Lossless Compression of Seismic Data

Caryl V. Peterson and C. Robert Hutt
USGS Albuquerque Seismic Laboratory
Kirtland AFB East
Albuquerque, NM 87115-5000

Abstract

This paper consists of a description and comparative study of three different waveform data compression schemes. All three schemes are lossless, that is, decompression results in exact recovery of the original data samples. The first technique uses linear prediction followed by residue sequence encoding. The second and third techniques use single-stage differencing followed by two different methods of encoding. The three schemes are first described. The compression ratio and CPU time required for compression and decompression are studied using an extensive set of seismic data.

1. Introduction

Data compression has become increasingly important recently in the area of seismic data. There are many different schemes for data compression but unfortunately, most of them deal with lossy data compression. Exact recovery of data is a requirement in certain fields such as medical research, geophysics, and other types of instrumentation and telemetry data communication. Two different lossless data compression schemes applied to seismic data are considered here.

With improved instrumentation, the frequency band in which seismic phenomena can be accurately measured has increased and hence the amount of data gathered has increased. Currently, sixty seismic stations report data to the Albuquerque Seismic Laboratory. Without data compression, we would be receiving more than a gigabyte of data per day assuming that each sample is represented as a 4-byte signed integer value. Since the trend is toward increasing the sampling frequency of continuously recorded data, the estimated amount of data will triple.

2. Lossless Data Compression

We are considering two types of lossless data compression in our study. The first uses linear predictive encoding followed by a bi-level encoding scheme. The second method uses a first difference followed by two different encoding schemes. Each of the methods can be described as a two stage process. Each scheme provides exact recovery of the original data.

2.1 Linear Predictor and bi-level encoding (LPBLC)

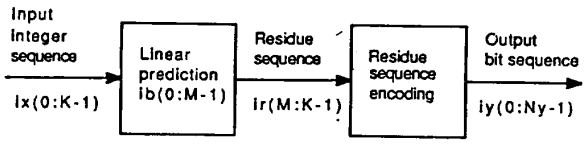

Figure 1 LPBLC data compression

A compression scheme using a linear predictor with bi-level encoding (LPBLC) is due to Stearns and Tan [1-3]. In figure 1, linear prediction is used in the traditional manner to produce a decorrelated residue sequence, *ir(M:K-1)*, from an input integer sequence. The difference from usual linear prediction is that the predictor coefficients, *ib(0:M-1)*, are integers and the prediction and recovery algorithms are designed to reverse each other exactly with no round off errors.

The input sequence, *ix*, has a fixed number (K) of samples that are assumed to be in the range from -2^{31} to $+(2^{31} - 1)$, which are represented as 4-byte signed integers. The residue sequence, *ir*, begins at M rather than 0 because the first M-1 samples are included in the compressed data frame and are used as initial values for decompression. The output from the compression algorithm, *iy*, is a variable length bit sequence from which the exact input sequence can be recovered knowing the parameters found in the keyword.

1058-6393/92 $03.00 © 1992 IEEE

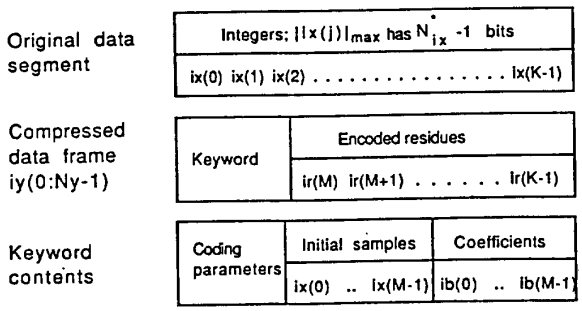

Figure 2 Data format of LPBLC

Figure 2 depicts the data format for the compression algorithm. The compressed data frame consists of a keyword and the sequence of residues, *ir*. This sequence is an integer sequence given by

$$ir(j) = ix(j) - NINT\left[\sum_{k=0}^{M-1} b(k)\, ix(j-k-1)\right];\ M \le j < K \qquad (1)$$

The residue sequence, *ir(j)*, is the error or residue when *ix(j)* is predicted using the first *M* past values. The summation is done in double precision and *NINT* means the nearest integer.

