Modelling Discrete Sequences with Fractal Interpolation Functions of higher order

P. Bouboulis*
Department of Informatics and Telecommunications,
Telecommunications and Signal Processing, University of Athens,
Panepistimiopolis 157 84, Athens, Greece
bouboulis@di.uoa.gr

February 6, 2010

Abstract

We present a method that increases the accuracy of approximation of discrete sequences from fractal interpolation functions. Instead of previously presented models which use affine mappings, we employ polynomials of order $d$. Comparative results confirm the improvements in the accuracy of the proposed method.

1 Introduction

Fractal interpolation functions have drawn a great deal of attention in many scientific areas. They pose as an alternative to traditional interpolation techniques, giving a broader set of interpolants. In fact, many traditional interpolation techniques (splines, hermite polynomials et c.) are included as special cases. Its main differences from standard methods consist: a) in the presence of a self similarity in small scales, b) in the constructive way (through iterations), that it is used to compute the interpolant, instead of the descriptive one (usually a formula) provided by the classical methods and c) in the usage of some parameters, which are usually called vertical scaling factors, that are strongly related with the fractal dimension of the interpolant. They have been used to construct wavelets ([13], [14], [12]), to model one-dimensional signals ([18], [19], [21]), in medical applications ([10], [20]), in remote sensing ([15]), in computer graphics ([17], [16], [22]), in image compression ([7], [8], [9]) and several other applications.

This paper deals with the modelling of discrete sequences using fractal interpolation, a subject pioneered by Mazel and Hayes in [18]. They used piecewise self-affine fractal interpolation functions and devised an algorithm to compute the parameters of the model. Most applications dealing with approximation of rough objects using fractal interpolation still use variants of their algorithm. However, their approach has several drawbacks. The use of affine mappings for the construction of the fractal function that approximate the given sequence makes the model rather inflexible. Still, there hasn’t been proposed a method that overcomes this obstacle. This is the objective of the present paper. We employ polynomials of order $d$ and gain significant improvements.

The structure of the paper is as follows. In section 2 the main aspects of IFS and RIFS theory are outlined and a little more detailed description of the concept of fractal interpolation is given. The methodology of Mazel and Hayes in a broader context suitable for our purposes is also found there. Section 3 deals with fractal interpolation functions of higher order. The procedure of the computation of the new model parameters using convex optimization and some issues regarding the implementation of the method are given. Finally we conclude with a comparison between the new method and the Mazel-Hayes approach.

*Partially supported by the Special Account for Research Grants of the University of Athens # 70/4/5626.
2 Background

2.1 IFS-RIFS

An Iterated Function System (or IFS for short) \{X; w_{1-N}\} is defined as a pair consisting of a complete metric space \((X, \rho)\), together with a finite set of continuous, contractive mappings \(w_n : X \to X\), with respective contraction factors \(|s_n| < 1\), for \(n = 1, 2, \ldots, N\) (\(N \geq 2\)). The attractor of an IFS is the unique set \(E\), for which \(E = \lim_{k \to \infty} W^k(A_0)\) for every starting compact set \(A_0\), where

\[
W(A) = \bigcup_{n=1}^{N} w_n(A) \text{ for all } A \in \mathcal{H}(X),
\]

and \(\mathcal{H}(X)\) is the complete metric space of all nonempty compact subsets of \(X\) with respect to the Hausdorff metric \(h\) (for the definition of the Hausdorff metric, properties of \((\mathcal{H}(X), h)\) and examples of IFS, see [3] and [2] among others). Since their introduction by Barnsley and Demko in the 80’s (see [3]) they have been used in numerous applications, due to their ability to construct highly complex structures with only a handful of mappings. A notion closely related with IFS is that of the Recurrent Iterated Function System, or RIFS for short, that allows the construction of even more complicated sets, by utilizing an irreducible row-stochastic matrix \(P\): \((p_{n,m} \in [0,1]: n, m = 1, \ldots, N)\), such that

\[
\sum_{m=1}^{N} p_{n,m} = 1, \quad n = 1, \ldots, N. \quad (1)
\]

The concept of the RIFS resembles that of a discrete time Markov process. The matrix \(P\) contains the transition probabilities for the process (i.e. \(p_{n,m}\) gives the probability of transfer into state \(m\) given that the process is in state \(n\)). Condition (1) says that whichever state the system is in (say \(n\), a set of probabilities is available that sum to one and describe the possible states to which the system transits at the next step. We will return to these issues later, when we will use RIFS for interpolation (see algorithm 2). For the construction of the attractor of the RIFS a little more effort is needed. The interested reader is addressed to [2], [4], [5] and [6] for more details.

