Linear Predictive Source Coding for Sonar Data

Vladimir Burstein*† and Werner Henkel*
* Transmission Systems Group (TrSyS), Jacobs University Bremen, Germany
Email: {v.burstein and w.henkel}@jacobs-university.de
† ATLAS ELEKTRONIK GmbH, Bremen, Germany
Email: vladimir.burstein@atlas-elektronik.com

***Abstract*—Large arrays of acoustical sensors are very common for all kinds of sonars, used to navigate and detect the obstacles in the underwater environment. The large volume of data, generated by such an array has to be transmitted and stored, giving a great motivation for applying compression methods. This paper concentrates on source coding, based on the well-known linear predictive coding methods, optimized for such sources. In particular, the natural time and space correlations of a passive sonar are taken into account by applying the multichannel linear prediction. One fixed variable-length universal code, namely the subexponential code, is found to perform better than the typical Golomb-Rice codes, used in audio applications. Its average redundancy is comparable to an optimum Huffman code, with much less implementation complexity.**

I. Introduction

The non-transmitting passive sonar arrays play an important role in localization of sources under water. Particularly in the submarine context, different passive sonars are for tactical reasons the only sources of information during a mission. In general an array consists of multiple sensors, arranged in a certain way to build the array structure. Most common geometries include linear, circular, and planar (rectangular grid) arrays.

The discrete samples, generated by a hydrophone, may have natural time and space correlations, such that this digital data is expected to be redundant. The source coding process removes the redundancy of the original data, reducing the binary representation size of the compressed data either in a lossless way (the original data can be fully recovered), or discarding some information, generally denoted as lossy compression.

Most of the existing approaches for the reduction of amount of data, produced by a sonar, consider the final image after the signal processing is done and thus concentrate on image compression techniques, such as based on wavelet transform [1], discrete cosine transform [2], or even inspired by compressive sensing [3]. There are only a few examples for using the well-established compression methods for the raw sonar data [4], [5].

These lossy image compression methods can be considered feature extraction methods, as they try to keep only the relevant source information. They clearly depend on the array processing being used and cannot provide the unaltered raw data for alternative processing (e.g. adaptive beamforming). This is only possible with lossless source coding, which maps the discrete source symbols into codewords in a fully reversible manner.

A well-known assumption of independence and identical distribution (i.i.d.) for a given source is theoretically comfortable, but usually not applicable to the sonar data. The statistical dependencies of the input signal are difficult to exploit in an entropy encoder for sources with memory in practice [6]. One of the concepts allowing a rather low complexity solution to this problem is predictive coding.

II. Linear Predictive Coding

The basic idea behind the predictive coding (Fig. 1) is to guess the value of the next symbol and to apply an entropy encoding to the difference (also called residual or error) $e(k)$ between the input samples $x(k)$ and their prediction $\hat{x}(k)$

$$e(k) = x(k) - \hat{x}(k) \ . \tag{1}$$

The measure of the effectiveness of a prediction is the

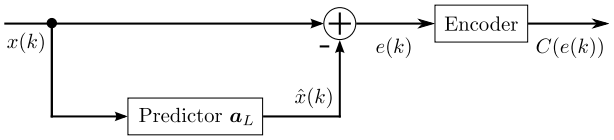

Fig. 1: Basic structure of lossless predictive coding

prediction gain, defined as

$$G_P = \frac{\sigma_x^2}{\sigma_e^2} \ . \tag{2}$$

A. Linear Prediction

Linear prediction (LP) is a widely used subclass of predictors, which uses a linear combination of the past values $x(k-i), i \in [1, L]$ to predict the current sample $x(k)$ [7]. The forward linear predictor of an order L is given by

$$\begin{aligned} \hat{x}(k) &= \sum_{i=1}^{L} a_{L,i} x(k-i) \ , \\ &= \mathbf{a}_L^T \mathbf{x}(k-1) \ , \end{aligned} \tag{3}$$

where $\mathbf{a}_L = [a_{L,1} \ a_{L,2} \ ... \ a_{L,L}]^T$ is a vector, containing L predictor coefficients and $\mathbf{x}(k-1) = [x(k-1) \ x(k-2) \ ... \ x(k-L)]^T$ are the past samples. An optimum linear predictor minimizes the mean-squared error

$$\begin{aligned} MSE(\mathbf{a}_L) &= E\{e^2(k)\} = E\{(x(k) - \hat{x}(k))^2\} \\ &= E\{(x(k) - \mathbf{a}_L^T \mathbf{x}(k-1))^2\} \ . \end{aligned} \tag{4}$$

