Lossless Data Compression in Real Time

F. Livingston†, *N. Magotra*†, *W. McCoy*†, *S. Stearns*‡

† Dept. of EECE, University of New Mexico, Albuquerque, NM 87131
‡ Dept. 9311, Sandia National Laboratory, Albuquerque, NM 87185

Ph.:505-277-0808/email:magotra@houdini.eece.unm.edu

Abstract

This paper describes the research effort currently in progress to develop lossless data compression algorithms for seismic, speech, and image data sets. For many applications, such as transmitting and archiving research data bases, using lossy compression algorithms is not advisable. In situations where critical data (e.g. research instrumentation) is to be transmitted or archived, a real time lossless data compression algorithm is desirable.

1. INTRODUCTION

We have developed a lossless data compression algorithm consisting of two stages, a lossless prediction stage followed by an encoding stage. A block diagram of the algorithm is presented in Figure 1. We have shown that various predictor structures can be used for lossless data compression with only a few critical restrictions. The predictor structures may be linear or nonlinear, fixed or adaptive, and can be implemented using floating point arithmetic [1]. These algorithms are incrementally processed as opposed to block processed. They have been tested on various seismic data sets and testing using speech and image data sets is presently underway.

This paper presents a version of the algorithm using a recursive least squares a priori adaptive lattice structure followed by an arithmetic coding stage. The real time effectiveness of this algorithm is being verified by coding the technique to run on a TMS320C3x card custom developed for our applications.

2. PREDICTION STAGE

The following discussion describes the lossless prediction stage in detail. Figure 2 shows a block diagram of a predictor which can be implemented by the following integer equation,

$$e_n = x_n - H(\mathbf{X}_n^-, \theta_n^-) \tag{1}$$

where x_n is the current integer sample value and $H(\mathbf{X}_n^-, \theta_n^-)$ is the predicted value. $\mathbf{X}_n^-$ is a sequence of vectors representing the past samples of the signal source and possibly past values of the error sequence. θ_n^- is the sequence of weight vectors and e_n is the difference between the current sample value and the predicted value. The receiver processor recovers the current sample using the following equation.

$$x_n = e_n + H(\mathbf{X}_n^-, \theta_n^-) \tag{2}$$

In order to implement Equation (2), the receiver must have access to e_n and must also be able to reconstruct $H(\bullet)$. e_n will be decoded from the transmitted bit stream at the receiver. The receiver processor will be able to reconstruct $H(\bullet)$ under two conditions. First, the receiver processor must have access to $\mathbf{X}_n^-$ and θ_n^-. Therefore, a keyword may need to be transmitted prior to decompression which contains information such as a signal model, or the number of transmitted bits per symbol. Second, it must also be able to generate, bit for bit, the exact same $H(\bullet)$ as the compression processor. As mentioned before, the predictor stage can have a variety of configurations; we have had the best results with the recursive least squares lattice (RLSL) predictor.

The lattice filter structure shown in Figure 3 combines the tasks of forward and backward prediction into one structure and consists of identical stages connected in cascade. The principle on which the lattice prediction is based is the Levinson-Durbin recursion which allows higher order lattice coefficients to be computed from lower order coefficients. There are different algorithms which can be applied to the lattice structure just as there are different algorithms which can be applied to the transversal filter structure. The algorithm used here is the a priori RLSL as described by Haykin [2]. The key equations are summarized below.

RLSL structures minimize the energy functions given by Equations (3) and (4) at each instant, n.

$$F_m(n) = \sum_{i=0}^{n} \lambda^{n-i} \eta_m^2(i) \tag{3}$$

$$B_m(n) = \sum_{i=0}^{n} \lambda^{n-i} \psi_m^2(i) \tag{4}$$

$\eta_m(n)$ and $\psi_m(n)$ are the forward and backward prediction errors respectively at time n and of order m. The order update recursions are given by

$$\Delta_{m-1}(n) = \lambda \Delta_{m-1}(n-1) + \gamma_{m-1}(n-1)\psi_{m-1}(n-1)\eta_{m-1}(n) \tag{5}$$

where $\Delta_{m-1}(n)$ is a lattice coefficient for stage $m-1$ which has not been normalized, $\gamma_{m-1}(n)$ is as defined below in Equation (12), and λ is the recursive least squares "forgetting factor". The $\Delta_{m-1}(n)$ coefficient is the same for both forward and backward prediction coefficients of the same order.