2.2 Fractal Interpolation

Barnsley in [1] was the first to introduce functions that are derived as attractors of IFSs or RIFSs (see [1], [4]) and interpolate given data points. Here we briefly describe this construction based on RIFSs. The resulting functions are called Recurrent Fractal Interpolation Functions (or RFIF for short).

Let \(X = [0, 1] \times \mathbb{R}\) and \(\Delta = \{(x_n, y_n) : n = 0, 1, \ldots, N\}\) be an interpolation set with \(N + 1\) interpolation points such that \(0 = x_0 < x_1 < \cdots < x_N = 1\). The interpolation points divide \([0, 1]\) into \(N\) intervals \(I_n = [x_{n-1}, x_n]\), \(n = 1, \ldots, N\), which we call domains. In addition, let \(Q = \{ (\hat{x}_j, \hat{y}_j) : m = 0, 1, \ldots, M \}\) be a subset of \(\Delta\), such that \(0 = \hat{x}_0 < \hat{x}_1 < \cdots < \hat{x}_M = 1\). We also assume that for every \(m = 0, 1, \ldots, M - 1\) there is at least one \(n\) such that \(\hat{x}_m < x_n < \hat{x}_{m+1}\). Thus, the points of \(Q\) divide \([0, 1]\) into \(M\) intervals \(J_m = [\hat{x}_{m-1}, \hat{x}_m]\), \(m = 1, \ldots, M\), which we call regions. Finally, let \(\mathcal{J}\) be a labelling map such that \(\mathcal{J} : \{1, 2, \ldots, N\} \to \{1, 2, \ldots, M\}\), with \(\mathcal{J}(n) = m\). Let \(x_n - x_{n-1} = \delta_n\), \(n = 1, 2, \ldots, N\), and \(\hat{x}_m - \hat{x}_{m-1} = \psi_m\), \(m = 1, 2, \ldots, M\). It is evident that each region contains an integer number of domains. In the special case where the interpolation points are equidistant (that is \(x_n - x_{n-1} = \delta\), \(n = 1, 2, \ldots, N\), and \(\hat{x}_m - \hat{x}_{m-1} = \psi\), \(m = 1, 2, \ldots, M\)), each region contains exactly \(\psi/\delta \in \mathbb{N}\) domains.

We define \(N\) mappings of the form:

\[
w_n \left( \frac{x}{y} \right) = \left( \frac{L_n(x)}{F_n(x, y)} \right), \quad \text{for } n = 1, 2, \ldots, N, \quad (2)
\]

where \(L_n(x) = a_n x + b_n\) and \(F_n(x, y) = s_n y + q_n(x)\) \((q_n(x)\) is a continuous function). Each map \(w_n\) is constrained to map the endpoints of the region \(J_{\mathcal{J}(n)} = J_m\) to the endpoints of the domain \(I_n\) (see figure 1). That is,

\[
w_n \left( \frac{\hat{x}_{m-1}}{\hat{y}_{m-1}} \right) = \left( \frac{x_{n-1}}{y_{n-1}} \right), \quad w_n \left( \frac{\hat{x}_m}{\hat{y}_m} \right) = \left( \frac{x_n}{y_n} \right), \quad \text{for } n = 1, 2, \ldots, N. \quad (3)
\]
Solving for \(a_n, b_n\) we obtain \(a_n = (x_n - x_{n-1})/(\tilde{x}_n - \tilde{x}_{n-1})\) and \(b_n = (\tilde{x}_m x_{n-1} - \tilde{x}_{m-1} x_n)/(\tilde{x}_m - \tilde{x}_{m-1})\). Vertical segments are mapped to vertical segments scaled by the factor \(s_n\). The parameter \(s_n\) is called the \textit{vertical scaling factor} of the map \(w_n\). It is easy to show that if \(|s_n| < 1\), then there is a metric \(d\) equivalent to the Euclidean metric, such that \(w_n\) is a contraction (i.e., there is \(\delta_n : 0 \leq \delta_n < 1\) such that \(d(w_n(\tilde{x}), w_n(\tilde{y})) \leq \delta_n d(\tilde{x}, \tilde{y})\), see [2]).