Such a minimization problem can be solved, setting the partial derivatives of the expression in Eq. (4) to zero and obtaining a system of linear equations (called normal or Yule-Walker eq.)

$$\mathbf{R}_L \mathbf{a}_L = \mathbf{r}_L \ , \tag{5}$$

with $\mathbf{r}_L$ and $\mathbf{R}_L$ defining correlation vector and matrix, respectively, which contain the autocorrelation coefficients R_{xx} of the input signal $x(k)$

$$\mathbf{r}_L = [R_{xx}(1)\ R_{xx}(2)\ ...\ R_{xx}(L)]^T \ ,$$
$$\mathbf{R}_L = \begin{bmatrix} R_{xx}(0) & R_{xx}(1) & \cdots & R_{xx}(L-1) \\ R_{xx}(1) & R_{xx}(0) & \cdots & R_{xx}(L-2) \\ \vdots & \vdots & \ddots & \vdots \\ R_{xx}(L-1) & R_{xx}(L-2) & \cdots & R_{xx}(0) \end{bmatrix} .$$

If the input signal is stationary or locally stationary and not fully deterministic – a valid assumption in sonar signal processing – the correlation matrix is nonsingular (or invertible) and the optimal linear predictor coefficients can be finally determined as

$$\mathbf{a}_L = \mathbf{R}_L^{-1} \mathbf{r}_L \ , \tag{6}$$

The complexity of the direct method of Eq. (6) can be reduced by exploiting a very regular structure of the correlation matrix $\mathbf{R}_L$, which is in fact a Toeplitz matrix. Robinson, Levinson, and finally Durbin proposed a recursive procedure, which can be found in e.g., [7].

B. Multichannel Linear Prediction

A sonar array is a great example of an application with multiple, possibly highly correlated, channels. The multichannel data can be viewed as a two-dimensional array – with one time and one spatial dimension. It is possible to apply the previously described linear prediction to both dimensions independently. A better way to take the inter-channel correlation into account is, however, to use the multichannel linear prediction (MLP) [8].

Adapting Eq. (3) from the previous section, the linear prediction residual vector of order L in a multichannel case with M channels is given by

$$\begin{aligned} \hat{\mathbf{x}}_M(k) &= \sum_{i=1}^{L} \mathbf{A}_{L,i} \mathbf{x}_M(k-i) \ , \\ &= \mathbf{A}_L^T \mathbf{x}_{ML}(k-1) \ , \end{aligned} \tag{7}$$

where

$$\mathbf{x}_M(k) = [x_1(k)\ x_2(k)\ ...\ x_M(k)]^T \ ,$$
$$\mathbf{x}_{ML}(k-1) = \left[\mathbf{x}^T(k-1)\ \mathbf{x}^T(k-2)\ ...\ \mathbf{x}^T(k-L)\right]^T \ ,$$
$$\mathbf{A}_L = [\mathbf{A}_{L,1}\ \mathbf{A}_{L,2}\ ...\ \mathbf{A}_{L,L}]^T \ .$$

Following the derivations resulting in equations (4) and (5), the multichannel version of the normal equations can be shown to be

$$\mathbf{R}_{ML} \mathbf{A}_L = \mathbf{R}_{1/L} \ , \tag{8}$$

with the inter-correlation matrix $\mathbf{R}_{1/L}$ and block-Toeplitz covariance matrix $\mathbf{R}_{ML}$ defined as

$$\mathbf{R}_{1/L} = [\mathbf{R}_{xy}(1)\ \mathbf{R}_{xy}(2)\ ...\ \mathbf{R}_{xy}(L)]^T \ ,$$
$$\mathbf{R}_{ML} = \begin{bmatrix} \mathbf{R}_{xy}(0) & \mathbf{R}_{xy}(1) & \cdots & \mathbf{R}_{xy}(L-1) \\ \mathbf{R}_{xy}(1) & \mathbf{R}_{xy}(0) & \cdots & \mathbf{R}_{xy}(L-2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{R}_{xy}(L-1) & \mathbf{R}_{xy}(L-2) & \cdots & \mathbf{R}_{xy}(0) \end{bmatrix} .$$

