Square Wave Demodulation of Frequencies in Depth Data

by Ben Johnson

Advisor: Lucas Koerner

Problem Statement:

A time-of-flight sensor measures the arrival of individual photons at a picosecond resolution. This generates a depth signal when combined with pulses from a laser. Changes in depth data indicate object motion or vibrations. What are the different ways that this data can be processed to extract frequencies? A Fourier transform extracts the amplitudes of the frequencies in a time series. However, it requires too much computational power to be implemented at each pixel of a time-of-flight sensor. We are developing a method that uses less computational power. A Fourier transform finds the coefficients of frequencies in a sequence of finite length (aperiodic). These are analogous to the coefficients of an infinite Fourier series. A Fast Fourier Transform (FFT) performs this operation on a discrete time series. It does this by multiplying the depth data by sine waves of various frequencies, then integrating these multiplied waves. Instead of multiplying by sine waves, square waves may be used to sample the incoming data. Square wave sampling is easily implemented by an electronic device using a switch (on-off). Working with square waves is less computationally intensive than working with sine waves. Here, we consider the on portion of a square wave as a bucket. The depth data is discrete and is "modulated" by the frequencies at which an object vibrates. Multiplying the depth data and square waves together will select samples from the positive portion of the frequencies present in the depth data. These samples can be summed together, and the value of this sum will be highest when the period of the square wave matches the period of the frequency. The ability to measure the vibration frequencies of an object is useful when monitoring the wear and tear of industrial components, and when monitoring the body of a patient while using medical sensors.

Previous work:

Previous papers that we found during our research measured the depth of an object by observing the change in the phase of a square wave or sine wave that bounced off an object. They did not attempt to measure the vibration frequencies of the object. However, they did deal with similar problems to what we dealt with. To deal with the problem of phase misalignment, adding a 90º phase square wave was done in one of the papers [A]. The offset of the depth data can be dealt with by adding 2 more waves with 180º and 270º phases [B]. One method of removing the influence of subharmonics that was used is “equivalent waveform shaping,” either emulating a trapezoidal demodulation wave [E], or a low-resolution sine wave [C][D]. There were also other methods of dealing with subharmonics that we could try besides waveform shaping, including modifications to the duty cycle and sampling rate [F].

Definitions:

TOF: An acronym for Time of Flight, describing a sensor that measures depth using the time it takes for a photon to bounce off an object and return to the sensor.

Nyquist frequency: The highest frequency that can be accurately measured in a signal, which depends on the sampling rate. A sampling rate of 0.1 seconds results in a Nyquist frequency of 5Hz.

Odd Harmonics: Frequencies that are odd multiples of a given frequency. For a frequency of 50Hz, the third harmonic is 150Hz and the fifth harmonic is 250Hz.

Subharmonics: A subharmonic of a frequency occurs at any odd fraction of that frequency. A frequency of 105Hz has subharmonics at 35Hz, 21Hz, 15Hz, (105/9)Hz, (105/11)Hz, and so on.

Fourier series: Any periodic function can be decomposed into a series of sine and cosine waves of different frequencies. A triangle wave can be expressed as a series of cosine waves with odd frequencies.

Modulation: This is when information is added to a wave, either through changes in the amplitude, phase, or frequency of the wave. Depth measurements are modulated (a frequency component is added) by object vibration.

Demodulation: Information is extracted from the depth data, finding the amplitudes and frequencies of the signals that modulate the depth data. Frequencies of 2Hz, 3Hz, and 5Hz may be measured in the data for example.

"Bucket": The peak of a square wave, which controls when samples will be added to the sum. For any frequency, the matching bucket size is half the period, and samples inside the bucket will be summed.

Target Frequency: This is the frequency which is having its amplitude measured. When the length of the bucket is half a second, then the target frequency is 1Hz.

Aliasing: This occurs when a frequency in a signal is measured incorrectly due to the under sampling of data. A frequency of 3Hz may appear identical to a frequency of 1Hz if there are not enough samples.

Research Goals:

Traditional methods of extracting the amplitudes of the frequencies present in depth data require too many computational resources to process the high-bandwidth data from TOF sensors. Therefore, we explored new methods to extract frequencies with lower computational requirements. To achieve this goal, we studied the effects of demodulating depth data with square waves. The depth data has qualities that may negatively affect the frequency measurement, such as an offset that obscures frequency amplitudes, and a misalignment of the phase of the depth data with the phase of the demodulating wave. To remedy these effects, alterations to the demodulating waves and amplitude calculations must be made. An additional challenge is that square wave demodulation produces false amplitude measurements at subharmonics of the actual data [C]. Many different methods may be used to combat this, most of which are centered around mimicking a sine wave with the demodulation method. While simulated depth data may contain clean frequencies, the measurement environment adds Gaussian noise to the depth data. We will mathematically describe the effects of this noise on our frequency measurements, and we will alter

our methods to counteract it. Once the dominant frequencies can be detected with the correct amplitudes while avoiding false measurements, there is an additional challenge of creating efficient search algorithms that reduce computational demands by minimizing the number of measurements required. We will explore algorithms that detect the general area of a peak by sampling at a lower frequency resolution, and then find the precise position of this peak by sampling at a higher frequency resolution. Finally, after developing a simulation of depth data demodulation, each step of this process will be converted into a component of an electrical circuit. Square waves will be created using a signal generator and multiplied with depth data using a mixer.

Results and Methodology:

Description of Our Approach:

The general approach that we are taking starts with discrete depth data that contains various frequencies. This depth data is then multiplied by a square wave, with values of one and zero. We refer to the peak of this square wave as a "bucket." By multiplying the two waves together, the samples that are inside the buckets will be selected, and these samples can be summed together. This sum is greatest when the length of the buckets is half the period of a frequency in the data. Using this approach, we have devised two slightly different methods that can be used to demodulate the frequencies present in the depth data.

Fixed Timestep Buckets:

The first method specifies a certain sampling rate at which depth samples will be taken. Then a width, w, is specified. Following this, "odd" and "even" buckets are constructed. The first w samples will be collected in an odd bucket, and the second w samples will be collected in an even bucket. This is repeated for the full length of the depth data. All the samples inside the odd buckets will be summed together, and the same is done for the even buckets. Then the absolute values of the odd and even bucket sums are added together. This process is repeated for many different bucket widths while keeping the sampling rate fixed, as shown in Figure 1 in the Appendix. The absolute value of the sum (the amplitude) for each bucket width can be plotted, as shown in Figure 2. To convert from bucket widths to frequencies, the width of the bucket must be calculated by multiplying the variable w by the timestep. This is equal to half the period of the target frequency. Then we double this width and take the reciprocal using the formula in Figure 3 to convert to frequency. When plotting the amplitudes after converting from bucket widths to frequencies, the high amplitude at the main frequency can be seen in Figure 4, as well as the subharmonics, which occur at 1/nth the main frequency with 1/nth the amplitude for all odd denominators.

Variable Timestep Buckets:

While the first method can measure high amplitudes at the correct frequencies, the square wave generation with this method is difficult. The square waves that are generated are discrete, so the first method generates square waves with varied numbers of samples per bucket. These numbers are difficult to track, especially when examining and altering the mathematical behavior of this method, so it would be more convenient to have a single number of samples per bucket. Additionally, the difference between any two target frequencies must be a multiple of the sampling rate dt, and this bounds the resolution of the

frequency sampling. A more flexible sample and bucket generation method would fix this problem, allowing for uniform frequency resolution. The second method fixes the problems from the first method by changing the process of setting the bucket width. A fixed number of samples per bucket is chosen, four for example, as in Figure 5, and the width of the bucket is varied by changing the timestep. Only the "odd" buckets are summed together. The square waves that are generated using the second method are easier to manipulate and keep track of, which will allow us to modify this method to solve problems that cause incorrect amplitude measurements.

Effects of Phase:

If the square wave peaks are aligned with the positive portion of the frequency, the correct amplitude will be measured. However, if the phase of the two waves is not properly aligned, then this measurement will be incorrect. In the worst-case scenario, the phase alignment will be off by 90º, resulting in positive and negative samples that cancel out when summed together like in Figure 6. Depending on the phase alignment, the amplitude will vary, as shown in Figure 7. To ensure that the correct amplitude is measured regardless of the difference in phase, we create a new square wave visible in Figure 8 with a phase of 90º [A]. This wave is also multiplied by the depth data, and then the two multiplied waves are combined using the Pythagorean theorem. Any frequency can be thought of as having a sine and cosine component on a unit circle, like in Figure 9, with these two components having a phase difference of 90º. These two components can be combined using the Pythagorean theorem in Figure 10 to find the amplitude of the frequency. After this change, the phase difference between the square waves and the depth data will have no effect on the amplitude measurements.

$$Amplitude = \sqrt{Q_1^2 + Q_3^2}$$

Figure 10

The amplitude formula combines the 0º wave and the 90º wave using the Pythagorean theorem.

Effects of Offset:

Until now, it was assumed that every depth signal had an average value of zero. However, the object that is being scanned for vibrations will sit at a certain distance from the sensor, and this distance will be the average value of the depth signal, or the offset. This offset is also summed and will obscure frequency measurements, so the offset must be removed for correct results. To remove this offset, two new square waves shown in Figure 11 are created, with phases of 180º and 270º [B]. We update our amplitude formula to subtract the pairs waves which are offset by 180º from each other, and then use these results as the components in the Pythagorean theorem formula shown in Figure 12. This subtracts the offset while leaving the frequencies behind, as demonstrated in Figure 13. This will completely

remove all the effects of the offset on the frequency measurements.

$$Amplitude = \sqrt{(Q1 - Q2)^2 + (Q4 - Q3)^2}$$

Figure 12

In the amplitude formula, the 180º and 270º waves are subtracted from the 0º and 90º waves to remove the offset.

