A General Technique for Filtering Random Noise

Balas Natarajan
Computer Systems Laboratory
HPL-94-55
June, 1994

non-linear filters,
random noise

We present a novel technique for the design of filters for random noise, leading to a class of filters called Occam filters. The essence of the technique is that when a lossy data compression algorithm is applied to a noisy signal with the allowed loss set equal to the noise strength, the loss and the noise tend to cancel rather than add. We give two illustrative applications of the technique to univariate signals. We observe that an Occam filter can outperform the Wiener filter, but unlike the Wiener filter, does not require prior knowledge of the spectral properties of the noise-free signal. We also prove asymptotic convergence bounds on the effectiveness of Occam filters.

1. Introduction

It is well known that random noise is hard to compress, while ordered information is not. We exploit this property to filter random noise from signals.

Suppose that we are given a signal that is corrupted with additive random noise of known strength. The strength may be measured as, say, the amplitude or the power of the noise. Assume that we have at our disposal a lossy data compression routine in the form of a black box. The black box has a knob that controls the loss allowed of the routine, and we shall pay attention to the position of the knob when using the box to compress the noisy signal. To begin, we set the knob at a large loss. Then, the output of the box tracks the signal weakly, much compression has been achieved in that the size of the output is small, and a lot of information has been lost. As we reduce the allowed loss, the output of the box tracks the signal increasingly. When the loss tolerance is equal to the strength of the noise, the output of the box just begins to track the noise. As the loss tolerance is further reduced, the output of the box tracks more of the noise. When the loss tolerance is equal to the noise strength, will the noise and the loss add, so that the decompressed output is further from the noise-free signal than the noisy signal? Or, will the loss cancel the noise, so that the decompressed output closer to the noise-free signal than the noisy signal. We show that the loss tends to cancel the noise, with the extent of the cancellation depending on the compression achieved and how often the signal is sampled.

The above experiment suggests the following, which is the essence of our technique. *Compress the noisy signal with a lossy compression algorithm, with the allowed loss set equal to the noise strength. The decompressed signal is the filtered signal.*

We use the term Occam filters to refer to the class of filters realizable by our technique. This is because the essence of our technique is the principle of Occam's Razor–"the simpler explanation of the observed phenomenon is more likely to be correct".

In the first part of the paper we discuss application issues. Readers interested in understanding the spirit of the technique rather than its theoretical foundations will find sufficient material in this part of the paper. We illustrate the application of the technique using two examples involving univariate functions. The first example concerns a broad-band signal corrupted with uniformly distributed random noise. Using a compression algorithm that operates in terms of the piecewise linear functions, we construct an Occam filter for the problem. We then compare the performance of our filter with the Wiener filter, which is known to be the optimal linear filter in the least-squares sense. We find that our non-linear Occam filter outperforms the Wiener filter. Also, our filter remains effective even when the distribution of the noise is skewed so as to have non-zero mean, a difficult situation for any linear filter including the Wiener filter. The second example concerns a smooth signal that is corrupted with normally distributed random noise. Using a compression algorithm that operates in the spectral representation, we construct an Occam filter for the problem. Since the Wiener filter operates in the spectral domain as well, a comparison between the two filters is particularly interesting. While its performance is slightly worse than that of the Wiener filter, unlike the Wiener filter, the Occam filter does not require a priori knowledge of the spectral properties of the noise-free signal.

In the second part of the paper, we examine the theoretical underpinnings of our technique and obtain convergence bounds for the residual noise in terms of the compression achieved and the sampling rate. Let s be the compressed size in bits of the n noisy samples, when the allowed loss is equal to the noise strength. Broadly speaking, we show that the residual noise in the filtered signal varies as $(s/n)^{½}$.

The results of this paper appeared in preliminary form in several papers. Natarajan (1993a) introduces Occam filters with emphasis on application; Natarajan (1993b) presents a theoretical justification in the learning theoretic context; Natarajan (1994) offers a convergence bound on Occam filters under some restrictions.

Of related interest are the papers by Donoho (1992), Donoho et al (1993), and DeVore and Lucier (1992), that examine the filtering of random Gaussian noise using wavelets, by shifting and thresholding the transform coefficients by an amount dependent on the noise strength, and then inverting to obtain the filtered signal. Also of interest is the paper by Saito (1994) who observes that data compression and filtering are related and attempts to filter normally distributed random noise by using the wavelet basis that compresses the input data best, selected from among a preselected class of wavelet bases. Saito justifies his technique via Rissanen's Minimum Description Length principle. Each of these results can be viewed as specializations of our results to the context of wavelets.

Part 1: Application

2. Preliminaries

We consider functions on the unit interval $[0,1]$. The limitation to the unit interval is in the interest of convenience and is without loss of generality. For a function f and natural number n, f_n refers to the sequence of uniform samples of f spaced $1/n$ apart. Specifically, f_n is the sequence of n samples $\{f(0), f(1/n), f(2/n), \ldots, f((n-1)/n)\}$. We call f_n a *sample sequence*.

A metric over the space of sample sequences is a measure of distance between any two sample sequences f_n and g_n, and is denoted by $\|f_n,g_n\|$. The power or L_2 metric is defined as

$$\|f_n,g_n\|_2 = \frac{1}{n}\sum_{i=0}^{n-1}\Big(f(i/n)-g(i/n)\Big)^2 . \tag{1}$$

Note that the power metric is the square of the Euclidean metric. The amplitude or L_∞ metric is defined as

$$\|f_n,g_n\|_\infty = \max_{i=0}^{n-1}\Big|f(i/n)-g(i/n)\Big| . \tag{2}$$

The L_1 metric is defined as

$$\|f_n,g_n\|_1 = \frac{1}{n}\sum_{i=0}^{n-1}\Big|f(i/n)-g(i/n)\Big| , \tag{3}$$

and measures the average absolute difference between two sequences.

The power of a sequence f_n is its distance from the sequence of zeros, i.e., $\|f_n,0\|_2$. Similarly the amplitude of a sequence f_n is $\|f_n,0\|_\infty$. In general, the strength of a sequence f_n in metric $\|\cdot\|$ is $\|f_n,0\|$. In the interest of brevity we will write $\|f_n\|$ to denote $\|f_n,0\|$.

With respect to a metric $\|\cdot\|$, a *lossy compression* algorithm C is a program that takes as input a sequence f_n and a loss tolerance $\epsilon \geq 0$, and produces as output a binary string s representing a sequence g_n such that $\|f_n,g_n\| \leq \epsilon$. C is said to obey the metric $\|\cdot\|$. A *decompression* algorithm D takes as input a binary string and produces as output a sample sequence. In particular, the decompression algorithm D corresponding to C would output g_n on input s. We use $C(f_n,\epsilon)$ to denote the string s obtained by running C on input f_n and ϵ, and we use $D(s)$ to denote the sequence g_n obtained by running D on string s.

Let ν be a random variable representing the noise. We use $\hat{f}_n$ to denote the sequence f_n corrupted with noise, i.e., $\hat{f}_n = \{f(0)+\nu, f(1/n)+\nu, f(2/n)+\nu,...\}$.

3. The General Algorithm

Using the notation of the preceding section, we can state our filtering technique in the form of an algorithm. In the following, the strength of the noise is measured in the same metric as that obeyed by the compression algorithm.

```
Filtering Algorithm
input $\hat{f}_n$
begin
    Let $\|\nu\|$ be the strength of the noise.
    Run $C(\hat{f}_n, \|\nu\|)$;
    Decompress to obtain the filtered sequence $g_n$;
end
```

As stated above, the filtering algorithm we requires the strength of the noise $\|\nu\|$ to be known, where $\|\cdot\|$ is the metric obeyed by the compression algorithm. In the next section we present a heuristic for estimating $\|\nu\|$.

4. Estimating the Noise Strength

The heuristic is best illustrated via the following experiment. Suppose we had direct access to the noise source and were able to obtain n samples of the noise ν. Run the compression algorithm on these samples, for various values of the allowed loss ϵ. Plot the size of the output of the compression algorithm as a function of $\log(\epsilon)$. This is essentially the rate-distortion plot of the noise function. At values of ϵ greater than the noise strength $\|\nu\|$, the compressed size will be small, since at such large tolerances the noise can be approximated by the constant function. As ϵ is reduced below the noise strength, the compressed size rises sharply, as shown conceptually in Figure 4.1. Now suppose that we had access to the noise-free signal f. Run the compression algorithm on f_n, for various values of ϵ and plot compressed size against $\log(\epsilon)$, to obtain the rate-distortion plot of the signal. At large values of ϵ the compressed size is small, and as ϵ is reduced the compressed size increases, as shown conceptually in Figure 4.2. Lastly, run the compression algorithm on the noisy samples $\hat{f}_n$ for various value of ϵ and plot compressed size against $\log(\epsilon)$, to obtain the rate-distortion plot of the noisy signal. As shown in Figure 4.3, at values of ϵ greater than the noise strength $\|\nu\|$, the plot follows the rate-distortion plot for the noise-free signal as in Figure 4.2. At values of ϵ less than the noise strength, the noise dominates the signal f and the plot follows the rate-distortion plot of the noise as in Figure 4.1. At $\epsilon = \|\nu\|$, the rate-distortion plot of the noisy signal $f+\nu$ displays a sharp "knee point", i.e., a point at which the plot rises sharply. The knee point can be more precisely identified as the point at which the second derivative of the plot attains a maximum. The second derivative plot is also depicted in Figure 4.3. This suggests the following strategy for determining the noise strength $\|\nu\|$: Run the compression algorithm on samples of the noisy signal for various values of the allowed loss ϵ. Plot compressed size versus $\log(\epsilon)$. The knee point of this plot is the strength of the noise in the metric obeyed by the compression algorithm. The knee point may be determined by inspection or as the value of ϵ at which the second derivative of the rate-distortion plot attains a maximum.