The main building block of these matrices is a cross-correlation matrix $\mathbf{R}_{xy}(i)$, which contains autocorrelation coefficients along the main diagonal and generally might not be symmetrical

$$\mathbf{R}_{xy}(i) = \begin{bmatrix} R_{x_1x_1}(i) & R_{x_1x_2}(i) & \cdots & R_{x_1x_M}(i) \\ R_{x_2x_1}(i) & R_{x_2x_2}(i) & \cdots & R_{x_2x_M}(i) \\ \vdots & \vdots & \ddots & \vdots \\ R_{x_Mx_1}(i) & R_{x_Mx_2}(i) & \cdots & R_{x_Mx_M}(i) \end{bmatrix} .$$

Similar to Eq. (6), the predictor matrix $\mathbf{A}_L$ of size $ML \times M$ can be directly calculated, if the matrix $\mathbf{R}_{ML}$ is nonsingular. It is also possible to reduce the computation cost by applying a generalization of the Levinson-Durbin algorithm to the multichannel case [8].

C. Universal Entropy Codes

Besides a predictor, an entropy encoding, shown in Fig. 1, is another key component of the (linear) predictive coding. The basic idea behind any entropy encoding is to map fixed-length input symbols to variable-length codewords, such that the most probable source symbols become the shortest codes. The Huffman code is a classical example of an asymptotically optimum entropy code. This variable-length code is used in many applications (e.g. JPEG or MP3), especially if no assumptions about the probability distribution can be made. The PMF of the source is then either estimated once or block-by-block (adaptive) in the first coding step, followed by the Huffman codeword tree generation, based on the estimated symbol frequencies. More details about Huffman codes and another very popular form of entropy coding – arithmetic coding – can be found in [9] or [10].

In contrast to Huffman or arithmetic coding, universal codes can be used without the exact knowledge of the PMF, if the ranking of symbols' priorities is non-increasing and known a priori. A universal code assumes some fixed probability distribution, such that encoding and decoding complexities are quite low. However, the code is only optimum, if the assumed distribution matches the actual one of the source symbols.

The compression ratio is an ultimate performance measure for the final application, which also takes the overhead into account, introduced by the L_N (normalized order) LP coefficients of length b_{LP} bits and variable number of source coding parameters P ($P \geq 0$) of the length b_{par} bits. Essentially, this is a ratio of the compressed block size to its original size B_{LP}, defined as

$$CR_{C(S)} = \frac{\sum_{s \in S} l_C(s) + L_N \cdot b_{LP} + P \cdot b_{par}}{B_{LP}} \ , \tag{9}$$

where S is the source sequence of an arbitrary size B_{LP} bits.

1) Golomb-Rice Codes: The original Golomb family of universal codes was proposed for sources with a geometric distribution [11] and uses a single parameter m to better match the actual distribution of the source. In the Golomb code a source symbol s (a non-negative integer) is mapped to a bit-valued codeword $C_G(s,m)$ in two steps, corresponding to prefix and suffix parts:

- The **prefix part** of a Golomb code is the value of the quotient
$$q = \left\lfloor \frac{s}{m} \right\rfloor \ ,$$
encoded in the unary fashion as a $q+1$ bits long $C_U(q)$
- The **suffix part** consists of the remainder
$$r = s - mq \ ,$$
coded as an unsigned integer in binary representation (denoted here as a beta code $C_\beta(r)$) with a bit-length l_{C_β} depending on the value of m. If m is a power of 2, then the remainder will always be represented by $\log_2(m)$ bits and has a constant length $l_{C_\beta} = \log_2(m)$. Otherwise the encoding (and decoding) is more complicated and codeword length depends on the value of r. Let $c = \lceil \log_2 m \rceil$ bits, which would be the upper bound to accommodate all possible values of r. It is possible, though, to use one bit less for the values of r between 0 and $2^c - m$ (corresponding to the binary range of $\lfloor \log_2 m \rfloor$ bits), such that
$$\begin{aligned} l_{C_\beta(r)} &= c-1, \qquad r \in [0,\ 2^c - m] \ ; \\ l_{C_\beta(r)} &= c, \qquad\quad r \in [2^c - m + 1,\ 2^c - 1] \ . \end{aligned}$$

A final m-parameter Golomb codeword for a symbol s is the concatenation of prefix and suffix parts $C_G(s,m) = \{C_U(q),\ C_\beta(r)\}$. Obviously, a great simplification of the original Golomb code can be achieved by choosing the parameter $m = 2^k$ for some non-negative k. In this case the second part of the Golomb code, $C_\beta(r)$, does not need any additional calculations, it is simply the k least significant bits of the binary representation of s. This was recognized by Rice in [12], who developed an algorithm for such codes, now known as Golomb-power-of-2 (GPO2), Golomb-Rice, or simply Rice codes. Because of its implementation simplicity, this kind of codes is widely used in various applications (e.g. JPEG-LS), but even more frequently for audio compression (Shorten, FLAC, and MPEG-4 ALS to name a few).

