

# Seismic data compression: a tutorial

*Tong Chen*

## ABSTRACT

A data compression system generally has three building blocks: the transformation, quantization and coding.

In this paper, I use the discrete wavelet packet transform (DWPT) as an example to introduce these building blocks. Specifically, I discuss issues as to why DWPT can be particularly helpful in compressing seismic data, how quantization and coding compresses data, and how to perform quantization and coding optimally, in the sense that might be appropriate for seismic data.

## INTRODUCTION

Seismic data volumes, these days, are huge and growing. With the emergence of 3D technology, the data volume is particularly large ( $> 10^{12}$  bytes are common in 3D surveys). Simply archiving these data will require a vast amount of storage. Moreover, as more data are processed and interpreted on workstations, more data transfer among the workstations through local area networks is required. It is therefore desirable to compress the data, in order to reduce the costs of storage and transmission.

There are two categories of data compression techniques: lossless and lossy. Lossless compression means no information is lost during the cycle of compression and decompression, and the original signal can be perfectly reconstructed from the compressed one. Lossy compression, on the other hand, means some information is lost during compression. Cost aside, lossless compression is what every customer would like since it provides a flawless reproduction of the original. Unfortunately, because seismic data are generally represented by floating point numbers throughout processing, true lossless compression of seismic data is inefficient for that compact representation. On the other hand, when data are sampled and recorded, some amount of error is already introduced. So, instead of trying to perfectly reproduce the original signal, it is realistic to make compromises that yield reproductions that are satisfactory for our purposes. Therefore, I focus the discussion here on lossy compression and specifically on transform-based lossy compression.

A transform-based, lossy compression technique consists of three building blocks: transformation, quantization and coding. First, some transform is applied to the

signal. After an appropriate transform, the energy of the signal can be concentrated into a relatively smaller region in the transformed domain than in the original data domain. Then the transformed coefficients (real numbers) are converted into a finite set of integer numbers through quantization. After quantization, fewer bits are used to approximate the coefficients, though at the expense of introducing some errors. Finally, the integer numbers are encoded to further reduce the number of bits required to represent the data.

Much of the original work in lossy compression can be found in the area of speech and image processing (e.g. Bellamy, 1991; Wallace, 1991; Gall, 1991). For seismic data, Wood (1974) discussed compression by truncating the Walsh transform of each trace. Bordley (1983), on the other hand, used linear predictive coding (LPC) to compress marine seismic data. Spanias et al. (1991) compared LPC with some of the transform-based compression techniques, such as the Karhunen-Loeve transform (KLT), the Walsh-Hadamard transform (WHT) and the discrete cosine transform (DCT). More recently, Luo and Schuster (1992) applied the wavelet packet transform to the compression of seismic data by discarding the small coefficients of the transform. Bosman and Reiter (1993) studied how the errors in the wavelet transform-based compression propagate through some processing modules. Reiter and Heller (1994) compared the compression errors for NMO-corrected CDP gathers and stacked sections and found that stacking can actually reduce the compression errors.

Different from the previous publications, in this paper I focus on gaining an understanding of data compression, particularly as it relates to the peculiarities of typical reflection seismic data. I will use the discrete wavelet packet transform (DWPT) as an example to discuss the issues as to why DWPT can be particularly helpful in compressing seismic data, how quantization and coding compresses data, and how to perform quantization and coding optimally, in the sense that might be appropriate for seismic data.

## DISCRETE WAVELET PACKET TRANSFORM

Wavelet packets, introduced by Coifman and Wickerhauser (1992) to compress speech signals, are closely related to the theory of wavelet transformation (Daubechies, 1992). Here, instead of giving rigorous mathematical definitions, I describe discrete transforms from a signal processing point of view.

Both the discrete wavelet transform and discrete wavelet packet transform involve two important filters: a high-pass filter  $D$  and a low-pass filter  $A$ . In the discrete wavelet transform, a signal  $x(n)$  is first decomposed by applying the two filters, and then the filtered data are subsampled — retaining only one sample in two so that the total number of samples remains unchanged. For notation purpose, I call the operation of high-pass filtering followed by subsampling  $G$  and the corresponding one for low-pass filtering  $H$ . After the first-stage decomposition, the output from  $H$  is further decomposed, and the process goes on until only one sample is left for the  $H$  operator, as illustrated by Figure 1. In the discrete wavelet packet transform, both