$$\Gamma_{f,m}(n) = -\frac{\Delta_{m-1}(n)}{B_{m-1}(n-1)} \tag{6}$$

$$\Gamma_{b,m}(n) = -\frac{\Delta_{m-1}(n)}{F_{m-1}(n)} \tag{7}$$

$\Gamma_{f,m}(n)$ and $\Gamma_{b,m}(n)$ are the normalized forward and backward lattice coefficients. The following two equations are the basic equations that actually implement the lattice structure.

$$\eta_m(n) = \eta_{m-1}(n) + \Gamma_{f,m}(n-1)\psi_{m-1}(n-1) \tag{8}$$

$$\psi_m(n) = \psi_{m-1}(n-1) + \Gamma_{b,m}(n-1)\eta_{m-1}(n) \tag{9}$$

The following two equations are updated energy functions for the forward and backward predictors.

$$F_{m-1}(n) = \lambda F_{m-1}(n-1) + \gamma_{m-1}(n-1)\eta_{m-1}^2(n) \tag{10}$$

$$B_{m-1}(n) = \lambda B_{m-1}(n-1) + \gamma_{m-1}(n-1)\psi_{m-1}^2(n) \tag{11}$$

$$\gamma_m(n) = \gamma_{m-1}(n) - \frac{\gamma_{m-1}^2(n)\psi_{m-1}^2(n)}{B_{m-1}(n)} \tag{12}$$

$\gamma_m(n)$ is a conversion factor which allows the lattice filter to quickly adapt to sudden changes in the input data.

As described in [1], the final stage lattice output is computed as a single summation equation to prevent excessive roundoff errors that might prevent the scheme from working losslessly. As long as the same lattice equations are implemented at the time of uncompressing the data (on a computer of similar architecture as the one used for compression) the scheme is lossless.

Columns 2 and 3 of Table 1 present the input and output variances of the RLSL prediction stage for several test seismic data files. As can be seen, the variance of the RLSL output is in each case lower than that of the corresponding RLSL input.

3. ARITHMETIC CODING

The residue (error) sequence generated by the prediction stage is encoded in the second stage. We use an arithmetic coding algorithm as described below. Arithmetic coding is able to incrementally compress data at near optimal rates and is also able to adapt to changing statistics [3]. The arithmetic coding algorithm includes two independent parts, an adaptive probability model, and an incremental transmission implementation.

The probability model consists of an array of integer frequency counts which is adapted with the arrival of each new symbol. The higher the frequency of a particular symbol, the larger its bin in the probability model. The total frequency count in the array must not exceed a predetermined value in order to prevent overflow. Each frequency count must be at least one.

The basic principal behind arithmetic coding is to map a sequence of integers from a random source onto a real number with precision related to the sequence length. Upon the arrival of each new symbol a new range is computed which is a function of the current range and the limits of the bin in the probability model associated with that symbol. Key equations [3] for encoding are given below.

$$h(n) = l(n-1) + bin\_hi(n) * pcn \tag{13}$$

$$l(n) = l(n-1) + bin\_low(n) * pcn \tag{14}$$

where h, l are the high and low of the current range respectively, $bin\_hi(n)$, $bin\_low(n)$ are the current limits of the probability model corresponding to the current symbol, and pcn is a number inversely related to the precision of the current range limits.

