**Lab 4：Frame Synchronization and Frequency Offset Correction**

|  |  |
| --- | --- |
| **Author** | Name：宋宜航 张皓东  Student ID:12112717 12113010 |
| **Introduction**  **Frame Synchronization and** **Frequency Offset Correction**  In the field of communication and computer science, the frame typically refers to a data block or packet in data transmission. It contains information about the transmission, such as the actual data, control information, synchronization details, and more.  Frame synchronization is a crucial aspect of communication systems, especially in scenarios where data is transmitted in frames or packets. In practice, due to propagation and signal processing delays, the location of the beginning of the frame is unknown. Therefore, the goal is to ensure that the receiver correctly identifies the boundaries of each frame and extracts the data accurately. Frames typically have a unique pattern or delimiter at the beginning or end, indicating the start or end of a frame. The synchronization process ensures that the receiver can properly interpret the transmitted frames, reducing the risk of errors in data extraction.  Frequency offset correction is essential when there is a mismatch in the carrier frequencies between the transmitter and the receiver. This mismatch can result from various factors, including oscillator inaccuracies and Doppler shifts. Once the frequency offset is determined, the receiver adjusts its local oscillator to compensate for the difference, ensuring that the received signal aligns with the intended carrier frequency. Correcting frequency offsets is crucial for maintaining the integrity of the received signal, as frequency deviations can lead to symbol misinterpretation and communication errors.    The system model is as below. The received signal should be processed by frame synchronization and then be processed through frequency synchronization.    **Training Sequence Designing and principles of Frame Synchronization and Frequency Offset Correctio**  The achievement of the above frame synchronization and frequency offset correction is mainly accomplished through a training sequence. The training sequence is a known sequence for both the transmitter and the receiver. The receiver can determine the frame position and frequency information of the signal by manipulating this training sequence. The format of a data packet is as follows, with the training sequence following the packet header and preceding the key bit sequence containing information. Therefore, by identifying the position of the training sequence, the frame position and frequency information can be determined.    Therefore, the design of training sequences is a critical step in digital communication systems, involving the transmission of specific sequences through the channel for training the receiver. Training sequences should possess sufficient uniqueness to enable accurate identification and discrimination by the receiver. This can be achieved by introducing distinctive patterns, encoding, or spectral features within the sequences. The spectral characteristics of training sequences are crucial for accurate estimation and compensation of frequency-selective fading in the channel. Appropriate spectral distribution enhances sensitivity to channel properties, aiding in precise adjustment of receiver parameters. And the length and complexity of training sequences depend on system requirements. Longer training sequences generally provide more accurate channel estimation but come with increased overhead. Complexity is contingent on system processing capabilities and complexity constraints. And the training sequences should possess two key characteristics: 1. It should have a strong autocorrection for frame detection. 2. It should have periodic structure for frequency correction.  Firstly, the training sequence requires strong autocorrelation but weak correlation with other sequences. In this way, the receiver can convolve the entire data packet with the same sequence, and the positions with strong correlation indicate the location of the training sequence, enabling frame synchronization. Many sequences exhibit such characteristics, and Barker codes are an example, as illustrated in the following diagram:    The correlation of two sequences can be represented as:    Then, the frame synchronization method is to find the location where the correlation is max, that is:    This algorithm above is also called as “the sliding correlation algorithm”.  Secondly, receiver can implement frequency offset correction using the periodic structure of the training sequence. The signal propagation process is that:    So the relationship among the transmitted signal x(t) and the received signal y(t) and he demodulated signal are:    Because the training sequence is periodical, x(t) = x(t+N), so      One way to solve the frequency offset estimation problem is to formulate and solve a least-squares problem. Because appears in the exponent, we solve a modified least squares problem. Consider the squared error:    Using the concept of liner least squares, the coefficient a is:    Since only the phase of aˆ is of interest, there is no need to compute the denominator. A simple estimate of the frequency offset is then:      This algorithm is also called as “Moose algorithm”.  **Lab results & Analysis**：  **2.1 Sliding Correlator Algorithm**  **2.1.1 Block Diagram**      First graph is the block diagram in sliding correlator algorithm, and the second picture is the proof that I use my own block while simulation.  **2.1.2 Program Process**  We can divide the into several parts, each part serves as a specific function.  Snipaste_2022-11-14_17-49-52  In this part, we want to assert zero into the original array, the number of zero equals to the zero-pad length in the modulation information.  Snipaste_2022-11-14_17-52-36  This is the key step and key block. Just as what we said previously, we should generate the training sequence in the receiver side and then make a cross correlation. And the maximum value will be the sum of all the number’s square.  Snipaste_2022-11-14_17-57-54  After cross correlation, we want to find out where is the largest energy, the blocks above provide this function. This is where the training sequence is.  Snipaste_2022-11-14_17-59-55  Last, we use Array Subset block to get the desired array, and input to a Moose.vi.  **2.1.3 Simulation Result**          The first graph is the result without correlating frequency offset, we can see the frequency offset from the constellation  The second graph is the result after correlating, we can see obviously that there is a maximum point and the x coordinate corresponding to the length of the training sequence.  The next two graph shows that the signal was received properly and the proper eye diagram.  **2.2 Moose Algorithm**  **2.2.1 Block Diagram**      First graph is the block diagram in sliding correlator algorithm, and the second picture is the proof that I use my own block while simulation.  **2.2.2 Program Process**  Since the Moose algorithm splits the training sequence after four repetitions into two periodic sequences for frequency bias estimation. So we can get the by divided 2 as below.    Then according to the equation:    Take as the interval and multiply it with its conjugate, and take the obtained result in phase. The above equation can be completed by dividing the training sequence into two arrays by Array Subset Vi. Then Conjugate one of the subarrays and multiply it with the other to find the sum.    According to the formula, the summation result is divided by 2 to obtain the frequency offset. Finally, the input is frequency shifted in the opposite direction to complete the frequency offset correction.    **2.2.3 Simulation Result**        The first image does not use frequency bias correction, the second image does. By comparison, it can be seen that the frequency offset correction is successfully completed by the moose algorithm. The calculation result of frequency offset estimation in the lower right corner also agrees with the frequency offset of our setup. The signal was successfully recovered.  **2.3 Analysis the performance of the Moose algorithm**  According to the previous analysis, the frequency bias value f:  that can be accurately estimated when estimating frequency bias using the Moose algorithm, i.e., the estimation range, is limited by the symbol rate and the length of the training sequence. To verify this conclusion, we can generate a segment of data containing only the training sequence, add tunable frequency We can generate a piece of data containing only the training sequence, input it to the Moose algorithm for frequency bias estimation, and then verify the relationship between the estimated range and the symbol rate and the length of the training sequence. To verify the relationship between the estimated range and the symbol rate and the length of the training sequence.  Here is the block diagram of the program we tested.    **2.3.2 Program Process**  First, a training sequence is generated, and the 5-bit and 11-bit Barker code sequences are repeated four times as the training sequence. This is shown below. And add a frequency offset.    The frequency offset is calculated using the same block diagram of the moose algorithm as in 2.2    Use the following method to implement the display of the constellation chart.    **2.3.2 Analysis of relationship between frequency bias angle and frequency bias**  Before exploring the factors influencing the estimated range of frequency bias, it is necessary to use a set of benchmark parameters for the estimated frequency bias range to be tested. Here we set the symbol rate Ts= 1MHz and the training sequence uses 11-bit Barker codes. Since the Moose algorithm splits the training sequence after four repetitions into two periodic sequences for frequency bias estimation, the actual window length Nt = = 22.    The three constellation plots are, in order, the original training sequence, the sequence after adding frequency bias, and the resultant plot recovered using the moose algorithm. We can see that the calculated frequency bias estimate is consistent with our settings, and the frequency bias correction is successful.  After testing, when the frequency bias f\_0 = 22728Hz, the Moose algorithm can no longer estimate the frequency bias, and its output star diagram with the estimated results is shown below. (22728Hz is the Criticality)      Based on the above tests, we obtained the actual range of the frequency bias estimate as [-22727Hz, 22727Hz]. And substituting the benchmark parameters 𝑇𝑠 = 1𝑀𝐻𝑧, 𝑁𝑡 = 22 into the theoretical range, the obtained theoretical estimate range of [-22727.27Hz, 22727.27Hz], which is consistent with the results of the benchmark test. Next we will further verify the relationship between the parameters and the estimated range by varying the input parameters.  **2.3.2.1** After changing the symbol rate to 4MHz, which is 4 times the original rate of 1MHz, we then use the benchmark The effective range of frequency bias estimation was tested by gradually adjusting the frequency bias in the test, and the test results          The left graph in Figure 4-32(b) shows the correction result when the frequency bias is 90909Hz, and the right graph shows the correction result when the frequency bias is The right graph shows the correction result when the frequency bias is 90910Hz. From the test results, it can be seen that the actual range of frequency bias estimation at this time is [-90909Hz, 90909Hz], which is the same as the calculated theoretical range [-90909.09Hz, 90909.09Hz] is basically essentially consistent, higher symbol rate increases the range of frequency bias estimation, and the symbol rate on the range of frequency bias estimation The effect of symbol rate on the range of frequency bias estimation is verified.  **2.3.2.2** By changing the training sequence to a 5-bit Barker code sequence, the length of the comparison window of Moose's algorithm is reduced from 22 bits to 10 bits. We then use the method of gradually adjusting the frequency bias in the benchmark test to test the effective range of the frequency bias estimation. We then test the effective range of the bias estimation by gradually adjusting the frequency bias in the benchmark test          the left picture is the correction result when the frequency offset is 50000Hz, and the right picture is the correction result when the frequency offset is 50001Hz. By the test result, the scale of the frequency offset estimation for [- 50000 Hz, 50000 Hz], and into the theoretical scope of parameters are calculated. It can be seen that the longer the training sequence is, the smaller the range of frequency offset estimation is. The relationship between the length of the training sequence and the range of frequency offset estimation is further verified.  Through the above benchmark test and the test after adjusting parameters, we have successfully verified that the effective range of frequency offset estimation by Moose algorithm is:    **2.4 USRP Verification**  **2.4.1 Frequency offset=0**  First, we want to verify the correction of Sliding Correlator algorithm, so first we set the frequency offset=0, then we have the result:    From the result, we can see that when the frequency offset=0, we can correctly get four points in the constellation graph, indicating that we the sliding correlator algorithm is right.  **2.4.2 Frequency offset=100**  When we do not use frequency correction:    When we use frequency correction:    From the results, we can see that the frequency offset makes the shift of the points in the constellation graph, here when we set the frequency as 100, the tail of each point will be long. However, if we use Moose algorithm, we still get bit error rate=0.  **2.4.3 Frequency offset=150**  When we do not use the frequency correction:    When we use frequency offset:    From the results, we can see that the frequency offset makes the shift of the points in the constellation graph, here when we set the frequency as 150, the tail of each point has cover a lot of space in the plane. However, if we use Moose algorithm, we still get bit error rate=0.  **2.4.4 Frequency offset=200**  When we do not use the frequency correction:    When we use frequency offset:    From the results, we can see that the frequency offset makes the shift of the points in the constellation graph, here when we set the frequency as 150, the tail of each point has cover all the space in the plane. However, if we use Moose algorithm, we still get bit error rate=0.  **2.4.5 Comparison between different frequency offset**  From the above result we can see that while the frequency offset becomes larger, the deviation angles also become large, we can see when the frequency offset=200, there is almost a circle in the constellation graph. | |
| **Experience**  **1.Screen shot on course:**            **2.Problems we meet and experience**  When we use usrp to verify the moose algorithm, we found that the constellation point shift is too small by increasing frequency from 100Hz to 200Hz.  100Hz:    150Hz:    200Hz:  According to the formula of phase degree, it is obviously to find that f\_s should be smaller to make difference dramatically.    So, we choose 20 as sample factor, 4MHz as sample rate. And we get better diagram in the section 2.4 USRP Verification.  In addition, through this experiment, we deepened our understanding of frame synchronization and frequency offset issues in digital communication. We mastered the sliding correlation algorithm for addressing frame synchronization problems and the Moorse algorithm for mitigating frequency offset issues. We are also capable of implementing these processes in LabVIEW and conducting prototype verification using USRP. Our major takeaway is that while the principles of these algorithms are relatively simple and easy to comprehend and learn, the task becomes more challenging when it comes to programming implementation in LabVIEW. There are many details to consider, and the overall framework is quite complex. Therefore, our ability to translate algorithms into programming code is an area where we need to enhance our skills.  **3.Contribution distribution**  We two finish the whole LabVIEW program together. Song Yihang complete procedure design including sliding correlation and Moose algorithm, performance analysis of Moose algorithm, and validate the effectiveness of the sliding correlation algorithm and the moose algorithm using USRP. The introduction of basic principle of frame synchronization and frequency offset correction, and the principle and details of sliding correlation and Moose algorithm were elaborated by Zhang Haodong. | |
| **Score** | 98 |