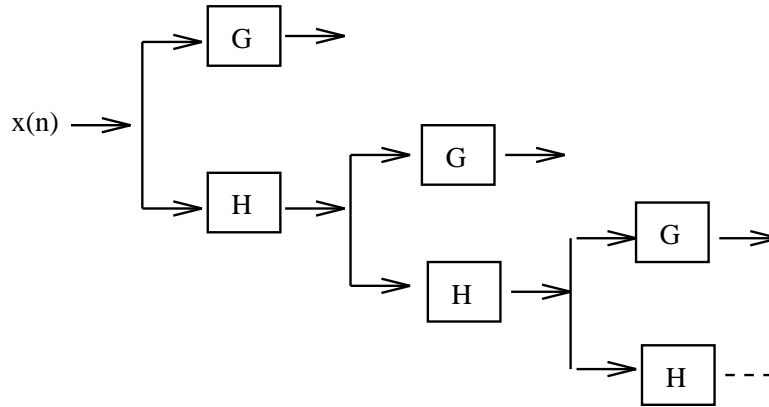


FIG. 1. The diagram for the discrete wavelet transform. The  $G$  and  $H$  operators correspond to first filtering by high-pass (for  $G$ ) and low-pass (for  $H$ ) filters and then subsampling the output by a factor of two.

the outputs from  $H$  and  $G$  are further decomposed, as shown in Figure 2. Consider

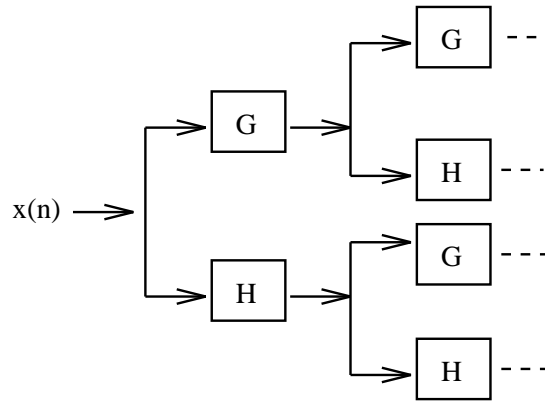


FIG. 2. The diagram for the discrete wavelet packet transform. The  $G$  and  $H$  operators correspond to first filtering by high-pass (for  $G$ ) and low-pass (for  $H$ ) filters and then subsampling the output by a factor of two.

the structure in this figure as a tree. For each node in the tree, there exists the choice of decomposing further or not. Therefore, there results a huge collection of possible valid decompositions, with the discrete wavelet transform being one of them. Here, for simplicity, I just decompose all the components to some fixed level.

Since the discrete wavelet packet transform (DWPT) involves iteratively applying the low- and high-pass filters, it achieves frequency partitioning as a result of the decomposition. Moreover, it is an unitary transform so that the root-mean-square (RMS) amplitude remains the same before and after the transform. Also, the process can be reversed to obtain the reconstruction or inverse transform using the same filters  $D$  and  $A$  (called *conjugate quadrature filters* in signal processing literature).

It is not difficult to extend DWPT to higher dimensions. A straightforward way to generate a 2D DWPT is to apply two 1D transforms separately along the two

dimensions, i.e., to cascade two 1D DWPTs. As can be imagined, 2D DWPT partitions the data into different frequency bands, along both dimensions. Figure 3 shows a stacked section, and Figure 4 shows the section after 2D DWPT. Here, I did three

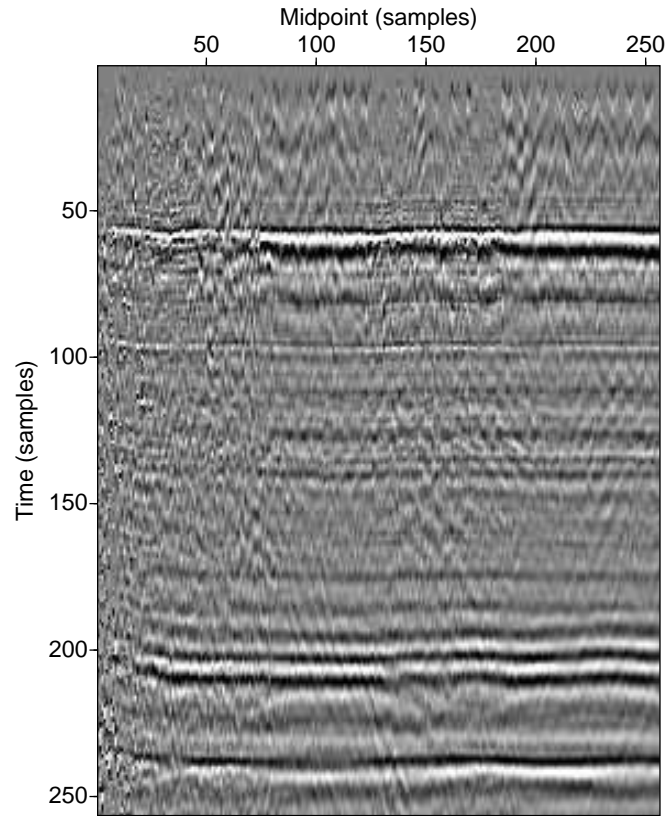


FIG. 3. A stacked section.