Mathematical Description of Variable Timestep Buckets:

So far, we have added two major modifications to the variable timestep method, that have removed the influence of phase and offset. We now discuss the mathematics of the method up to this point. Assume that our depth data visible in Figure 14 has only a single frequency, F_d. Then we create four square waves visible in Figure 15 expressed as Fourier series, which have distinct phases and a specified frequency of F_s, the target frequency being measured. Using the sine multiplication identities found in Figure 16, the depth signal can be multiplied by these four square waves, creating four new Fourier series with these identities present in their components, an example of which is present in Figure 17. When the depth data and square waves are multiplied, as in Figure 18, the portion of the depth data where the square wave is one (the bucket) is left unaltered, while the portion where the square wave is zero is set to zero. Then the four multiplied waves in Figure 19 will be integrated from zero to an arbitrary value I. In reality, the wave is discrete, not continuous, so summation will be used instead of integration. Finally, these four summations are plugged into the amplitude formula shown in Figure 20. This process is repeated with many different values of F_s to measure the frequencies present in the depth signal.

$$DS_1 = \frac{1}{2}\sin\left(F_d(2\pi x)\right) + \sum_{n=1}^{100} \frac{1}{(2n-1)\pi}\left(\cos\left(\left(F_d - \left(F_s(2n-1)\right)\right)(2\pi x)\right) - \cos\left(\left(F_d + \left(F_s(2n-1)\right)\right)(2\pi x)\right)\right.$$

Figure 17

Multiplying the depth data by one of the square waves will result in a new Fourier series with the sine multiplication identity inside.

$$Q_1 = \int_0^I DS_1 dx \quad Q_2 = \int_0^I DS_2 dx \quad Q_3 = \int_0^I DS_3 dx \quad Q_4 = \int_0^I DS_4 dx$$

Figure 19

The four multiplied waves are integrated over a range specified by the variable I.

Amplitude Scaling:

With the problems from offset and phase removed, the frequency amplitudes will now have the correct ratios when compared to each other. However, these amplitudes will not have the correct measurements, instead having the improperly scaled measurements seen in Figure 21. The integration time is linearly proportional to the value of the (incorrectly scaled) amplitude at each frequency. An integration time of three seconds will measure a value three times higher than an integration time of one second, as shown in Figure 22. To remove this influence, all measurements must be divided by the integration time. Secondly, the frequency being measured is linearly proportional to the value of the measurement. Given an integration time of one second while sampling three times per bucket, a signal with a frequency of 1Hz will only have one bucket summed, with three samples in total. A frequency of 3Hz will have three buckets summed over the same length of time, with nine samples in total, as illustrated in Figure 23. By dividing the value by the target frequency, this influence can be removed. Finally, there is the influence of the number of samples per bucket on the amplitude measurement. A higher number of samples per bucket will result in a higher (incorrect) amplitude, multiplying the real amplitude by a scaling factor. This factor can be determined by taking evenly spaced samples (the same number as the number of samples per bucket) on the positive half of a period of a sine wave with an amplitude of one. We determined that the sum of these samples can determine the scaling factor for an even number of samples per bucket, as is demonstrated in Figure 24. Odd numbers of samples per bucket were not necessary for our method to work. The scaling factor can be thought of as the area of a sine wave integrated from zero to pi, but with a finite number of samples instead of an integral. Dividing by the scaling factor will remove the influence of the number of samples per bucket on the amplitude measurement. After dividing by these three values, the amplitude measurements will be the correct ones visible in Figure 25. However, there are subharmonics that give false frequency measurements, that are not immediately distinguishable from real measurements. There are various methods to suppress these subharmonics.

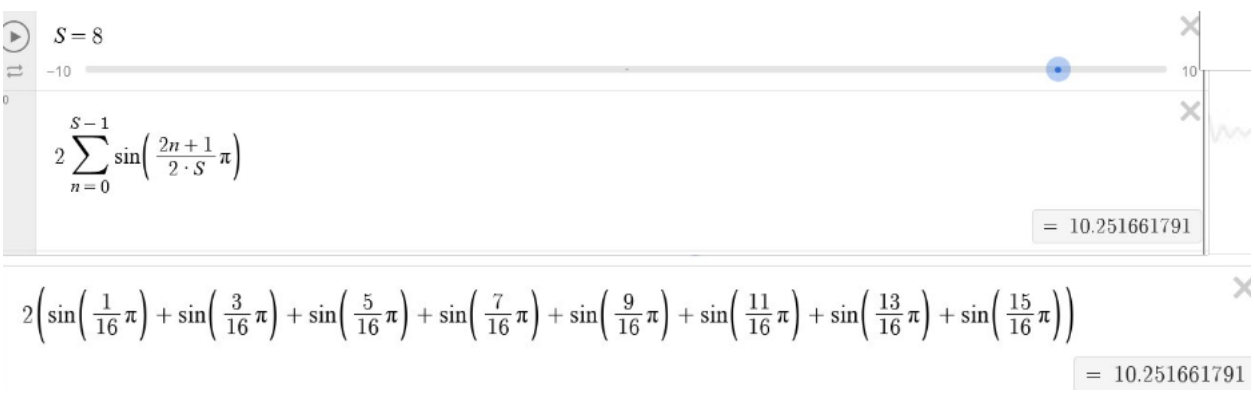

Figure 24

By taking evenly spaced samples on half the period of a sine wave and summing them, the scaling factor can be found for any number of samples per bucket.

Subharmonics:

Subharmonics show up since we are demodulating depth data using a square wave rather than a sine wave [C]. Figure 26 exhibits how a square wave can be written as a Fourier series, a sum of sine waves with odd frequencies that have matching odd denominators in the coefficients. This square wave is multiplied by the depth data. For a simple example, consider depth data with only one frequency, a sine wave. Then we will multiply this sine wave by the multiple other sine waves in the square wave and

integrate these multiplied waves. First, we use the sine multiplication identity to find that the multiplication of two sine waves can be expressed as the difference of two cosine waves. Then each of the components of the square wave can be multiplied with this identity to express the multiplication of the depth and square wave as a Fourier series, as shown in Figure 27. Then this Fourier series with cosine components will be integrated. When a cosine wave is integrated over a long length, it will measure a value of zero, unless the frequency of that cosine wave is zero. Then its value will be one, and it will be integrated as a constant, giving a strong amplitude measurement. The first cosine will be zero when F_d matches F_s, as in Figure 28. This will result in the coefficient of the first component of the square wave, which is one, being measured as the amplitude of the main frequency. When F_s is a third of the main frequency, the first cosine will be zero again since F_s is now multiplied by three. The coefficient of the next component of the square wave has a value of a third, resulting in the third subharmonic having an amplitude that is a third of the main frequency's size. This same process occurs at a fifth of the frequency, a seventh of the frequency, and for every odd fraction of the original frequency.

An alternate way of explaining the origin of these subharmonics is to examine the aliasing which occurs during their measurement. A given frequency, F_0, will alias onto all frequencies that are odd fractions of it. The nth subharmonic of F_0 is located at F_0/n. Some of the samples are perfectly aligned so that the subharmonic frequencies have samples taken that appear identical to the samples of F_0 but are stretched out in time. If this does not happen, then there will still be a pattern of samples present that will result in an amplitude being measured where there should be nothing. With four samples per bucket, two patterns are observable in Figure 29. A wave pattern is sampled at $1/9^{th}$ of F_0, which results in an amplitude measurement identical to F_0. The second pattern is a sort of tooth-shaped pattern sampled at 1/3rd of F_0, which results in an amplitude measurement that is half of that of F_0. The aliasing phenomenon is highly dependent on the number of samples per bucket, and a higher number of samples will produce sampling patterns that approach amplitudes that are 1/nth the size of the main amplitude for the nth subharmonic.

Suppressing Subharmonics:

To remove the influence of these subharmonics, there are multiple methods that could be pursued, so we only chose a few of them. One immediately apparent method to reduce the amplitude of these subharmonics is to increase the number of samples per bucket [F]. With a small number of samples per bucket, four for example, the amplitudes of all the subharmonics will be comparable to the actual frequency. However, they will match the sizes described by the mathematical theory, 1/nth the size of the main frequency for the nth subharmonic, once the number of samples per bucket is increased to a larger number, such as the 32 samples per bucket displayed in Figure 30. The second method we employed to reduce the amplitude of these subharmonics was to introduce a phase shift into the square waves [C]. When multiplied with the depth data, this will produce a result that is equivalent to a waveform closer to a sine wave, as can be seen in Figure 31. The resolution of this wave can be increased to approximate a sine wave to any level desired, but is limited by the sampling rate, imposing obstacles for small levels of phase shift. We shifted each of the square waves using the $1{:}\sqrt{2}{:}1$ ratio illustrated in Figure 31 to split the square wave into three portions, where the portions with a length of one were shifted to the left or right by 45º each. The effect that this phase shift will have on the amplitude measurements at the main frequency and the subharmonics is visualized in Figure 32 using vectors on a unit circle. The two portions of the square wave being shifted by 45º will result in a smaller amplitude measurement for the main frequency than without the subharmonic rejection activated. The phase shift of 45º is multiplied by the number of the

subharmonic, so it is 135º for the third subharmonic and 225º for the fifth. For these two subharmonics, the real and imaginary components of the measurements will cancel out, eliminating their influence. This process can be repeated to suppress any number of subharmonics. To suppress the kth subharmonic, there are mathematical formulas shown in Figure 33 that can determine the number of segments to shift (n), the size of the phase shift (p), and the weight of each segment (a_L) [C]. The results of this method are promising. Turning the suppression on will completely remove the third and fifth subharmonics, as seen in Figure 34. However, there are still artifacts present around the subharmonics. When comparing the results with the subharmonic suppression turned on and off, shown in Figure 35, there are two noticeable features. Firstly, the amplitude of the main frequency at thirty is smaller with the suppression turned on, as expected. Secondly, even though there are still artifacts around the subharmonics, these artifacts are smaller with the subharmonic suppression turned on, when the ratio of the subharmonics to the main frequency is considered.

$$k = 5 \quad n = \frac{(k+1)}{2} \quad p = \frac{180}{(n+1)} \quad a_L = \sin\left(\frac{L\pi}{n+1}\right) \quad L = 3$$

Figure 33

Formulas are used to calculate the phase shift necessary to suppress a given subharmonic k. The number of segments, the size of the shift, and the length of each segment each have their own formula.

Description of Code:

The code for the second method shown in Figure 36 was written in Python using the Numpy, SciPy, and Matplotlib libraries. The code starts with a section of control parameters that set the behavior of the program. The bucket_samples variable controls the number of samples taken during the peaks of the square wave. The duty_cycle variable controls the percentage of time that the peak of the square wave equals one. The default value is 0.5, with half the values of the square wave equaling one. The supharm variable turns the phase shift that suppresses subharmonics on and off. Offset and noise control the offset of the depth data and the level of random noise that is present in the data (uniform distribution currently, gaussian in the future). The Min_freq and Max_freq variables control the range of target frequencies that the program will measure in the depth data. Num_T controls the number of target frequencies that will be measured by the program, in between the two limits. Max_time is the integration/generation time for the depth data in seconds. A Max_time of 10 will generate depth data from zero to ten seconds. The whole signal will be multiplied and integrated.

