad j = 1, \ldots, M_R(B_1/2). \tag{5.58}$$

2. **Zoom-in:**

$$\mathbf{f}^1((i-1)B_1 + j) = a_i \mathbf{f}^{\frac{1}{2}}((m_i - 1)D_h{}^{\frac{1}{2}} + j) + b_i, \tag{5.59}$$

$$i = 1, \ldots, M_R, \qquad j = 1, \ldots, B_1. \tag{5.60}$$

where $D_h{}^{\frac{1}{2}} \equiv \frac{D_h{}^1}{2} = \frac{B_1}{2}$.

Definition 5.2 *Let* $G(x) \in L^\infty[0,1]$. *Define* $G_r^s(i)$ *by*

$$G_r^s(i) \equiv G\left((i-1)\frac{1}{r}\right), \qquad i = 1, \ldots, r. \tag{5.61}$$

$G_r^s(i)$ *denotes the* function $G(x)$ at point sampling resolution r. *We say that* $G_{r_1}^s(i)$ *is* finer (*i.e., having higher resolution*) *than* $G_{r_2}^s(i)$ (*which is* coarser) *if* $r_1 > r_2$.

Theorem 5.5 (PIFS Embedded Function) [1] *Given a PIFS code with* $\varphi^s(\mathbf{d}_l)(j) \equiv \mathbf{d}_l(2j-1)$, *there exists a unique function* $G(x) \in L^\infty[0,1]$ *such that a vector* $\mathbf{v}_N{}^s \in \mathbb{R}^N$ *is a fixed point of the PIFS iff it is equal to the function* $G(x)$ *at point sampling resolution* $r = N$, *i.e.,*

$$\mathbf{v}_N{}^s(j) = G_N^s(j), \qquad j = 1, 2, \ldots, N. \tag{5.62}$$

The function $G(x)$ *is called the* PIFS embedded function.

The importance of formulating and treating different sampling methods lies in the fact that the coding problem can usually be stated as follows: One has a model for a continuous signal, be it a voice signal, an image, etc. The model states that the continuous signal is of fractal type. This signal is then sampled, and the work of finding the code is performed on the discrete signal.

As we have shown, the proper method for coding, namely, the method which enables finding the correct code, depends on the sampling method. So, given a model and the method of sampling, one can use the corresponding coding method.

It is worth noting that the embedded function for *both* sampling methods is the same. This should not be surprising, since as N grows, $g_N^s(i)$ and $g_N(i)$ approach each other, as seen from the definitions in Equations (5.56) and (5.54).

5.7 Conclusions

This chapter describes a hierarchical interpretation of the PIFS coding problem. Decoding the PIFS code at different resolutions results in a *pyramid of fixed points*. The ideas presented suggest many directions for future research. Some of them are:

- *Fractal image-interpolation* - This subject was briefly described in Figure 5.5.
- *Tighter Collage Theorem bound* - The Collage Theorem 2.1 (see also [4]) uses the distance between the image and its collage to bound the distance between the image and the related fixed point. However, the pyramidal structure suggests that the distance between the image and its collage in *different resolutions* should also be considered. Namely, instead of using

  $$d(\boldsymbol{\mu}_o^1, \mathbf{f}^1) \le \frac{1}{1-s} d(\boldsymbol{\mu}_o^1, W^1(\boldsymbol{\mu}_o^1)),$$

  one can use

  $$d(\boldsymbol{\mu}_o^1, \mathbf{f}^1) \le d(\boldsymbol{\mu}_o^1, W^1(\boldsymbol{\mu}_o^1)) + s \cdot d(\boldsymbol{\mu}_o^{\frac{1}{2}}, W^{\frac{1}{2}}(\boldsymbol{\mu}_o^{\frac{1}{2}})) + \cdots.$$

  For further details, see [65].

[1] The reason for using the same name for the two sampling methods will be justified below.

- *Fractal image segmentation* - The embedded function relates a certain fractal dimension to the image. However, there may be cases when more then one fractal dimension can be related to the image. This may be done by investigating more thouroughly the matrix structre of the transformation. Thus, the image can be segmented into regions with different fractal dimensions.

We believe that the combination of the evolving area of fractal image coding and the established use of pyramids in signal processing [13] will provide a useful common ground for future activity in this area.

Addendum A
Proof of Theorem 5.1 (Zoom)

In this addendum, we will prove Theorem 5.1 (zoom) stated in Section 5.2. The proof is divided into two parts, according to the two parts of the theorem.