levels of decomposition along both the time and space dimensions. The filters  $D$  and  $A$  used here correspond to a specific type of wavelets called fourth-order *coiflets* (Daubechies, 1992). As shown in the figure, the transformed section consists of small blocks. Each block represents one frequency-wavenumber partition. From the figure, the DWPT concentrates the energy of the data in Figure 3 in the lower-right corner, which corresponds to the partition of low frequency and small wavenumber. This is not surprising, because the original data contain mainly horizontal events resulting in the strong small-wavenumber components.

After DWPT, therefore, the coefficients are concentrated in a region of the new (frequency-wavenumber) domain. But how this action of DWPT help in compressing seismic data?

Figures 5 and 6 may help give understanding of this. Figure 5 shows a histogram obtained from the stacked section shown in Figure 3. Since the data are represented

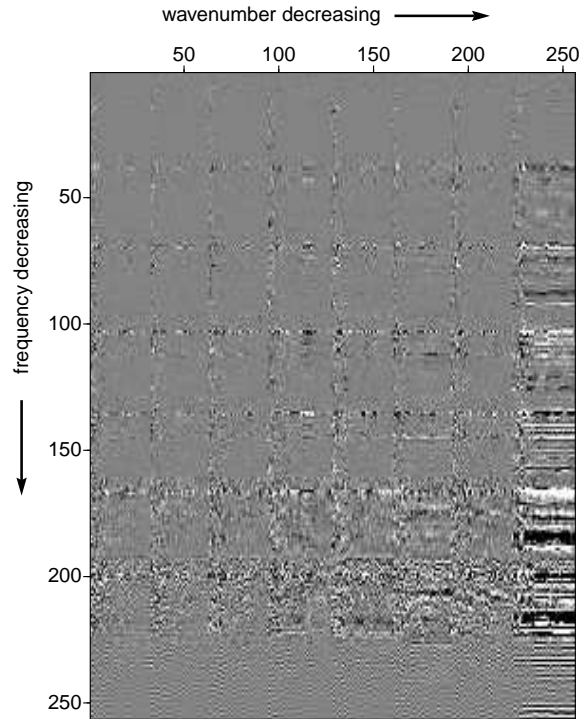


FIG. 4. The stacked section after 2D DWPT. It is obtained by three levels of decomposition along both space and time dimensions using the filters corresponding to the fourth-order coiflet. Each block represents a frequency-wavenumber partition.

as floating point numbers, they are first converted into integers based on a chosen quantization, thus incurring some error. In generating this figure, I fixed the RMS amplitude of this error to be 1% of the RMS amplitude of the signal. After the conversion, the occurrence of each integer is then counted and normalized to generate the histogram. Figure 6 shows the histogram of the section after DWPT, as shown in Figure 4. Here again, 1% relative RMS error is allowed. Since the RMS amplitude of the data remains the same before and after DWPT, the absolute errors are the same in both cases as well. However, the number of integer levels for the data before and after the transform might differ. Here, for plotting purpose, the integer levels are truncated to the range of  $-128$  to  $127$ . Clearly from the figures, the data after the transform have a much narrower distribution than do those before the transform. As I define later, the information *entropy* is lower for narrower distributions. Entropy is a quantity that determines the average bits per sample needed to represent a signal. Therefore, the lower the entropy, the fewer bits needed to represent the signal, and the more compression that can be obtained. It turns out that the entropy of the signal in Figure 3 is 6.7 bits while that of the signal in Figure 4 is 5.8 bits.

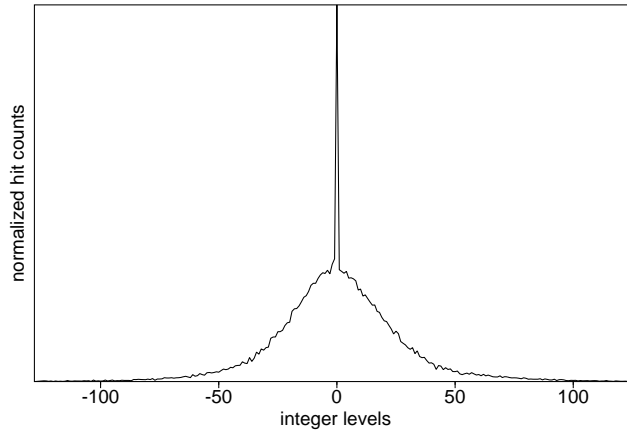


FIG. 5. Histogram for the stacked section, shown in Figure 3.