A for loop runs for the number of specified target frequencies, Num_T. The difference between each of these target frequencies is calculated, and the target frequency is iterated at the beginning of each run of the loop by this difference. The T variable is calculated based on the target frequency to be half the period of the square wave of that frequency. Then the T_sub variable is calculated to be the t variable divided by the number of samples per bucket. A depth signal is then generated by sampling a depth function at the rate specified by T_sub and the length specified by Max_time. Then the four square waves are generated. A pattern of ones and zeros based on the number of samples per bucket is generated, and this pattern is repeated to match the length of the depth data. This pattern is then shifted three times to generate the other four square waves. These waves can be plotted to see which samples of the depth data are being summed, a process which is visible in Figure 37.

After the waves have been generated, there is an option to add a phase shift to them using the subharmonic rejection function. This will split the square waves into three parts according to the ratio specified by the shift_rat variable and shift the end portions left or right by 45º each. Then the q1-q4 waveforms are set to equal the depth data multiplied by each of the four square waves. These waveforms are then summed together, used to calculate the amplitude, and then the loop repeats for the next target frequency to measure. After all frequencies have been measured, the measurements are divided by max time, the frequencies, and the scaling factor to correctly scale each amplitude. These results are then plotted according to the settings specified by the markers and semi log variable, as shown in Figure 38.

Future Directions:

The influence of the subharmonics may be successfully removed by shifting the phase of the square wave in a specific ratio. As a next step, we could use the equations that specify the phase shift ratio to suppress more subharmonics. To do this properly, we would examine the inner workings of these equations, which we have not done in depth yet. However, there are other methods that could be further explored to find their advantages and disadvantages compared to the suppression method we are currently using. One direction that could be further explored is to change the duty cycle of the square wave [F]. By reducing the duty cycle from 50% to 29%, the amplitude of the third subharmonic can be reduced. While this method initially appears less promising than shifting the phase of the square wave, this could change with further exploration. A second method that could be used is a look up table to remove the influence of the subharmonics. Since the value of the subharmonics can be modeled ahead of time, the simulated data could be used to apply an offset to the real measurements to remove the influence of the subharmonics. However, it might magnify other sources of error. Another option is to measure the effects of multi-path interference. By measuring all the harmonics that alias onto the target frequency at the same time, their contributions can be calculated and removed. This may be limited by the fact that only a finite number of harmonics for a given frequency can realistically be measured. Finally, another method of removing subharmonics may be to use bitwise multiplication to emulate a sine wave. Doubling a value is a multiplication operation that computers can perform very quickly. This could be used to generate a demodulation wave form that is like a sine wave, as shown in Figure 39. These methods could be combined, but that is not something we have explored yet.

We implemented a basic search algorithm for finding frequencies which has room for further improvement. However, this method requires the storage of the general locations of the frequencies (peaks in amplitude) in memory so they can be more precisely determined later, adding additional complexity to our electrical circuit. Another approach would be to detect the general location of an amplitude peak, then increase the resolution of the frequency sampling to determine its precise location right away. This could potentially be done by detecting when the measurements are approaching a peak, or whether the measurements have passed a peak, and then increasing the sampling rate. The version of this algorithm that detects the peak after it has been passed would require reversing the direction of the sampling to go back and find the precise location of the peak. During the testing of this method, the initial frequency resolution and all subsequent increases in resolution would be set to be as low as possible to maximize the sampling speed while still detecting all the amplitude peaks.

We did not reach the point in our project where we were able to explore the effects of gaussian noise on our signals. We did simulate this at a basic level, as shown in Figure 40. From the literature that we read, we understand that certain subharmonic suppression methods may reduce the amplitude

measurement of the main frequency, not just the subharmonics of that frequency [C][D]. Therefore, the effects of noise on the main amplitude measurement will be amplified. One method to deal with this problem may be to separate the detection and measurement of the frequencies. A sweep of all the frequencies in the targeted range may be conducted with subharmonic suppression activated to find all the frequencies in the depth data, which can be simultaneously measured with subharmonic suppression turned off to obtain a more accurate amplitude measurement.

We have not tried to implement our method as an electrical circuit yet. It will most likely be similar to the system found in Figure 41. Signal generators will be used to generate square waves. Mixers, or simply sampling switches, will multiply the depth data by these square waves. If we emulate a sine wave using square waves with amplitudes scaled by powers-of-two, we can use bit shifts for multiplication as opposed to general purpose multipliers. We will need to write the computer program that will control this hardware. It is likely that we will encounter unforeseen issues when constructing this circuit and will have to alter our method.

Bibliography:

[A]Orozco, Luis. “Synchronous Detectors Facilitate Precision Low-Level Measurements.” Analog Dialogue, Nov. 2014, https://www.analog.com/en/resources/analog-dialogue/articles/synchronous-detectors-facilitate-precision.html. Accessed 1 Aug. 2024.

[B]“Time-of-Flight Basics.” Melexis.Com, Melexis, Aug. 2022, www.melexis.com/en/documents/documentation/application-notes/application-note-time-of-flight-basics.

[C]Payne, Andrew & Dorrington, Adrian & Cree, Michael & Carnegie, Dale. (2010). Improved measurement linearity and precision for AMCW time-of-flight range imaging cameras. Applied optics. 49. 4392-403. 10.1364/AO.49.004392. https://www.researchgate.net/publication/45583875_Improved_measurement_linearity_and_precision_for_AMCW_time-of-flight_range_imaging_cameras

[D]Payne, Andrew & Dorrington, Adrian & Cree, Michael & Carnegie, Dale. (2008). Improved linearity using harmonic error rejection in a full-field range imaging system. Proceedings of SPIE - The International Society for Optical Engineering. 6805. 10.1117/12.765930. https://www.researchgate.net/publication/43294141_Improved_linearity_using_harmonic_error_rejection_in_a_full-field_range_imaging_system

[E]Keel, Min-Sun, et al. “Wiggling Error Self-Calibration for Indirect ToF Image Sensors.” Imagesensors.Org, 2021 International Image Sensor Workshop, 20 Sept. 2021, https://imagesensors.org/Past%20Workshops/2021%20Workshop/2021%20Papers/R21.pdf

[F]Whyte, Refael. “Harmonic Aliasing (Wiggle Error) in Indirect Time-of-Flight Depth Cameras.” Medium.Com, Chronoptics Time-of-Flight, 17 Sept. 2020, https://medium.com/chronoptics-time-of-flight/harmonic-aliasing-wiggle-error-in-indirect-time-of-flight-depth-cameras-efa1632d4d1b. Accessed 10 Aug. 2024.

Appendix:

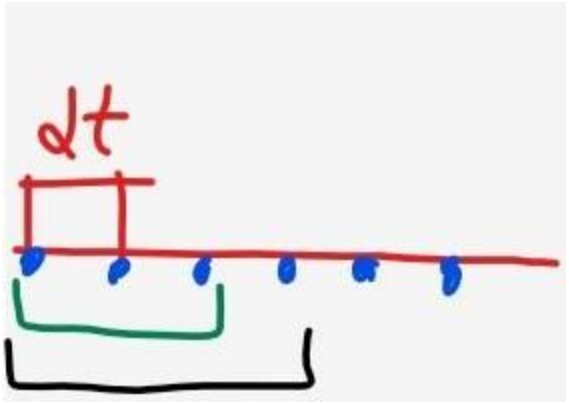

Figure 1

With a fixed timestep (dt), the width of the bucket depends on the number of samples in that bucket.

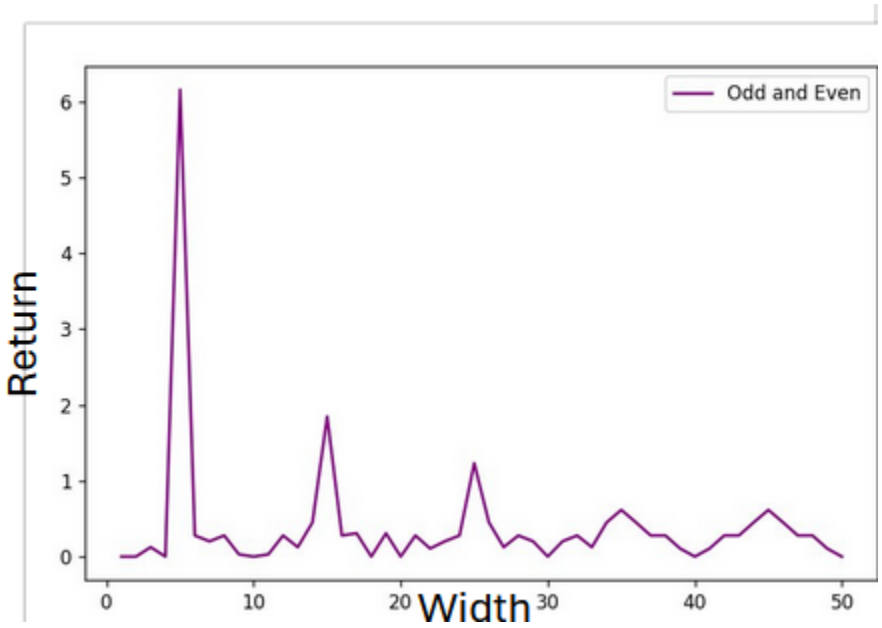

Figure 2

Each bucket width will result in a different sum. The largest sums occur when the bucket is half the period of the actual frequency or its subharmonics.

```
freq_x = [1/(w*2*T) for w in freq_x]
```

Figure 3

To read which frequencies have the highest amplitudes, the widths of the buckets (the samples w times the timestep T) must be converted to the frequencies they match with using this formula.

