A lossless compression method for logging data while drilling

Shan Song\textsuperscript{a}, Taiwei Lian\textsuperscript{b,}* , Wei Liu\textsuperscript{b}, Zhengbing Zhang\textsuperscript{a}, Mingzhang Luo\textsuperscript{a} and Aiping Wu\textsuperscript{a}

\textsuperscript{a}Electronic Information School, Yangtze University, Jingzhou, People’s Republic of China; \textsuperscript{b}Directional Drilling Technology Service Company, CCDC Drilling and Production Engineering Technology Research Institute, Deyang, People’s Republic of China

ABSTRACT

Mud pulse transmission technology is widely used in logging while drilling (LWD), while its low data transmission efficiency has restricted the development of LWD technology for a long time. In order to improve the equivalent data transmission efficiency of LWD system, a data compression method, namely, Frame Prediction Huffman Coding (FPHC) with a group of frames, is proposed in this paper. Several factors including the data accuracy, the need for real-time transmission, and the existence of transmission errors are comprehensively considered in the proposed method. Through the probability statistical analysis of LWD data, it is proved that the data have a relatively stable probability distribution after predictive coding. In addition, the Huffman code table generated from predictive coding is universally applicable to other LWD data. Through data compression experiments on logging data, it is verified that the proposed method has higher compression efficiency compared with Huffman coding, Lempel–Ziv–Welch algorithm, adaptive Huffman coding and adaptive arithmetic coding. This lossless compression method will have a good application prospect because of its ability to prevent error diffusion, simple implementation and easy to be embedded in down-hole equipment.

1. Introduction

Logging while drilling (LWD) technology can monitor the drilling and formation conditions in real time during bit drilling. Therefore, it is an important technology to obtain formation parameters in oil drilling activities, and it is widely used in various drilling and logging projects (Ivanov et al., 2020; Mwachaka et al., 2019; Yang et al., 2013; Zhu et al., 2013). In the process of LWD activities, it is necessary to transmit the logging information to the surface in real time. The mud pulse is a stable, reliable and the most widely used method of LWD information transmission. But the data transmission rate is very low (Bernasconi et al., 1999; Su et al., 2013; Zheng et al., 2012). In addition, with the development of LWD technology, measuring instruments are constantly improved and measuring parameters are also increasing. From the initial measurement of geometric parameters such as well inclination, tool orientation and tool face, to the measurement of multiple geological environment parameters such as pressure, temperature, natural gamma-ray and resistivity, and the amount of data to be transmitted increases sharply. So traditional mud pulse data transmission technology is difficult to meet the needs of real-time logging engineering. However, the efficiency of data transmission in LWD will be greatly improved if the data compression technology is applied (Hutin et al., 2001). Thus, the real-time performance and safety in LWD engineering can be improved while carrying out LWD projects.

In order to apply data compression technology to LWD engineering, many studies are mainly focused on lossy compression technology and lossless compression technology. In the lossy data compression method for LWD data processing, wavelet transform and differential pulse code modulation are the main techniques. In terms of the application of wavelet transform, Bernasconi and Rampa paused applied wavelet transform to compress LWD data in 2004 (Bernasconi & Rampa, 2004). The compression ratio reached 15:1, the signal-to-noise ratio was maintained at more than 10–15dB, which greatly improved the ability of downhole equipment to store logging data while drilling. In 2018, Jarrot et al. proposed the directional wavelet transform method to compress the downhole image data by studying the sparse representation characteristics of the downhole image. The comparative experiments showed that the inclination angle in the image compressed by directional wavelet transform is more easy to be identified than the image compressed...
In terms of differential pulse code modulation, Zhang et al. proposed a compression method based on differential pulse code modulation, which greatly improved the compression efficiency and controlled the information distortion rate below 5% in 2010 (Zhang et al., 2010). In 2019, Yao et al. designed a data compression algorithm based on interframe difference and applied it to the data transmission process of LWD, in which the compression ratio of well deviation data is 1.72 and that of tool face data is 1.2. This method effectively improved the speed of data updating (Liang et al., 2019). Both wavelet transform and differential pulse code modulation are lossy compression methods. Although the lossy compression method improves the compression efficiency of LWD data, it sacrifices the accuracy of data. Therefore, this kind of data compression algorithm is obviously unreliable for some LWD projects which require high accuracy of data.

In terms of lossless data compression of LWD data, Lempel–Ziv–Welch (LZW) algorithm (Welch, 1984) and Huffman algorithm (Huffman, 1952) are the main technologies. In the application of LZW algorithm, Lin et al. in 2012 applied LZW algorithm to compress the acoustic signal of LWD and used hash search algorithm to locate the position of string code quickly. The results showed that the average compression ratio of LZW algorithm for acoustic signals of LWD is more than 2:1 (Lin et al., 2012). In 2013, Zou et al. improved LZW algorithm by introducing XOR hash function to further improve the efficiency of dictionary queries. The improved LZW algorithm can achieve a compression ratio of more than 50% (Zou et al., 2013). In 2016, Li et al. analyzed the mud pulse channel and noise characteristics and proposed a joint coding method of LZW-RS, which realized lossless data compression without increasing the system’s complexity. The compression ratio was 2:1–3:1. This method improved the data transmission efficiency, reduced the bit error rate, and ensured the reliability of data transmission of the channel (Li & Shi, 2016). Burrows–Wheeler transform algorithm (Burrows & Wheeler, 1994) was used by Wang to improve LZW algorithm. The improved algorithm is called DB–LZW coding, which made the process of creating a dictionary in LZW algorithm simpler. In the data compression experiment on stationary data, the average compression ratio of the improved LZW algorithm was reduced by 10% to 15% compared with the classical LZW algorithm. The average compression time was reduced from 58 to 54 ms. This method not only improved the compression effect but also reduced the compression time (Wang, 2016). Although LZW algorithm improves the compression efficiency, the dictionary management of LZW algorithm has always been a difficult problem to handle. Because LZW algorithm creates the string table in the process of compression and decompression adaptively, it will lead to a wrong dictionary in the decoding process if there is a bit error in the process of transmission, and then affect the decoding of all subsequent data. This kind of error diffusion will be disastrous. In terms of the Huffman coding application, Li et al. designed a data compression protocol based on Huffman coding according to the stability of LWD parameters in 2014. Through the simulation experiments to real well history data, the average compression ratio reached 43%, which improved the data transmission efficiency of wireless LWD channel (Li et al., 2014). Huffman coding is a classical lossless data compression method, which is based on the characteristics of data statistics, making it more suitable for LWD data compression. However, the traditional Huffman coding needs to find the probability distribution of the original data in advance, and it has not been optimized in the existing study. So it cannot be directly applied to the real-time data compression and transmission while bit drilling.