The coefficients *b(0:M-1)* are derived using a standard least-squares design routine to find optimal coefficients, $b^*(0:M-1)$, that minimize the total squared error (TSE) given by

$$TSE = \sum_{j=M}^{K-1} ir^2(j) \qquad (2)$$

The optimal coefficients $b^*(0:M-1)$ are then converted to an integer sequence by

$$ib(j) = NINT[2^{N_{ib}-I-1} b^*(j)];\ 0 \le j < M \qquad (3)$$

where N_{ib} is the number of bits in each coefficient *ib(k)* and *I* is the number of bits it takes to represent $|ib(j)|_{max}$. *I* is chosen so that *ib(0:M-1)* is an integer sequence that represents the sign and magnitude of $b^*(0:M-1)$ as accurately as possible. The coefficients *b(0:M-1)* used in the predictor are found by

$$b(j) = \frac{DBLE[ib(j)]}{2^{N_{ib}-I-1}};\ 0 \le j < M \qquad (4)$$

With the initial data sequence, *ix(0:M-1)*, the sequence *b(0:M-1)* can be reproduced from the above equation on any standard computer. Since *ix(0:M-1)* and *ib(0:M-1)* are known from the keyword, the original data can be reproduced using

$$ix(j) = ir(j) + NINT\left[\sum_{k=0}^{M-1} b(k)\, ix(j-k-1)\right];\ M \le j < K \qquad (5)$$

The second stage of the compression algorithm in figure 1 is called bi-level encoding. Short sequences of residues are encoded using two different sample sizes, N_0 and N_1. N_0 is the number of bits it takes to represent all residues (magnitude plus sign). N_1 bits can represent those residues that are smaller in magnitude. This results in two levels of encoding; level-0 samples have N_0 bits and level-1 samples have N_1 bits.

As an example of a bi-level short sequence, let N_0=5 and N_1=3. λ bits are required for designating the length, *n*, of each level-0 or level-1 sequence. A minimum sequence length, μ, is established such that

$$\mu \le length(n) < 2^{\lambda} + \mu - 1 \qquad (6)$$

The complete rules for bi-level encoding are described in [1], [3], and [5]. In our example, λ=4 and μ=3. The binary encoding of the short sequence begins with λ bits encoded as (*length-*μ), or binary 2=0010. The example is shown in Figure 3.

residues	bits	level	short sequence	encoded in binary
7	4	0		
10	5	0		
2	3	1	2	010
3	3	1	3	011
0	1	1	0	000
-2	3	1	-2	110
-1	2	1	-1	101
-4	4	0		
8	5	0		
...				

Figure 3 Example of a encoding a short sequence

Figures 4 and 5 describe the parameters found in the keyword, shown in figure 2, as well as the position of each parameter in the keyword.

Parameter	Definition
K	Number of input samples
M	Number of initial data samples included in the keyword
N_{ix}	Number of bits in *ix(k)*
I	Number of bits to represent $\|NINT[ib(j)]\|_{max}$
N_{ib}	Number of bits allocated to *ib(j)*
λ	Number of bits for designing the short sequence for bi-level encoding
N_0	Size of level-0 in bi-level encoding
N_1	Size of level-1 sequence

Figure 4 Keyword Parameter definition for LPBLC

Bit Position	No. of bits	Contents	Range
0-15	16	K	0-65535
16-20	5	M	1-31
21-25	5	N_{ix}	2-31
26-29	4	I	0-15
30-34	5	N_{ib}	3-29
35-38	4	λ	1-15
39-43	5	N_0	0-31
43-48	5	N_1	0-31
	Keyword Parameter bits = 49		

Figure 5 Keyword Parameters for LPBLC

In addition to the above parameters, the keyword, as shown in figure 2, also contains the M initial samples, *ix(0:M-1)*, and the predictor coefficients, *ib(0:M-1)*. The number of keyword bits in the LPBLC encoding scheme is given by

$$\text{Keyword bits} = 49 + M\,(N_{ix} + N_{ib}) \qquad (7)$$

The bits in the keyword are the "overhead" for LPBLC.

2.2 First differences and encoding schemes

The second data compression technique, due to J. Steim, is a first difference followed by methods of encoding called "Steim1" or "Steim2" as documented in [4]. Steim1 is the encoding scheme that USGS is currently using in most of their global seismic stations. In both of these encoding schemes, the input number of samples is a variable and the output is a fixed number of compressed bytes. Recall that in the LPBLC scheme, we have a fixed number of samples compressed into a variable length bit sequence.

First differencing is based on the fact that, in general, seismic data is dominated by low frequency components. The differences between consecutive samples tend to be quite small when compared to the full scale 32-bit signed integer representation of the data. The first differences in the Steim schemes are analogous to the residues in the LPBLC scheme of Stearns and Tan. The encoding of the differences enters the length of the differences into the keyword as will be described later.