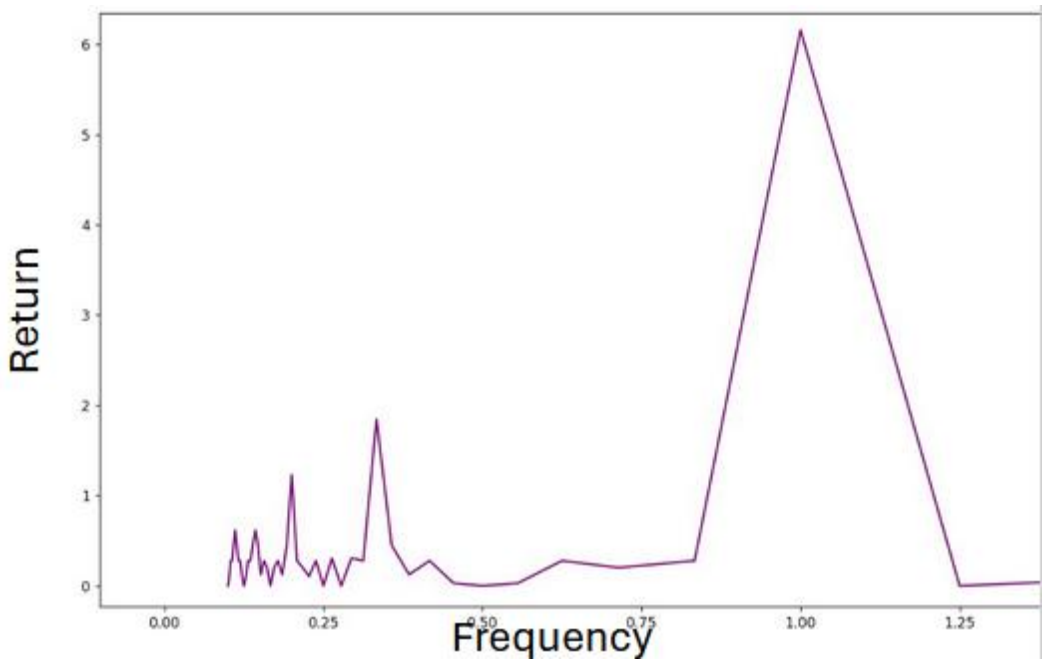

Figure 4

After converting from timesteps to frequencies, the sum is largest at the main frequency of 1Hz and its subharmonics.

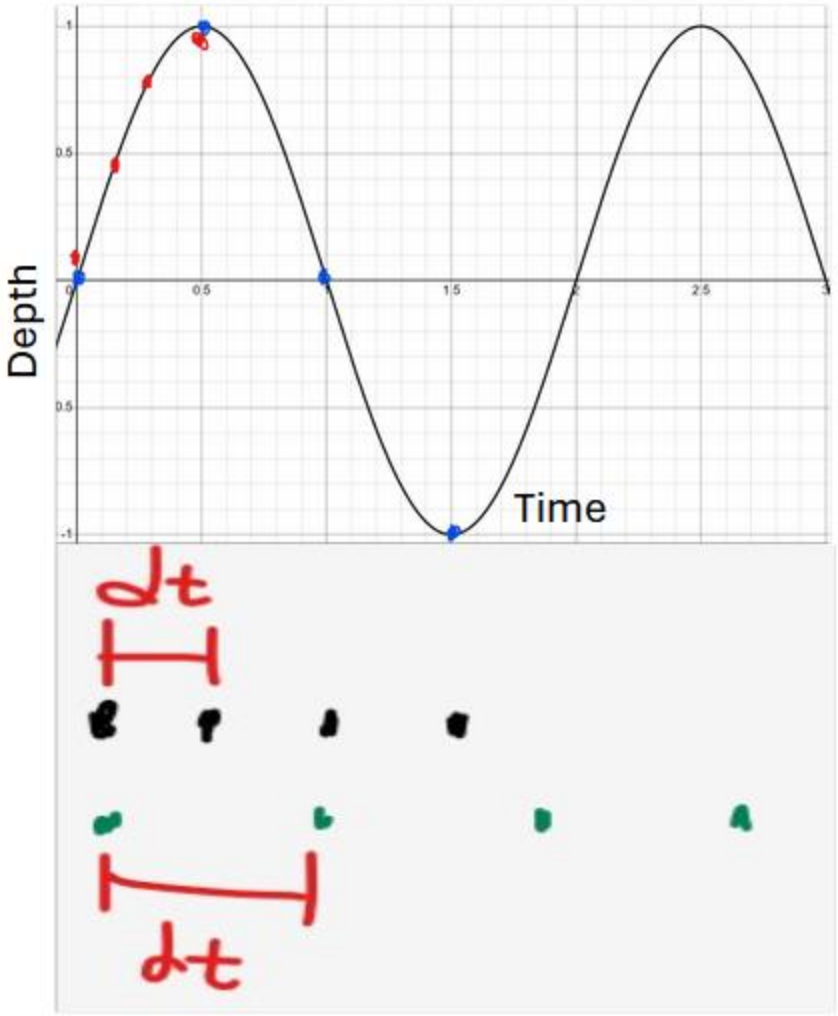

Figure 5

With variable timestep buckets, a number of samples per bucket is chosen (4 in this case), and the timestep is changed to vary the width of the bucket.

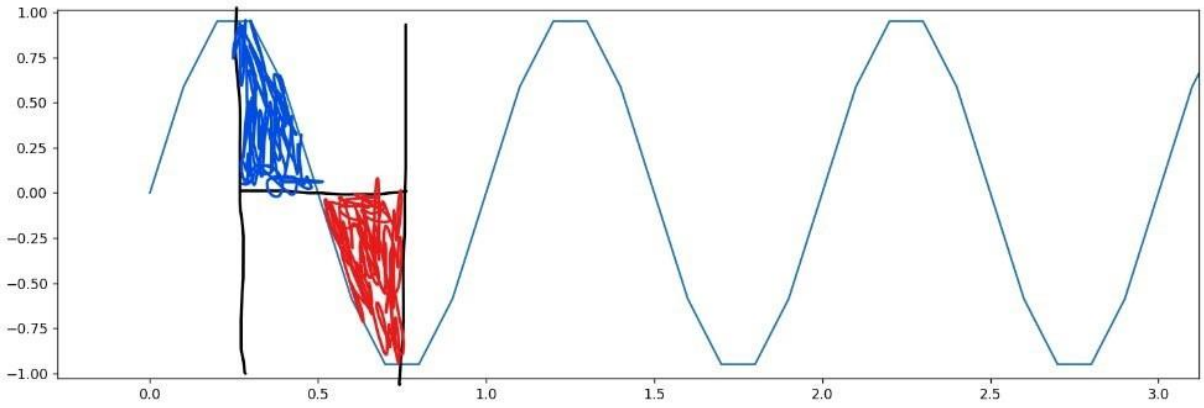

Figure 6

When the phase of the square wave is misaligned by 90º, the positive (blue) samples cancel out negative (red) samples, and the measurement of the amplitude is zero.

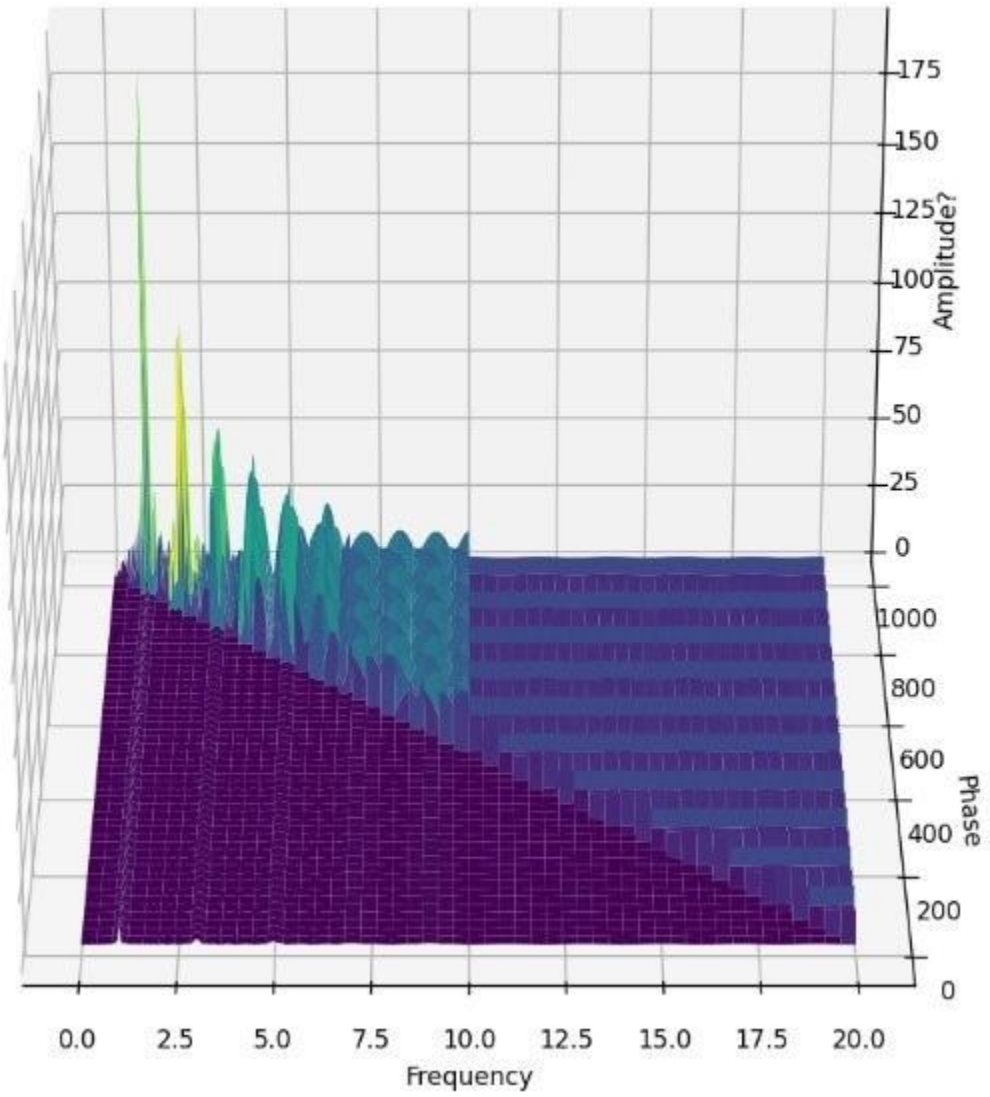

Figure 7

When changing the phase offset of the square wave, the measured amplitude will vary.