To conclude, we must define the \(N \times N\) stochastic matrix \(P\):

\[
p_{nm} = \begin{cases} \frac{1}{s_n} & \text{if } I_n \subseteq I_{J(m)} \\ 0, & \text{otherwise.} \end{cases}
\]

where \(\gamma_n\) is the number of positive entries of the line \(n, n = 1, 2, \ldots, N\). This means that \(p_{n,m}\) is positive, if and only if there is a transformation \(L_m\), which maps the region containing the \(n\)th domain (i.e. \(I_n\)) to the \(m\)th domain (i.e. \(I_m\)). Usually, it is more efficient to describe the stochastic matrix through the connection vector \(V = (J(1), J(2), \ldots, J(N))\), that describes which region is mapped to each domain. Assuming that we have chosen a set of interpolation points \(\Delta\), its subset \(Q\), the vertical scaling factors \(s_n\), \(V\), and some arbitrary mappings that satisfy (3) we can construct the RFIF using the Deterministic Iterative Algorithm (Algorithm 1) or the Random Iterative Algorithm (Algorithm 2).

**Algorithm 1: Deterministic Iterative Algorithm (DIA)**

\textbf{Input:} \(\Delta, Q, \{s_1, \ldots, s_N\}, V, \{q_1(x), \ldots, q_N(x)\}, \nu\) (the number of steps).

\textbf{Output:} \(f\)
\[
f_0 = \Delta; \\
\text{for } k = 1 \text{ to } \nu \text{ do} \\
\quad \text{foreach domain } I_n \text{ do} \\
\quad \\
\quad \quad \text{Find the corresponding region } J_{V(n)}; \\
\quad \quad \text{Denote } R_{V(n)} \text{ as the set containing the points of } f_{k-1} \text{ that lie inside } J_{V(n)}; \\
\quad \quad \text{Map the points of } R_{V(n)} \text{ using the map } w_n, \text{ i.e. } D_n = w_n(R_{V(n)}); \\
\quad \quad f_k = f_k \cup D_n; \\
\quad \text{end}
\]

**Algorithm 2: Random Iterative Algorithm (RIA)**

As mentioned above, while the theory allows the use of arbitrary functions \(\{q_1(x), \ldots, q_N(x)\}\) that satisfy (3), only the linear case has been considered in most applications, i.e. \(q_n(x) = c_n x + d_n, n = 1, \ldots, N\). In this case (3) gives a linear system which can be easily solved to obtain:

\[
c_n = \frac{y_n - y_{n-1}}{\tilde{x}_m - \tilde{x}_{m-1}} - s_n \frac{\tilde{y}_m - \tilde{y}_{m-1}}{\tilde{x}_m - \tilde{x}_{m-1}}, \\
d_n = \frac{\tilde{x}_m y_{n-1} - \tilde{x}_{m-1} y_n}{\tilde{x}_m - \tilde{x}_{m-1}} - s_n \frac{\tilde{x}_m \tilde{y}_m - \tilde{x}_{m-1} \tilde{y}_{m-1}}{\tilde{x}_m - \tilde{x}_{m-1}},
\]

for all \(n = 1, \ldots, N\). Thus, once the contraction factor \(s_n\) for each map has been chosen, the remaining parameters may be easily computed. This is the reason why the linear case is so popular. Mazel and Hayes (see [18]) used this model (which they called piecewise self-affine fractal model) to approximate discrete sequences. Their methodological is briefly described in algorithm 3 in a more general setting. The computation of the vertical scaling factor is done via unconstrained optimization of the mean square error or using some geometric criteria ([18]). Both methods, however, may well lead to non acceptable values for the vertical scaling factor (\(|s_n| > 1\)). This is another drawback of their approach which we attempt to fix.
Input: The discrete sequence \( \{(t_i, f_i), i = 0, 1, \ldots, N \} \), \( \delta, \psi \) such that \( \psi/\delta \in \mathbb{N} \) and the error tolerance \( E \).

