Lossless Seismic Data Compression using Adaptive Linear Prediction

Giridhar Mandyam and Neeraj Magotra

Department of Electrical and Computer Engineering
The University of New Mexico
Albuquerque, NM 87131

Wes McCoy

Motorola Incorporated
5401 North Beach Street
Fort Worth, TX 76137

ABSTRACT

This paper presents a comparison of adaptive linear predictors as applied to the area of lossless compression of seismic waveform data. Three methods are explored: the normalize least-mean square (NLMS) algorithm, the gradient adaptive lattice (GAL) algorithm, and the recursive least squares lattice (RLSL) algorithm. When compared to standard linear prediction techniques, all three of these methods require little overhead, are more computationally efficient, and can be implemented using floating point techniques. With respect to a standard seismic database, the RLSL filter outperforms the other two methods in nearly all cases tested.

1. INTRODUCTION

Lossless compression of data is extremely important in cases where immense amounts of data are present and the integrity of this data must be preserved. One finds such an instance in the case of seismic data, since not only large amounts of seismic data are collected by many different organizations, but this data must fully retain its information for accurate analysis. For instance, vast quantities of seismic data are collected and processed for monitoring purposes related to the Comprehensive Test Ban Treaty.

Usually, lossless compression is achieved in two stages: the first stage is a *decorrelator* which exploits redundancy between neighboring samples in a data sequence, and the second stage is an *entropy coder* which takes advantage of the decreased variance of the data. In this paper, we will be concerned with the first stage. Normally, *linear prediction* is used for decorrelation. Given a sample sequence of length K, $ix(n)$ $(0 \leq n \leq K-1)$, one can design a linear predictor of order M by using M predictor coefficients $\{b_i\}$ to find an estimate $\hat{ix}(n)$ for each sample in $ix(n)$:

$$\hat{ix}(n) = \sum_{i=0}^{M-1} b_i ix(n-i-1) \qquad (1)$$

Obviously, M should be much less than K to achieve compression, because $\{b_i\}$ must be included with the compressed data. The estimate $\hat{ix}(n)$ is not the same as the original value; therefore a residue sequence is formed to allow exact recovery of $ix(n)$:

$$ir(n) = ix(n) - \hat{ix}(n) \qquad (2)$$

If the predictor coefficients are chosen properly, the entropy of $ir(n)$ should be less than the entropy of $ix(n)$. Choosing the coefficients $\{b_i\}$ involves solving the *Wiener-Hopf* equations:

$$\begin{array}{lcl} R_{0,0}b_0 + R_{0,1}b_1 + \cdots + R_{0,M-1}b_{M-1} & = & R_{M,0} \\ R_{1,0}b_0 + R_{1,1}b_1 + \cdots + R_{1,M-1}b_{M-1} & = & R_{M,1} \\ & \vdots & \\ R_{M-1,0}b_0 + R_{M-1,1}b_1 + \cdots + R_{M-1,M-1}b_{M-1} & = & R_{M,M-1} \end{array} \qquad (3)$$

where $R_{i,j}$ is the average over n of the product $ix(n)ix(n+(i-j))$. This can be represented as the matrix-vector product

$$\mathbf{Rb} = \mathbf{p} \qquad (4)$$

where $\mathbf{R}$ is the M by M matrix defined in (3), $\mathbf{b}$ is the M by 1 vector of predictor coefficients, and $\mathbf{p}$ is the M by 1 vector in (3).

While the method of linear prediction is effective, it suffers from the problem of finding a solution to the Wiener-Hopf equations in (3), which becomes increasingly computationally expensive with larger data block sizes. Adaptive FIR filters have been proposed and used successfully as a way of solving this problem [2]. With a fixed filter size, M, there is a variety of stochastic gradient methods to adapt the filter.