On the one hand, lossless data compression method must be used when an LWD project requires high data accuracy. Moreover, the advantage of LWD is its real-time performance. The higher the real-time performance is, the more efficiently it can help the surface operators to adjust the drilling strategy. On the other hand, the underground environment is complex; it is difficult to avoid bit error in the process of data transmission. Unfortunately, after the analysis above, there is no compression method, which can take into account the accuracy of data, high compression efficiency, real-time data transmission and a certain anti error ability at the same time. In order to address these issues, a coding method, namely, Frame Prediction Huffman Coding (FPHC) with group of frames, is proposed in this paper. This method is a lossless real-time compression method. It is easy to implement and has a certain of ability to anti the error’s propagation. In order to verify the effectiveness of this method, we compare this method with adaptive Huffman coding (Knuth, 1985), adaptive arithmetic coding (Moffat et al., 1998), LZW coding and other real-time data compression methods (Nelson & Gailly, 1995). The results show that the proposed method outperforms previous approaches.

The remainder of this paper is arranged as follows: In Section 2, the probabilistic model of LWD data is analyzed as a theoretical foundation in order to propose a new compression method; In Section 3, an LWD data compression method, namely, FPHC with a group of frames is proposed, the encoding and decoding principles and algorithms are also described in detail; In Section 4, logging data are compressed and decompressed with the proposed method, the experimental results demonstrate
that the proposed method has higher compression efficiency compared with Huffman coding, LZW algorithm, adaptive Huffman coding and adaptive arithmetic coding; at last the conclusion is obtained in Section 5.

2. The analysis of probability model of LWD data

In order to improve the real-time transmission rate of LWD data, it is necessary to embed the compression algorithm into downhole instruments. Therefore, two important factors should be considered in the design of the compression algorithm:

(1) Downhole instruments require low power consumption and small size. Downhole instruments generally use multiple high-temperature lithium batteries for power supply, and the space for data compression module is narrow, which requires that the data compression module must be low-power and small size (Chao et al., 2015; Fang et al., 2019; Li, 2018; Qiu et al., 2012; Zhang et al., 2009; Zhang et al., 2015). Therefore, when designing the compression algorithm, we should choose the data compression algorithm which is simple with low computational complexity, and easy to implement in small devices;

(2) The coding should be error-resilient. Due to the interference of noise, the channel has a high bit error rate when using mud pulse to transmit data (Hu et al., 2019; Liang et al., 2020; Qu et al., 2018; Tu et al., 2014). Therefore, when designing compression algorithm, it is required to have a certain bit error resistance.

The existing lossless data compression techniques for logging data transmission include Huffman coding, arithmetic coding (Witten et al., 1987) and LZW algorithm. Among them, LZW algorithm is simple to implement, but it needs to establish the dictionary at the encoder and decoder simultaneously. So, its error resilience is poor, which cannot meet the requirements of resilience. Arithmetic coding and Huffman coding are both coding methods based on probability and statistics. They all use the probabilistic model to compress data. In terms of the compression effect, the compression efficiency of the two is close to the entropy of the information source, and there is no big difference (Moffat, 2019). In terms of computational complexity, arithmetic coding requires a lot of multiplication, addition and subtraction methods. It also has high requirements for the accuracy of floating-point calculation (Bodden et al., 2007). So arithmetic coding has high requirements on the processor, which does not meet the requirements of low power consumption and small size of compression module. Huffman coding is divided into non-adaptive Huffman coding and adaptive Huffman coding. Adaptive Huffman coding is to dynamically establish and update the probability model of data in the process of encoding and constantly update the Huffman tree and Huffman codeword at the same time. Moreover, in order to decode correctly, the encoder and decoder must update the probability model and Huffman tree synchronously. They have to ensure that the probability model and Huffman tree are completely consistent. If the compressed data appears error bits in the process of data transmission, the decoder can’t decode all the subsequent data correctly. Therefore, adaptive Huffman coding not only has higher computational complexity than non-adaptive Huffman coding does but also has poor error resilience. Instead of adaptive Huffman coding, non-adaptive Huffman coding uses the fixed probability model, so the Huffman codeword of each symbol is independent. Thus, compared with LZW algorithm, arithmetic coding, adaptive Huffman coding and other adaptive coding algorithms, the error resilience performance of non-adaptive Huffman coding is better.

To sum up, among LZW coding, arithmetic coding, adaptive Huffman coding and non-adaptive Huffman coding, the non-adaptive Huffman coding has the best performance of error resistance and the lowest computational complexity. Therefore, the non-adaptive Huffman coding algorithm is undoubtedly the best choice to realize lossless compression of LWD data in downhole instruments, which require low power consumption, small size and error resistance. However, the great challenge of using non-adaptive Huffman coding is that the fixed probability model used in the algorithm may not match the statistical characteristics of the current data. This will result in a poor compression effect. To tackle this challenge, we study the statistical characteristics of actual drilling data, which lays a theoretical foundation for effectively compressing LWD data by using non-adaptive Huffman coding algorithm.

2.1. Probability model analysis of LWD data

In order to describe the research process of the probability model of LWD data, the average code length is used to represent the compression potential of data. The types of the average code length are divided into the average code length based on self-probability model and the average code length based on the other-probability model. The average code length based on the self-probability model is the sum of the product of the probability distribution of the data set itself and the code length of the corresponding Huffman codewords, while the average code length based on the other-probability model is the sum of the product of the probability distribution of
one data set and the code length of Huffman codewords of another data set.

Assume data set \( g(i) (i = 1, 2, 3, \ldots) \) has \( N \) kinds of values, \( p_{g(i)}(n) (n = 0, 1, 2, 3, \ldots N - 1) \) is the probability of \( n \) in the data set, and \( L_{g(i)} (n) (n = 0, 1, 2, 3, \ldots N - 1) \) is the length of the Huffman codeword of the value of \( n \) according to the probability model of \( g(i) \). Then the average code length based on the self-probability model of \( g(i) \) is as follows:

\[
\bar{L}_{g(i), g(j)} = \sum p_{g(i)}(n) L_{g(i)} (n).
\]

(1) where \( i = 1, 2, 3, \ldots n = 0, 1, 2, 3, \ldots N - 1. \)

The average code length based on another probability model of \( g(i) \) is as follows:

\[
\bar{L}_{g(i)} = \sum p_{g(i)}(n) L_{g(i)} (n)
\]

where \( i = 1, 2, 3, \ldots j = 1, 2, 3, \ldots \) and \( i \neq j, n = 0, 1, 2, 3, \ldots N - 1. \)

Obviously, the average code length based on the self-probability model corresponds to the common usage of non-adaptive Huffman coding. The calculation process of the average code length based on the other probability model is equivalent to using the Huffman code generated by the probability model of one group of data to compress another group of data.