Steim1 divides the residues into three different sizes of one byte, two bytes, and four bytes. If the difference between two samples is between -128 and +127, then one "signed" byte can be used to express that difference. Four consecutive differences are scanned to be sure that all four will fall into this range in order to express the four byte differences as a 32-bit "word". However, if any of the differences is greater than +127 but less than +32,767 or less than -128 but greater than -32768, then two bytes can be used to represent the difference. Two consecutive differences in this second range would be represented in a "word". If the difference does not fall into these ranges, then 4 bytes must be used to represent the difference.

The 4-byte (32-bit) longword differences are put into a data frame that is 64 bytes long. Normally, a data record is made of 63 data frames as shown in figure 7. If there are not enough samples to complete a data record, a shorter record can be formed with as little as one data frame.

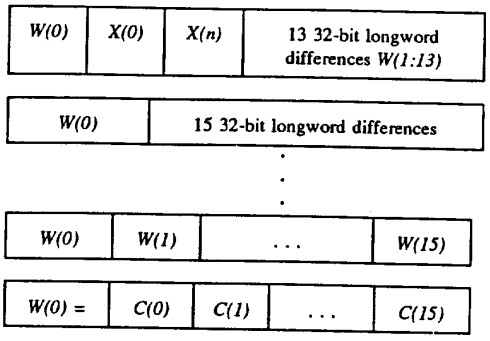

Figure 6 Compressed data format for Steim1

The first longword, *W(0)*, in each data frame consists of 16 2-bit "nibbles", *C(k)*. Each *C(0:15)* corresponds to *W(0:15)* in the data frame and contains the code to

determine what is found in each *W(k)* in the 64 byte frame. The code is determined by

$C(k) = 00_2$: Special: *W(k)* contains non-data information, such as *W(0)*
$C(k) = 01_2$: four 1-byte differences contained in *W(k)*
$C(k) = 10_2$: two 2-byte differences contained in *W(k)*
$C(k) = 11_2$: one 4-byte difference contained in *W(k)*

C(0) corresponds to *W(0)* which always contains the code bits. Therefore, *C(0)* always is 00_2.

In the first frame of a data record, $C(0) = C(1) = C(2) = 00_2$ because *W(1)* contains the first sample, *X(0)*, and *W(2)* contains the last sample, *X(n)*. *C(3)* through *C(15)* correspond to *W(3)* to *W(15)*. In subsequent frames, *W(0)* still contains the 2-bit code *C(0:15)*, but *W(1:15)* contain sample differences. The first sample, *X(0)* called the "forward integration constant", and the last sample, *X(n)* called the "reverse integration constant", are in *W(1)* and *W(2)* of the first data frame in a data record. The reverse integration constant is included in the data frame to ensure data integrity. Also, if the data is corrupted, knowing the last sample, one can recompute the differences and correct a simple mistake in the data. The number of samples contained in a full data record is

$$943 \le \text{number of samples} \le 3,772 \quad (8)$$

The actual number of samples is known globally and is considered as part of the overhead in our analysis of data compression ratio. The overhead for Steim1 is

$$32\text{bits} * 63\text{frames} + 64\text{bits} + 16\text{bits} = 2096\text{bits} \quad (9)$$

The 16 bits are assuming an integer*2 representation for the number of samples and the 64 bits are for the forward and reverse integration constants. The total number of bits per data record are

$$\frac{63 \text{ data frames}}{\text{data record}} * \frac{64 \text{ bytes}}{\text{data frame}} * \frac{8 \text{ bits}}{\text{byte}} = 32256 \quad (10)$$

Of the 32256 bits in the data record, 30160 are the actual differences.

Steim2 allows for a greater variety in the number of bits used to represent the first difference. Figure 7 shows the number of differences that can be packed into a 32-bit longword and the range that the difference covers.

Number of differences	Number of bits	Range
7	4	$-8 \le d(k) \le +7$
6	5	$-16 \le d(k) < -16$ or $+7 < d(k) \le +15$
5	6	$-32 \le d(k) < -16$ or $+15 < d(k) \le 31$
4	8	$-128 \le d(k) < -32$ or $+31 < d(k) \le +127$
3	10	$-512 \le d(k) < -128$ or $+127 < d(k) \le +511$
2	15	$-16384 \le d(k) < -512$ or $+511 < d(k) \le +16383$
1	30	$-2^{29} \le d(k) < -2^{14}$ or $+2^{14}-1 < d(k) \le +2^{29}-1$