A.1 Proof of Theorem Zoom-out

In order to prove that $\mathbf{f}^{\frac{1}{2}}$, as given by Equation (5.19), is indeed the fixed point of $W^{\frac{1}{2}}$, all we need to show is that it satisfies

$$\mathbf{f}^{\frac{1}{2}}(l) = W^{\frac{1}{2}}(\mathbf{f}^{\frac{1}{2}})(l), \qquad l = 1, \ldots, M_R B_1/2 . \tag{5.63}$$

In the following, it is convenient to express l as follows:

$$l = (i-1) \cdot B_1/2 + k, \qquad i = 1, \ldots, M_R; \quad k = 1, \ldots, B_1/2 . \tag{5.64}$$

This expression for l emphasizes that the l-th element of $\mathbf{f}^{\frac{1}{2}}$ is actually the k-th element of the i-th range block of $\mathbf{f}^{\frac{1}{2}}$. Also, let the superscript 1 or $\frac{1}{2}$ denotes a symbol as belonging to $\mathbf{f}^1$ or $\mathbf{f}^{\frac{1}{2}}$, respectively. Thus, for example, we let $D_h{}^{\frac{1}{2}} = B_1/2$ (in accordance to our basic assumptions in Section 5.1); it denotes the shift between two adjacent domain blocks in $\mathbf{f}^{\frac{1}{2}}$. We start by substituting $\mathbf{f}^{\frac{1}{2}}$ into the right-hand side of Equation (5.63), and taking the relevant w_i,

$$\begin{aligned} W^{\frac{1}{2}}(\mathbf{f}^{\frac{1}{2}})(l) &= W^{\frac{1}{2}}(\mathbf{f}^{\frac{1}{2}})((i-1)\frac{B_1}{2} + k) \\ &= a_i \cdot \varphi(\mathbf{d}_{m_i}^{\frac{1}{2}})(k) + b_i . \end{aligned} \tag{5.65}$$

Substituting for $\varphi(\cdot)$ from Equation (5.15), Equation (5.65) becomes:

$$\begin{aligned} & a_i \cdot \frac{1}{2}\left\{\mathbf{f}^{\frac{1}{2}}((m_i-1)D_h{}^{\frac{1}{2}} + 2k) + \mathbf{f}^{\frac{1}{2}}((m_i-1)D_h{}^{\frac{1}{2}} + 2k - 1)\right\} + b_i \\ = \; & \frac{1}{2} \cdot \left\{[a_i \mathbf{f}^{\frac{1}{2}}((m_i-1)D_h{}^{\frac{1}{2}} + 2k) + b_i] \right. \\ & \left. + [a_i \mathbf{f}^{\frac{1}{2}}((m_i-1)D_h{}^{\frac{1}{2}} + 2k - 1) + b_i]\right\} . \end{aligned} \tag{5.66}$$

Now, let us further explore the first term in the last equation:

$$\begin{aligned} & a_i \cdot \mathbf{f}^{\frac{1}{2}}((m_i-1)D_h{}^{\frac{1}{2}}+2k)+b_i \\ &= a_i \cdot \frac{1}{2}\left\{\mathbf{f}^1((m_i-1)\cdot 2D_h{}^{\frac{1}{2}}+4k)+\mathbf{f}^1((m_i-1)\cdot 2D_h{}^{\frac{1}{2}}+4k-1)\right\}+b_i \\ &= a_i \cdot \frac{1}{2}\left\{\mathbf{f}^1((m_i-1)D_h{}^1+4k)+\mathbf{f}^1((m_i-1)D_h{}^1+4k-1)\right\}+b_i \\ &= a_i \cdot \varphi(\mathbf{d}^1_{m_i})(2k)+b_i\ . \end{aligned} \tag{5.67}$$

but since $\mathbf{f}^1$ is the fixed point of W^1, it follows that the last equation is equal to

$$\mathbf{f}^1((i-1)B_1+2k). \tag{5.68}$$

Treating the second term in Equation (5.66) the same way, leads to:

$$a_i \cdot \mathbf{f}^{\frac{1}{2}}((m_i-1)D_h{}^{\frac{1}{2}}+2k-1)+b_i = \mathbf{f}^1((i-1)B_1+2k-1). \tag{5.69}$$

Thus, Equation (5.66) can be written as:

$$\frac{1}{2}\left\{\mathbf{f}^1((i-1)B_1+2k)+\mathbf{f}^1((i-1)B_1+2k-1)\right\} \tag{5.70}$$

which, by the theorem statement (5.19), is just $\mathbf{f}^{\frac{1}{2}}((i-1)\frac{B_1}{2}+k)$ which, in turn, is the left side of Equation (5.63).