In order to measure the difference of average code length, we proposed the increment ratio of code length to represent the percentage of the increment of the average code length of the compressed data using the other probability model relative to the average code length of the compressed data using the self-probability model. The calculation formula is as follows:

\[
\Delta LR_{g(i), g(j)} = \frac{\bar{L}_{g(i), g(j)} - \bar{L}_{g(i)} g(j)}{\bar{L}_{g(i), g(j)}} \times 100% \tag{3}
\]

where \( i = 1, 2, 3, \ldots j = 1, 2, 3, \ldots \).

The smaller the absolute value of the increment ratio of code length is, the closer the other probability model is to the self-probability model. In other words, if the absolute value of the increment ratio of code length is very small, it means that the effect of using other probability model is almost as good as that of using the self-probability model.

The purpose of the experiment is to solve the problem that the fixed probability model used by the Huffman algorithm may not match the statistical characteristics of the compressed data, which results in a poor compression effect. If the following two points can be proved by experiments, the problem can be solved well:

1. The residual data after predictive coding has a stable probability distribution, and the probability model distribution of residual data is more concentrated, which is more conducive to compression.
2. The average code length of the residual data after predictive coding based on other probability model is close to that of the residual data after predictive coding based on the self-probability model.

In order to prove the above two arguments, we designed and carried out the following experiments:

1. Collect 19 LWD data files as a data set which is named DS. These files are collected in different time periods in the same survey area;
2. Divide the files in DS into two groups randomly, and the group names are Group 1 and Group 2, the two groups contain 10 and 9 files, respectively;
3. Count the frequency of the original data in all the files in each group by byte, and then generate two probability distribution models according to the frequency, respectively;
4. Fit the probability distribution model of the original data obtained in step (3) with the normal distribution curve, and get the fitting parameters \( \mu_o \) and \( \sigma_o \);
5. Code the original data in all files in each group with adjacent interframe predictive coding, respectively, that is, in addition to the data in the first frame, the value of each byte of the original data in the current frame is subtracted from the value of the original data in the previous frame corresponding to its position. The value of the residual data is taken as the value of the data in the current position. Generate the corresponding probability distribution models of residual data according to their frequency;
6. Fit the probability distribution models obtained in step (5) with the normal distribution. Get the fitting parameters \( \mu_p \) and \( \sigma_p \) of each group. Then, calculate the average code length based on the self-probability model \( \bar{L}_{g(i), g(j)} \) and the average code length based on other probability model \( \bar{L}_{g(i)} \), then, calculate the code length increment ratio \( \Delta LR_{g(i), g(j)} \);
7. Repeat steps (2), (3), (4), (5) and (6) above and make sure that the files in Group 1 and Group 2 are random and not repeated in each experiment. Until a total of eight experiments are completed. Count the fitting parameters \( \mu_o \) and \( \sigma_o \) of the probability distribution of the two groups of original data in each experiment and the fitting parameters \( \mu_p \) and \( \sigma_p \) of the probability distribution of residual data, and the average code length based on the self-probability model \( \bar{L}_{g(i), g(j)} \) and the code length based on other probability model \( \bar{L}_{g(i)} \), and code length increment ratio \( \Delta LR_{g(i)} \) of residual data in each experiment.
Calculate the average value and standard deviation of each kind of data above.

After performing the above experimental steps, the probability distribution and normal distribution fitting curve of the original data of Group 1 are obtained, as shown in Figure 1.

In Figure 1, in each experiment of Group 1, the probability distribution shape of the original data is quite different, which indicates that the probability distribution of the original data is unstable.

The probability distribution and normal distribution fitting curve of the original data of Group 2 are obtained as shown in Figure 2.

In Figure 2, the probability distribution of the original data of each experiment in Group 2 is similar to that in Group 1, that is, the probability distribution shape of each experiment in Group 2 is quite different.

The probability distribution and normal distribution fitting curve of prediction residual data in Group 1 are shown in Figure 3.

In Figure 3, the probabilities of values in the interval $[-255, -30]$ and $[30,255]$ are very small. For the convenience of observation, only the probability distribution and its normal distribution fitting curve in the interval $[-30,30]$ are shown. In Figure 3, the probability distribution shape of the residual data in each experiment of Group 1 is basically the same, which indicates
that the residual data in Group 1 has a stable probability distribution.

The probability distribution and normal distribution fitting curve of residual data in Group 2 are shown in Figure 4.

Figure 4 is similar to Figure 3. For the convenience of observation, only the probability distribution of $[-30,30]$ interval and its normal distribution fitting curve are shown. In Figure 4, the probability distribution of residual data in each experiment of Group 2 is similar to that in Figure 3. The shape of the probability distribution of predictive residual data of every experiment of Group 2 is basically the same, so the probability model of predictive residual data in Group 2 is stable.

The parameters fitted by the normal distribution curve of each experiment of each group were counted, and the statistical results are shown in Table 1.

It can be seen from Table 1 that the standard deviation of the fitting parameters of the probability distribution of original data is large, while the standard deviation of the residual data after predictive coding becomes very small. It shows that after the data being coded with predictive coding, the probability distribution of it becomes stable.