After computing $h(n)$ and $l(n)$ the arithmetic coder transmits all common leading bits between $h(n)$ and $l(n)$ and recomputes the range as follows [3].

```
if (l ≥ Half)
    Transmit 1
    h = (h – Half)*2
    l = (l – Half)*2
    pcn = pcn*2
if (h < Half)
    Transmit 0
    h = h*2
    l = l*2
```

$pcn = pcn*2$

In this pseudo-code, *Half* represents the midpoint of the range of long integer values on the computer architecture. Conditions of underflow must be avoided and are explained in detail in [3].

The coding scheme described above has been shown to give near optimal compression for white gaussian residue sequences with dynamic ranges less than 14 bits [3]. At ranges larger than 14 bits, overflow can occur, and the symbol set becomes large and cumbersome. Improvements have been made recently in the arithmetic coding algorithm which solve this problem [4]. This paper presents results incorporating these improvements. The results should give an idea of the kind of compression ratios to be expected from this algorithm.

Columns 4, 5, and 6 of Table 1 give compression ratios for the test seismic data files considered in Section 2. CR1 is computed as the size of a given input file in bytes over the compressed output file size in bytes assuming the input data is represented in 16 bits. CR2 is computed in the same fashion but assuming the input data is represented in the minimum number of bits needed to represent its maximum absolute value. CR3 is computed like CR1 and CR2 but assuming the input data is represented in the minimum number of bytes needed to represent its maximum absolute value (i.e. no fractional bytes as in the case of CR2). These compression ratios were obtained using an off-line implementation of the compression algorithm.

4. REAL TIME IMPLEMENTATION

A real time version of the compression algorithm is currently being developed for use with the Texas Instruments TMS320C30 floating-point digital signal processor. This real time implementation is being developed according to the general DSP design procedure shown in Figure 4. As can be seen, this process involves not only formulating an algorithm and implementing it using DSP hardware, but also involves simulating the algorithm using a mainframe computer. This is done so that the algorithm can be tested and verified on field acquired data.

The real time version of the algorithm is being written specifically for seismic data compression. Although this does not change the algorithm as thus far presented, it does require some method of segmenting seismic inputs into event and non-event portions. With this in mind, a real time seismic event detection algorithm was implemented [5]. Using this algorithm it is possible to discriminate event data from non-event data in test seismic inputs and hence allow the compression algorithm to compress only the data associated with the seismic events.

At present, the RLSL prediction stage has been implemented on the 'C30 and the arithmetic coding stage of the algorithm is being implemented. The results obtained from the real time and off-line versions of the RLSL are identical and it is expected that the same will be true for the complete compression algorithm once its real time implementation is finished.

5. RESULTS AND CONCLUSIONS

Adaptive algorithms have been demonstrated in application to the lossless prediction problem. The RLSL algorithm has been demonstrated to perform the best over a seismic data base. Compression results have been given for the RLSL algorithm followed by an arithmetic coding algorithm. At present various other processing techniques are being investigated for the first stage and the entire algorithm is undergoing trials using speech and image (Synthetic Aperture Radar (SAR)) data. A real time version of the algorithm for seismic event data compression is currently being implemented with only the arithmetic coding portion of the algorithm yet to be implemented.

References

[1] McCoy, J.W., Magotra, N., and Stearns, S., "Lossless Predictive Coding", IEEE Midwest Symposium on Circuits and Systems, Lafayette, LA, August 1994.

[2] Haykin, S. Adaptive Filter Theory, Prentice Hall, 2^{nd} edition, 1991.

[3] Witten, Neal, and Cleary, "Arithmetic Coding for Data Compression", Communications of the ACM, pp. 520-540, June 1987.

[4] Stearns, S.D., "Arithmetic Coding in Lossless Waveform Data Compression", Submitted for publication, August 1994.

[5] Livingston F., Magotra, N., and McCoy, J.W. "Real Time Seismic Event Detection and Lossless Waveform Compression Using the TMS320C30", Fourth Annual TMS320 Educators Conference, Dallas, TX, October 1994.

[6] TMS320C3x User's Guide, Texas Instruments Inc., 1992.