Output: A list of interpolation points \( L_p \), a list of domain numbers \( L_d \), a list of addresses \( L_a \) (describing which region is mapped to each domain) and a list of vertical scaling factors \( L_v \).

1. Store the points \( (t_0, f_0), (t_{0+k}, f_{0+k}), (t_{0+2k}, f_{0+2k}) \) etc. to the interpolation points list \( L_p \);
2. Store the numbers of all domains, defined by those points, to \( L_d \);
3. foreach domain \( n \) do
   4.     foreach region \( m \) do
      5.         Compute the vertical scaling factor \( s \) associated with region \( m \) and domain \( n \) (this is done via some geometric arguments or optimization);
      6.         if \( |s_n| > 1 \) then
          7.             Continue to the next region;
      8.         Compute the rest of the parameters of the map \( w_n \) (i.e. \( c_n, d_n \));
      9.         Denote by \( D_n \) the points of the sequence that lie inside the domain \( n \) and by \( R_m \) the points of the sequence that lie inside the region \( m \);
      10.       Map \( R_m \) through \( w_n \), i.e. \( \hat{D}_n = w_n(R_m) \);
      11.       Compute the SNR between \( D_n \) and \( \hat{D}_n \);
      12.       if \( \text{SNR} \geq E \) then
          13.           Continue to the next domain;
      14.     end
   15.  Find the maximum of the computed SNRs for the domain \( n \);
   16.  if \( \text{maximum SNR} \geq E \) then
      17.         Store the corresponding \( m \) to the addresses-list \( L_a \);
      18.      Store the corresponding \( s \) to the vertical scaling factors list \( L_v \);
      19.   end
   20. if \( \text{maximum SNR} < E \) then
      21.     Split the domain \( D_n \) into \( \psi/\delta \) smaller domains (adding the respective interpolation points to the list \( L_p \)) and repeat the region-search for each one;
      22.   end
   23. end

Algorithm 3: Mazel - Hayes Methodology

Note that Algorithms 1, 2 and 3 work even if \( \psi/\delta \notin \mathbb{N} \). In the following, however, we limit our interest in the case where there are numbers \( \alpha, \beta \in \mathbb{N} \) such that \( \delta = \alpha^\beta \), \( \psi = \alpha^{\beta+1} \). In this case, for the reconstruction step, the DIA will produce exactly the values of the discrete sequence. Otherwise, the algorithm will produce more points than needed and we have to add an additional step to the procedure (for details see [18], [8], [9]).

3 Fractal Interpolation Functions of higher order

3.1 Definition

Although the method of Mazel and Hayes is quite popular and has been already implemented in many applications it has some serious disadvantages. If we try to approximate a smooth signal, even as simple as the points of a parabola, with the piecewise self affine model, we will obtain in the output much more information than needed. This motivates us to wonder what will happen if, instead of affine functions the \( q_n \)'s are polynomials of degree greater than one. Here we consider this case, i.e.

\[ q_n(x) = \sum_{k=0}^{d} a_{n,k} x^k, \]

for \( n = 1, \ldots, N, \; d \geq 1 \). Considering \( s_n \) and \( a_{n,k} \) for \( k = 2, \ldots, d \), as free parameters and taking into account conditions (3) after some algebra we obtain:

\[ a_{n,1} = \frac{y_n - y_{n-1}}{\hat{x}_m - \hat{x}_{m-1}} - \frac{\sum_{k=2}^{d} a_{n,k} (\hat{x}_m^k - \hat{x}_{m-1}^k)}{\hat{x}_m - \hat{x}_{m-1}} = s_n \frac{\hat{y}_m - \hat{y}_{m-1}}{\hat{x}_m - \hat{x}_{m-1}}, \]