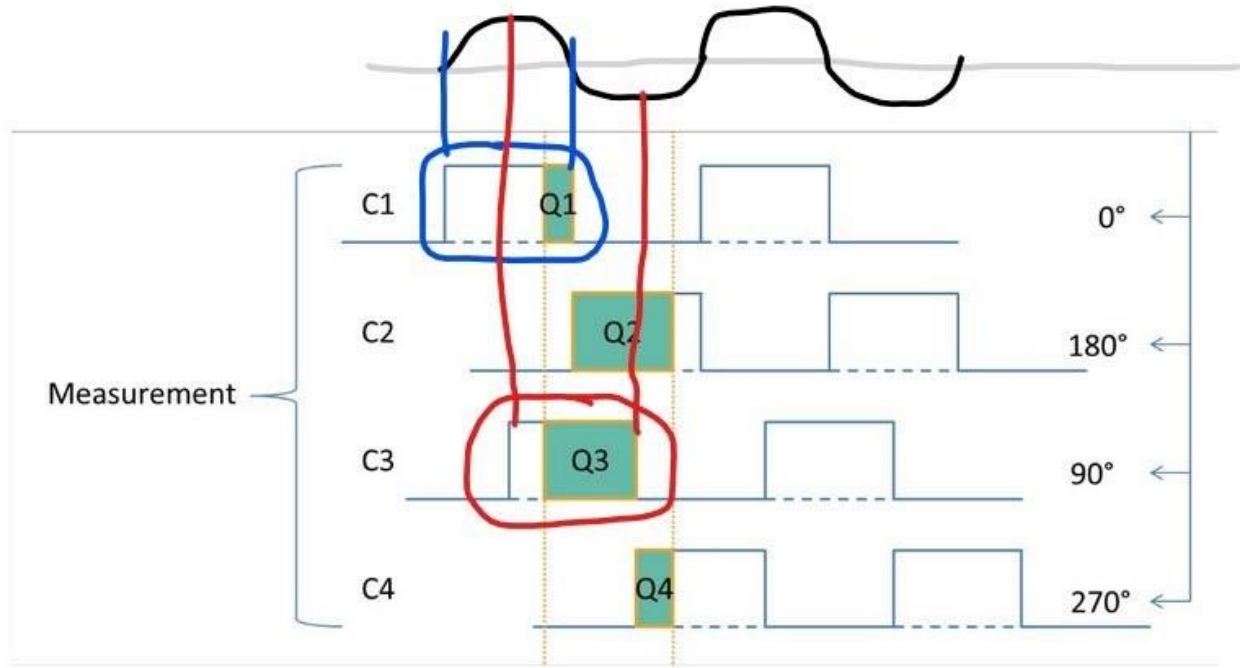

Figure 8

Creating a new 90º phase wave will sample a different part of the depth data. [B]

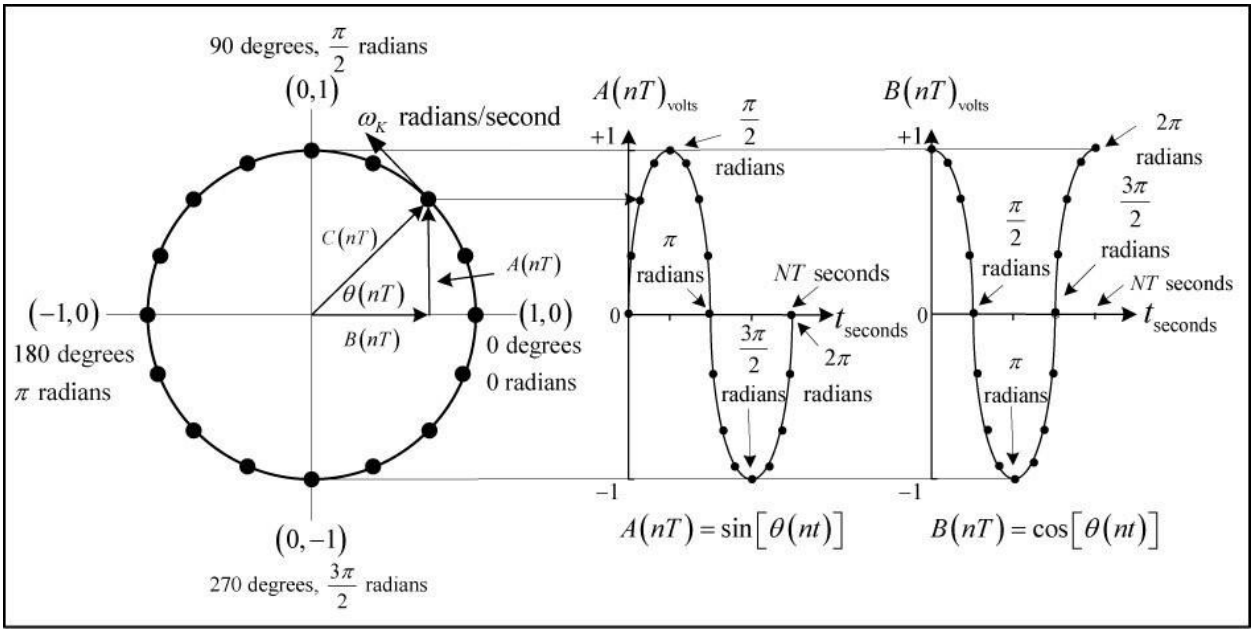

Figure 9

The 0º phase waves and 90º phase wave measurements are analogous to the sine and cosine components of a signal visualized on a unit circle, so they can be combined with the Pythagorean theorem to find the correct amplitude.

https://www.informit.com/articles/article.aspx?p=1967020&seqNum=2

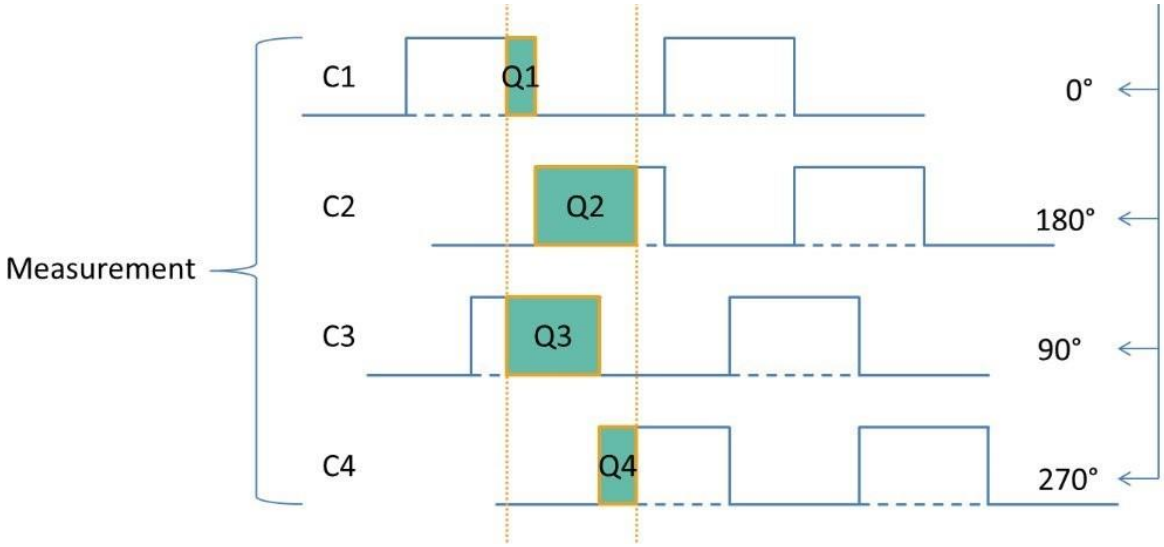

Figure 11

Two new square waves have been created with phases 180º and 270º, so there are now four in total. [B]

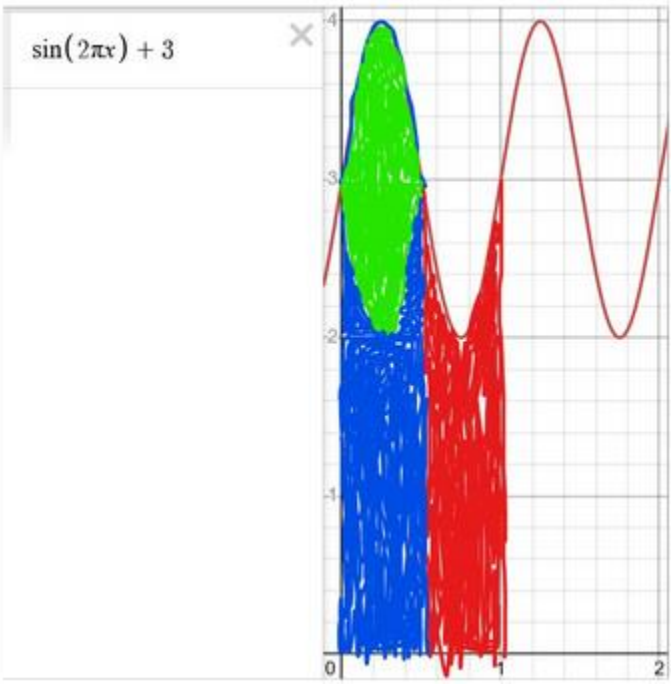

Figure 13

The red square wave is offset by 180º from the blue square wave, and subtracting the red samples from the blue samples removes the offset, leaving the green area behind as a measurement.