The average code length based on the self-probability model and the average code length based on other probability models of two groups of data in each experiment are calculated, and the results are shown in Table 2.
Table 1. Statistical table of fitting parameters of the normal distribution curve.

| Data     | Parameter | Group 1 | 1st  | 2nd  | 3rd  | 4th  | 5th  | 6th  | 7th  | 8th  | Average value | Standard deviation |
|----------|-----------|---------|------|------|------|------|------|------|------|------|----------------|-------------------|
| Original | \(\mu_o\) | 17.463  | 19.524 | 16.161 | 22.991 | 9.363 | 21.234 | 20.276 | 17.966 | 18.247 | 4.336         |                   |
|          | \(\sigma_o\) | 2.568  | 2.105  | 2.416  | 2.324  | 2.339 | 2.504  | 2.521  | 2.426  | 2.400  | 0.147         |                   |
| Predictive | \(\mu_p\) | 1.973  | 2.397  | 2.014  | 2.169  | 2.101 | 1.962  | 2.016  | 2.097  | 0.144  | 0.079         |                   |
|          | \(\sigma_p\) | 0.120  | 0.112  | 0.122  | 0.090  | 0.083 | 0.110  | 0.129  | 0.130  | 0.112  | 0.017         |                   |

Table 2. Statistics results of average code length of the probability distribution of predictive.

| Group     | Data | 1st  | 2nd  | 3rd  | 4th  | 5th  | 6th  | 7th  | 8th  | Average value | Standard deviation |
|-----------|------|------|------|------|------|------|------|------|------|----------------|-------------------|
| Group 1   | \(\bar{L}_{g(1,g(1))}\) (bit) | 4.843 | 4.584 | 4.721 | 4.822 | 4.681 | 4.695 | 4.716 | 4.664 | 4.716 | 0.084          |                   |
| Group 2   | \(\bar{L}_{g(1,g(2))}\) (bit) | 4.884 | 4.592 | 4.732 | 4.871 | 4.690 | 4.707 | 4.725 | 4.674 | 4.734 | 0.098          |                   |
| Group 1   | \(\bar{L}_{g(2,g(1))}\) (bit) | 4.438 | 4.467 | 4.511 | 4.507 | 4.583 | 4.591 | 4.541 | 4.582 | 4.553 | 0.071          |                   |
| Group 2   | \(\bar{L}_{g(2,g(2))}\) (bit) | 4.408 | 4.666 | 4.500 | 4.470 | 4.576 | 4.583 | 4.533 | 4.572 | 4.539 | 0.079          |                   |

In Table 2, the average value of \(\bar{L}_{g(1,g(1))}\) is slightly greater than \(\bar{L}_{g(2,g(1))}\), and \(\bar{L}_{g(2,g(2))}\) is a little bigger than \(\bar{L}_{g(2,g(1))}\). In sum, the average code length of the other-probability model is very close to that of the self-probability model.

The code length increment ratio of each group of predictive residual data is calculated, and the results are shown in Table 3.

The increment ratio of code length represents the percentage of the increment of the average code length of the compressed data using the other-probability model relative to that of the compressed data using the self-probability model. In Table 3, the average value of \(\Delta \bar{L}_{g(1)}\) is \(-0.390\)% , which indicates that the average code length of data in Group 1 using other-probability model is only 0.39% larger than that using the self-probability model. Similarly, the average code length of the data in Group 2 using the other-probability model is 0.328% larger than that using the self-probability model.

2.2. Analysis and summary of probability model of LWD data

The probability distributions in Figures 1 and 2 are disordered and uneven, which are not suitable for data compression by Huffman coding. On the contrary, the shapes of the probability distributions of the predictive residual data in Figures 3 and 4 are basically the same in each experiment, and these distributions are relatively concentrated. In addition, it can be seen from Table 1 that the values of standard deviations of fitting parameters \(\mu_o\) and \(\sigma_o\) obtained are much larger than that of the fitting parameters \(\mu_p\) and \(\sigma_p\), and the average values of fitting parameters \(\mu_p\) and \(\sigma_p\) are very small. These results indicate that the residual data has the advantage of a stable and centralized probability distribution, this makes it more suitable for data compression by Huffman coding.

It can be seen from Tables 2 and 3 that the average code length based on the other-probability model is very close to that based on the self-probability model. This indicates that the Huffman code table generated by using the probability model of residual data of one data set will have a very similar compression effect for the residual data of another data set.

The above results show that the probability distribution of predictive residuals is basically invariable. Moreover, the probability distribution of prediction residuals is relatively concentrated. It will be more suitable to be compressed by Huffman coding. It can be inferred that, the probability model of predictive residuals based on the existing LWD data volume with a certain scale will have a similar compression effect on the predictive residual data of other LWD data. That is to say, the probabilistic model for the residual data will be universally applicable. This lays a foundation for effective real-time data compression of LWD data by using an independent Huffman code table. Therefore, the Huffman code table based on the probability distribution of the last residual data will be suitable for compressing the current and future residual data collected in the same logging area. The data

Table 3. Statistics results of code length increment ratio of two groups of predictive residual data.

| Group     | Data | 1st  | 2nd  | 3rd  | 4th  | 5th  | 6th  | 7th  | 8th  | Average value | Standard deviation |
|-----------|------|------|------|------|------|------|------|------|------|----------------|-------------------|
| Group 1   | \(\Delta \bar{L}_{g(1)}\) (%) | -0.847 | -0.175 | -0.233 | -1.016 | -0.192 | -0.256 | -0.191 | -0.214 | -0.390 | 0.338          |
| Group 2   | \(\Delta \bar{L}_{g(2)}\) (%) | -0.681 | -0.150 | -0.244 | -0.828 | -0.153 | -0.175 | -0.176 | -0.219 | -0.328 | 0.268          |
compression method of FPHC with a group of frames proposed in this paper is based on the facts above.

3. FPHC with group of frames

LWD data is uploaded with a certain frame format. In a logging project, according to the actual work needs, it is necessary to upload the logging data with the frame format shown in Table 4 to the ground (For the sake of confidentiality, the actual parameter name is represented by simple letters and numbers). In Table 4, the number of significant data bits of each parameter is given by the accuracy of the measuring instrument. In order to improve the efficiency of data transmission, it is necessary to compress the LWD data, which has the frame format shown in Table 4 according to the actual needs and then upload it to the ground. This paper will take the logging data in the format shown in Table 4 as an example to explain the content of the work.

3.1. Predictive coding

The coding process of predictive coding is to code the difference \( e(k) = y(k) - x(k) \), \( k = 0, 1, 2 \ldots \) of the current sampling signal \( x(k) \), \( k = 0, 1, 2 \ldots \) and predictive signal \( y(k) \), \( k = 0, 1, 2 \ldots \) which is based on the value of the previous sampled signal.

In order to meet the real-time requirements of LWD engineering and reduce the computational complexity of the compression algorithm, this paper uses the simplest predictive coding method for logging data of the same type.

Let the value of predictive signal by:

\[
y(k) = x(k-1), k = 0, 1, 2, \ldots
\]

Then the difference \( e(k) \) between the actual measured value \( x(k) \) and the predicted value \( y(k) \) is:

\[
e(k) = \begin{cases} x(0) & k = 0 \\ x(k-1) - x(k) & k = 1, 2, 3, \ldots \end{cases}
\]

The decoding process of predictive coding is to use the actual measured value of the previously sampled signal to subtract the difference between it and the current measured value.