A.2 Proof of Theorem Zoom-in

The procedure here is similar to the previous one, so we skip most of the details. It is needed to show that $\mathbf{f}^1$ obtained by Equation (5.20) is a fixed point of W^1:

$$\begin{aligned} W^1(\mathbf{f}^1)((i-1)B_1+j) &= a_i \cdot \varphi(\mathbf{d}^1_{m_i})(j)+b_i \\ &= a_i \cdot \frac{1}{2}\left\{\mathbf{f}^1((m_i-1)D_h{}^1+2j)\right. \\ &\quad \left.+\mathbf{f}^1((m_i-1)D_h{}^1+2j-1)\right\}+b_i\ . \end{aligned} \tag{5.71}$$

By Equation (5.19), which we have just proved, the last equation reduces to

$$\begin{aligned} a_i\mathbf{f}^{\frac{1}{2}}\left((m_i-1)\frac{D_h{}^1}{2}+j\right)+b_i &= a_i\mathbf{f}^{\frac{1}{2}}((m_i-1)D_h{}^{\frac{1}{2}}+j)+b_i \\ &= \mathbf{f}^1((i-1)B_1+j). \end{aligned} \tag{5.72}$$

Addendum B
Proof of Theorem 5.2 (PIFS Embedded Function)

The proof is constructive. It is based on finding the PIFS embedded function $G(x)$ and then showing that $G_N(x)$ is indeed a fixed point of the PIFS at resolution N.

Suppose that the PIFS is composed of M_R transformations. Let (a_i, b_i, m_i) denote the transformation parameters of each range block $\mathbf{r}_i$, $i = 1, \ldots, M_R$,

For operating with the PIFS code on a function of a continuous argument $x \in [0, 1]$, we look at the following analogies between the current continuous variable case (functions) and the previously discussed discrete case (vectors):

discrete	$\mathbf{v}_N(i)$	$i \in (1, 2, \ldots N)$	B	$D = 2B$	$D_h = B$
continuous	$f(x)$	$x \in [0, 1]$	$\frac{B}{r}$	$2 \cdot \frac{B}{r}$	$\frac{B}{r}$

discrete	i-th range block : $\mathbf{v}_N(j)$, $j = (i-1)B + 1, \ldots, iB$
continuous	i-th range block : $f(x)$, $x \in [\frac{(i-1)B}{r}, \frac{iB}{r})$

Such a configuration is demonstrated in Figure 5.7, where the i-th range block is

$$x \in [a, b), \qquad a = (i-1)\frac{B}{r} \text{ and } b = i\frac{B}{r}, \tag{5.73}$$

and the m_i-th domain block is

$$x \in [c, d), \qquad c = (m_i - 1)\frac{B}{r} \text{ and } d = (m_i - 1)\frac{B}{r} + 2 \cdot \frac{B}{r}. \tag{5.74}$$

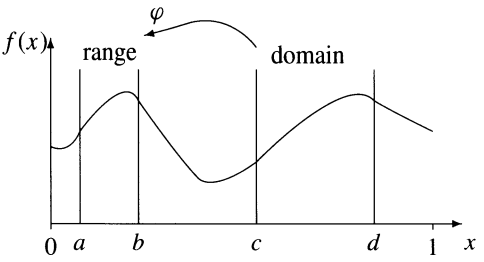


Figure 5.7: Domain to range transformation.

Define the following transformation $w_i : L^\infty[0, 1] \to L^\infty[0, 1]$:

$$w_i(f(x)) = \begin{cases} a_i f(2(x-a) + c) + b_i & a \le x < b \\ 0 & \text{otherwise}. \end{cases} \tag{5.75}$$

Then, since $|a_i| < 1$, w_i is contractive. Next, define the transformation

$$W = \bigcup_{i=1}^{M_R} w_i, \tag{5.76}$$

which is a contractive PIFS [45], and thus has a unique fixed point defined to be the function $G(x)$.