2) Exp-Golomb Codes: The Golomb codes as well as the Rice codes provide quite effective low redundancy coding for geometric (e.g. Laplace) distribution sources. However, changes in the source distribution can lead to a very quick redundancy growth. The more robust parametrized Exp-Golomb codes (also called Elias-Teuhola [13]) have a very similar structure to the Golomb codes and supposed to better fit exponential sources:

- The variable-length **prefix part** is a unary code of
$$d = \left\lfloor \log_2 \left(1 + \frac{s}{2^k}\right) \right\rfloor \ .$$
- The **suffix part** of the length $d+k$ bits is the binary representation of
$$r = s - 2^k \left(2^d - 1\right) \ , \quad r \in \left[0, 2^{d+k}\right] \ .$$
This part has constant length if d does not change. Similar to the suffix of the Golomb-Rice code, it represents the "rest" information (remainder), not included in d.

The length of a codeword $C_{EG}(s,2^k) = \{C_U(d),\ C_\beta(r)\}$ is given by $l_{C_{EG}(s,2^k)} = 1 + k + 2\left\lfloor \log_2\left(1 + \frac{s}{2^k}\right)\right\rfloor$. Due to truncation of the logarithm, the length increases by 2 (logarithmically) each time s is a power of 2.

3) Subexponential Codes: The subexponential codes may seem to be just another set of Rice codes. They also depend on a non-negative parameter k and consist of two parts. In contrast to the previous codes, all integer symbols with values smaller than 2^k are coded in binary and mapped to fixed-length codewords of length $l_{C_{SE}(s,k)} = k+1$. For larger values of s the code length increases logarithmically, similar to the Exp-Golomb code.

- The variable-length **prefix part** is either zero, if $s < 2^k$, or otherwise a unary code of d
$$d = \left\lfloor \log_2 \left(\frac{s}{2^{k-1}}\right) \right\rfloor \ , \quad s \geq 2^k \ .$$
- The **suffix part** has the same dependency on the value of s as the prefix: it is either the k bits long binary representation of s if $s < 2^k$, or $d + k - 1$ bits long remainder r:
$$r = s - \left(2^{d+k-1}\right) \ , \quad s \geq 2^k \ .$$
Similar to the Exp-Golomb code, this remainder has constant length if d does not change.

Hence, the structure of the codewords is slightly more complicated, than the previous codes and depends on the value of s:
$$\begin{aligned} C_{SE(s,k)} &= \{'0', C_\beta(s)\} \ , \qquad s < 2^k \ ; \\ C_{SE(s,k)} &= \{C_U(d), C_\beta(r)\} \ , \qquad s \geq 2^k \ . \end{aligned}$$
Consequently, the codeword lengths are then given by
$$\begin{aligned} l_{C_{SE}(s,k)} &= k+1 \ , \qquad s < 2^k \ ; \\ l_{C_{SE}(s,k)} &= 2d + k \ , \qquad s \geq 2^k \ . \end{aligned}$$
As a consequence of this two-fold structure, also the optimal source distribution consists of two parts: the uniform part for the sample values smaller than 2^k corresponds to the uniform distribution due to the fixed length of the codewords; the exponential part for the $s \geq 2^k$ behaves exactly like the Exp-Golomb codes.

III. Simulations

Within the scope of this work two sonar datasets have been used. Both resulted from simulating different scenarios of a submarine mission with multiple moving targets, very close, but not identical to the real world classified data. Each set contains 2 hours of raw data from 96 single hydrophones of a flank array, sampled at $T = 7.8125$ kHz and uniformly quantized at 16 bit.