3.2. Intraframe predictive coding and interframe predictive coding based on grouping

Previous experiments have verified that after preprocessing LWD data with predictive coding, the residual data has a stable probability distribution, and the Huffman code table generated by using this probability distribution will have a very similar compression effect on the residual data of other LWD data. This part will take the LWD data in the format shown in Table 4 as an example to elaborate on the predictive coding method used in our proposed method.

In Table 4, there are multiple B2 parameter measurement values in a frame. Therefore, B2 parameters should be coded with adjacent interframe predictive coding and adjacent intraframe predictive coding. That is, in addition to the first frame, the original value of the first B2 parameter of the previous frame is subtracted by byte from the value of the first B2 parameter of the current frame, and the residual is taken as the value of the current position. Then, for the following B2 parameters of the current frame, the original value of the previous B2 parameter adjacent to the current position is subtracted by byte from the original value of the B2 parameter of the current position. The residual value is taken as the value of the current position. As for the other data (A1, C3, D4, E5, and F6) in the frame, it is necessary to be coded with adjacent interframe predictive coding, that is, subtract the value of the corresponding position of the previous frame by byte from the value of a current position of the current frame, and the residual is taken as the data value of the current position.

If all LWD data are coded according to the above method, the correlation redundancy between the data can be fully eliminated. However, in the process of mud pulse transmission of LWD data, it is inevitable that there will be some bit errors, especially, when a data such as A1, C3, D4, E5, and F6 will be coded with predictive coding, appears error. The wrong bit will affect all such data behind it, which will lead to disastrous consequences.

In order to prevent this kind of situation, we need to adopt the method of grouping according to the number of frames to restrain the spread of the error code. Specifically, it is to group the collected data according to a fixed number of frames. We called a group of frames as Group of Frames (GOFs). Only the data within the GOF is allowed to be coded with predicted coding, and the GOFs are independent of each other. For example, when the data

---

**Table 4. Upload sequence format of LWD data.**

| Item       | A1 | B2 | C3 | B2 | D4 | B2 | E5 | B2 | F6 | B2 | Total |
|------------|----|----|----|----|----|----|----|----|----|----|-------|
| Significant data bits (Bit) | 12 | 8  | 11 | 8  | 7  | 8  | 8  | 8  | 8  | 8  | 86    |
| Bytes occupied (Byte)       | 2  | 1  | 2  | 1  | 2  | 1  | 1  | 1  | 1  | 1  | 13    |
is grouped according to six frames, and the A1 data of the second frame of the first GOF appears error code, only the subsequent A1 data of the third, fourth, fifth, and sixth frames will be affected, and the data in the seventh frame (belonging to the second GOF) and the subsequent GOFs will not be affected.

### 3.3. Huffman coding based on independent code table

The predictive coding method can reduce the correlation redundancy of the original LWD data and improve compression efficiency. However, the amount of data after the predictive coding does not decrease. In order to realize the data compression truly, it is necessary to compress the residual data after predictive coding with the entropy coding method.

In Section 2, we have shown that residual data have a stable probability distribution after doing predictive coding to the original LWD data. The Huffman code table generated from the probability distribution, can be used to compress other LWD data encoded by predictive coding. Therefore, we propose a Huffman compression method based on an independent code table. The specific method is described as follows: encode the previous logging data in the same survey area according to the method in Section 3.2, and then count the frequency of every kind of residual data. Then, obtain the probability distribution of residual data and generate a fixed Huffman tree and a Huffman code table. Then, use the Huffman code table to compress the current or the future LWD data. Because a fixed Huffman code table is obtained in advance, the newly collected LWD data can be compressed in real time, which will have a good compression effect.

Moreover, the Huffman code table is fixed, so the Huffman codewords of every symbol are independent of each other, when the Huffman codeword of one symbol appears error code in the transmission process. It will not affect the others’ Huffman codewords. Therefore, the Huffman coding method based on the independent code table has a certain error resistance.

### 3.4. The implementation of FPHC with group of frames

After the above discussion, we comprehensively consider the real-time performance and error resistance of data compression in LWD engineering and propose a method, namely, FPHC with group of frames. The implementation of the proposed coding method is divided into three parts: making code table, compression coding, and decoding and reconstruction.

#### 3.4.1. Making code table

The process of making a code table is shown in Figure 5. The specific steps of implementation are as follows:

1. Obtain a file set which the data collected at different periods in the past of a certain logging area, then open a file, and set the number (4, 5, or 6) of frames contained in a GOF. The first frame in a GOF is referred to as the reference frame, and the frames other than the reference frame are referred to as the predictive frames.
2. Read the data of a frame in the opened file. If the current frame is the reference frame in the current GOF, continue to read the data of the next frame. If the current frame is a predictive frame, encode it with adjacent intraframes predictive coding and adjacent interframe predictive coding according to the following steps:
3. Code the data of the predictive frame with intraframe predictive coding and interframe predictive coding if the data has multiple identical items in the frame (Like B2 in Table 4). First, code the first data of the item in the current frame with interframe predictive coding, that is, subtract the original value of the data at the corresponding position in the previous frame from the value of the current data by byte, and the residual value obtained is taken as the value of the data at the current position. Then, code the subsequent data of the item in the current frame with intraframe predictive coding. That is, subtract the original value of the adjacent previous data of the same item from the value of the current data by byte, and the residual value is taken as the value of the current data in the current position. Finally, reserve all the data of the item after interframe predictive coding and intraframe predictive coding, and it will be used to make Huffman code table in step (7);
4. Encode the other data which does not have multiple data of the same item in the current frame (Like A1, C3, D4, E5, and F6 in Table 4) with interframe predictive coding. That is, subtract the original value of the data at the same position of the previous frame from the value of the data in the current position by byte and take the difference as the value of the data in the current position. Keep all the data encoded by adjacent interframe predictive coding, and it will be used in step (7) to make the code table;
5. Repeat the above steps (2), (3), and (4) for the data in subsequent frames in the current file until all data in the current file are processed.
6. Get the next file in the current file set, and repeat steps (2), (3), (4), (5) until all files in the current file set are processed.
Get all the data after predictive coding, count the probability of each data value according to 2 bytes (Subtract data from two single bytes, the difference needs two bytes to save) and generate a Huffman tree based on it. Record the value of each data contained in the Huffman tree and its corresponding codeword and frequency, then form a code table which is called the Huffman code table.