## QUANTIZATION

We will find that such reduction in the entropy of a signal is sufficient to yield cost-effective compression of seismic data. But no compression has occurred so far, because the number of coefficients is the same as the number of samples in the original signal. To achieve compression, we need to approximate the transform coefficients using fewer bits. In their approach, Luo and Schuster (1992) applied an approximation by discarding the small coefficients. In order to reconstruct the data, however, they also had to store the locations of the remaining coefficients. An alternative, more practical approach, however, is to approximate the coefficients by a set of integers; this is called quantization.

Since quantization is the only step where approximations are made in representing the signal, how one designs a *quantizer* — an algorithm performing the quantization — will determine how the error is distributed in the approximation. This, in turn, will have direct influence on how the approximated signal looks, how the waveform in the approximation differs from that in the original, and how the approximation error propagates through different processing modules. Therefore, a good quantizer is one that is tuned for a specific type of signal and processes that will be applied to it. For example, the quantizers used in speech signal compression are different from those used in image compression. Applying the quantizers (and the compression techniques using those quantizers) designed for one type of signal to another type is thus generally inappropriate.

In order to design quantizers that might be appropriate for seismic signals, it is necessary to understand the theory of quantization, a subject that is more complicated than it appears. In their book, Gersho and Gray (1992) discussed many different quantizers and therefore provided many options. However, after some transformation,

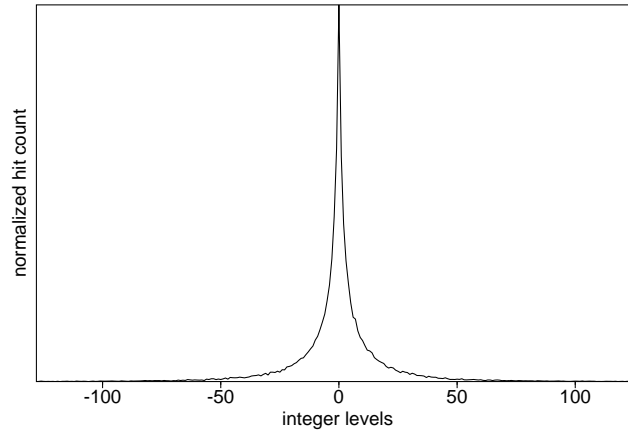


FIG. 6. Histogram for the stacked section after DWPT, shown in Figure 4.

*scalar quantizers* — where each sample is quantized independently — are often used for simplicity.

A scalar quantizer is an operator  $Q$  that maps real numbers  $x$  within a range  $(a, b)$  into a finite set of *output levels*  $y_1, y_2, \dots, y_N$ . After quantization, a real number  $x$  can be uniquely approximated by the nearest output level  $y_i$  and therefore represented by the integer  $i$ . Depending on how the  $y_i$ 's are distributed, scalar quantizers are further categorized as *uniform* — where the  $y_i$ 's are uniformly distributed, as shown in Figure 7 — and *nonuniform*, otherwise (Figure 8). The distance  $\Delta_i = |y_i - y_{i-1}|$  is

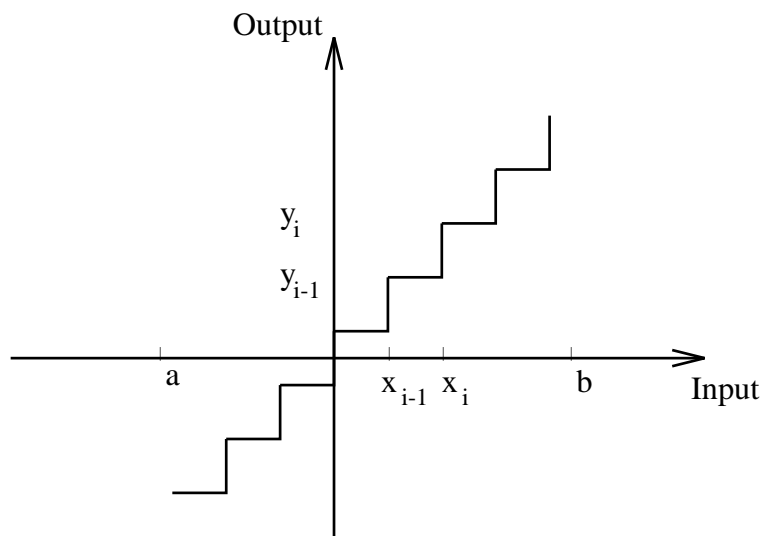


FIG. 7. A uniform quantizer where the output levels  $y_i$  are uniformly distributed. called the *stepsize*. Therefore, the *maximum error* of a quantizer is just  $\max_i \frac{\Delta_i}{2}$ . The