We can state the above strategy in the form of an algorithm.

```
Calibration Algorithm
input $\hat{f}_n$
begin
    Run $C(\hat{f}_n, \epsilon)$ for various values of
    $\epsilon$, and plot output size versus $\log(\epsilon)$;
    Let $\epsilon^*$ be the knee point of this plot, i.e the point
    at which its second derivative attains a maximum;
    Output $\epsilon^*$ as an estimate for the strength of the noise;
end
```

5. The Piecewise Linear Representation

In this section we select a compression algorithm that operates in terms of the piecewise linear functions and obeys the L_∞ or amplitude metric. Using this compression algorithm we build an Occam filter and examine its properties.

The compression algorithm does the following. Given a sequence f_n and a tolerance ϵ, the algorithm constructs a piecewise linear function g such that $\|g_n, f_n\|_\infty \leq \epsilon$, and g consists of the fewest number of pieces over all such piecewise linear functions.

The output of the compression algorithm is the sequence of break points of the piecewise linear function g. Decompression is achieved by linear interpolation of the break points.

It happens that the compression scheme described above can be implemented optimally as an algorithm requiring time linear in the number of input points, using visibility techniques. Details can be found in Imai and Iri (1986), or in Natarajan (1991) where the case of univariate functions in higher dimensions is also examined. Also, Konstantinides and Natarajan (1994) describe a simplified form of the general algorithm that is amenable to hardware implementation.

We now use the above compression algorithm to construct a filter. As the noise-free signal, we select the function

$$f(x) = \begin{cases} 0 & x \leq 0.2 \\ \sin\left(\dfrac{1}{(x-0.2)+0.03}\right) & \text{otherwise}. \end{cases} \tag{4}$$

This is a broad-band signal, i.e., it has broad spectral support, and is difficult to filter with a classical spectral filter. Figure 5.1 is a plot of 1000 uniformly spaced samples of the function.

We now add random noise to the function. We select the noise to be a uniformly distributed random variable in the range [-0.1,+0.1]. Figure 5.2 shows the 1000 samples of Figure 5.1, corrupted with noise generated by a pseudo-random number generator obeying the above distribution.

We now run the Calibration Algorithm given earlier on the noisy samples. Specifically, we run the piecewise linear compression algorithm on the noisy samples of Figure 5.2 for various values of the tolerance ϵ, and plot the compressed size of the samples versus $\log(\epsilon)$ in Figure 5.3. In the interest of simplicity, we measure the compressed size in terms of the number of vertices defining the piecewise linear representation output by the compression algorithm, rather than the number of bits required to store these vertices. Since each vertex is stored as a fixed and constant number of bits in the machine representation, this simplification does not affect our calculations. Figure 5.3 also shows the second derivative of the plot, calculated as central differences of the first derivative, which in turn was calculated as the central differences of the values themselves. The second derivative curve attains a maximum at $\epsilon^* = 0.0985$. This is the knee point of the curve, and is the estimate of the noise strength in the L_∞ metric. Compare this estimate with the true value of .1 as dictated by our selection of the noise variable to be uniformly distributed over the range [-0.1, +0.1].

Using the knee point value as the estimate for the noise strength in the Filtering Algorithm, we now run the compression algorithm on the noisy samples of Figure 5.2, with tolerance $\epsilon = 0.0985$. The result is the piecewise linear function of Figure 5.4. Visually, the function of Figure 5.4 appears to have much less noise than the noisy signal of Figure 5.2.

We now run the compression algorithm for various values of the tolerance ϵ. At each value, we decompress the output of the compression algorithm to obtain the sequence g_n. We then measure the residual noise in g_n by computing $\|g_n, f_n\|_1$. Figure 5.5 is a plot of this residual noise against $\log(\epsilon)$. Notice that the plot exhibits a minimum at about the knee point ϵ^*. This minimum value of roughly 0.01 corresponds to the residual noise in the filtered sequence shown in Figure 5.4, and should be compared to the average absolute value of the noise of 0.05.

Next we consider the effect of increasing the sampling rate n. Figure 5.6 shows the plots of output size versus $\log(\epsilon)$ for three different sampling rates, $n = 1000$, 2000 and 4000. In essence, Figure 5.6 is a parametrization of Figure 5.3 It is clear that the knee point does not change appreciably as the sampling rate is increased, but it is better defined. Similarly, we plot residual noise versus log(ϵ) for the three different sampling rates to obtain Figure 5.7. In essence, Figure 5.7 is a parametrization of Figure 5.5. Notice in Figure 5.7 that for each of three sampling rates, the minimum residual noise occurs at roughly the knee point, i.e., $\epsilon = \epsilon^*$, but as the sampling rate increases, the minimum residual noise decreases.

Next we change the distribution of the noise variable ν from the uniform distribution to a distribution with non-zero mean. Specifically, the noise variable has the probability density function shown in Figure 5.8: ν is uniform in [-0.1, +0.02] with probability 1/8, ν is uniform in [0.02,0.05], with probability 6/8, and ν is uniform in [0.05,0.1] with probability 1/8. The mean of the distribution is 0.030625. Figure 5.9 shows the 1000 samples of the function of Figure 5.1 corrupted with random noise under the new distribution, generated by a pseudo-random number generator. Figure 5.10 shows the knee point plot for these 1000 samples with the knee point occurring at 0.0985 again. Figure 5.11 shows the filtered function obtained by compressing and decompressing at the knee point. Note that the filtered function of Figure 5.11 is not displaced from the noise-free function of Figure 5.1 by the mean value of the noise, i.e., the filter has eliminated the mean value of the noise as well.

We now compare the performance of the Occam filter we constructed above with a classical filter–the Wiener filter, Tretter (1976). The Wiener filter is known to be optimal among linear filters in the least square sense.

The Wiener filter requires that the spectral properties of the noise-free signal and the noise be known in advance. Its transfer function is given by

$$H(\omega) = \frac{S(\omega)}{S(\omega)+N(\omega)} . \tag{5}$$

where $S(\omega)$ is the power spectral density of the noise-free signal and $N(\omega)$ is the power spectral density of the noise. Since we assume the noise variable to be statistically independent at each sample point, it has uniform power spectral density and $N(\omega)$ is a constant that depends only on the variance of the noise distribution.

We implement the Wiener filter as follows: We take the discrete Fourier transform of the noise-free sequence. The sum of the squares of the imaginary and the real part of the transform at each frequency is the power spectral density at that frequency. Also, the power spectral density of the noise at each frequency is equal to the average power of the noise over all frequencies. Using these estimates we compute the transfer function $H(\omega)$ at each frequency. We then compute the discrete Fourier transform of the noisy sequence, and multiply each term of the transform with the corresponding term of the transfer function. Inverting the modified discrete Fourier transform gives the filtered sequence.

Figure 5.12 shows the result of filtering the sequence of Figure 5.2 with a Wiener filter. The residual noise in the filtered sequence is 30×10^{-3} in the L_1 sense and 15×10^{-4} in the L_2 sense. These numbers should be compared to the residual noise in Figure 5.4 achieved with the Occam filter of 8.3×10^{-3} in the L_1 sense and 5×10^{-4} in the L_2 sense.

To verify that the residual noise values of the Occam and Wiener filter reported above are statistically significant, we repeated the filtering experiment a 100 times, with randomly chosen values for the seed of the pseudo-random number generator generating the random noise values. At each experiment, we filtered the noisy samples with both the Occam and the Wiener filters. In the L_2 metric, the average value of the residual noise over these experiments was 4.1×10^{-4} for the Occam filter, and 15×10^{-4} for the Wiener filter. In the L_1 metric, the average value of the residual noise over these experiments was 9.0×10^{-3} for the Occam filter, and 30×10^{-3} for the Wiener filter.

6. The Spectral Representation

In this section we select a compression algorithm that operates in terms of the sine and cosine functions and obeys the L_2 or power metric. Using this compression algorithm we will build an Occam filter and examine its properties.