Transform the Huffman code table and make a Huffman code table file. First, extract the value of each data is Huffman code table in step (7) and its corresponding codeword length, codeword, and statistical frequency. Then, suppose the codeword information consists of codeword length, codeword, and statistical frequency, then arrange the codeword information of all data from small to large according to the value of the data. Lastly, write the arranged codeword information into a file, and the Huffman code table file is generated. It should be noted that, if it is found that there is a missing integer between the minimum value and the maximum value, the missing value can be added artificially, and the frequency is set to 1. Then make the Huffman code table file according to the method above. This will prevent the coding error in the process of compression when there is a logging data whose value is not counted before.

3.4.2. Compression coding
The main idea of compressing coding is to groups the frames according to the number of frames of a GOF set in the making code table part. Then output the effective digits directly if the LWD data is in reference frames in a GOF. Then, perform adjacent intraframe predictive coding and adjacent interframe predictive coding for the data in predictive frames in the GOF, and finally perform Huffman coding for the data encoded by predictive coding and outputs the Huffman codewords. The flow of real-time data compression is shown in Figure 6.

The specific implementation steps are as follows:

(1) Obtain a collected data and save a copy, the copy will be used for predictive coding with subsequently collected data;

(2) Judge whether the currently collected data is in the reference frame, if it is, extract the data of effective digits from each data in the reference frame according to the effective data bits given by the accuracy of measuring instruments (similar to the effective digits indicated in Table 4), and then execute step (5). Otherwise, proceed to the next step;

(3) The current data are in the predicted frame. If the current data do not have multiple data of the same item in the frame, or the current data have multiple data of the same item and is the first one in the current frame, use the data and the data at the corresponding position of the previous frame for interframe predictive coding by byte, and then perform the next step. If the current data have multiple data of the same item in the frame, but it is not the first one in the current frame, then proceed to the next step.

Figure 5. Flow chart of making code table.
Figure 6. Flow chart of compressing coding.

(1) Set the default decoding state as the ‘decode reference frame data’ state.

(2) Get the compressed data and check the current decoding status. If the current decoding state is the ‘decode reference frame data’, according to the known effective digits of the measuring instrument, read the corresponding effective data bit by bit and put it into the memory. For the data less than 8bit, fill 0 in the high bit address until there is 8bits in the data, then go to the next step. If the current decoding state is ‘decode predictive frame data’, jump to step (5).

(3) Outputting the decoded data of the reference frame.

(4) Check whether all the reference frame data of the current GOF has been decoded. If not, skip to step (9). If all the reference frame data of the current GOF has been decoded, set the current decoding state to ‘decode predictive frame data’, and then jump to step (9).

(5) Decode the compressed data with Huffman decoding. If a decoded parameter data is obtained, continue with the next step. Otherwise, go back to step (2).

(6) Confirm whether the current decoded data have multiple data of the same item in the frame. If the current decoded data do not have multiple data of the identical item in the frame, or it has and it is the first one of the item in the frame, use the current data and the data at the corresponding position of the previous frame to perform intraframe predictive decoding by byte (the two value are added together), then jump to step (7). If the current decoded data has multiple data of the identical item in the frame, but it is not the first one of the same item in the frame, perform intraframe predictive decoding for the data and its
Figure 7. Flow chart of decoding and reconstruction.

adjacent previous data of the same type, and then skip to step (7);
(7) Output the current decoded data, then continue with the next step.
(8) If all the predictive frame data of the current GOF has been decoded, set the decoding state to ‘decode reference frame data’ state and jump to step (9), otherwise, jump to step (9) directly;
(9) Check whether the current measurement project is completed, if not, return to step (2) Otherwise, end the process of decoding and reconstruction.

4. Experiment and analysis

In order to verify the effectiveness of the proposed method, we conduct two experiments. The first experiment is ‘the verification experiment of practicability of FPHC with group of frames’, which validate that the proposed method has universal applicability from the actual compression effect. The second experiment is ‘the data compression experiment of LWD data’, which is used to verify the actual compression effect of the proposed compression method.

The evaluation index of compression in this part is compression ratio, and the formula is as follows:

\[ CR = \frac{S_O}{S_C} \]  

where \( S_O \) is the amount of original data, \( S_C \) is the amount of compressed data.

4.1. The verification experiment of practicability of FPHC with group of frames

Collecting two file sets of LWD data in different periods of the same survey area, which are named DS1 and DS2. The data in both file sets have the data format shown in Table 4. But it is different from the file set DS used in the previous experiment. Use the method in Section 3.4.1 to generate two code tables for DS1 and DS2, respectively, the generated code table are named T1 and T2, then use T1 and T2 to compress the data in DS1 and DS2 with the method of Section 3.4.2, respectively, and count the compression ratio \( C_R \).

The data compression results of the data in DS1 using the code table T1 and T2, are shown in Table 5.

| Number | File name | \( D_{O1} \) (Byte) | \( D_{C11} \) (Byte) | \( D_{C12} \) (Byte) | \( C_{R1} \) | \( C_{R2} \) |
|--------|-----------|----------------------|----------------------|----------------------|------------|------------|
| 1      | F001.datx | 1325                 | 926                  | 946                  | 1.43       | 1.40       |
| 2      | F002.datx | 1363                 | 908                  | 920                  | 1.50       | 1.48       |
| 3      | F003.datx | 2138                 | 1348                 | 1358                 | 1.59       | 1.57       |
| 4      | F004.datx | 3300                 | 1952                 | 1941                 | 1.69       | 1.70       |
| 5      | F005.datx | 8288                 | 6605                 | 6780                 | 1.25       | 1.22       |
| 6      | F006.datx | 8875                 | 6717                 | 6834                 | 1.32       | 1.30       |
| 7      | F007.datx | 10,200               | 7882                 | 8073                 | 1.29       | 1.26       |
| 8      | F008.datx | 10,575               | 8294                 | 8478                 | 1.28       | 1.25       |
| 9      | F009.datx | 12,013               | 8973                 | 9079                 | 1.34       | 1.32       |
| 10     | F010.datx | 12,525               | 9242                 | 9342                 | 1.36       | 1.34       |
| 11     | F011.datx | 13,550               | 9630                 | 9713                 | 1.41       | 1.40       |
| 12     | F012.datx | 13,800               | 10256                | 10357                | 1.35       | 1.33       |
| 13     | F013.datx | 17,500               | 11058                | 10968                | 1.58       | 1.59       |
| 14     | F014.datx | 18,475               | 14185                | 14421                | 1.30       | 1.28       |
| 15     | F015.datx | 34,625               | 25170                | 25675                | 1.38       | 1.35       |
| 16     | F016.datx | 36,350               | 24747                | 24690                | 1.47       | 1.47       |
| 17     | F017.datx | 60,888               | 38589                | 38111                | 1.58       | 1.60       |
| 18     | F018.datx | 64,313               | 43568                | 43670                | 1.48       | 1.47       |
| 19     | F019.datx | 67,950               | 47451                | 48065                | 1.43       | 1.41       |

The average value of compression ratio: \( 1.42 \) \( 1.41 \)
Figure 8. Comparison of effects of real-time logging data compression methods.