Compared to the underwater survey data in, e.g., [4] or typical speech data [14], the empirical probability distribution of the source in Fig. 2, based on the above described datasets

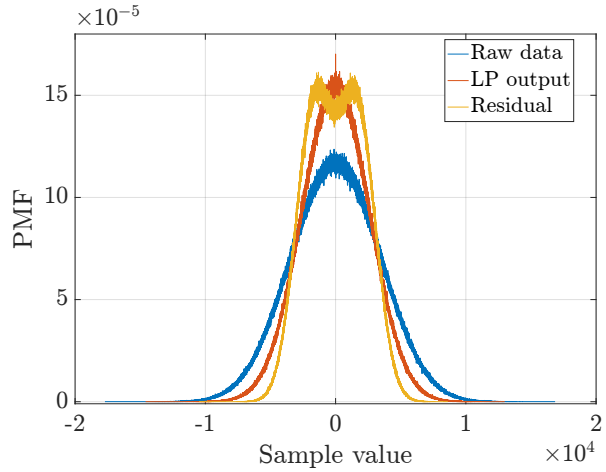

Fig. 2: Probability mass functions for the fixed LP order (16)

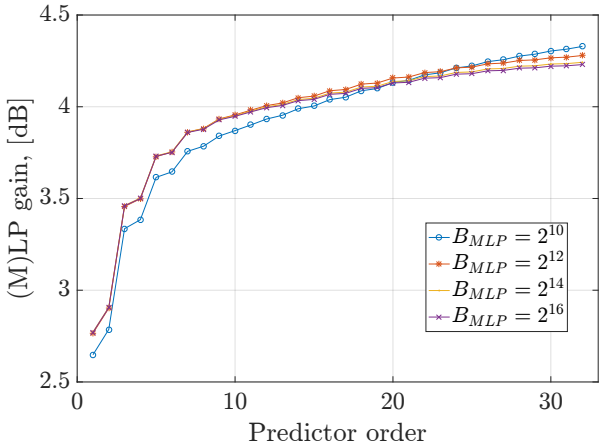

Fig. 4: Mean prediction gain for MLP with $M = 2$

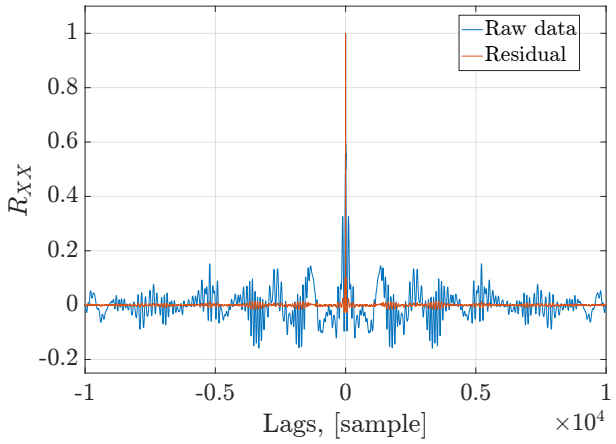

Fig. 3: Autocorrelation function of residual signal of the first sensor before and after applying linear prediction of order 16

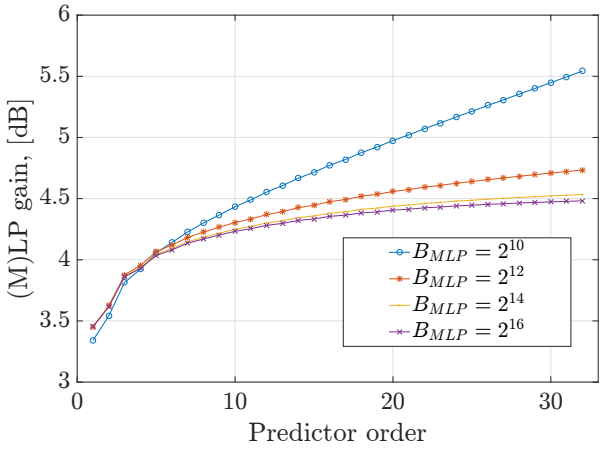

Fig. 5: Mean prediction gain for MLP with $M = 8$

(assuming the ergodicity of the random process), tends to be very close to an ideal normal distribution. Despite a slight difference in probability distribution between the original data and the prediction, applying linear prediction of rather moderate orders (up to 32) results in significant reduction of the time-domain correlation of the acoustic data, received by a sonar. The autocorrelation function of the residuals in Fig. 3 has a single peak at the origin, meaning the absent periodicity in the noise-alike residual sequence.

A. Linear Prediction

The optimum prediction parameters, satisfying Eq. (6) can be calculated either for the whole data set, assuming the ideal stationarity of the source or for the blocks of data. The latter approach was used here, as real sources are seldom perfectly stationary and are better modeled as locally block-stationary processes.