The compression algorithm that we use does the following. Given a sample sequence f_n and a tolerance ϵ, the algorithm computes a sequence g_n such that $\|g_n, f_n\|_2 \leq \epsilon$, and the discrete Fourier transform of g_n has the fewest non-zero terms over all such sequences. The non-zero terms of the transform is the compressed representation of g_n.

The above spectral compression algorithm is straightforward to implement by exploiting the orthogonality of the Fourier transform. Specifically, compute the discrete Fourier transform of f_n. In order of increasing size, set to zero terms of the transform so that the sum of the squares of the terms set to zero does not exceed ϵ. To decompress, simply invert the transform to obtain the sequence g_n.

As the noise-free signal f we choose

$$f(x) = x\sin(6\pi x) - x\cos(40\pi x)\,. \tag{6}$$

A sequence of 1000 samples of f is shown in Figure 6.1. We now add random noise to these samples. We select the noise to be Gaussian distributed with zero mean and variance .003. Figure 6.2 shows the 1000 samples of Figure 6.1, corrupted with random noise, generated by a pseudo-random number generator.

We now run the Calibration Algorithm on the noisy sequence. Specifically, we run the spectral compression scheme described above on the noisy samples of Figure 6.2 for various values of the tolerance ϵ, and plot compressed size versus $\log(\epsilon)$ in Figure 6.3. In the interest of simplicity, we measure the compressed size in terms of the number of non-zero frequency terms in the discrete Fourier transform output by the compression algorithm, rather than the number of bits required to store these terms. Since each term is stored as a fixed and constant number of bits in the machine representation, this simplification does not affect our calculation of the knee point. Figure 6.3 also shows the second derivative of the plot, calculated as central differences of the first derivative, which in turn was calculated as the central differences of the values themselves. The second derivative plot attains a maximum at $\epsilon^* = .00394$, and this is the knee point of the plot. This is the estimate of the noise strength in the L_2 metric as determined by the Calibration Algorithm.

Using $\epsilon = \epsilon^*$ in the Filtering Algorithm, we get the filtered sequence of Figure 6.4. The residual noise in the filtered sequence is 8.2×10^{-4} in the L_2 metric.

Figure 6.5 shows the result of filtering the noisy sequence of Figure 6.2 with a Wiener filter. The residual noise in the filtered sequence is 6.5×10^{-4} in the L_2 metric.

To verify that the residual noise values of the Occam and Wiener filter reported above are statistically significant, we repeated the filtering experiment a 100 times, with randomly chosen values for the seed of the pseudo-random number generator generating the random noise values. At each experiment, we filtered the noisy samples with both the Occam and the Wiener filters. In the L_2 metric, the average value of the residual noise over these experiments was 8.3×10^{-4} for the Occam filter, and 6.8×10^{-4} for the Wiener filter.

Part 2: Theory

We now establish convergence bounds on Occam filters. Specifically, we estimate the residual noise in the filtered sequence as a function of the sampling rate, and the compression achieved by the compression algorithm. The bounds that we establish are largely of asymptotic interest, and are too loose to be used as quality measures for applications.

Our proof technique requires that the metric obeyed by the compression algorithm be linear in that the measure of the noisy sequence be the sum of the measures of the noise and the noise-free signal, in the limit as the sampling rate becomes infinite.

$$\lim_{n \to \infty} \|\hat{f}_n\| = \|f_n\| + \|\nu\| . \tag{7}$$

Since ν is statistically independent of f_n, the amplitude metric L_∞ is linear. If ν is of zero mean the power metric L_2 is linear, since the variance of the sum of two random variables is the sum of their variances, provided that at least one of the random variables is of zero mean.

We assume that the noise-free signal is a function from the interval $[0,1]$ to the interval $[-a, +a]$, and that the noise ν is a random variable in the interval $[-b, +b]$. Later we mention how these restrictions may be removed.

For a random event A, $\mathbf{Pr}\{A\}$ denotes the probability of A occurring. For a random variable x, $\mathbf{E}\{x\}$ denotes the expected value of x.

7. The L_∞ Metric

Let C be a compression algorithm that obeys the L_∞ metric. Let S_n denote the expected length in bits of the output of the compression algorithm when given $\hat{f}_n$ and $b = \|\nu\|_\infty$ as input. The expectation is taken over all possible values of the noise corrupting the sequence. Specifically,

$$S_n = \mathbf{E}\left\{ \left| C(\hat{f}_n, b) \right| \right\} . \tag{8}$$

With respect to the probability distribution of the noise variable ν, the tail estimate $\tau(\zeta)$ denotes the minimum probability that ν will fall within ζ of $+b$ or $-b$. Specifically,

$$\tau(\zeta) = \min\left\{ \mathbf{Pr}\{\nu \in [b-\zeta, b]\}, \mathbf{Pr}\{\nu \in [-b, -(b-\zeta)]\} \right\} . \tag{9}$$

Theorem 1: Let g_n be the sequence output by the filtering algorithm, and let $\epsilon = \|g_n, f_n\|_1$. For any $0 \le \delta \le 1$, with probability $1-\delta$,

$$\epsilon\tau(\epsilon/2) \le \frac{4a}{n} \left((2/\delta) S_n \ln(2) + \ln(2/\delta) \right) . \tag{10}$$

Proof: Let h_n be a sequence such that $\|h_n, f_n\|_1 > \epsilon$. Then, there is a set of at least $n\epsilon/(4a)$ distinct integers i, $0 \le i < n$ such that $|f(i/n) - h(i/n)| > \epsilon/2$. Let i be one such integer. Suppose $f(i/n) - h(i/n) > \epsilon/2$. If the noise variable ν takes on a value in $[-b, (b - \epsilon/2)]$ at this sample point, then $|f(i/n) + \nu - h(i/n)| \le b$. The probability of this occurring is at most $1 - \tau(\epsilon/2)$, by the definition of τ. On the other hand, suppose $h(i/n) - f(i/n) > \epsilon/2$. Then, if the noise variable ν takes on a value in $[-(b - \epsilon/2), b]$, $|f(i/n) + \nu - h(i/n)| \le b$. The probability of this occurring is also at most $1 - \tau(\epsilon/2)$.

Thus, the probability that $|f(i/n) + \nu - h(i/n)| \le b$ at a particular value of i for which $|f(i/n) - h(i/n)| > \epsilon/2$, is at most $1 - \tau(\epsilon/2)$. It follows that the probability that $|f(i/n) + \nu - h(i/n)| \le b$ at all of the $n\epsilon/(4a)$ values of i for which $|f(i/n) - h(i/n)| > \epsilon/2$, is at most $(1 - \tau(\epsilon/2))^{n\epsilon/(4a)}$. In other words, the probability that $\|\hat{f}_n, h_n\|_\infty \le b$ is at most

$$(1 - \tau(\epsilon/2))^{n\epsilon/(4a)} \le e^{-n\tau(\epsilon/2)\epsilon/(4a)} . \tag{11}$$

Thus, if h_n is a particular sequence satisfying $\|h_n, f_n\|_1 > \epsilon$, the probability that $\|\hat{f}_n, h_n\|_\infty \le b$ is at most $e^{-n\tau(\epsilon/2)\epsilon/(4a)}$. Since the expected length of the output of the compression algorithm is S_n bits, with probability at most $1 - \delta/2$, the output of the compression algorithm is of length at most $(2/\delta)S_n$. Otherwise, the expected length would be greater than $\delta/2(2/\delta)S_n$, which is greater than S_n. We work on the condition that the output length is at most $(2/\delta)S_n$ and later account for this conditionality. That is, we only consider sequences that are representable with at most $(2/\delta)S_n$ bits. The probability that any such sequence h_n will be such that $\|h_n, f_n\|_1 > \epsilon$ but $\|\hat{f}_n, h_n\|_\infty \le b$ is at most

$$2^{(2/\delta)S_n} e^{-n\tau(\epsilon/2)\epsilon/(4a)} . \tag{12}$$

Since the sequence g_n output by the filtering algorithm satisfies $\|g_n, \hat{f}_n\|_\infty \le b$, it follows that the probability that $\|g_n, f_n\|_1 > \epsilon$ is at most the quantity in (12). Setting this probability to be at most $\delta/2$ and taking logarithms on both sides, we get that with probability $1 - \delta/2$,

$$\epsilon\tau(\epsilon/2) \le \frac{4a}{n}\Big((2/\delta)S_n \ln(2) + \ln(2/\delta)\Big) . \tag{13}$$

The above estimate is conditional on the output of the compression algorithm being of length at most $(2/\delta)S_n$. Eliminating this condition will lower the probability of the above estimate by a factor of $(1 - \delta/2)$. Hence, the estimate holds with probability at least $(1 - \delta/2) \times (1 - \delta/2)$. Since $(1 - \delta) > (1 - \delta/2)^2$, the proof is complete. □

Corollary: If the noise variable is uniformly distributed over $[-b, +b]$, then

$$\epsilon^2 \leq \frac{16ab}{n}\left((2/\delta)S_n \ln(2) + \ln(2/\delta)\right). \tag{14}$$

Proof: If the noise is uniformly distributed in the range $[-b, +b]$, $\tau(\zeta) = \zeta/2b$. Substituting in Theorem 1 we get the corollary. □

Remark 1: According to the Corollary, when the noise is uniformly distributed, the L_1 measure of the residual noise varies as the square root of the sampling rate. This is roughly the case in Figure 5.7, where the residual noise at the knee point is 8×10^{-3}, 5×10^{-3} and 4×10^{-3}, at sampling rates of 1000, 2000 and 4000.