\[ a_{n,0} = \frac{y_n - y_{n-1}}{\hat{x}_m - \hat{x}_{m-1}} - \frac{\sum_{k=2}^{d} a_{n,k} (\hat{x}_m^k - \hat{x}_{m-1}^k)}{\hat{x}_m - \hat{x}_{m-1}} = s_n \frac{\hat{y}_{m-1} \hat{x}_m - \hat{y}_m \hat{x}_{m-1}}{\hat{x}_m - \hat{x}_{m-1}}. \]
We will call the attractor of an RIFS of this form as RFIF of order $d$.

### 3.2 Modeling discrete sequences

In order to use RFIFs of order $d$ to model discrete sequences, we need to devise a method for the efficient computation of the free parameters. To this end, we consider the discrete sequence $\{(t_i, f_i), i = 0, 1, \ldots, N_0\}$ and the sets $D_n = \{(t_{i(m)}), (t_{i(m)}), \ldots, (t_{i(m)}+\delta)\}$ and $R_m = \{(t_{i(m)}), (t_{i(m)}+\psi), (t_{i(m)}+\psi)\}$, containing the points of the signal that lie inside the $n$-th domain and $m$-th domain respectively. Since we limit our interest in the case where $\delta = \alpha^3$ and $\psi = \alpha^{3+1}$, we do not need all the points of $R_m$, but only those mapped at the corresponding points of $D_n$, therefore we redefine $R_m$ as $R_m = \{(t_{i(m)}+\alpha^l, f_{i(m)+\alpha l}), l = 0, \ldots, \delta\}$. Mapping the set $R_m$ through a $w_n$ of the form (2) we obtain the set $\hat{D}_n = w_n(R_m) = \{(t_{i(m)+l}, f_{i(m)+\alpha l}), l = 0, \ldots, \delta\}$, where $f_{i(m)+\alpha^l} = s_n f_{i(m)+\alpha^l} + g_n(t_{i(m)+\alpha^l})$. After some lines of algebra we conclude that $f_{i(n)+l} = \sum_{k=2}^d B_{m,l,k} \cdot a_{n,k} - s_n \Gamma_{m,l}$, where

\[
\begin{align*}
A_{n,m,l} &= A_{n,m}^{(2)} + A_{n,m}^{(1)} \cdot \hat{t}_{i(m)+\alpha^l}, \\
A_{n,m}^{(1)} &= \frac{y_{n-1} - y_n}{x_n - x_m}, \\
P_{m,k}^{(1)} &= \frac{x_m^{k} - x_{m-1}^{k}}{x_m - x_{m-1}}, \\
\Gamma_{m,l} &= \Gamma_{m}^{(2)} + \Gamma_{m}^{(1)} f_{i(m)+\alpha^l} - \hat{f}_{i(m)+\alpha^l}, \\
\Gamma_{m}^{(2)} &= \frac{y_{m-1} - y_m}{x_m - x_m},
\end{align*}
\]

for $l = 0, \ldots, \delta$, $k = 2, \ldots, d$. We should choose the free parameters $s_n$, $a_{n,k}$, $k = 2, \ldots, d$ such that the square error

\[
E(s_n, a_{n,2}, \ldots, a_{n,d}) = \sum_{l=0}^{\delta} \left( f_{i(n)+l} - \hat{f}_{i(n)+l} \right)^2
\]
is minimized. To simplify the notation, we define:

\[ u = \begin{pmatrix} s_n \\ s_{n,2} \\ \vdots \\ s_{n,d} \end{pmatrix}, \quad A = \begin{pmatrix} \Gamma_{m,0} & B_{m,0,2} & \ldots & B_{m,0,d} \\ \Gamma_{m,1} & B_{m,1,2} & \ldots & B_{m,1,d} \\ \vdots & \vdots & \vdots & \vdots \\ \Gamma_{m,\delta} & B_{m,\delta,2} & \ldots & B_{m,\delta,d} \end{pmatrix}, \quad b = \begin{pmatrix} A_{n,m,0} - f_i(n) \\ A_{n,m,1} - f_{i(n)+1} \\ \vdots \\ A_{n,m,\delta} - f_{i(n)+\delta} \end{pmatrix}. \]