The dependency of the prediction gain on different block sizes B_{LP}, predictor order, and number of channels M is shown in figures 4 and 5. A more pronounced dependency on the block size with increasing M – smaller blocks are better – can be explained as a direct consequence of non-stationarity. It has a very limited influence on the single-dimensional prediction gain for the provided acoustical data.

The mean prediction gain in Fig. 6, calculated as the average over gains for all channels and blocks, approaches the flat region (the upper bound is known as asymptotic prediction gain, given in e.g., [6]) at the LP order of roughly 25, getting as high as 4 dB for the single-dimensional prediction. An additional dimension ($M > 1$) clearly improves the performance of the linear prediction (1 dB gain improvement for $M = 16$). However, the total number of coefficients, given by $L \cdot M^2$, also increases noticeably.

B. Optimal Code for Residual Compression

The PMFs of the codes, with parameters best-fitted to the distributions of LP residuals of the available data sets, are shown in Fig. 7. The residuals were first mapped to nonnegative integers by bijection [15], which first multiplies the absolute value of input by a factor of two and then adds one

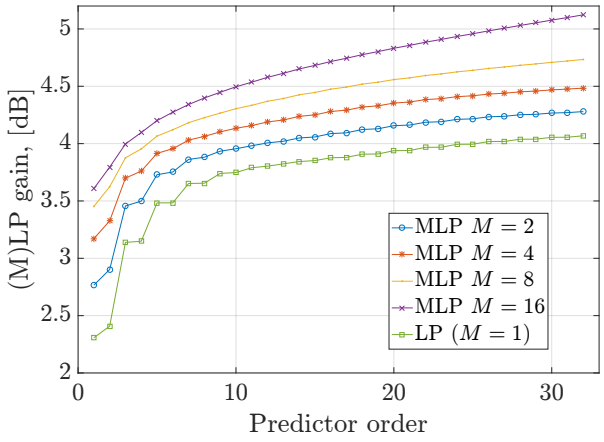

Fig. 6: Mean prediction gain of single- and multichannel linear predictors, $B_{(M)LP} = 2^{12}$

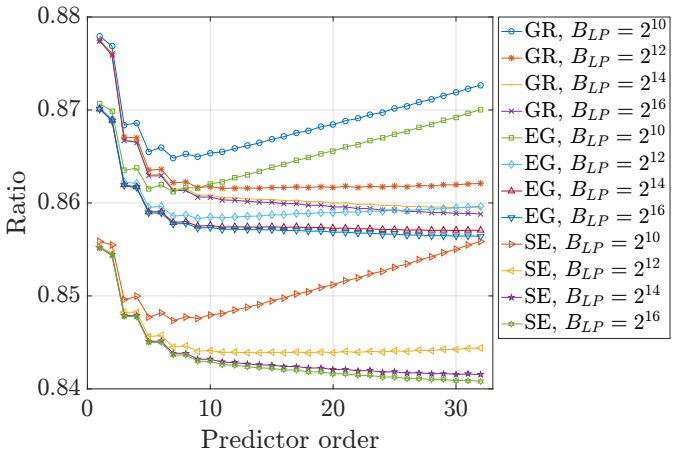

Fig. 8: Mean compression ratio over all channels of different variable-length entropy codes

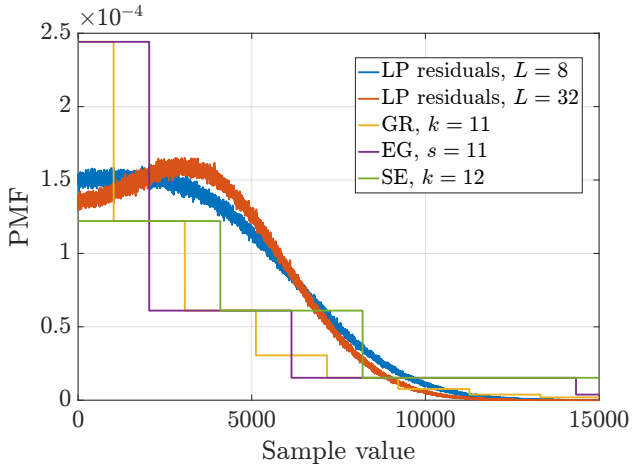

Fig. 7: Probability mass functions of linear prediction residuals (block size $B_{LP} = 10^{16}$) and different variable-length entropy codes