mean-squared-error (MSE, which is the square of RMS) between the original signal  $x$  and the approximation  $Q(x)$  is also called the  $L^2$ -average distortion (I will simply call it *average distortion* throughout the rest of the paper.), which is given by

$$D = \int |x - Q(x)|^2 f_X(x) dx, \quad (1)$$

where  $f_X(x)$  is the probability density function of  $x$ .

Given these definitions, following are some results important in designing a quantizer. The proofs for most of the observations can be found in Gersho and Gray (1992).

1. *For a given number of output levels  $N$ , the uniform quantizer minimizes the maximum error.*
2. *The average distortion of the uniform quantizer is*

$$D = \frac{\Delta^2}{12}, \quad (2)$$

*provided that  $f_X(x)$  is smooth and the stepsize  $\Delta$  (all the stepsizes are the same for the uniform quantizer) is small.*

3. *For a given number of output levels  $N$ , a nonuniform quantizer that matches the input probability density function  $f_X(x)$  minimizes the average distortion.*

The next two observations are related to entropy. The entropy of a discrete-alphabet random variable  $f$  (i.e., random variable that can take on a discrete number of values) is defined as

$$H_Q \equiv - \sum_{i=1}^N P(i) \log_2 P(i), \quad (3)$$

where  $P()$  is the *probability mass function* of  $f$ ; less rigorously,  $P(i)$  is the frequency of occurrence of the symbol  $i$ . There is a continuous-alphabet analog of  $H_Q$  called the *Shannon's differential entropy*  $h(X)$  of the random variable  $X$ .

4. *For a fixed entropy, the uniform quantizer minimizes the average distortion. Equivalently, for a fixed average distortion, the uniform quantizer achieves the minimum entropy.*
5. *The minimum entropy for a fixed average distortion is given approximately by*

$$H_Q \approx h(X) - \frac{1}{2} \log_2 12D. \quad (4)$$

From these observations, there are different **optimal** quantizers for different purposes. For example, in digital telephone communication, it is desirable to have a fixed number of output levels  $N$  (*fixed rate codes*). From Observation 3, a nonuniform quantizer is therefore required that matches the amplitude distribution of speech signals. Although the shape might be different, most of the natural (as opposite to synthetic) signals have an amplitude distribution similar to the one shown in Figure 5; i.e., there are more small-amplitude samples than large-amplitude samples. Intuitively, to have as small an average distortion as possible, the nonuniform quantizer will allocate smaller errors for the small amplitude values than for the large amplitude values (Figure 8), because there are more of them. A nonuniform quantizer such as this is what is used in the current North American standard for digital telephony (CCITT G.711, e.g. Bellamy, 1991). It happens that, besides minimizing the average

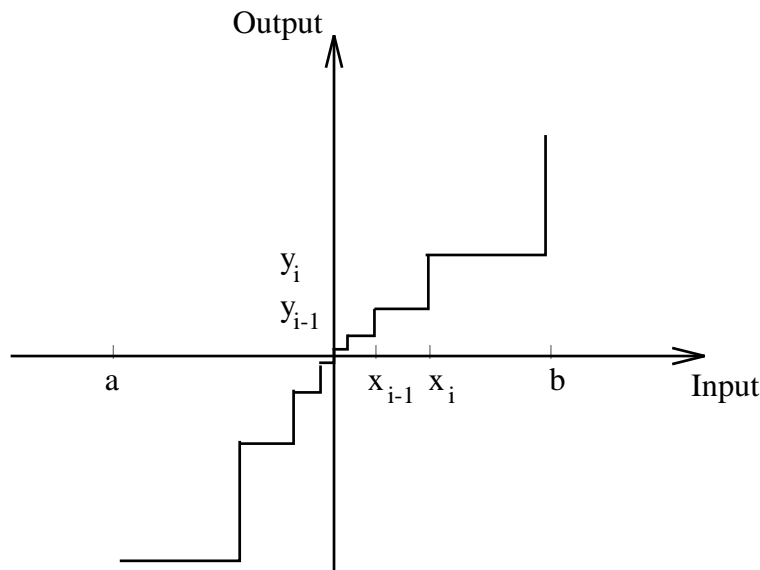


FIG. 8. Nonuniform quantizer. The output levels  $y_i$  are nonuniformly distributed so that the stepsizes for smaller input values are smaller than those for larger input values.