2. ADAPTIVE FILTERS

Adaptive filters encompass many classic stochastic gradient techniques, such as the LMS or RLS algorithms. One common method is discussed here, the normalized least mean square algorithm (NLMS) [3]. With the sample sequence index denoted by n, the set of predictor coefficients $\{b_i\}$ in (1) is now time-varying, and is represented by the

0-7803-3068-4/96$5.00©1996 IEEE

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 27,2022 at 02:12:51 UTC from IEEE Xplore. Restrictions apply.

column vector $\mathbf{b}(n) = [b_0(n) \cdots b_{M-1}(n)]^{\mathbf{T}}$, where $[]^{\mathbf{T}}$ is the transpose operator. If the input to the filter is the vector $\mathbf{ix}(n) = [ix(n-1) \cdots ix(n-M)]^{\mathbf{T}}$, then a time-varying residue [4] is given by

$$ir(n) = ix(n) - NINT\{\mathbf{b}^{\mathbf{T}}(n)\mathbf{ix}(n)\} \quad (5)$$

If two fixed parameters, a smoothing parameter β and a convergence parameter u, are specified, then $\mathbf{b}(n)$ is computed iteratively as follows:

$$\begin{aligned}\mathbf{b}(n+1) &= \mathbf{b}(n) + \mu(n) ir(n)\mathbf{ix}(n) && (6)\\ \mu(n) &= \frac{u}{\sigma_M^2(n)} && (7)\\ \sigma_M^2(n) &= \beta\sigma_M^2(n-1) + (1-\beta)(ir^2(n-1)) && (8)\end{aligned}$$

In order to reverse the algorithm without loss [4], the following equation may be used along with (6)-(8):

$$ix(n) = ir(n) + NINT\{\mathbf{b}^{\mathbf{T}}(n)\mathbf{ix}(n)\} \quad (9)$$

Therefore, one needs the initial predictor coefficients $\mathbf{b}(0)$, the initial data vector $\mathbf{ix}(0)$, and the error sequence $ir(n)$ to reconstruct the original $ix(n)$ sequence exactly. Moreover, in this approach, the coefficients $\mathbf{b}(n)$ and the data vectors $\mathbf{ix}(n)$ do not have to be transmitted at all after start-up. We note that the functions used in encoding must be repeated exactly in decoding. Therefore all quantities used in (6)-(8), that is, u, β, $\mathbf{b}(0)$, and $\mathbf{ix}(0)$, must be stored in the compressed data frame exactly as they are used in the encoding process, so that they may be used in the same way in the decoding process. We also note that here it is not necessary to break the data up into frames as was the case with the linear predictor method described previously, and that in general this method requires less overhead than the linear predictor method.

3. LATTICE FILTERS

Although the NLMS adaptive filter is effective, it sometimes displays slow convergence, resulting in a residue sequence of high-variance. This has lead to the use of a class of adaptive filters known as *lattice filters*. Although the implementation of lattice filters is more involved than that of the NLMS algorithm, faster convergence usually makes up for the increased complexity. A simple M-stage adaptive lattice filter is shown in Figure 1; this filter is known as the *gradient adaptive symmetric lattice* (GAL) filter. The update equations for this filter are [4]

$$\begin{aligned}b_i(n) &= b_{i-1}(n-1) + k_i(n)f_{i-1}(n) && (10)\\ f_i(n) &= f_{i-1}(n) + k_i(n)b_{i-1}(n-1) && (11)\\ k_i(n) &= k_i(n-1) + && (12)\\ & \alpha\frac{f_i(n-1)b_{i-1}(n-2) + f_{i-1}(n-1)b_i(n-1)}{\sigma_{i-1}^2(n-1)}\\ \sigma_{i-1}^2(n-1) &= \beta\sigma_{i-1}^2(n-2) + && (13)\\ & (1-\beta)(f_{i-1}^2(n-1) + b_{i-1}^2(n-2))\\ f_M(n) &= ix(n) + && (14)\\ & NINT\{\sum_{i=0}^{M-1} b_i(n-1)k_i(n-1)\}\end{aligned}$$