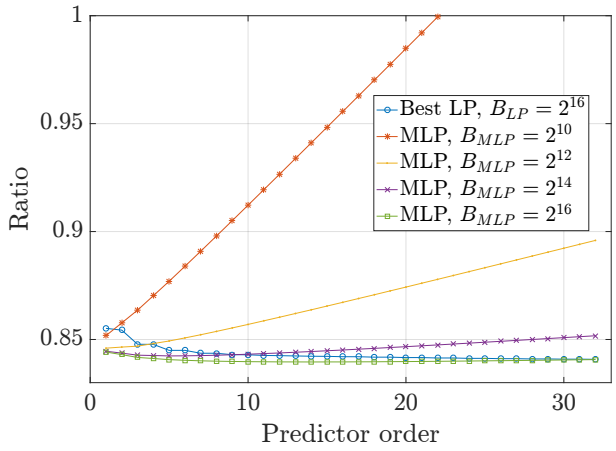

Fig. 9: Mean compression ratio of the subexponential code, used with the multichannel ($M = 16$) linear predictor

to the output, if the input was a positive integer. Calculation of the optimum parameters of the Golomb-Rice (GR), Exp-Golomb (EG), and subexponential (SE) codes can be found in [16] and [17].

In general, the subexponential code achieves the closest match to the source PMF. Its structure has a very useful property for the given source distributions – the code PMF can be divided into uniform and exponential parts. The uniform part is in fact the best fit for the source PMFs up to the sample values of roughly 3000. The tendency of the residual PMF to form two peaks with increasing predictor order, as discussed in the previous section, is handled quite well by this code: at low values of predictor order it is close to optimum and the redundancy at higher values is rather low, compared to other codes, which all assume a non-increasing PMF.

The compression ratio, calculated by eq. (9), is smaller and thus the performance is better, when larger block sizes are used, as shown in Fig. 8. Obviously, this is due to the overhead of transmitting the LP coefficients. As a function of the predictor order, the ratios will have a single minimum (which is also the global optimum) not necessarily at the highest value of order. Not surprisingly, the subexponential code performs best, because of the better PMF match of the source, and is used in the following simulations. The acoustical signal, sampled by a sonar, is generated in a very noisy environment and hence has less potential for lossless source coding, compared to to the best examples of text or speech coding. Nevertheless, roughly 15% data rate reduction is a welcomed improvement.

Even better performance in terms of compression ratio can be achieved by applying multichannel linear prediction, as can be seen in Fig. 9. The number of the coefficients, and thus the overhead, grows faster with L for M-LPC, which results in a low performance at high values of L and small block sizes. As the block size increases, M-LPC outperforms LPC by a small margin. Increasing L does not automatically decreases the minimum ratio, such that the optimum value is around 10.

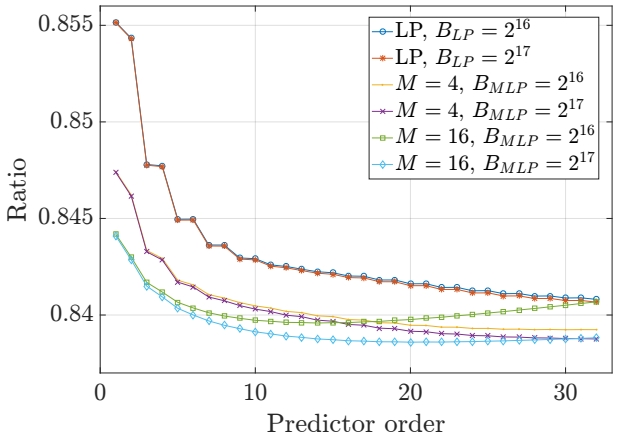

Fig. 10: Mean compression ratio of the subexponential code, used with the multichannel linear predictor and larger block sizes

Compression method	block size	max. compression ratio
ZIP	n.a.	0.932
BZIP2	100 kbit	0.908
BZIP2	900 kbit	0.888
FLAC	130 kbit	0.859
FLAC	1 Mbit	0.858
LP+SE	130 kbit	0.841
LP+SE	1 Mbit	0.840
4-MLP+SE	130 kbit	0.839
16-MLP+SE	130 kbit	0.838
16-MLP+SE	1 Mbit	0.837

TABLE I: Mean compression ratios of different source codes

Reducing the number of channels in M-LPC (see Fig. 10) results in a slightly increased compression ratio. Although the best performance can only be achieved with more channels, using less channels for M-LPC has an advantage of lower implementation complexity and higher flexibility in terms of the block size. It is usually limited in practice by memory or initial delay requirements. It can be seen in Fig. 10 that reducing the block size for $M = 4$ does not increase the ratio by a large margin, as it is the case for $M = 16$. Still, compared to the single channel LPC, the 4-channel M-LPC performs noticeably better.