distortion, the nonuniform quantizer fits the purpose of telephony as well. This is because the human auditory system is not very sensitive to the volume of the sound. For a range of large-amplitude events, the content is already known, and the volume does not make too much difference (it might make some difference in expressing emotions though). In contrast, for the small-amplitude events (the whispers) only small errors can be tolerated in order that the content be understandable.

This nonuniform quantizer, however, might not be appropriate for seismic signals. When a nonuniform quantizer is used, more error is allocated to large-amplitude events, because they occur less frequently, as shown in Figure 5. However, in seismic data large-amplitude events (the stand-outs) are what we are often most interested in. Those are the events from which we estimate various earth parameters. Keeping those

events in position and their amplitudes as accurate as possible, intuitively, will help alleviate the possible exaggeration of the quantization errors in further processing. On the other hand, the small-amplitude events have a good chance of being random noise. The nonuniform quantizer therefore might expend too much effort in approximating possible random noise.

Until a better quantizer is found, the uniform quantizer might be a safe choice. From Observation 1, the uniform quantizer minimizes the maximum error for a given number of output levels  $N$ . Therefore, the uniform quantizer is robust in that good performance can be maintained for a wide variety of input signals. With the error allocated uniformly, the targeting features (large-amplitude events) are approximated accurately, while the small-amplitude events are treated with some care as well. The above reasoning remains valid for the transformed domain in a transform-based compression technique, as well as in the original data domain.

The uniform quantizer might be a safe choice for allocating the error. But will it provide enough compression for a given amount of average distortion? From Observation 4, the uniform quantizer minimizes the entropy. Therefore, if the uniform quantizer is followed by an *entropy coder*, it will provide the most compression for a given average distortion. The entropy coder will be discussed in the next section, but before going to that, let me show an example.

The approach of discarding the small coefficients discussed in some of the literature can be considered as a special form of quantizer. In this quantizer, small amplitudes are set to zero while the large amplitudes are kept intact. Therefore, large-amplitude events are treated with extreme care (with no approximation at all) while small-amplitude events are totally ignored. Though it might be difficult to argue the possible disadvantages of this error allocation approach in terms of further processing, the compression ratios can be evaluated. For the stacked section shown in Figure 3, I tried the compression technique of quantization with coding, as well as the method of discarding small coefficients. The transformations used are identical for both cases, with five levels along the time direction and four levels along the space dimension of wavelet packets decomposition using the fourth-order coiflet. Under 1% relative RMS error (RMS amplitude of the error is 1% of the RMS amplitude of the signal), the quantization (with a uniform quantizer) and coding technique gives about 5.75:1 compression. To achieve this same amount of compression, the method of discarding small coefficients would have to throw away more than 80% of the smallest coefficients and also store the indices of the remaining coefficients. This however gives an RMS error as large as 20% even though the coefficients discarded are smaller than 2% of the largest coefficient. This result gives support to Observation 4: the uniform quantizer minimizes the entropy for a given average distortion.

## CODING

As suggested in the previous section, entropy coders are needed after using a uniform quantizer, in order to achieve good compression. Entropy coding is a lossless

compression step. It attempts to compress the data so that the average number of bits per symbol is close to the entropy of a sequence of symbols, defined by equation (3). The literature contains extensive study on entropy coding, and detailed accounts may be found in many books and papers, e.g., Gersho and Gray (1992).

Example of the many forms of entropy coders include Huffman coders (Huffman, 1952), arithmetic coders (Witten et al., 1987) and dictionary-based coders (Welch, 1984). Here, I use a simple example to show how Huffman coders compress data.