where α is a convergence parameter, β is a smoothing parameter, $f_i(n)$ is the *forward prediction error*, and $b_i(n)$ is the *backward prediction error*. The recovery equation is [4]

$$ix(n) = f_M(n) - NINT\{\sum_{i=0}^{M-1} b_i(n-1)k_i(n-1)\} \quad (15)$$

An improvement on the GAL is the recursive least-squares lattice (RLSL) filter [3]. The Mth-order forward and backward prediction error coefficients, $\eta_M(n)$ and $\psi_M(n)$ respectively, are given by:

$$\begin{aligned}\eta_M(n) &= \eta_{M-1}(n) + \Gamma_{f,M}(n-1)\psi_{M-1}(n-1) && (16)\\ \psi_M(n) &= \psi_{M-1}(n-1) + \Gamma_{b,M}(n-1)\eta_{M-1}(n) && (17)\\ \Gamma_{f,M}(n) &= -\frac{\Delta_{M-1}(n)}{B_{M-1}(n-1)} && (18)\\ \Gamma_{b,M}(n) &= -\frac{\Delta_{M-1}(n)}{F_{M-1}(n)} && (19)\\ \Delta_{M-1}(n) &= \lambda\Delta_{M-1}(n-1) + && (20)\\ & \gamma_{M-1}(n-1)\psi_{M-1}(n-1)\eta_{M-1}(n)\\ F_{M-1}(n) &= \lambda F_{M-1}(n-1) + && (21)\\ & \gamma_{M-1}(n-1)\eta_{M-1}^2(n)\\ B_{M-1}(n) &= \lambda B_{M-1}(n-1) + \gamma_{M-1}(n)\psi_{M-1}^2(n) && (22)\\ \gamma_M(n) &= \gamma_{M-1}(n) - \frac{\gamma_{M-1}^2(n)\psi_{M-1}^2(n)}{B_{M-1}(n)} && (23)\end{aligned}$$

The parameter λ is a fixed constant arbitrarily close to, but not equaling 1. The error residuals are computed as [4]

$$ir(n) = ix(n) + NINT\{\sum_{i=1}^{M}\Gamma_{f,i}(n-1)\psi_{i-1}(n-1)\} \quad (24)$$

4. SIMULATIONS

The three methods described above were compared with respect to several seismic data files quantized to 32 bits per sample) which were provided by the United States Geological Survey (USGS). The stations where this data was collected and the corresponding sampling rates are given in Table 1.

The RLSL performed better than the GAL or the NLMS algorithms. The seismic waveforms given in were each subjected to all three methods; the residue variances are given in Table 2. As can be seen, the RLSL outperformed the NLMS and GAL algorithms in nearly every case, with the residue variances reducing to lower values. The residue files associated with the RLSL filter were then subjected to the modified arithmetic coding method given in [5], with the resulting compression ratios (denoted by CR) given in Table 3. The results are promising, with the minimum compression ratio still exceeding two to one compression.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 27,2022 at 02:12:51 UTC from IEEE Xplore. Restrictions apply.

File	Location	Latitude	Longitude	Sampling Rate
anmbhz89	Albuquerque, NM	34.95	-106.5	20sps
anmbhz92	Albuquerque, NM	34.95	-106.5	20sps
anmehz92	Albuquerque, NM	34.95	-106.5	100sps
anmlhz92	Albuquerque, NM	34.95	-106.5	1sps
kipehz13	Kipapa, Hawaii	21.42	-158	100sps
kipehz20	Kipapa, Hawaii	21.42	-158	100sps
rarbhz92	Rarotonga, Cook Islands	-21.21	-159.8	20 sps
rarlhz92	Rarotonga, Cook Islands	-21.21	-159.8	1 sps