The comparison of different source codes, including popular dictionary methods (e.g. ZIP and BZIP2) and the lossless audio-codec FLAC [15], given in Table I, clearly indicates that the combination of linear prediction and subexponential codes has an application potential in sonar data transmission. The multichannel prediction shows a better performance in absolute numbers, but this fact is rather of theoretical interest, as the implementation complexity would outweigh its advantages in practice.

IV. Conclusion

The source data, generated by the acoustical sensors, was shown to be redundant. Using the methods of (audio) lossless compression, particularly linear prediction coding, a source coding scheme was introduced. The single channel linear predictor case was extended to cope with the multi-channel data, resulting from the physically combined sensors. A subexponential code was proven to be the most efficient universal variable-length code for both cases. It provides better matching of the probability distribution of the residuals, generated by the linear prediction, than the other codes. The new methods were shown to be more effective than the existing compression schemes and can achieve an average lossless compression ratio of roughly 15 % for this particular type of data. Further compression can be achieved by lossy compression methods, but this requires adaptation to the signal processing algorithms (e.g., direction-of-arrival estimation) used.

Acknowledgment

The authors would like to thank ATLAS ELEKTRONIK GmbH in Bremen for sponsoring the project and providing the acoustic data.

References

[1] L. Bin and M. Qinggang, "An improved spiht wavelet transform in the underwater acoustic image compression," in *Proceedings of 2013 2nd International Conference on Measurement, Information and Control*, vol. 02, pp. 1315–1318, Aug 2013.

[2] D. H. Kil and F. B. Shin, "Reduced dimension image compression for remotely distributed underwater signal processing," in *OCEANS '95. MTS/IEEE. Challenges of Our Changing Global Environment. Conference Proceedings.*, vol. 2, pp. 1183–1188, Oct 1995.

[3] W. Chen, F. Yuan, and E. Cheng, "Adaptive underwater image compression with high robust based on compressed sensing," in *2016 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, pp. 1–6, Aug 2016.

[4] L. S. Wong, G. E. Allen, and B. L. Evans, "Sonar data compression using non-uniform quantization and noise shaping," in *2014 48th Asilomar Conference on Signals, Systems and Computers*, pp. 1895–1899, Nov 2014.

[5] A. Chybicki, M. Moszynski, and P. Pocwiardowski, "Applications of compression techniques for reducing the size of multibeam sonar records," in *2008 1st International Conference on Information Technology*, pp. 1–4, May 2008.

[6] T. Wiegand and H. Schwarz, "Source coding: Part I of fundamentals of source and video coding," *Foundations and Trends in Signal Processing*, vol. 4, no. 12, pp. 1–222, 2011.

[7] J. Makhoul, "Linear prediction: A tutorial review," *Proceedings of the IEEE*, vol. 63, pp. 561–580, April 1975.

[8] J. Benesty, M. M. Sondhi, and Y. A. Huang, *Springer Handbook of Speech Processing*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[9] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.

[10] K. Sayood, ed., *Lossless Compression Handbook*. Communications, Networking and Multimedia, San Diego: Academic Press, 2003.

[11] S. Golomb, "Run-length encodings," *IEEE Transactions on Information Theory*, vol. IT-12, pp. 399–401, July 1966.

[12] R. Rice, "Some practical universal noiseless coding techniques," *JPL Publication 79-22*, March 1979.

[13] J. Teuhola, "A compression method for clustered bit-vectors," *Information Processing Letters*, vol. 7, no. 6, pp. 308–311, 1978.

[14] S. Gazor and W. Zhang, "Speech probability distribution," *IEEE Signal Processing Letters*, vol. 10, pp. 204–207, July 2003.

[15] D. Salomon and G. Motta, *Handbook of Data Compression*. Springer Publishing Company, Incorporated, 5th ed., 2009.

[16] A. B. Kiely, "Selecting the Golomb parameter in Rice coding," *IPN Progress Report*, vol. 42-159, Nov. 2004.

[17] S. Pigeon, "Start/stop codes," in *2001 Data Compression Conference (DCC 01)*, p. 511, 2001.