therefore the error to be minimized is \( E(u) = \| A \cdot u - b \|_2^2 \). Considering some positive regularization coefficients \( \gamma_1, \ldots, \gamma_d \) and taking into account the fact that \( |s_n| < 1 \) we end up with the following convex optimization problem:

\[
\text{minimize} \quad E(u) = \| A \cdot u - b \|_2^2 + \sum_{k=1}^d \gamma_k u_k^2,
\]

\text{subject to} \quad
\begin{align*}
    u_1 &\leq \epsilon, \\
    u_2 &\geq -\epsilon,
\end{align*}

where \( \epsilon \) should be chosen a priori close to 1 (say 0.99). We can easily check that problem (6) has a unique solution, which can easily be found. Before we give the solution we need the Lemma given below.

**Lemma 3.1** Consider the function \( f : \mathbb{R}^n \rightarrow \mathbb{R} : f(x) = x^\top Ax + b^\top x + c \), where \( A \) is a strictly positive matrix and the hyperplane \( a^\top x = \zeta \). The aforementioned hyperplane is a supporting hyperplane to the level set \( C_\eta = \{ x : f(x) \leq \eta \} \) at the point \( (x_0, f(x_0) = \eta) \), where

\[
x_0 = \frac{1}{2} A^{-1}(\lambda a - b), \quad \lambda = \frac{\zeta + a^\top A^{-1}b}{a^\top A^{-1}a}.
\]

**Proof.** The supporting hyperplane to the set \( C_\eta \) at a point \( (x_0, \eta) \) is \( \nabla f(x_0)^\top x = \nabla f(x_0)^\top x_0 \). Therefore, since we want this hyperplane to be identified with the hyperplane \( a^\top x = \zeta \), it is sufficient to find \( \lambda \) such that \( 2A x_0 + b = \lambda a \) and \( (2A x_0 + b)^\top x_0 = \lambda \zeta \). Solving for \( x_0 \) gives the result.

Now we focus our interest to problem (6). Since \( E(u) = u^\top \left( A^\top A + \text{diag}(\gamma) \right) u - 2b^\top Au + b^\top b \) is a strictly convex differentiable function (where \( \gamma = (\gamma_1, \ldots, \gamma_d) \)), it has a unique global minimum. Taking the gradient equal to 0, we can easily found that the global minimum is attained at \( u^{(m)} = (A^\top A + \text{diag}(\gamma))^{-1} b^\top A \). If \( |u_1^{(m)}| \leq \epsilon \) then the solution of problem (6) is \( u^* = u^{(m)} \). If \( u_1^{(m)} > \epsilon \) then (using lemma 3.1)

\[
u^* = \frac{1}{2} \left( A^\top A + \text{diag}(\gamma) \right)^{-1} (\lambda a + 2b^\top A),
\]

where \( \lambda = \frac{-2a^\top (A^\top A + \text{diag}(\gamma))^{-1} b^\top A}{a^\top A^{-1}a} \) and \( a = (1, 0, \ldots, 0)^\top \in \mathbb{R}^d \).

Finally, if \( u_1^{(m)} < -\epsilon \) we find similarly that

\[
u^* = \frac{1}{2} \left( A^\top A + \text{diag}(\gamma) \right)^{-1} (\lambda a + 2b^\top A),
\]

where \( \lambda = \frac{-2a^\top (A^\top A + \text{diag}(\gamma))^{-1} b^\top A}{a^\top A^{-1}a} \) and \( a = (1, 0, \ldots, 0)^\top \in \mathbb{R}^d \).