Remark 2: Notice that the proof of Theorem 1 did not require that the noise be of zero mean. Thus, an Occam filter constructed out of a compression algorithm that obeys the L_∞ metric can be used to filter noise that is not of zero mean. Indeed, this was the case in the application of part 1, Figures 5.8 through 5.11.

Remark 3: The convergence bound in Theorem 1 depends only on the tails of the noise distribution, via $\tau(\epsilon)$. Thus, if the noise distribution is tightly clustered about its mean, $\tau(\zeta)$ is small compared to ζ and the convergence of the filter is slower than for the uniform distribution. For the distribution of Figure 5.8, $\tau(\zeta) = \zeta/0.8$, while while for the uniform distribution, $\tau(\zeta) = \zeta/0.2$. Thus, according to Theorem 1, the L_1 measure of the residual noise in Figure 5.11 should be $\sqrt{(4)} = 2$ times the L_1 measure of the residual noise in Figure 5.4. Indeed, this is borne out by experiment, since the L_1 measure of the residual noise is 8.3×10^{-3} and 16.6×10^{-3} in Figures 5.4 and 5.11 respectively.

8. The L_2 Metric

Let C be a compression algorithm that obeys the L_2 metric, and let ν be of zero mean and variance σ^2. Let S_n denote the expected length in bits of the output of the compression algorithm when given $\hat{f}_n$ and σ^2 as input. The expectation is taken over all possible values of the noise corrupting the sequence. Specifically,

$$S_n = \mathbf{E}\left\{\left|C(\hat{f}_n, \sigma^2)\right|\right\}. \tag{15}$$

Theorem 2: Let g_n be the sequence output by the filtering algorithm, and let $\epsilon = \|g_n, f_n\|_2$. For any $0 \leq \delta \leq 1$, with probability $1 - \delta$,

$$\epsilon^2 \leq \frac{18b^2(a^2+b^2)}{n}\left((2/\delta)S_n \ln(2) + \ln(12/\delta)\right). \tag{16}$$

Proof: Let h_n be a sequence such that $\|h_n, f_n\|_2 > \epsilon$.

$$\|h_n, \hat{f}_n\|_2 = \frac{1}{n}\sum_0^{n-1}\Big(f(i/n) + \nu - h(i/n)\Big)^2 . \tag{17}$$

$$= \frac{1}{n}\sum_0^{n-1}\Big(f(i/n) - h(i/n)\Big)^2 + \frac{1}{n}\sum_0^{n-1}\nu^2 + \frac{1}{n}\sum_0^{n-1}2\nu\Big(f(i/n) - h(i/n)\Big) . \tag{18}$$

$$= \|h_n, f_n\|_2 + \frac{1}{n}\sum_0^{n-1}\nu^2 + \frac{1}{n}\sum_0^{n-1}2\nu f(i/n) - \frac{1}{n}\sum_0^{n-1}2\nu h(i/n) . \tag{19}$$

If $\|h_n, \hat{f}_n\|_2 \le \sigma^2$, then at least one of the following must hold.

$$\left|\frac{1}{n}\sum_0^{n-1}\nu^2\sigma^2\right| > \frac{\epsilon}{3} , \quad \left|\frac{1}{n}\sum_0^{n-1}2\nu f(i/n)\right| > \frac{\epsilon}{3} , \quad \left|\frac{1}{n}\sum_0^{n-1}2\nu h(i/n)\right| > \frac{\epsilon}{3} . \tag{20}$$

It follows that

$$\Pr\left\{\|h_n, \hat{f}_n\|_2 \le \sigma^2\right\} \le \Pr\left\{\left|\frac{1}{n}\sum_0^{n-1}\nu^2 - \sigma^2\right| > \frac{\epsilon}{3}\right\} +$$

$$\Pr\left\{\left|\frac{1}{n}\sum_0^{n-1}2\nu f(i/n)\right| > \frac{\epsilon}{3}\right\} + \Pr\left\{\left|\frac{1}{n}\sum_0^{n-1}2\nu h(i/n)\right| > \frac{\epsilon}{3}\right\} . \tag{21}$$

Noting that $|f(i/n)| \le a$ and $|h(i/n)| \le a$, and $\nu \in [-b, +b]$, we can rewrite the above as

$$\Pr\left\{\|h_n, \hat{f}_n\|_2 \le \sigma^2\right\} \le \Pr\left\{\left|\frac{1}{n}\sum_0^{n-1}\nu^2 - \sigma^2\right| > \frac{\epsilon}{3}\right\} +$$

$$2\Pr\left\{\left|\frac{1}{n}\sum_0^{n-1}\zeta\right| > \frac{\epsilon}{3}\right\} , \tag{22}$$

where ζ is a random variable in the range $[-ab, +ab]$. Furthermore, since ν is of zero mean so is ζ. Noting that ν^2 has a mean of σ^2 and lies in the range $[0, b^2]$, we can invoke Hoeffding's inequality, Pollard (1984), to get

$$\Pr\left\{\left|\frac{1}{n}\sum_0^{n-1}\nu^2 - \sigma^2\right| > \frac{\epsilon}{3}\right\} \le 2e^{-\frac{2n\epsilon^2}{9b^4}} . \tag{23}$$

Similarly, noting that ζ has a mean of zero and lies in the range [$-ab$, $+ab$], we can invoke Hoeffding's inequality to get

$$\Pr\left\{\left|\frac{1}{n}\sum_{0}^{n-1}\zeta\right| > \frac{\epsilon}{3}\right\} \leq 2e^{-\frac{n\epsilon^2}{18b^2a^2}} . \tag{24}$$

Substituting, we get

$$\Pr\left\{\|h_n,\hat{f}_n\|_2 \leq \sigma^2\right\} \leq 2e^{-\frac{2n\epsilon^2}{9b^4}} + 4e^{-\frac{n\epsilon^2}{18b^2a^2}} \leq 6e^{-\frac{n\epsilon^2}{18b^2(a^2+b^2)}} . \tag{25}$$

Since the expected length of the output of the compression algorithm is S_n bits, with probability at least $1-\delta/2$, the output of the compression algorithm is a sequence of length at most $(2/\delta)S_n$. Otherwise, the expected length would be greater than $\delta/2(2/\delta)S_n$, which is greater than S_n. We work on the condition that the output length is at most $(2/\delta)S_n$ and later account for this conditionality. That is, we only consider sequences that are representable with at most $(2/\delta)S_n$ bits. The probability that any such sequence h_n will be such that $\|h_n,f_n\|_2 > \epsilon$ but $\|h_n,\hat{f}_n\|_2 \leq \sigma^2$ is at most

$$2^{(2/\delta)S_n} 6e^{-\frac{n\epsilon^2}{18b^2(a^2+b^2)}} . \tag{26}$$

Since the sequence g_n output by the filtering algorithm satisfies $\|g_n,\hat{f}_n\|_2 \leq \sigma^2$, it follows that the probability that $\|g_n,f_n\|_2 > \epsilon$ is at most the quantity in (26). Setting this probability to be at most $\delta/2$ and taking logarithms on both sides, we get that with probability $1-\delta/2$,

$$\epsilon^2 \leq \frac{18b^2(a^2+b^2)}{n}\Big((2/\delta)S_n \ln(2) + \ln(12/\delta)\Big) . \tag{27}$$

The above estimate is conditional on the output of the compression algorithm being of length at most $(2/\delta)S_n$. Eliminating this condition will lower the probability of the above estimate by a factor of $(1-\delta/2)$. Hence, the estimate holds with probability at least $(1-\delta/2)\times(1-\delta/2)$. Since $(1-\delta) > (1-\delta/2)^2$, the proof is complete. □

Remark 4: In the foregoing, we assumed that the noise variable ν lies in the bounded ranged [$-b$, $+b$]. This assumption was necessary for Theorem 1, since we assumed that the L_∞ measure of the noise was bounded and known. For Theorem 2, it is only necessary that the L_2 measure of the noise be bounded and known.

The assumption that the noise lies in the range [$-b$, $+b$], where b is bounded but not necessarily known, was only in the interest of technical simplicity. In particular, the bounded range assumption allowed us to use Hoeffding's inequality. We can release the bounded range assumption if the noise distribution is such that empirical estimates of the variance and the mean of the noise variable can be shown to converge exponentially in the sampling rate. This is the case for the normal distribution for instance, which does not satisfy the bounded range assumption. In short, the bounded range assumption is solely in the interest of technical simplicity and is not an essential ingredient.