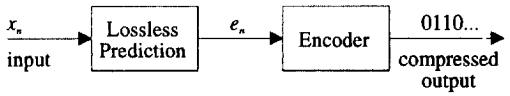

Figure 1. Algorithm Block Diagram

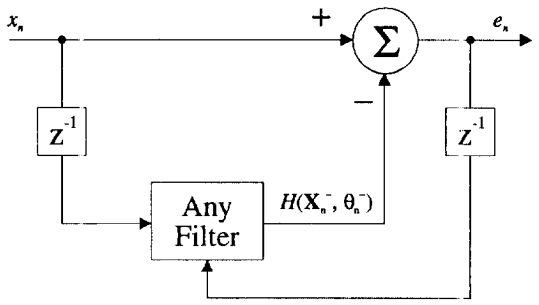

Figure 2. Block Diagram of a Lossless Predictor

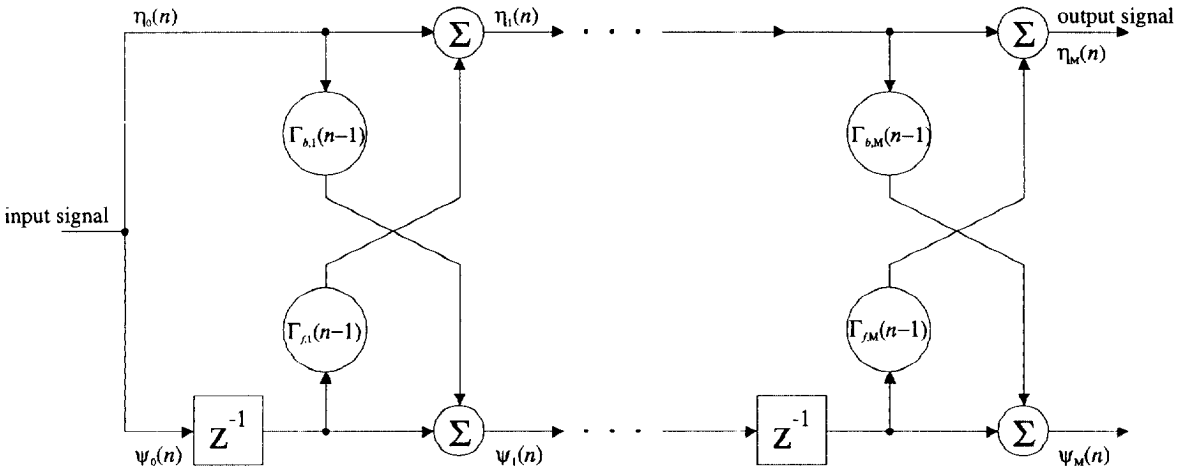

Figure 3. M-Stage Lattice Predictor

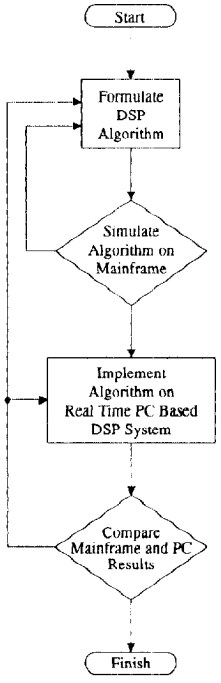

Figure 4. Design Cycle for Real Time Implementation

Event ID	i/p variance	o/p variance	CR1	CR2	CR3
anmbhz89	1.87e+09	2.07e+01	7.17	4.48	7.17
anmbhz92	3.18e+02	9.24e+00	7.88	2.96	3.94
anmehz92	7.84e+04	1.82e+04	5.25	2.13	2.63
anmlhz92	4.44e+05	4.54e+04	3.07	1.34	1.54
kipehz13	2.45e+03	5.67e+00	8.51	2.39	4.26
kipehz20	1.52e+04	4.45e+01	6.08	1.9	3.04
rarbhz92	1.02e+07	2.71e+02	4.96	2.33	2.48
rarlhz92	1.76e+08	2.34e+07	2.24	1.19	2.24

Table 1. RLSL Based Result Summary