Now it is left to show that $G_N(x)$ is indeed a fixed point of the PIFS code. We take

$$\mathbf{v}_N(i) = G_N(i) \tag{5.77}$$

and look for consistency with the fixed-point criterion, i.e., verify that $\mathbf{v}_N = W(\mathbf{v}_N)$. Take, for example, the j-th element in the i-th range block. Let (a_i, b_i, m_i) denote the appropriate transformation parameters, so the fixed-point equation for this element is:

$$\begin{aligned} \mathbf{v}_N((i-1)B+j) &= a_i \frac{1}{2}(\mathbf{v}_N((m_i-1)B+2j) + \\ &\quad \mathbf{v}_N((m_i-1)B+2j-1)) + b_i, \\ &\quad j = 1, 2, \ldots, B. \end{aligned} \tag{5.78}$$

This time we deal with elements rather then whole blocks, so that the following notation is used (see also Figure 5.8):

- Boundaries of the j-th element in the i-th range block are

$$\tilde{a} = ((i-1)B + j - 1)\frac{1}{r} \quad \text{and} \quad \tilde{b} = ((i-1)B + j)\frac{1}{r}. \tag{5.79}$$

- Boundaries of the $(2j-1)$-th and the $(2j)$-th elements in the m_i domain block are

$$\tilde{c} = ((m_i - 1)B + 2j - 2)\frac{1}{r}; \tag{5.80}$$

$$\tilde{e} = \frac{\tilde{c} + \tilde{d}}{2} = ((m_i - 1)B + 2j - 1)\frac{1}{r}; \tag{5.81}$$

$$\tilde{d} = ((m_i - 1)B + 2j)\frac{1}{r}. \tag{5.82}$$

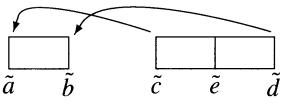


Figure 5.8: Elements of transformations.

Substituting $G_N(i)$ for $\mathbf{v}_N(i)$ in the right side of Equation (5.78) yields:

$$\begin{aligned} \mathbf{v}_N((i-1)B+j) &= a_i \frac{1}{2} r \left(\int_{\tilde{c}}^{\tilde{e}} G(x)\,dx + \int_{\tilde{e}}^{\tilde{d}} G(x)\,dx \right) + b_i \\ &= a_i \frac{1}{2} r \int_{\tilde{c}}^{\tilde{d}} G(x)\,dx + b_i\,. \end{aligned} \tag{5.83}$$

The left side of Equation (5.78) yields

$$\mathbf{v}_N((i-1)B+j) = r \int_{\tilde{a}}^{\tilde{b}} G(x)\,dx\,. \tag{5.84}$$

Since $G(x) = w_i(G(x))$, and since the range $x \in [\tilde{c}, \tilde{d})$ is mapped to $x \in [\tilde{a}, \tilde{b})$, we obtain:

$$\begin{aligned} & r \int_{\tilde{a}}^{\tilde{b}} [a_i G(2(x-\tilde{a})+\tilde{c}) + b_i]\, dx \\ = \; & a_i r \int_{\tilde{a}}^{\tilde{b}} G(2(x-\tilde{a})+\tilde{c})\, dx + r \int_{\tilde{a}}^{\tilde{b}} b_i\, dx \\ = \; & a_i \frac{1}{2} r \int_{\tilde{c}}^{\tilde{d}} G(x)\, dx + b_i \; . \end{aligned} \tag{5.85}$$

Thus, since Equations (5.83) and (5.85) agree, we've established the consistency of the fixed point equation for $\mathbf{v}_N$, under the theorem assumption that $\mathbf{v}_N(i) = G_N(i)$.

Addendum C
Proof of Theorem 5.3 (Fractal Dimension of the PIFS Embedded Function)

A related formal proof can be found in [3] and [34]. We will follow the informal proof, similar to the one given in [4] about the fractal dimension of a fractal interpolation function.

Let $\epsilon > 0$, and let $G(x)$ be superimposed on a grid of square boxes of side length ϵ. Let $N(\epsilon)$ denote the number of grid boxes of side length ϵ which intersect $G(x)$. Since $G(x)$ has fractal dimension $\mathcal{D}$, it follows that the following relation exits:

$$N(\epsilon) \sim \epsilon^{-\mathcal{D}} \quad \text{as} \quad \epsilon \to 0. \tag{5.86}$$

Our goal is to evaluate the value of $\mathcal{D}$.

Denote the number of boxes intersecting $G(x)$, for $x \in (\frac{(i-1)B}{r}, \frac{iB}{r})$, by

$$N_i(\epsilon), \qquad i = 1, \ldots, M_R. \tag{5.87}$$

Namely, $N_i(\epsilon)$ is the number of intersecting boxes in the i-th range block.

Since $\epsilon \to 0$, the contribution of boxes at the edges of range blocks can be ignored. Therefore we can write

$$N(\epsilon) \cong \sum_{i=1}^{M_R} N_i(\epsilon). \tag{5.88}$$

Now, let us investigate more thoroughly the value of $N_i(\epsilon)$. Let m_i denote the index of the domain block mapped to the i-th range block by the PIFS. This domain block is composed of two adjacent range blocks with indices denoted as m_i^1 and m_i^2.