9. Conclusion

We presented a novel technique for the design and analysis of a class of filters for random noise—Occam filters. These filters are in general non-linear. The technique exploits the property that random noise is hard to compress, as compared to useful signals. The technique is simple to use: run a lossy compression algorithm with the allowed loss set equal to the noise strength.

We presented two illustrative applications of the technique for univariate signals, and compared the performance of Occam filters with the Wiener filter that is known to be optimal in the least square sense over the class of linear filters. We found that Occam filters can perform as well or better than the Wiener filter, without requiring a priori spectral knowledge of the signal. We also presented analytical convergence bounds relating the residual noise achieved by an Occam filter, with the compression achieved and the sampling rate.

Since the noise rejection characteristic of an Occam filter is directly related to the effectiveness of the compression algorithm, the filter designer should pick the compression algorithm best suited to the noise free signal. This is equivalent to selecting the representation that is best suited to describing the noise-free signal, rather than always selecting the spectral representation as is the case with classical spectral filters. This degree of freedom is an important feature of an Occam filter, since a signal may have a compact description in one representation but not in another. For instance, a square wave requires broad support in the spectral representation, but can be compactly represented as a piecewise linear function. Better yet, a square wave can be represented in terms of the three parameters, frequency, phase and amplitude, and can be compressed highly by a compression algorithm that exploits this property. An Occam filter using such a compression algorithm would have powerful noise rejection characteristics. Indeed a phase-locked loop is such a filter, and the robustness of a phase-locked loop in the face of noise is widely known.

10. Acknowledgements

I thank V. Bhaskaran, D. Donoho, A. Lempel, and K. Konstantinides for the discussions.

11. References

DeVore, R.A., and Lucier, B. J., (1992) Fast wavelet techniques for near-optimal image processing, IEEE Military Comm. Conf., IEEE Comm. Society, 1992.
Donoho, D. L., (1992). De-noising by soft thresholding, Tech. Rep 409, Dept. of Statistics, Stanford University, Stanford, CA.
Donoho, D. L., Johnstone, I.M., Kerkyacharian, G., and Picard, D., (1993). Wavelet Shrinkage: Asymptopia?, Tech. Rep. 419, Dept. of Statistics, Stanford University, Stanford, CA.
Imai, H., and Iri, M., (1986). An optimal algorithm for approximating a piecewise linear function. J. of Information Processing, Vol. 9, No. 3, pp. 159-162.
Konstantinides, K., and Natarajan, B.K., (1994). An architecture for lossy compression of waveforms using piecewise linear approximation, IEEE Transactions on Signal Processing, to appear.
Natarajan, B.K., (1991). On piece-wise linear approximations to curves, Tech. Rep. HPL-91-36, Hewlett Packard Labs, Palo Alto, CA.
Natarajan, B.K., (1993a). Filtering random noise via data compression, Proc. IEEE Data Compression Conference, Snowbird, Utah, pp.60-69.
Natarajan, B.K., (1993b). Occam's Razor for functions, Proc. ACM Conf. on Comp. Learning Theory, Santa Cruz, CA.
Natarajan, B.K., (1994). Sharper bounds on Occam filters and application to digital video, Proc. IEEE Data Compression Conference, Snowbird, Utah, pp.440-449
Pollard, D., (1984). *Convergence of stochastic processes*, Springer Verlag, N.Y.
Saito, N., (1994). Simultaneous noise suppression and signal compression using a library of orthonormal bases and the minimum description length principle, in *Wavelets in Geophysics*, Foufoula-Georgiou and Kumar,Eds, Academic Press, San Diego, CA.
Tretter, S.A., (1976) *Introduction to discrete-time signal processing*, John Wiley and Sons, New York, NY.

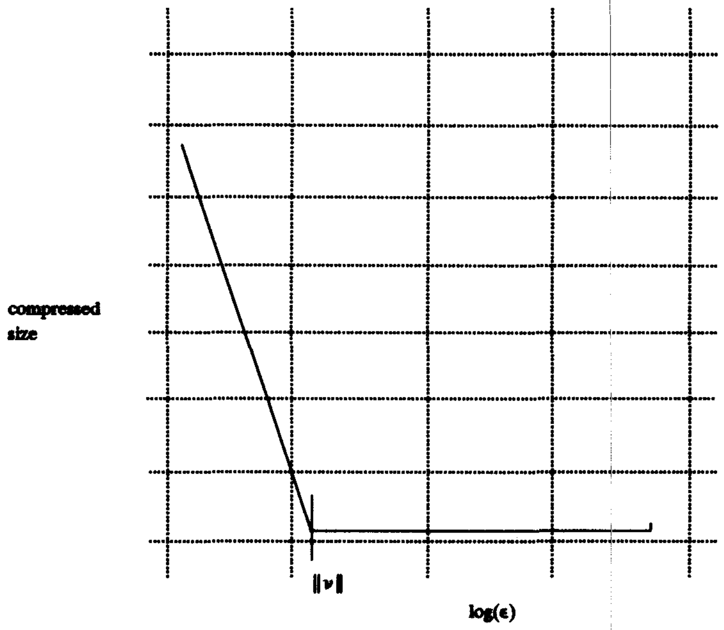

Figure 4.1: Illustrative Plot of compressed size versus allowed loss for the noise sequence.

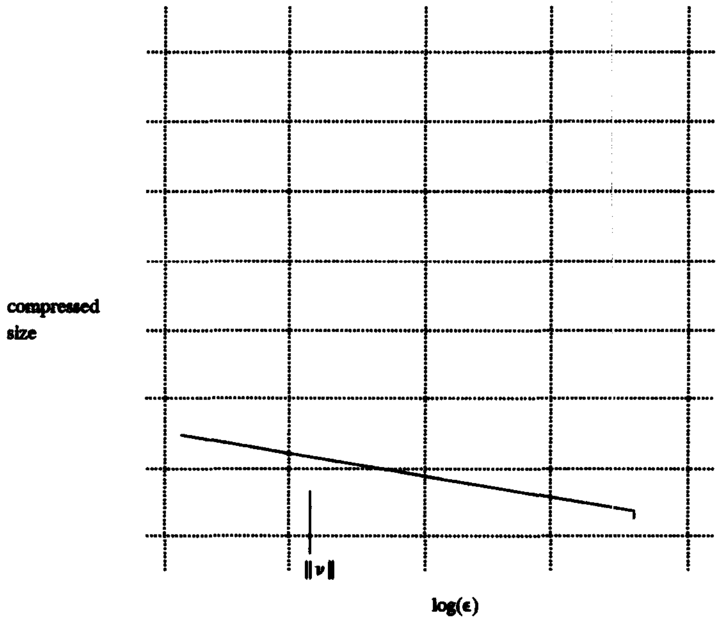

Figure 4.2: Illustrative Plot of compressed size versus allowed loss for the noise-free sequence.

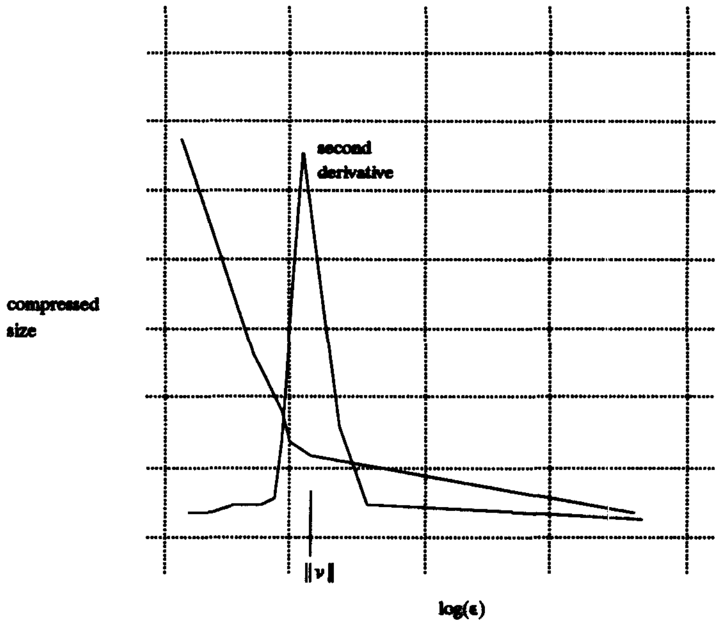

Figure 4.3: Illustrative Plot of compressed size versus allowed loss for the noisy sequence, with second derivative plot.