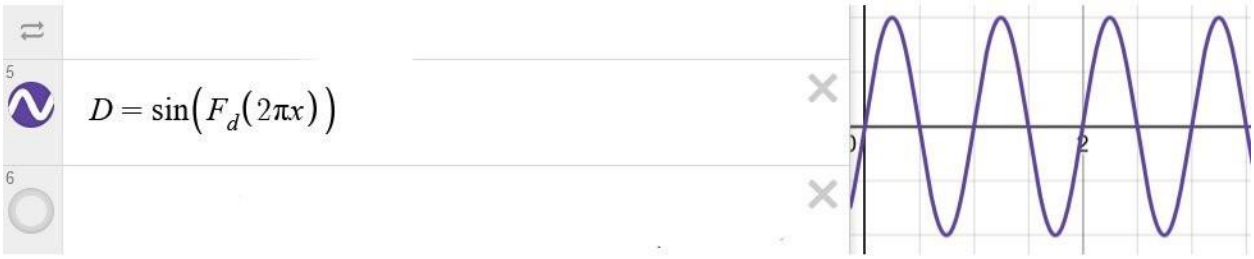

Figure 14

This depth data contains only one frequency which is specified by F_d.

$$L = \frac{2}{F_s}$$

$$L = 6$$

$$S_1 = \frac{1}{2} + \frac{2}{\pi}\sum_{n=1}^{100}\frac{1}{2n-1}\sin\left(\frac{(2n-1)\pi x}{L}\right)$$

$$S_2 = \frac{1}{2} + \frac{2}{\pi}\sum_{n=1}^{100}\frac{1}{2n-1}\sin\left(\left(\frac{(2n-1)\pi x}{L}\right)+\pi\right)$$

$$S_3 = \frac{1}{2} + \frac{2}{\pi}\sum_{n=1}^{100}\frac{(-1)^{(n)}}{2n-1}\cos\left(\left(\frac{(2n-1)\pi x}{L}\right)\right)$$

$$S_4 = \frac{1}{2} + \frac{2}{\pi}\sum_{n=1}^{100}\frac{(-1)^{(n-1)}}{2n-1}\cos\left(\left(\frac{(2n-1)\pi x}{L}\right)\right)$$

Figure 15

The four square waves with different phases can be represented as Fourier series, with a frequency specified by F_s.

$$\sin\alpha\sin\beta = \frac{\cos(\alpha-\beta)-\cos(\alpha+\beta)}{2}$$

$$\sin\alpha\cos\beta = \frac{\sin(\alpha+\beta)+\sin(\alpha-\beta)}{2}$$

Figure 16

Sine multiplication identities are used to multiply the depth data by the components of the square waves.

https://www2.clarku.edu/faculty/djoyce/trig/productidentities.jpg

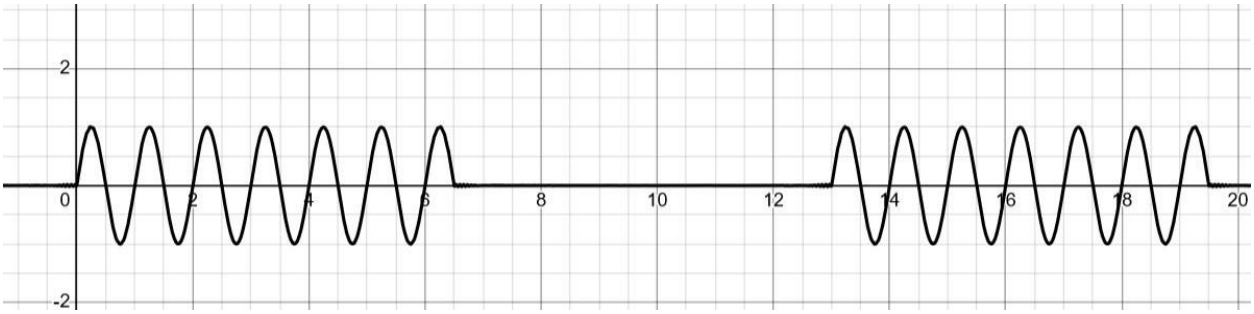

Figure 18

The Depth and square wave multiplied will result in some portions of the depth data being sampled, and other portions ignored.

$$A = \sqrt{(Q_1 - Q_2)^2 + (Q_4 - Q_3)^2}$$

Figure 20

The multiplied waves are inserted into the amplitude formula after they are summed, giving the measurement for that value of F_s.

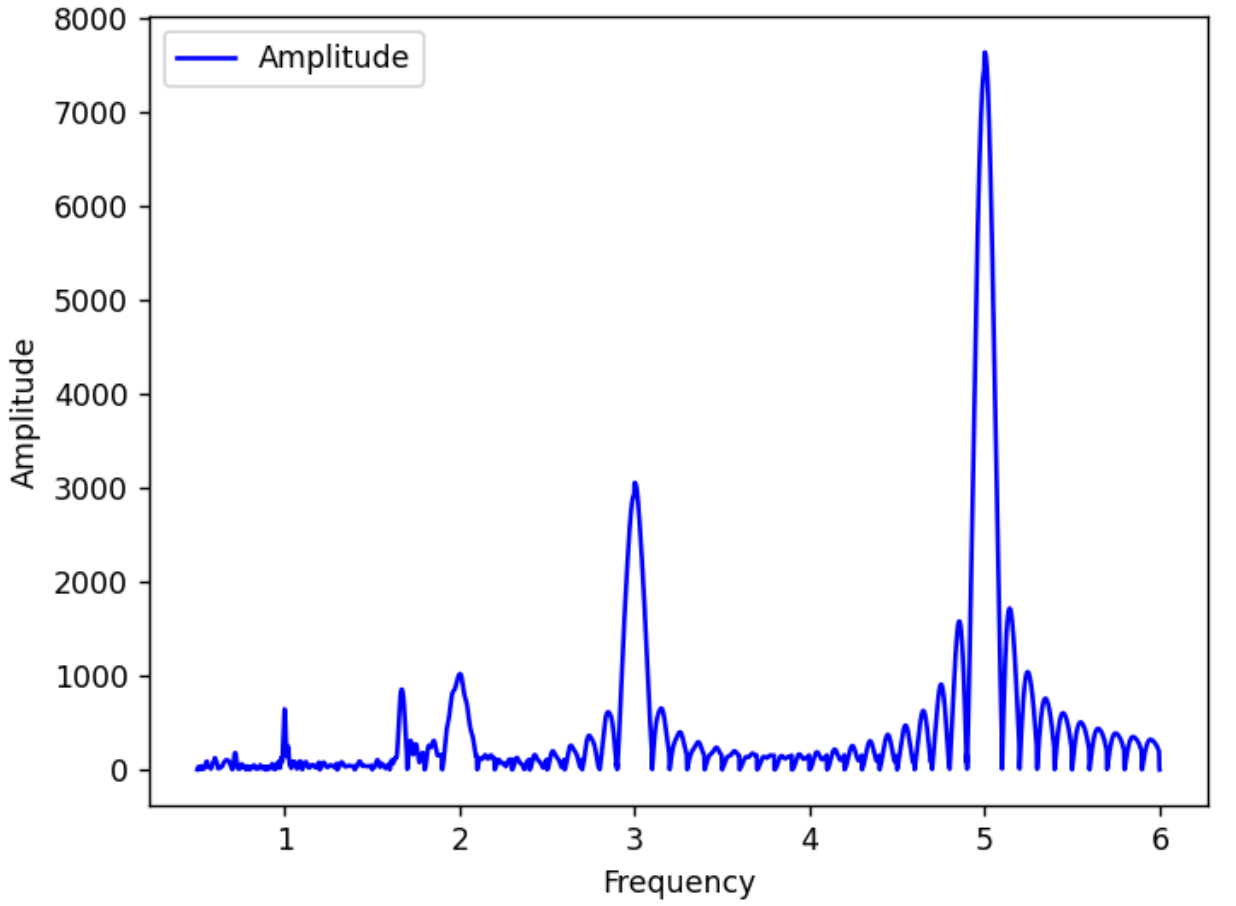

Figure 21

Incorrectly scaled amplitude measurements which must be corrected. The actual amplitudes are 1,2, and 3 at 2Hz, 3Hz, and 5Hz.

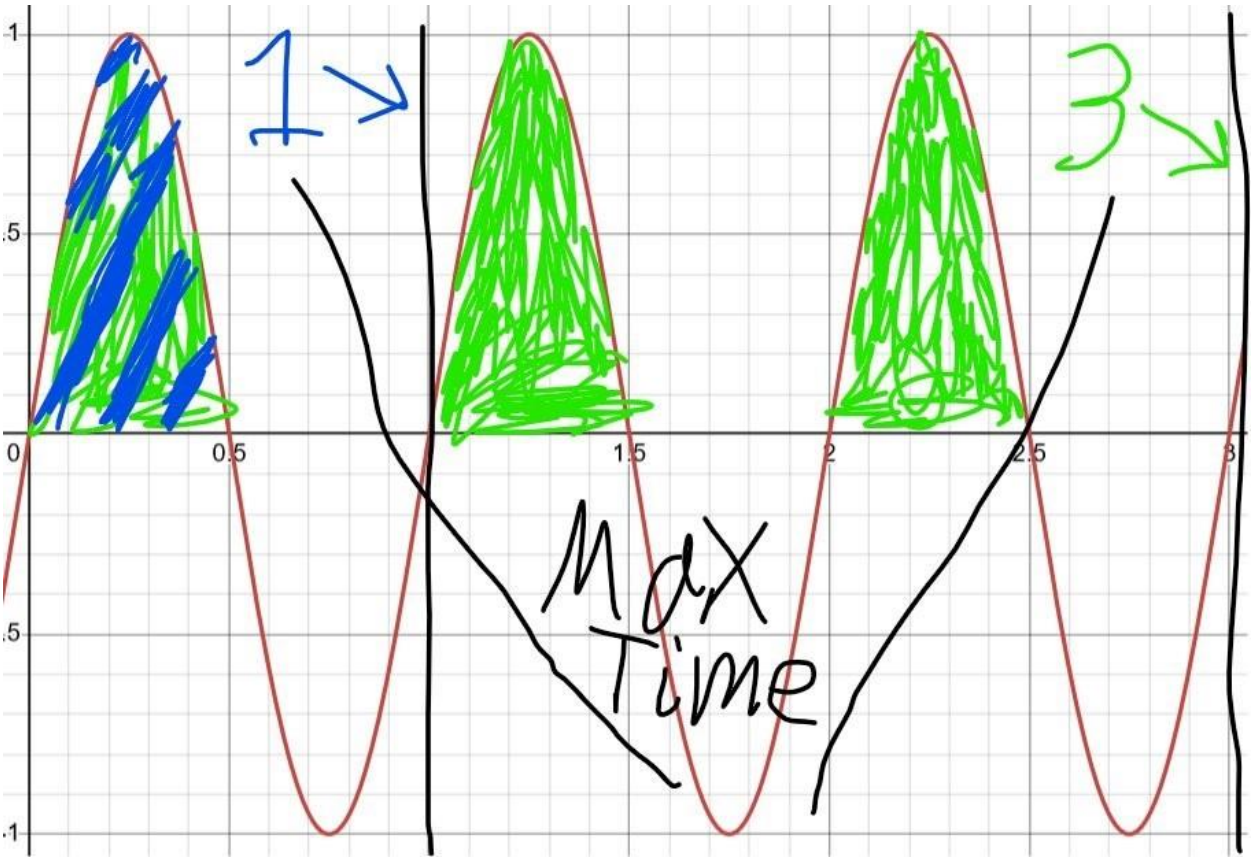

Figure 22

The sum in green is three times larger than the sum in blue because the integration time is three times longer.