$i$	$P(i)$	Natural Code	Huffman Code
0	1/2	000	0
1	1/4	001	10
2	1/8	010	110
3	1/16	011	1110
4	1/32	100	11110
5	1/64	101	111110
6	1/128	110	1111110
7	1/128	111	1111111

Suppose there is a sequence of symbols. Each symbol belongs to the set of  $\{0, 1, 2, \dots, 7\}$ . Their corresponding binary (natural) codes are shown in the above table. The binary code requires 3 bits per symbol, no matter what the distribution of the symbols in a sequence. Now suppose each symbol  $i$  has a frequency of occurrence or probability  $P(i)$  shown in the table. In Huffman coding, each symbol  $i$  is assigned a code according to its probability  $P(i)$ . The Huffman code length for symbol  $i$  approaches  $-\log_2 P(i)$ . Therefore, symbols occurring frequently will have shorter code length, as shown by their Huffman codes in the above table. Huffman code therefore requires  $1 \times \frac{1}{2} + 2 \times \frac{1}{4} + 3 \times \frac{1}{8} + 4 \times \frac{1}{16} + 5 \times \frac{1}{32} + 6 \times \frac{1}{64} + 7 \times \frac{1}{128} + 7 \times \frac{1}{128} \approx 1.98$  bits per sample on average. Therefore, for this example Huffman coding compresses the sequence by a factor of more than 3 : 2 relative to binary coding.

After entropy coding, the average number of bits per sample will approach the entropy. Therefore, the lower the entropy of the data, the fewer bits required per sample of the representation and the more compression will be achievable. From the definition [equation (3)], it is not difficult to show that the more evenly distributed is  $P(i)$ , the higher the entropy. If in the previous example, all the symbols  $i$  have the same probability  $P(i)$ , the entropy will be 3 bits, and no compression can be achieved. For the stacked section in Figure 3, we saw that DWPT helped reduce the entropy (from 6.7 to 5.8 bits). Therefore, applying DWPT in this case will result in more compression (for a given level of accuracy) than that achievable for the original data. (We can always compress data with quantization and coding, no matter whether a transform is applied or not.)

Quantization error is another factor that can change the entropy, as shown in Observation 5 in the previous section. From equation (4), it is easy to understand

the trade-off between the quantization error, or the average distortion  $D$ , and the *compression ratio*  $r$ , which is defined as the ratio of the average number of bits per sample before and after compression. For some given data, the number of bits per sample is a fixed quantity  $b$  before compression, while it can ideally be the entropy  $H_Q$  after compression. Therefore, the compression ratio

$$r \equiv \frac{b}{H_Q}.$$

Since  $H_Q$  is related to the average distortion  $D$  according to equation (4), the compression ratio is therefore a function of  $D$ , as

$$r(D) \equiv \frac{b}{H_Q} = \frac{b}{h(X) - \frac{1}{2} \log_2 12D}.$$

Generally, it is difficult to estimate  $h(X)$  and absolute  $D$  from the data. On the other hand,  $r(D)$  can be measured for some given initial value of  $D_0$  ( $D_0$  is related to the stepsize in the quantization according to equation (2).) to obtain  $r_0 = r(D_0)$ . Since

$$r_0 = r(D_0) = \frac{b}{h(X) - \frac{1}{2} \log_2 12D_0},$$

$r(D)$  can be represented using  $r_0 = r(D_0)$ , as

$$r(D) = \frac{b}{\frac{b}{r_0} - \frac{1}{2} \log_2 \frac{D}{D_0}}.$$

Defining the relative error  $e$  as the MSE (which is just the average distortion  $D$ ) divided by the mean squared amplitude of the data  $E$ ,

$$e \equiv \frac{D}{E},$$

the compression ratio can then be represented as a function of  $e$

$$r(e) = \frac{b}{\frac{b}{r_0} - \frac{1}{2} \log_2 \frac{e}{e_0}}, \quad (5)$$

where  $r_0 = r(e_0)$ .

Figure 9 shows how the compression ratio  $r(e)$  changes with the relative error  $e$  for two hypothetical data sets, one with  $r_0 = 6$  and the other with  $r_0 = 4$ , when  $e_0 = .01\%$  and  $b = 32$  assumed for both cases. It looks similar to the one shown in Reiter and Heller (1994), where they compared how the compression ratios change with the relative error for an NMO-corrected common-midpoint (CMP) gather and a stacked section. By comparison, they concluded that the error increases with compression ratio more rapidly for CMP gathers than for stacked sections. Figure 9 gives a possible explanation for this phenomenon. Compared to CMP gathers, stacked sections often