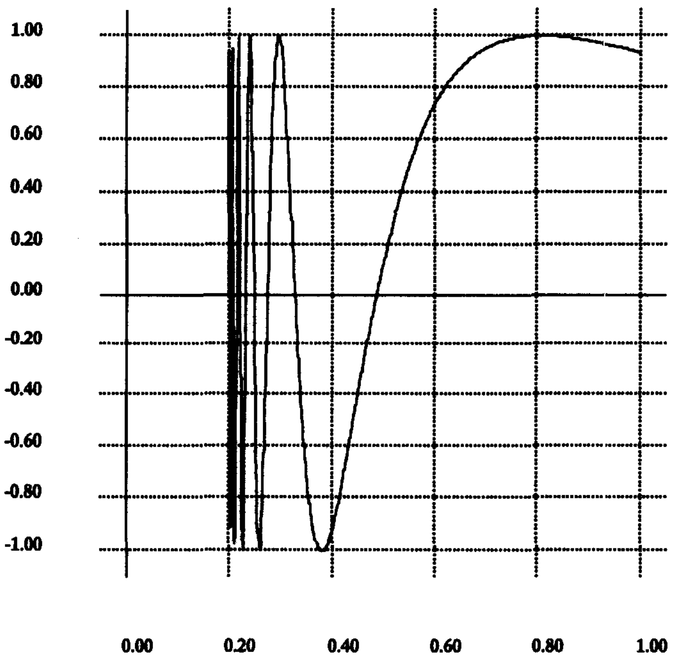

Figure 5.1: A sequence of 1000 samples.

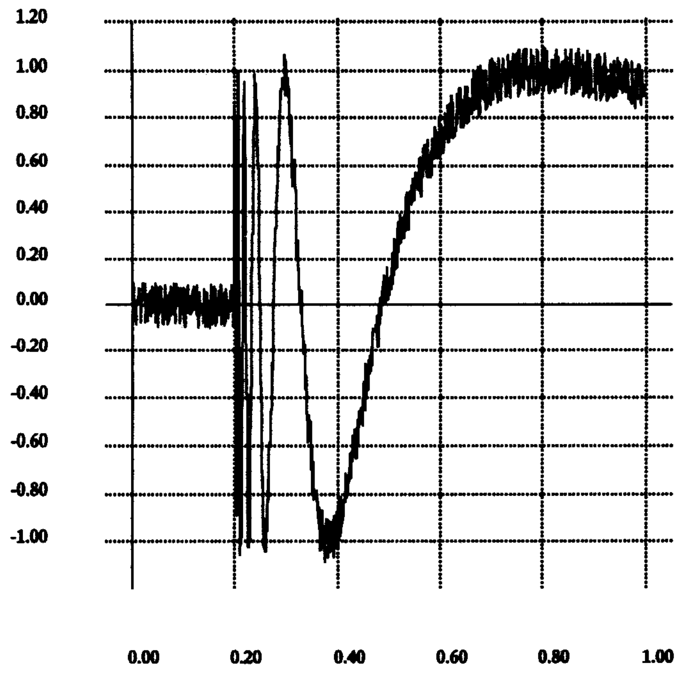

Figure 5.2: The sequence of Figure 5.1 corrupted with additive random noise uniformly distributed in the range [-.1,+.1].

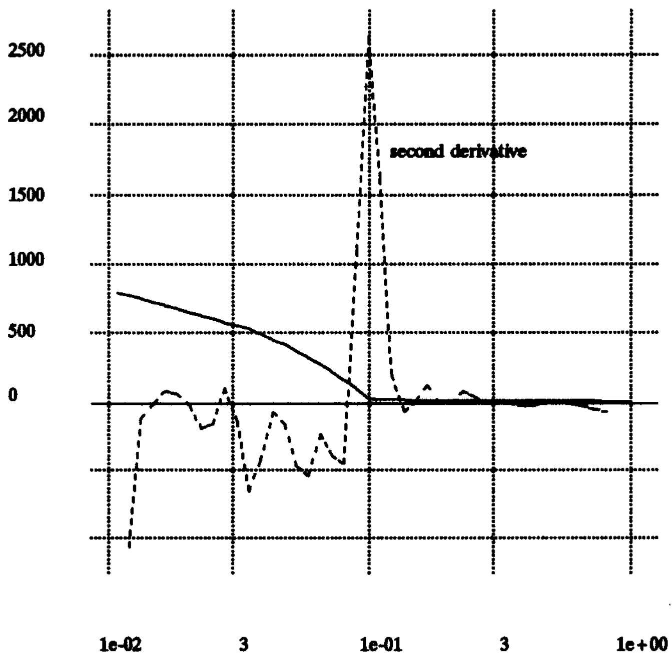

Figure 5.3: Plot of compressed size versus allowed loss ϵ (logarithmic scale) and the second derivative of the plot.

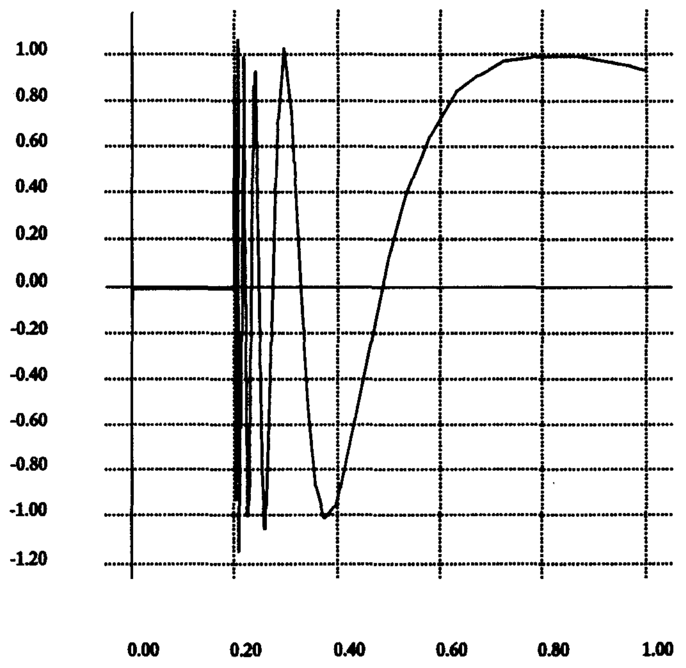

Figure 5.4: The sequence obtained by filtering the noisy sequence of Figure 5.2 using the piecewise linear Occam filter.

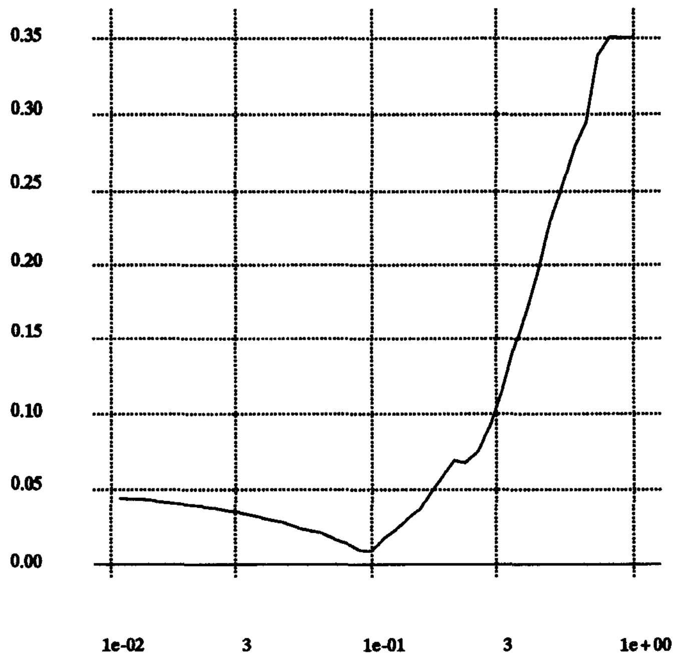

Figure 5.5: Residual noise versus allowed loss ϵ.

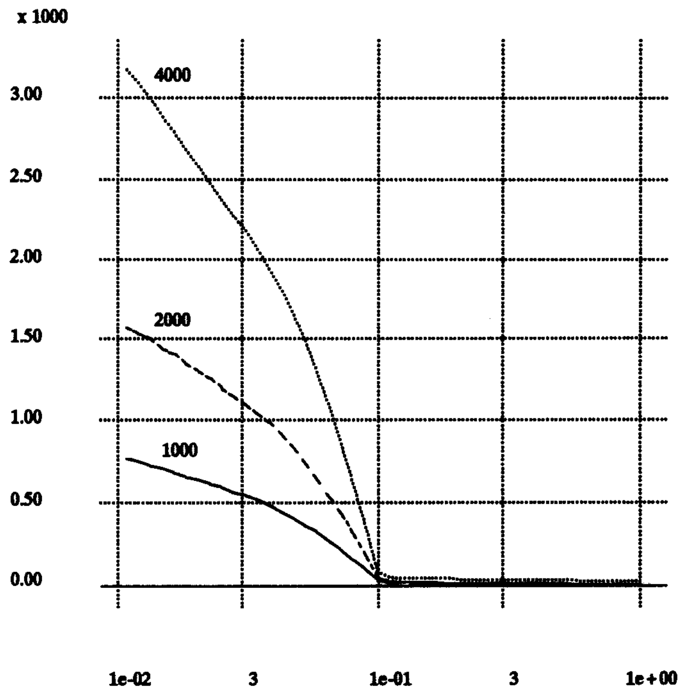

Figure 5.6: Plot of compressed size versus allowed loss ϵ (logarithmic scale) for sampling rates of 1000, 2000, and 4000.