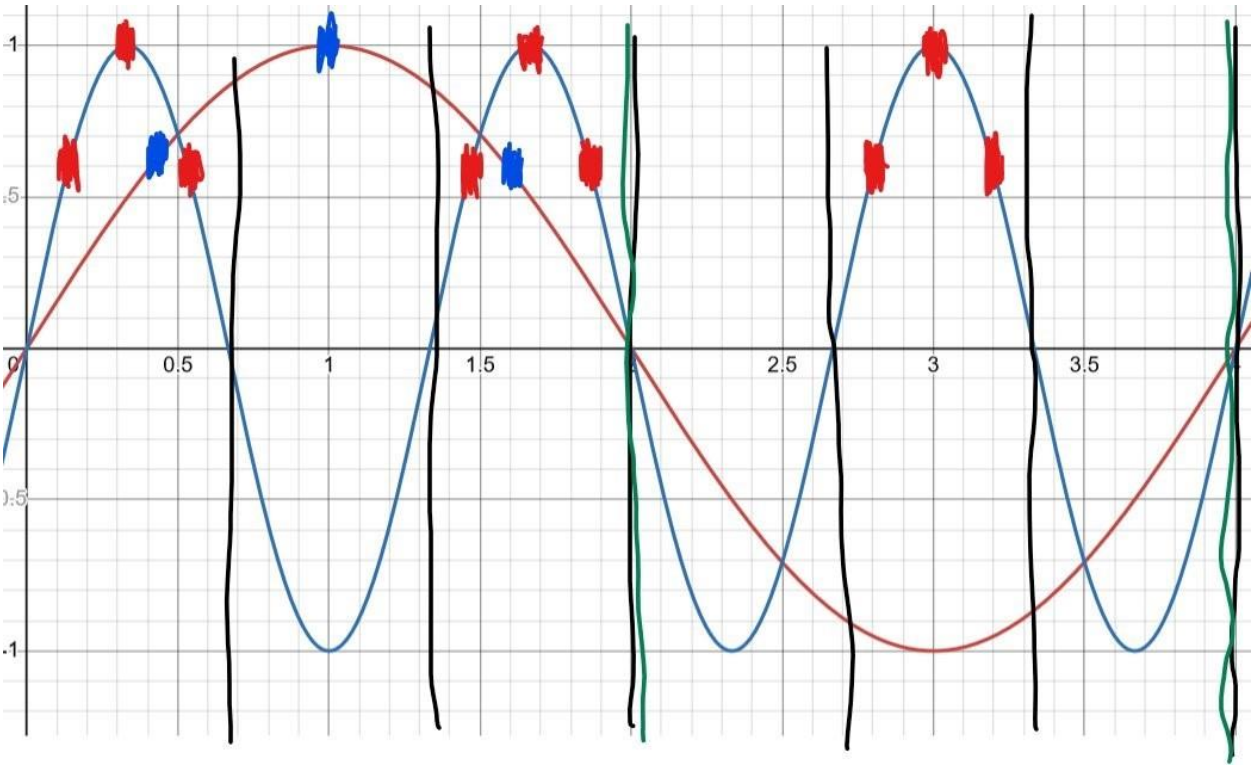

Figure 23

Samples for 1 Hz (red line, blue dots) versus samples for 3 Hz (blue line, red dots) with the same integration time. The 3hz line has three times as many samples.

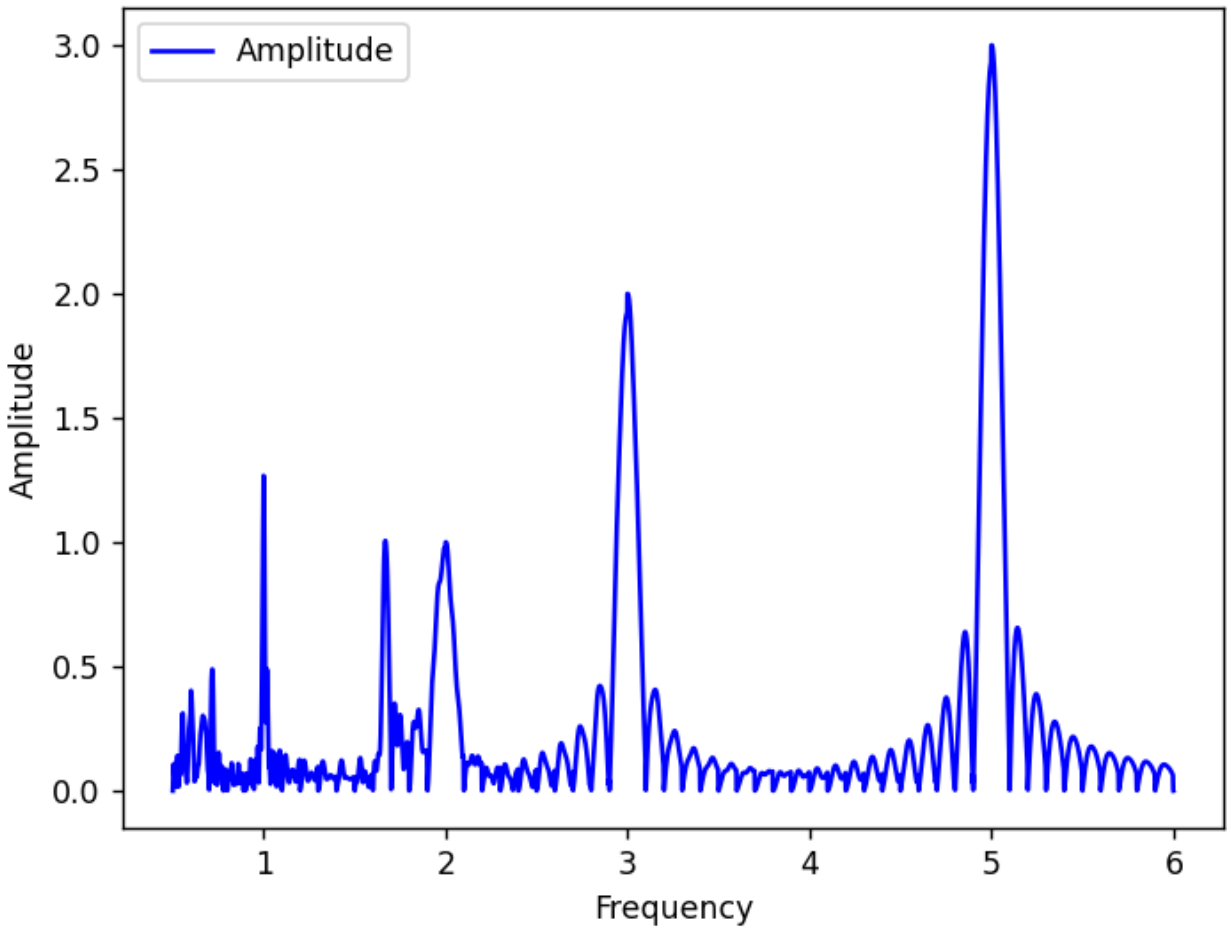

Figure 25

Correctly scaled amplitude measurements at 2Hz, 3Hz, and 5Hz.

$$S_1 = \frac{1}{2} + \frac{2}{\pi}\frac{1}{1}\sin\left(1F_s(2\pi x)\right) + \frac{2}{\pi}\frac{1}{3}\sin\left(3F_s(2\pi x)\right) + \frac{2}{\pi}\frac{1}{5}\sin\left(5F_s(2\pi x)\right) + \frac{2}{\pi}\frac{1}{7}\sin\left(\left(7F_s(2\pi x)\right)\right) \ldots$$

$$S_1 = \frac{1}{2} + \frac{2}{\pi}\sum_{n=1}^{100}\frac{1}{2n-1}\sin\left(\frac{(2n-1)\pi x}{L}\right)$$

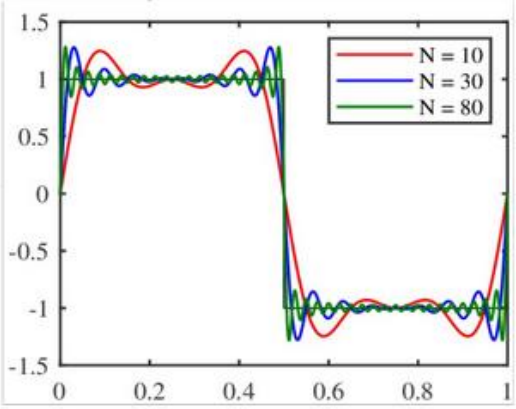

Figure 26

A square wave written as a Fourier series. N is the number of Fourier components included in the summation. The example equation sets N=100.

https://www.researchgate.net/Figure/Fourier-series-approximation-of-a-square-wave-N-is-the-number-of-number-of-terms-used-to_fig1_337768291

$$\sin\alpha\sin\beta = \frac{\cos(\alpha-\beta)-\cos(\alpha+\beta)}{2}$$

$$DS_1 = \frac{1}{2}\sin\left(F_d(2\pi x)\right) + \sum_{n=1}^{100}\frac{1}{(2n-1)\pi}\left(\cos\left(\left(F_d-\left(F_s(2n-1)\right)\right)(2\pi x)\right)-\cos\left(\left(F_d+\left(F_s(2n-1)\right)\right)(2\pi x)\right)\right.$$

Figure 27

Depth data and square waves are multiplied using the sine multiplication identity, resulting in a Fourier series containing this identity in its components.

https://www2.clarku.edu/faculty/djoyce/trig/productidentities.jpg

$$\int_0^I \frac{\cos\left(F_d t - F_s t\right) - \cos\left(F_d t + F_s t\right)}{2}\,dt$$

$$\cos\left(F_d t - F_s t\right) = 1$$

$$\cos\left(F_d t - 3F_s t\right) = 1$$

Figure 28

The components of the Fourier series are integrated as a constant of 1 when the frequencies lead to a cosine of zero at the main frequency and subharmonics.