Note that in the case where Algorithm 3 partitioned a domain to several smaller ones due to inadequate region mapping as described in the algorithm, the above relations still hold if we consider \( \alpha = \psi/(\text{domain-width}) \). Moreover, it is easy to modify the procedure to handle the case where the interpolation points are not equidistant.
3.3 Implementation - Comparison

Using Algorithm 3, with the modifications in the computation of the map parameters which we described above (i.e., we replace the steps of the vertical scaling factor computation and the $c_n, b_n$ computation with the minimization procedure) and removing the if statement regarding the value of the vertical scaling factor (which is not needed since the minimization procedure ensures that $|s| < 1$) one should expect to get better results than the Mazel-Hayes approach, since our methodology uses polynomials of degree higher than 1 and minimizes the error analytically. In fact, it is evident that in the trivial case where we try to approximate a sequence which has been derived as points of a piecewise self affine FIF, our approach will give the same parameters up to order one, filling the rest of the polynomial coefficients (up to order $d$) with zeroes. In this section we demonstrate the efficiency of the proposed model, comparing it with the approach of Mazel and Hayes in the area of one-dimensional signal compression.

To achieve compression, we must come up with a quantization method for the model parameters. For $d = 1$, Mazel and Hayes showed that a uniform quantizer with $2^b$ or $2^s$ levels for the quantization of the vertical scaling factors gives satisfactory results. This is expected since we can see from examples that the vertical scaling factors $s_n$ follow an almost uniform distribution. In the non-linear case a little more effort is needed. The positive (likewise for the negative) values of $a_{n,k}$ (i.e. the coefficients of $x^k$ terms in the model) now follows exponential distributions and there are some cases (luckily only a few) where the values deviate from the mean significantly (see figure 2). To confront this problem we add one more step to Algorithm 3, so that the domains with model parameters that deviate from the mean significantly (i.e. the values greater than a cutoff threshold for the positive coefficients or alternatively the values lower than a cutoff threshold for the negative coefficients) are fed back again to the algorithm. The cutoff threshold is taken as the value greater than the 99% of the positive coefficients (or alternatively the value lower than the 99% of the negative coefficients). The algorithm searches again for a region that is “best-mapped” to the respective domain. However, each time the minimization process yields model parameters that deviate from the mean as explained above, the deviated parameters are set equal to the respective cutoff threshold and the algorithm continues as before (i.e. the set $D_n$ is formed and the SNR is computed). Consequently, we apply different uniform quantizers for the model parameters of different orders. In addition for higher order coefficients we need more quantizer levels since the model becomes more sensitive to errors.

Figure 3 shows the results of the methodology applied to two sound samples for various values of error tolerance $E$ (from 10 to 50). The first is a sample from a man singing a ballad and the second a sample from an electric guitar. We can see in the figure that there is a point which the model (for various values of $d$) gets its maximum compression ratio and the decrease of error tolerance $E$ doesn’t contribute. This is due
to the fact that as \( d \) increases, we need to store more coefficients for each domain. There is a value for \( E \) such that almost none of the domains get split. If we decrease \( E \) more, the algorithm finds less suitable regions for each domain, which results to a decrease of PSNR while the compression ratio remains the same, since the same number of coefficients is stored. For the quantization of the coefficients we used 8 bits for the vertical scaling factors \( s_n \) of the Mazel-Hayes approach. For the higher order model with \( d = 2 \), we used 8 bits for the \( s_n \) and 8 bits for the \( a_{n,2} \) coefficients. For the same model with \( d = 3 \), we used 8 bits for the \( s_n \) and 9 bits for the \( a_{n,2} \) and \( a_{n,3} \) coefficients. Finally, when \( d = 4 \), we used 8 bits for the \( s_n \) and 11 bits for the \( a_{n,2} \), \( a_{n,3} \), \( a_{n,4} \) coefficients. For the regularization coefficients we chose \( \gamma = (1, \ldots, 1) \) and \( \epsilon = 0.99 \). The values of \( \delta \) and \( \psi \) were chosen to be 32 and 64 respectively.

Acknowledgments. The author would like to thank professors Leoni Dalla and Sergios Theodoridis for their continuous support.

References


Figure 3: Two different wav files consisting of 32769 samples have been compressed with the Mazel-Hayes methodology and the RFIF of order $d$ methodology, for various values of $d$. The second sound sample (b) is far more complex than the first (a).