Table 6. Data compression results of DS2 using T1 and T2.

| Number | File name | $D_{o2}$(Byte) | $D_{c21}$(Byte) | $D_{c22}$(Byte) | $C_{r21}$ | $C_{r22}$ |
|--------|-----------|----------------|----------------|----------------|----------|----------|
| 1      | G001.datx | 6700           | 4446           | 4508           | 1.51     | 1.49     |
| 2      | G002.datx | 8288           | 6605           | 6780           | 1.25     | 1.22     |
| 3      | G003.datx | 33,050         | 22791          | 22784          | 1.45     | 1.45     |
| 4      | G004.datx | 33,350         | 24383          | 24570          | 1.37     | 1.36     |
| 5      | G005.datx | 37,400         | 24179          | 23943          | 1.55     | 1.56     |
| 6      | G006.datx | 40,238         | 25715          | 25449          | 1.56     | 1.58     |
| 7      | G007.datx | 60,888         | 38589          | 38111          | 1.58     | 1.60     |
| 8      | G008.datx | 67,475         | 43313          | 42817          | 1.56     | 1.58     |
| 9      | G009.datx | 67,975         | 40333          | 40029          | 1.69     | 1.70     |

The average value of compression ratio 1.50 1.50

DS1 with the code table T1, and there is $C_{r11} = D_{o1}/D_{c11}$. $C_{r12}$ is the compression ratio obtained by compressing the data in DS1 with the code table T2, and there is $C_{r12} = D_{o1}/D_{c12}$.

The data compression results of DS2 using T1 and T2 are shown in Table 6.

In Table 6, $D_{o2}$ is the amount of the original data in DS2, $D_{c21}$ is the amount of compressed data obtained by compressing the data in DS2 with the code table T1, $D_{c22}$ is the amount of compressed data obtained by compressing the data in DS2 with the code table T2, $C_{r21}$ is the compression ratio obtained by compressing the data in DS2 with the code table T1, and there is $C_{r22} = D_{o2}/D_{c22}$.

Finally, get the statistical results of the average compression ratio obtained by compressing the two file sets with T1 and T2, and they are as shown in Table 7.

It can be seen from Table 7 that the FPHC with a group of frames can achieve a similar compression effect for logging data of different periods in the same survey area by using the probability model of measurement data and different periods. The result proves that the proposed method is effective and reliable.

Table 7. Statistical results of the average compression ratio of file sets DS1 and DS2 compressed with T1 and T2.

| Code table | File set | DS1 | DS2 |
|------------|----------|-----|-----|
| T1         | 1.42     | 1.50|
| T2         | 1.41     | 1.50|

4.2. The data compression experiment of LWD data

In order to verify the compression efficiency of the proposed method, i.e. (FPHC with a group of frames, we use the proposed method, Huffman coding, LZW algorithm, LZW coding based on grouping predictive coding, adaptive Huffman coding, adaptive Huffman coding based on grouping predictive coding, adaptive arithmetic coding and adaptive arithmetic coding based on grouping predictive coding to compress the data set DS1. Each GOF consists of six frames. All compression methods based on
grouping prediction coding are to compress the residual data after grouping prediction coding with corresponding entropy coding methods, while the common entropy coding methods does not preprocess the data with grouping predictive coding. The code table used is generated by the method of Section 3.4.1 with the data of DS1. After testing the above methods, the average compression ratio is calculated and the results are as shown in Figure 8.

It can be seen from the Figure 8 that all the compression methods using grouping predictive coding can achieve a higher compression ratio, especially, the proposed method, i.e. FPHC with group of frames, has achieved the best compression effect. Although the compression ratio is only slightly higher than that of adaptive Huffman coding and adaptive arithmetic coding, the proposed method has the advantages of simple implementation and better error resilience. Therefore, the proposed method will be more suitable for real-time logging engineering.

5. Conclusion

Mud pulse transmission technology is a stable, reliable data transmission technology, and it is easy to implement in LWD projects. But its problem of low data rates has always existed. On the basis of fully considering the data accuracy, real-time and error resilience of data transmission while drilling, we studied the probability distribution of the original data and the residual data obtained by predictive coding to the original data and verified that the residual data has a stable probability distribution. The Huffman code table generated by the probability model of the residual data can be used to compress other LWD data in the same survey area after predictive coding. We proposed a method, i.e. FPHC with a group of frames. This method is simple to implement and can be embedded in downhole equipment without using too much hardware resources. This method does not need to obtain the probability model of the current data, so it can carry out real-time data compression and meet the needs of real-time LWD engineering. At the same time, due to the characteristics of grouping predictive coding and independent code table, this method has a certain error resilience and general applicability. After compared with LZW algorithm, adaptive Huffman coding and adaptive algorithm coding, the experimental results showed that the proposed method has higher compression efficiency. Therefore, the compression method proposed in this paper is more suitable for the LWD engineering with the needs of real time and error resilience.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported in part by National Natural Science Foundation of China [grant number 51978079], and Hubei Province Excellent young and middle-aged scientific and technological innovation team Projects [grant number T2020007].