Figure 7 Steim2 differences

There are at least 2 extra bits in the 32-bit longword in each case with the exception of those differences represented by 4 8-bit differences. The first two bits in the longword are used in conjunction with the *C(k)* 2-bit nibbles in the keyword to tell what number of bit differences are found in each longword as follows

$C(k) = 01_2$: four 1-byte differences contained in *W(k)*
$C(k) = 10_2$: look at first two bits in *W(k)*, denoted as *C1*
$C1 = 01_2$: one 30-bit difference in *W(k)*
$C1 = 10_2$: two 15-bit differences in *W(k)*
$C1 = 11_2$: three 10-bit differences in *W(k)*
$C(k) = 11_2$: look in *C1*
$C1 = 00_2$: five 6-bit differences in *W(k)*
$C1 = 01_2$: six 5-bit differences in *W(k)*
$C1 = 10_2$: seven 4-bit differences in *W(k)*

The differences are put into the data frames and data records in the same way as they are in Steim1.

The overhead for Steim2 is only slightly more than for Steim1 because two bits out of every 32 are designated for further decoding of the differences.

The number of samples contained in a full data record for Steim2 is

$$943 \le \text{number of samples} \le 6,601 \quad (11)$$

3. Compression ratio and CPU time

The compression ratio is calculated by

$$\frac{\text{Number of Samples} * 32\ bits}{\text{number of bits in compressed data}} \qquad (12)$$

Each sample is assumed to be represented as a longinteger, which is 32 bits. The denominator in the above equation included any overhead that the algorithm needs to compress and decompress the data.

For Steim1, the maximum compression ratio is 3.74 when all differences can be expressed as 8-bit differences. For Steim2, the best compression ratio is 6.74 when all differences are expressed as 4-bit differences.

Figure 8 summarizes the compression ratio of each of the three compression schemes on different types of seismic data.

Compression Scheme	80 Hz	20 Hz Noise	20 Hz low noise	1 Hz large event	1 Hz noise
LPBLC	6.47	5.39	7.34	2.02	3.09
Steim1	3.77	3.63	3.67	1.57	1.94
Steim2	5.03	4.65	6.07	1.36	2.78

Figure 8 Compression Ratio

The compression ratio is better in every case of seismic data using LPBLC. The largest difference in compression ratio was between Steim1 and LPBLC for 20 Hz low noise seismic data where LPBLC was 2 times better than Steim1. The smallest difference in compression ratio was between Steim2 and LPBLC using 1 Hz noise data where LPBLC was 1.11 times better.

The compression and decompression CPU time (in seconds) was computed using 50,000 data samples whose frequency was 80 Hz. The computer was a SPARC2 40 MHz machine. SPARC2 is a RISC machine that has 25 MIPS (million instructions per second). Those results are in Figure 9.

Compression Scheme	Compression Time	Decompression Time
LPBLC	21.03	3.28
Steim1	0.4	0.24
Steim2	0.4	0.24

Figure 9 Compression, Decompression Time in seconds

The time to compress the data is faster using either Steim1 or Steim2 by a factor of 50. For decompression time, Steim1 and Steim2 are 13 times faster.

4. Conclusions

Since all three data compression schemes provide lossless data compression, any of the schemes can be used for geophysics. The version of LPBLC used for this report was coded in FORTRAN without any regard for computing speed. As the compression schemes are currently programmed, Steim2 is the optimum compression scheme for seismic stations. Since the compression ratio is better with LPBLC, it is worthwhile to consider optimizing the LPBLC code for speed in addition to compression ratio. One possibility to improve the speed would be to use integer arithmetic in determining the coefficients, *b(0:M-1)*, used in the predictor as suggested by L. Tan [5].

5. References

1. S. D. Stearns, "Predictive data compression with exact recovery", Sandia Nat. Labs., Albuquerque, NM, Sandia Rep., SAND 90-2583. UC-403, December, 1990.

2. S.D. Stearns, L. Tan, N. Magotra, "A Technique for Lossless Compression of Seismic Data", *1992 International Geoscience and Remote Sensing Symp.*, Houston, TX, May 26-29, 1992.

3. S. D. Stearns, L. Tan, N. Magotra, "A bi-level Coding Technique for Compressing Broadband Residue Sequences", *Digital Signal Processing, A Review Journal*, July 1992, pp. 146-156.

4. Federation of Digital Seismographic Networks et.al., *Standard for the Exchange of Earthquake Data Reference Manual*, March 1990, pp.103-107.

5. L. Tan, *Theory and Techniques for Lossless Waveform Data Compression*, Ph.D dissertation Dept. of Electrical Engineering, University of New Mexico, Albuquerque, NM 87131, May 1992.