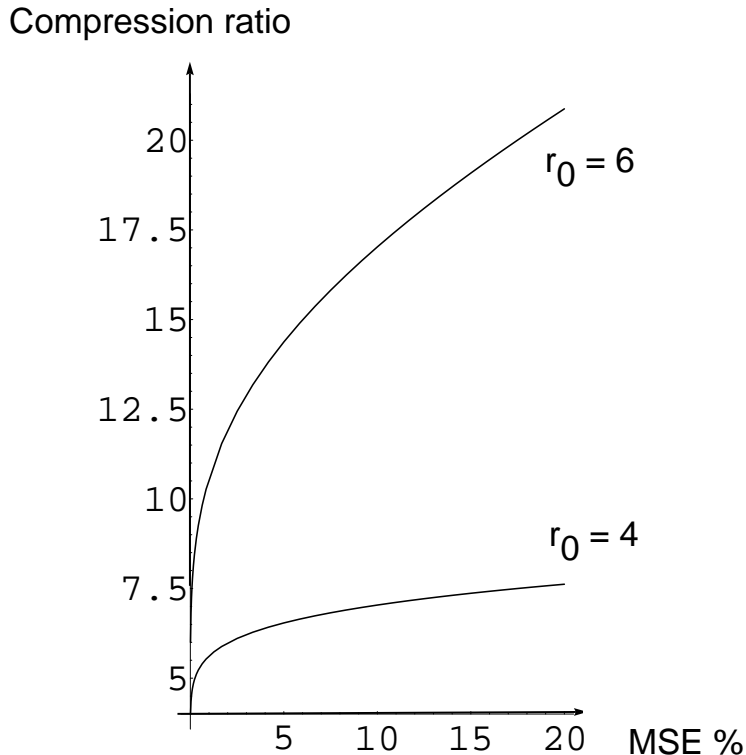


FIG. 9. Compression ratios changing with compression error.

have higher signal-to-noise ratios and therefore more coherency. After the transform, they will have less entropy. Since

$$r_0 = \frac{b}{h(X) - \frac{1}{2} \log_2(12D_0)},$$

the gathers with less entropy will have a larger compression ratio  $r_0$ . From equation (5), then, the error will increase more slowly for stacked sections, as illustrated by the example shown in Figure 9.

## CONCLUSION

Using DWPT based compression technique, I show that DWPT helps compressing seismic data because it reduces the entropy of the data. I reason that uniform quantizers might be more appropriate for seismic data, and they can achieve the same amount of compression with less error than simply throwing away the small transform coefficients. I also give a possible explanation for the phenomenon that the compression error grows more rapidly with compression ratio for CMP gathers than for stacked sections.

## ACKNOWLEDGEMENTS

Most of the work in this paper was done during my summer employment at Advance Geophysical Inc. and I thank Advance for allowing me to continue and publish this work after I came back to school. Thanks to Professor Ken Larner for his technical discussion and valuable suggestions as well as critical reviewing of the paper.

## REFERENCES

- Bellamy, J., 1991, Digital Telephony, 2nd edition, Wiley.
- Bordley, T. E., 1983, Linear predictive coding of marine seismic data: IEEE Trans. on ASSP, **31**, 828-835.
- Bosman, C. L., and Reiter, E. C., 1993, Seismic data compression using wavelet transforms: Expanded Abstracts, 63rd Annual International SEG Meeting, 1261-1264.
- Coifman, R. R., and Wickerhauser, M. V., 1992, Entropy based algorithms for best basis selection: IEEE Trans. on Information Theory, **38**, 713-719.
- Daubechies, I., 1992, Ten Lectures on Wavelets, SIAM.
- Gall, D. L., 1991, MPEG: A video compression standard for multimedia applications: Commun. ACM, **34**, 46-58.
- Gersho, A., and Gray, R. M., 1992, Vector quantization and signal compression, Kluwer Academic Publishers.
- Huffman, D. A., 1952, A method for the construction of minimum-redundancy codes: Proceedings of the IRE, **40**, 1098-1101.
- Luo, Y., and Schuster, G. T., 1992, Wave packet transform and data compression: Expanded Abstracts, 62nd Annual International SEG Meeting, 1187-1190.
- Reiter, E. C., and Heller, P. N., 1994, Wavelet transform-based compression of NMO-corrected CDP gathers: Expanded Abstracts, 64th Annual International SEG Meeting, 731-734.
- Spanias, A. S., Jonsson, S. B., and Stearns, S. D., 1991, Transform methods for seismic data compression: IEEE Trans. on Geoscience and Remote Sensing, **29**, 407-416.
- Wallace, G. K., 1991, The JPEG still-picture compression standard: Commun. ACM, **34**, 30-44.
- Welch, T. A., 1984, A Technique for high-performance data compression: Computer, June, 8-19.
- Witten, I. H., Neal, R. M., and Cleary, J. G., 1987, Arithmetic coding for data compression: Commun. ACM, **30** 520-540.
- Wood, L. C., 1974, Seismic data compression methods: Geophysics, **39**, 499-525.