After transforming the m_i-th domain block onto the i-th range block, each column of grid squares is mapped into a column of grid-rectangles, with height $|a_i|\epsilon$ and width $\frac{B}{D}\epsilon$. Let us define $q \equiv \frac{B}{D}$, and hence, according to our assumptions, $q = \frac{1}{2}$. Therefore, if a column of width ϵ in the domain block intersects $G(x)$ in L boxes, then after transformation the column will be of width $q\epsilon$, and the number of boxes of size $q\epsilon$ intersecting $G(x)$ in the transformed column is therefore at most, but possibly less than, $|a_i|\, L/q$ boxes.

Summing on all the domain columns yields

$$N_i(q\epsilon) \leq \frac{|a_i|}{q}(N_{m_i^1}(\epsilon) + N_{m_i^2}(\epsilon)). \tag{5.89}$$

Denoting by c_i the proportionality factor, i.e,

$$N_i(\epsilon) = c_i \cdot \epsilon^{-\mathcal{D}} \quad \text{as} \quad \epsilon \to 0, \tag{5.90}$$

we can write

$$c_i \cdot (q\epsilon)^{-\mathcal{D}} \leq \frac{|a_i|}{q}(c_{m_i^1}\epsilon^{-\mathcal{D}} + c_{m_i^2}\epsilon^{-\mathcal{D}}). \tag{5.91}$$

With the same reasoning for all others range blocks, we get

$$c_i \cdot (q)^{-\mathcal{D}+1} \leq |a_i|\,(c_{m_i^1} + c_{m_i^2}), \qquad i = 1, \ldots, M_R\,. \tag{5.92}$$

Recalling the definition of the matrix $\mathbf{D}$ in Equation (5.35), the last equation, for all the blocks, can be written as:

$$\mathbf{c} \cdot (q)^{-\mathcal{D}+1} \leq (\mathbf{A}_{abs} \cdot \mathbf{D}) \cdot \mathbf{c}, \qquad \mathbf{c} = [c_1, c_2, \ldots, c_{M_R}]^T\,. \tag{5.93}$$

It can be shown that $(q)^{-\mathcal{D}+1}$ is bounded by the spectral radius λ of $(\mathbf{A}_{abs} \cdot \mathbf{D})$, and hence

$$(q)^{-\mathcal{D}+1} \leq \lambda, \tag{5.94}$$

and hence,

$$\mathcal{D} \leq 1 + \log_{\frac{1}{q}}(\lambda). \tag{5.95}$$

Since we assume $q = \frac{B}{D} = \frac{1}{2}$, the theorem follows immediately.

It is worth noting that:

- Equation (5.93) holds even when $q \neq \frac{1}{2}$, and so does Equation (5.95). For example, it can be shown immediately that in the case of uniform fractal interpolation of functions [4], where the whole function is mapped to each interval, the following results:

$$D = N, \qquad q = \frac{D}{B} = M_R, \tag{5.96}$$

$$\mathcal{D} = 1 + \log_{M_R}\left(\sum_{i=1}^{M_R} |a_i|\right) \tag{5.97}$$

  which agrees with the result there.

- The matrix $q\mathbf{D}$ is a stochastic matrix. That is, the sum of the elements in each row equals 1. Thus, all its eigenvalues are equal or smaller then 1 (in magnitude), and at least one eigenvalue equals 1. For example, if

$$\mathbf{A} = a\mathbf{I}_{M_R \times M_R} \tag{5.98}$$

one can write

$$\mathbf{A}_{abs} \cdot \mathbf{D} = |a| \, \mathbf{I}_{M_R \times M_R} \cdot \frac{1}{q} q\mathbf{D} = \frac{|a|}{q}(q\mathbf{D}) \tag{5.99}$$

since $|a| / q$ is a scalar factor, and $q\mathbf{D}$ is stochastic, the largest eigenvalue is $|a| / q$. Thus, Equation (5.95) results in

$$\mathcal{D} = 2 + \log_{\frac{1}{q}} |a| \, . \tag{5.100}$$

If $q = \frac{1}{2}$, then

$$\mathcal{D} = 2 + \log_2 |a| \, , \tag{5.101}$$

which approaches $\mathcal{D} = 2$ as $|a|$ approaches 1, and $\mathcal{D} = 1$ if $|a| \leq \frac{1}{2}$ (by Equation (5.50)).