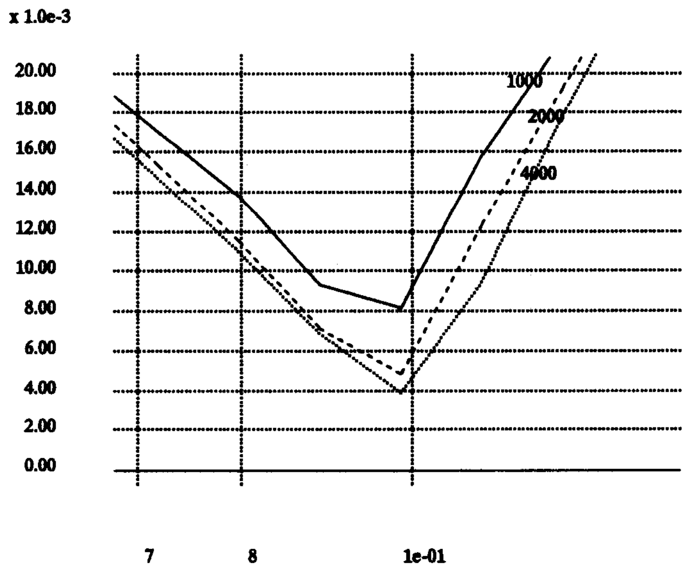

Figure 5.7: Residual noise versus allowed loss ϵ for sampling rates of 1000, 2000, and 4000.

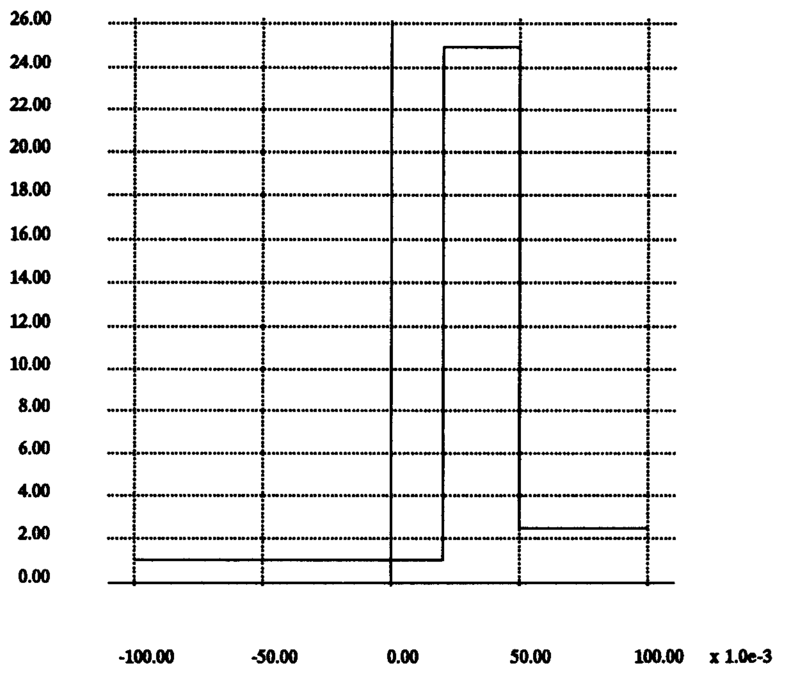

Figure 5.8: Probability density function of random noise variable with non-zero mean.

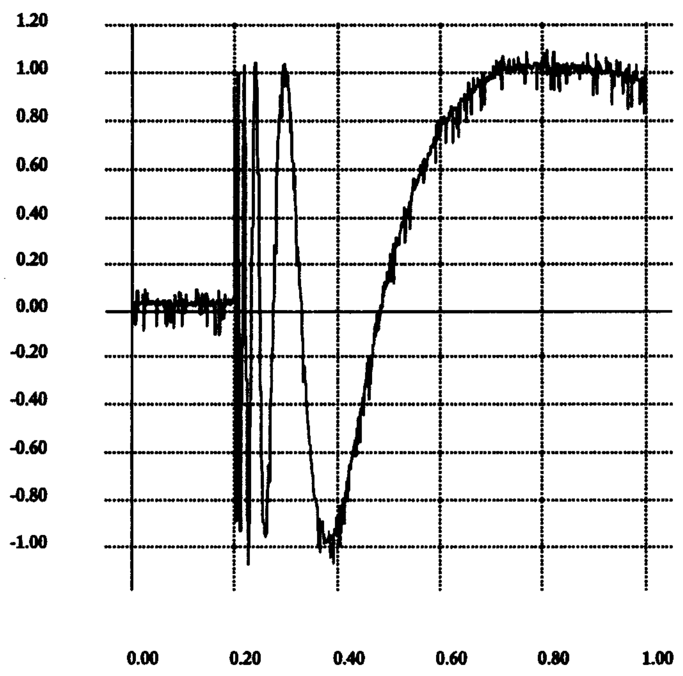

Figure 5.9: The sequence of Figure 5.1 corrupted with additive random noise distributed as in Figure 5.8.

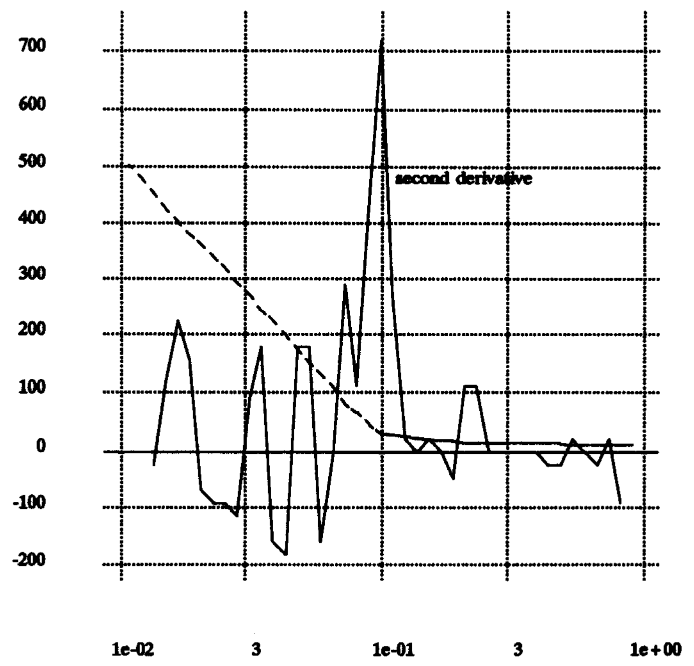

Figure 5.10: Plot of compressed size versus allowed loss ϵ (logarithmic scale) and the second derivative of the plot.

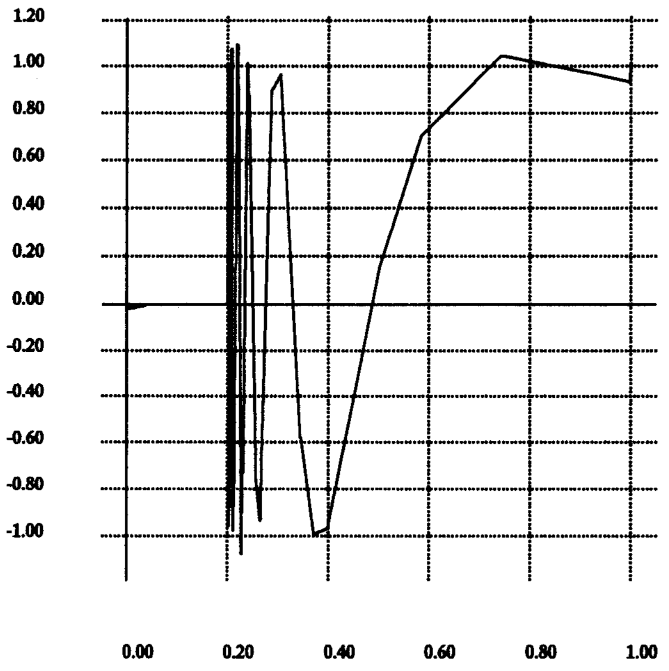

Figure 5.11: The sequence obtained by filtering the noisy sequence of Figure 5.9 using the piecewise linear Occam filter.