References

Bernasconi, G., & Rampa, V. (2004). High-quality compression of MWD signals. Geophysics, 69(3), 849–858. https://doi.org/10.1190/1.1759471
Bernasconi, G., Rampa, V., Abramo, F., & Bertelli, L. (1999). Compression of downhole data. SPE/IADC drilling conference. 9-11 March.
Boddien, E., Clasen, M., & Kneis, J. (2007). Arithmetic coding revealed. McGill University. Sable Research Group www.sable.mcgill.ca
Burrows, M., & Wheeler, D. (1994). A block-sorting lossless data compression algorithm. Digital SRC Research Report.
Chao, D., Zhuang, Y., & El-Sheimy, N. (2015). An innovative MEMS-based MWD method for directional drilling. SPE/CSUR unconventional resources conference. 20-22 October.
Fang, J., Gu, S., Shi, Z., Li, Q., & Ren, J. (2019). Study and experiment on mud pulse transmission characteristics of measurement while drilling signal in coal mine. Acta Coalae Sinica, 44(11), 3604–3613. https://doi.org/10.13225/j.cnki.jccs.2018.1458
Hu, Y., Huang, Y., & Li, X. (2019). Automatic de-noising and recognition algorithm for drilling fluid pulse signal. Petroleum Exploration and Development, 46(2), 393–400. https://doi.org/10.1016/S1876-3804(19)60019-4
Huffman, D. A. (1952). A method for the construction of minimum-redundancy codes. Proceedings of the IRE, 40(9), 1098–1101. https://doi.org/10.1109/JRPROC.1952.273898
Hutin, R., Tennent, R., & Kashikar, S. (2001). New mud pulse telemetry techniques for deepwater applications and improved real-time data capabilities. SPE/IADC drilling conference. February 27-March 1.
Ivanov, A., Lakontsev, D., Fisenko, A., & Ushakov, A. (2020). Soft decision decoding in Mud pulse telemetry system. 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring). 25-28 May.
Jarrot, A., Gelman, A., & Kusuma, J. (2018). Wireless Digital Communication technologies for drilling Communication in the bits/s regime. IEEE Signal Processing Magazine, 35(2), 112–120. https://doi.org/10.1109/MSP.2017.2781288
Knuth, D. E. (1985). Dynamic huffman coding. Journal of Algorithms, 6(2), 163–180. https://doi.org/10.1016/0196-7745(85)90036-7
Li, Q. (2018). Development and application of mine mud pulse wireless MWD device. Coal Geology and Exploration, 46(06(6)), 193–197 + 202. https://doi.org/10.3969/j.issn.1001-1986.2018.06.029
Li, T., & Shi, X. (2016). Research on joint coding of mud channel data compression and error correction. Electronic Measurement Technology, 39(04(4)), 133–135. https://doi.org/10.19651/j.cnki.emt.2016.04.031
Li, Z., Cui, X., & Wu, C. (2014). Data compression protocol research of wireless while drilling measurement system. *Microcomputer Application, 30*(12), 15–17.

Liang, Y., Ju, X., Li, A., Li, C., Dai, Z., & Ma, L. (2020). The process of high-data-rate Mud pulse signal in logging while drilling system. *Mathematical Problems in Engineering, 2020*. https://doi.org/10.1155/2020/3207087

Liang, Y., Ju, X., Li, A., Li, C., Dai, Z., & Ma, L. (2020). The process of high-data-rate Mud pulse signal in logging while drilling system. *Mathematical Problems in Engineering, 2020*. https://doi.org/10.1155/2020/3207087

Liang, Y., Ju, X., Li, C., Chen, G., Zuo, X., Sun, Y., & Dai, Z. (2019). Data compression algorithm for logging while drilling based on frame-to-frame difference. *Journal of Data Acquisition & Processing, 34*(2), 297–302. https://doi.org/10.16337/j.1004-9037.2019.02.012

Lin, C., Shi, Y., & Zhang, W. (2012). Design of signal Acquisition and Processing system of acoustic logging-while-drilling tool. *Measurement & Control Technology, 31*(3), 19–23. https://doi.org/10.19708/j.ckjs.2012.03.006

Moffat, A. (2019). Huffman coding. *ACM Computing Surveys, 52*(4), 1–35. https://doi.org/10.1145/3342555

Moffat, A., Neal, R. M., & Witten, I. H. (1998). Arithmetic coding revisited. *ACM Transactions on Information Systems (TOIS), 16*(3), 256–294. https://doi.org/10.1145/290159.290162

Mwachaka, S. M., Wu, A. P., & Fu, Q. Q. (2019). A review of mud pulse telemetry signal impairments modeling and suppression methods. *Journal of Petroleum Exploration and Production Technology, 9*(1), 779–792. https://doi.org/10.1007/s13202-018-0483-y

Nelson, M., & Gailly, J.-L. (1995). *The data compression book 2nd edition*. M & T Books.

Qiu, A., Li, Y., Liu, X., & Xiao, B. (2012). Design of power supply system while drilling. *Electronic Test, (01)*, 74–77.

Qu, F., Zhang, Z., Hu, J., Xu, J., Wang, S., & Wu, Y. (2018). Adaptive dual-sensor noise cancellation method for continuous wave mud pulse telemetry. *Journal of Petroleum Science and Engineering, 162*, 386–393. https://doi.org/10.1016/j.petrol.2017.12.058

Su, Y., Qi, X., Liu, Y., & Zhang, J. (2013). Electromagnetic measurement while drilling technology based on the carrier communication principle. *Petroleum Exploration and Development, 40*(2), 242–248. https://doi.org/10.1016/S1876-3804(13)60029-4

Tu, B., Li, D., & Lin, E. (2014). Design of ground decoding system and algorithm for wireless MWD. *Journal of Beijing University of Technology, 40*(01(1)), 25–31.

Welch, T. A. (1984). A Technique for high-performance data compression. *Computer, 17*(06), 8–19. https://doi.org/10.1109/MC.1984.1659158

Zhang, Y., Qiu, Z., Xiong, K., Wang, S., & Sun, D. (2009). A new method for real-time LWD data compression. (Ed.), (Eds.). Proceedings. The 2009 International Symposium on Information Processing (ISIP 2009).

Zhang, Z., Xin, P., Fu, X., & He, H. (2015). Optimization design of management system for high efficiency power while drilling. *Acoustic Technology, 34*(06(6)), 556–561. https://doi.org/10.16300/j.cnki.1000-3630.2015.06.017

Zheng, Y., Sun, X., Chen, J., & Yue, J. (2012). Extracting pulse signals in measurement while drilling using optimum denoising methods based on the ensemble empirical mode decomposition. *Petroleum Exploration and Development, 39*(6), 798–801. https://doi.org/10.1016/S1876-3804(12)60107-4

Zhu, K., Nie, Z., & Sun, X. (2013). Numerical modeling of the signal transmission by cables and electromagnetic coupling for logging while drilling. *Acta Physica Sinica, 62*(06(6)), 16–24.

Zou, X., Feng, Z., Zhang, S., & Han, F. (2013). On the sonic logging data compression based on LZW algorithm. *Well Logging Technology, 37*(3), 294–296. https://doi.org/10.16489/j.issn.1004-1338.2013.03.019