Table 1: Seismic Data Base

File	Input	NLMS	GAL	RLSL
anmbhz89	$1.87e^9$	$3.85e^4$	$6.81e^1$	$2.07e^1$
anmbhz92	$3.18e^2$	$2.26e^1$	$1.06e^1$	$9.24e^0$
anmehz92	$7.84e^4$	$2.17e^4$	$2.28e^4$	$1.82e^4$
anmlhz92	$4.44e^5$	$9.40e^4$	$7.61e^4$	$4.54e^4$
kipehz13	$2.45e^3$	$6.74e^0$	$8.22e^0$	$5.67e^0$
kipehz20	$1.52e^4$	$9.56e^1$	$7.17e^1$	$4.45e^1$
rarbhz92	$1.02e^7$	$2.05e^3$	$4.36e^2$	$2.71e^2$
rarlhz92	$1.76e^8$	$1.42e^7$	$2.21e^7$	$2.34e^7$

Table 2: Residue Variances

File	Input Size (bytes)	Output Size (bytes)	CR
anmbhz89	268000	40945	7.82
anmbhz92	288000	41305	8.72
anmehz92	56000	14995	5.24
anmlhz92	344000	122919	3.37
kipehz13	248000	30883	9.17
kipehz20	244000	40789	6.57
rarbhz92	288000	62025	5.20
rarlhz92	344000	153561	2.46

Table 3: Compression Results

5. CONCLUSIONS

Three adaptive linear prediction techniques have been applied to the area of lossless seismic waveform data compression. The RLSL filter in general outperformed the other two methods. This is encouraging, since the RLSL has practically no coefficient transmission overhead, and the data does not require blocking. Further analysis should be done into the statistical nature of the residuals for each respective adaptive filter, as suitable entropy coders can then be designed to achieved maximal compression.

6. REFERENCES

[1] Stearns, S.D., Tan, L.-Z., and Magotra, N. "Lossless Compression of Waveform Data for Efficient Storage and Transmission." *IEEE Transactions on Geoscience and Remote Sensing.* Vol. 31. No. 3. May, 1993. pp. 645-654.

[2] Widrow, Bernard and Stearns, Samuel D. *Adaptive Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1985.

[3] Haykin, Simon. *Adaptive Filter Theory.* Englewood Cliffs, NJ: Prentice-Hall, Inc., 1991.

[4] McCoy, J.W, Magotra, N., and Stearns, S. "Lossless Predictive Coding." *37th IEEE Midwest Symposium on Circuits and Systems.* Lafayette, LA. August, 1994.

[5] Stearns, Samuel D. "Arithmetic Coding in Lossless Waveform Compression." *IEEE Transactions on Signal Processing.* Vol. 43. No. 8. August, 1995. pp. 1874-1879.

[6] McCoy, J.W. *Lossless Waveform Data Compression.* Ph.D. Thesis. The University of New Mexico. 1995.

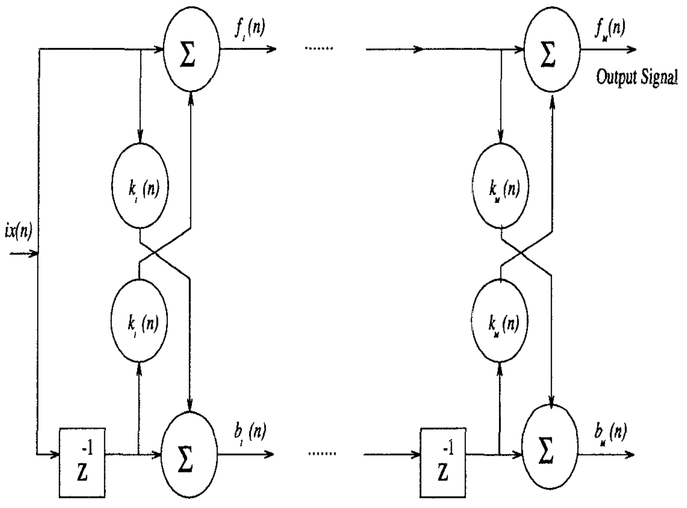

Figure 1: Gradient Adaptive Lattice (GAL) Filter

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 27,2022 at 02:12:51 UTC from IEEE Xplore. Restrictions apply.