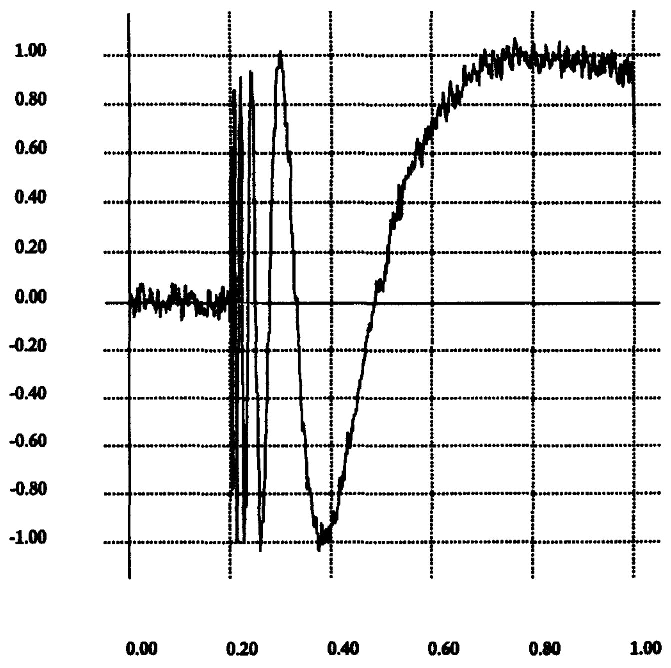

Figure 5.12: The sequence obtained by filtering the noisy sequence of Figure 5.2 using a Wiener filter.

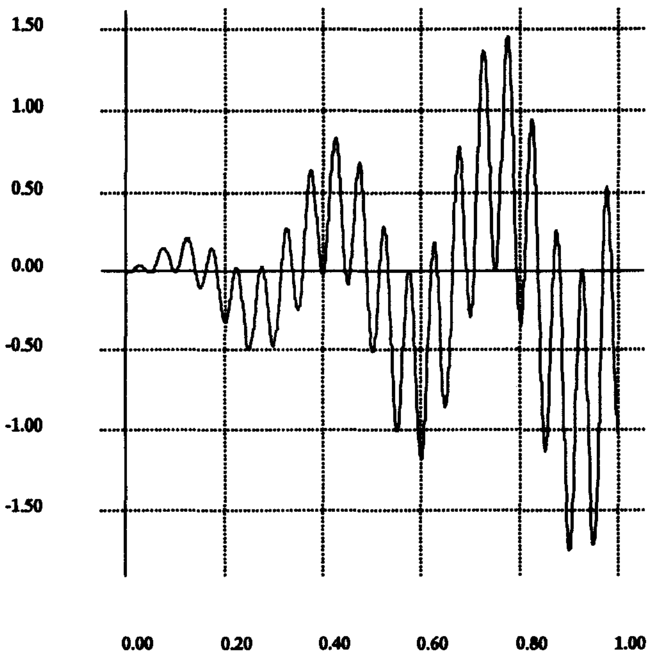

Figure 6.1: A sequence of 1000 samples.

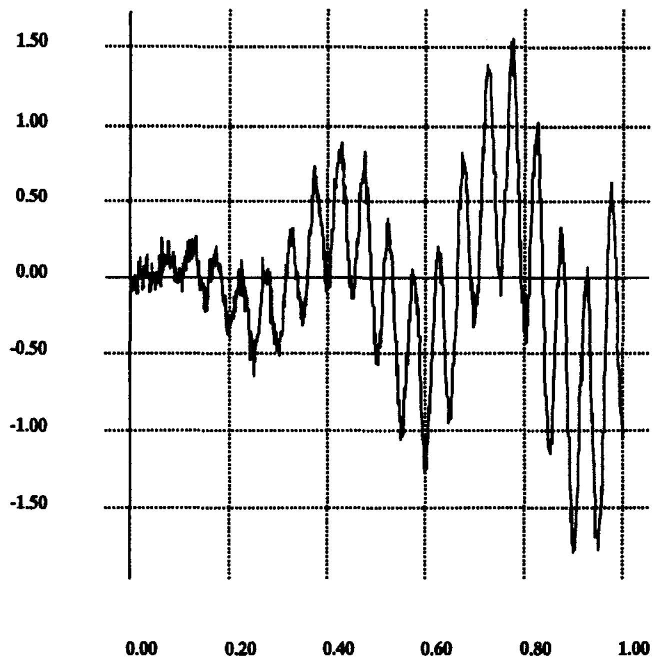

Figure 6.2: The sequence of Figure 6.1 corrupted with additive random noise normally distributed with zero mean and variance .003.

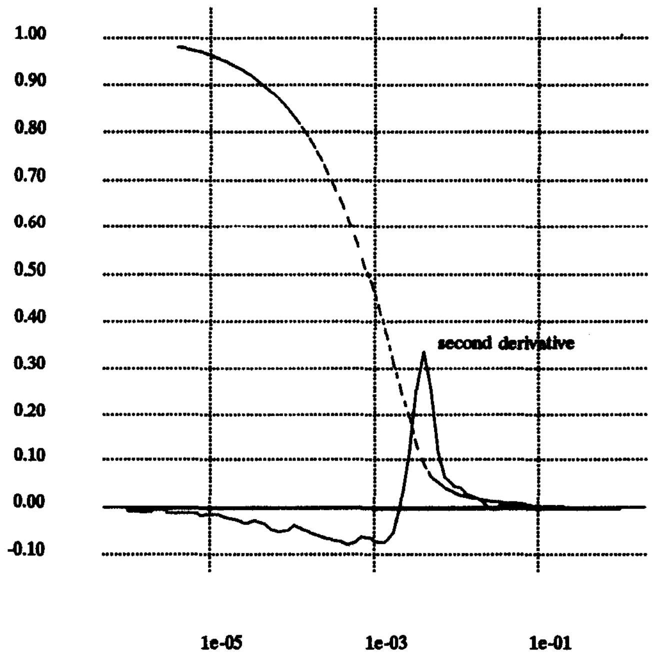

Figure 6.3: Plot of compressed size versus allowed loss ϵ (logarithmic scale) and the second derivative of the plot.

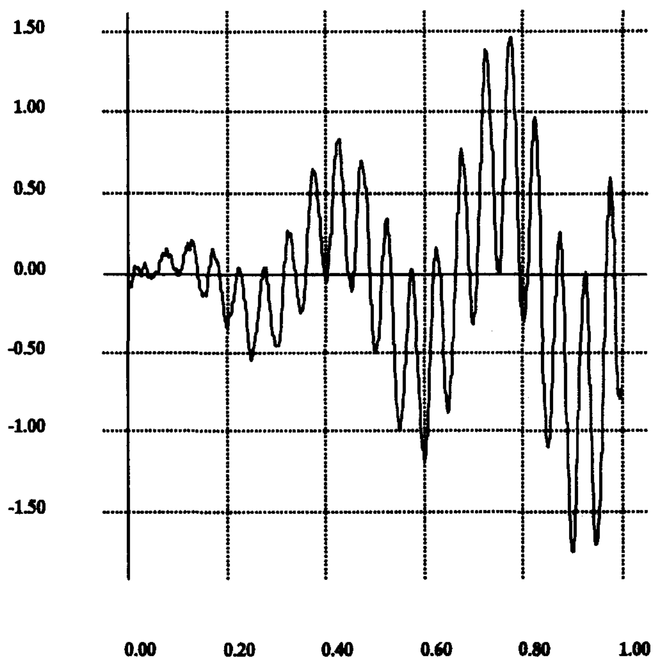

Figure 6.4 The sequence obtained by filtering the noisy sequence of Figure 3.2 using the piecewise linear Occam filter.

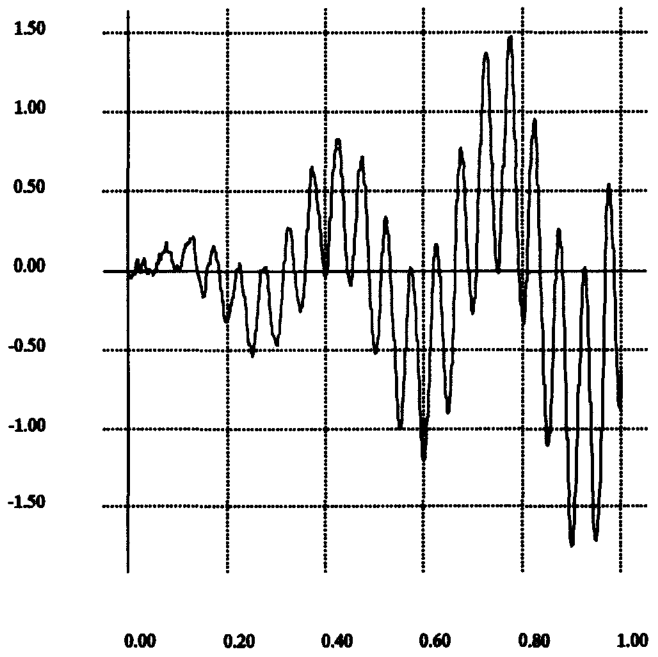

Figure 6.5: The sequence obtained by filtering the noisy sequence of Figure 3.2 using a Wiener filter.