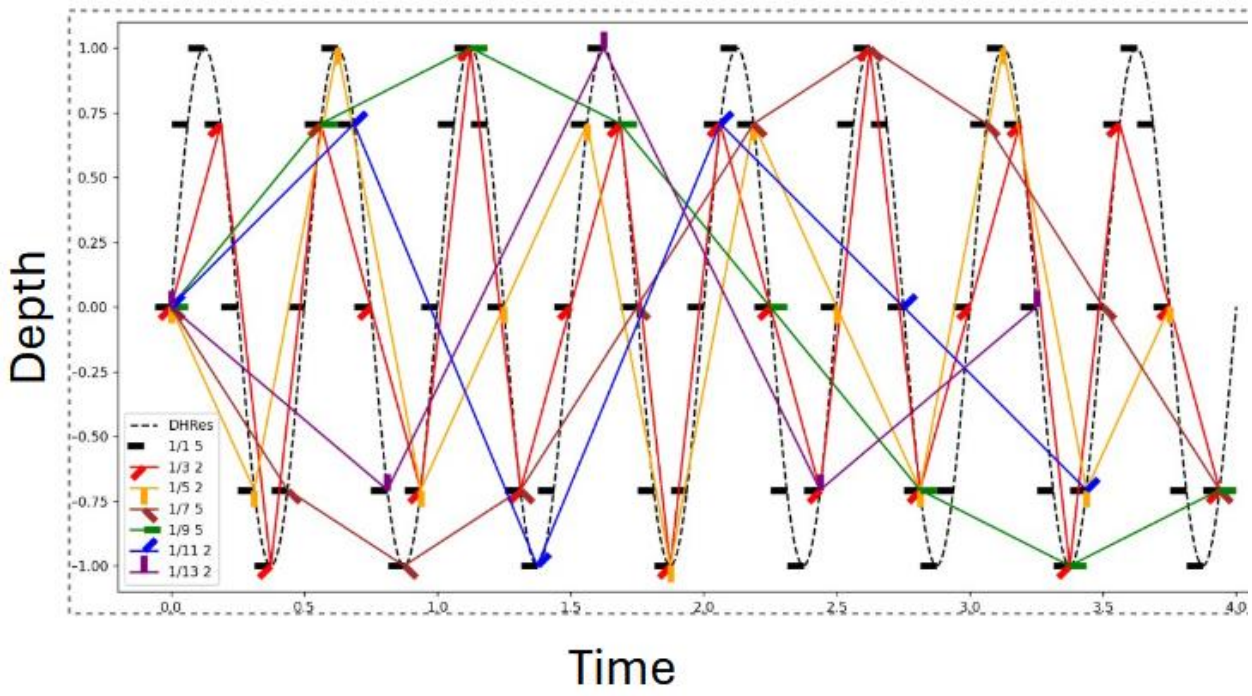

Figure 29

Aliasing of odd subharmonics, where different colors represent different subharmonics. There are four samples per bucket, resulting in two types of patterns and amplitude readings.

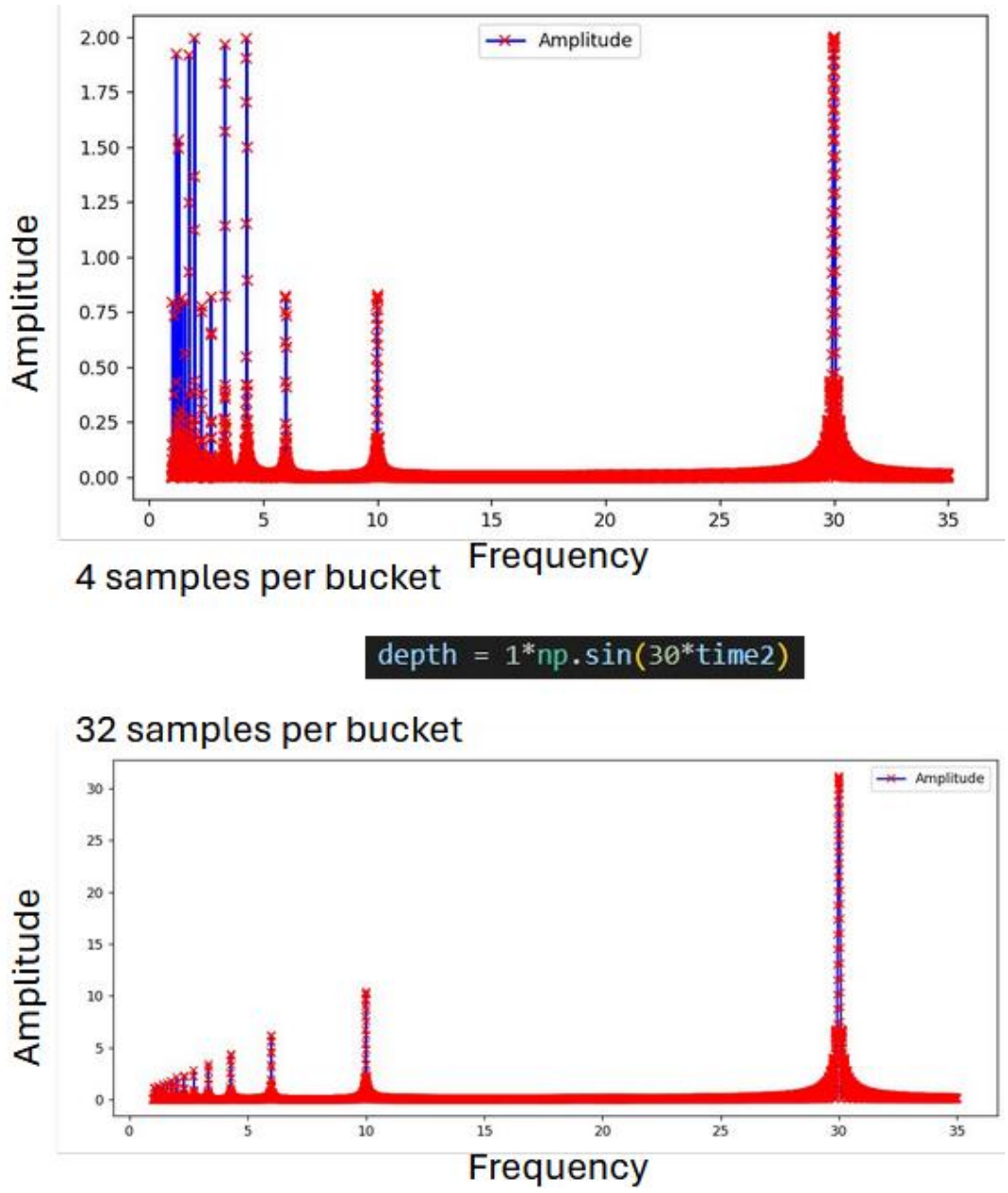

Figure 30

Increasing the number of samples per bucket will reduce the amplitude of the subharmonics, converging quickly to amplitudes matching their subharmonic frequencies (1/3rd, 1/5th).

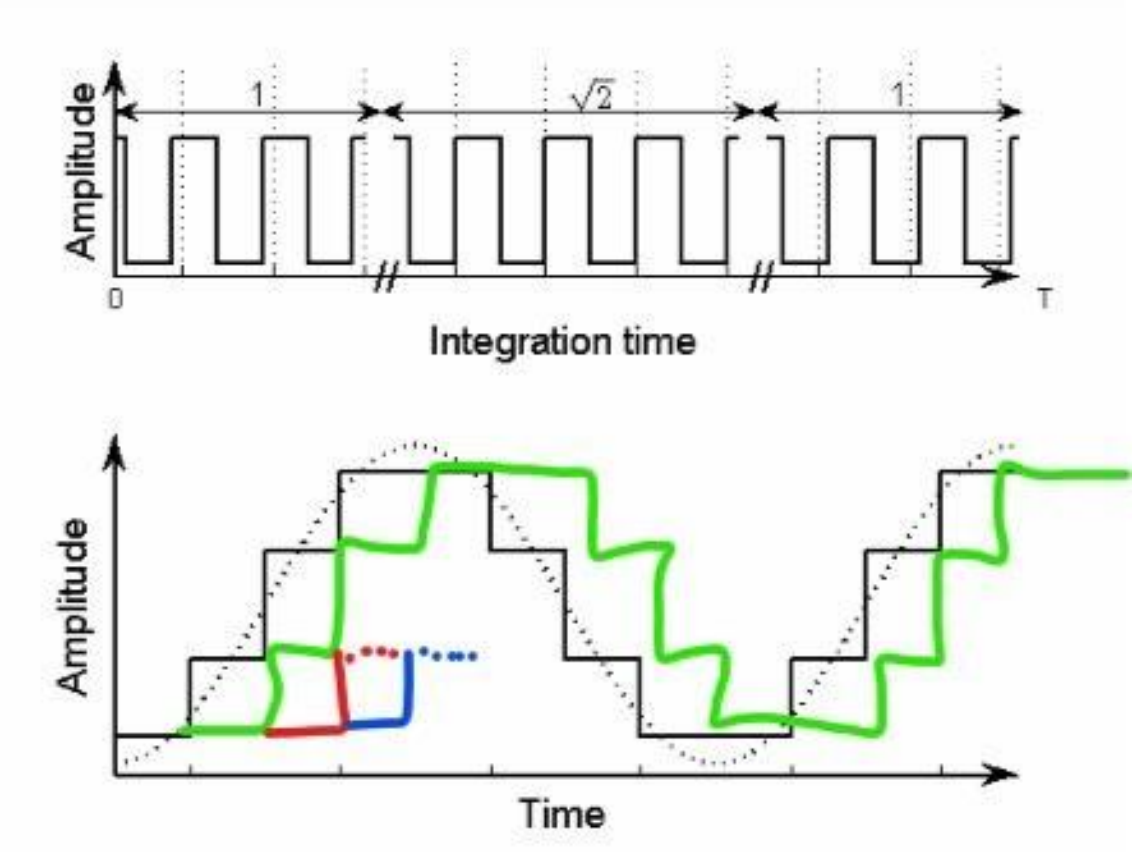

Figure 31

A ratio specifies which portions of the square wave will be shifted 45º to the left and right, resulting in the effective demodulation wave below. [D]

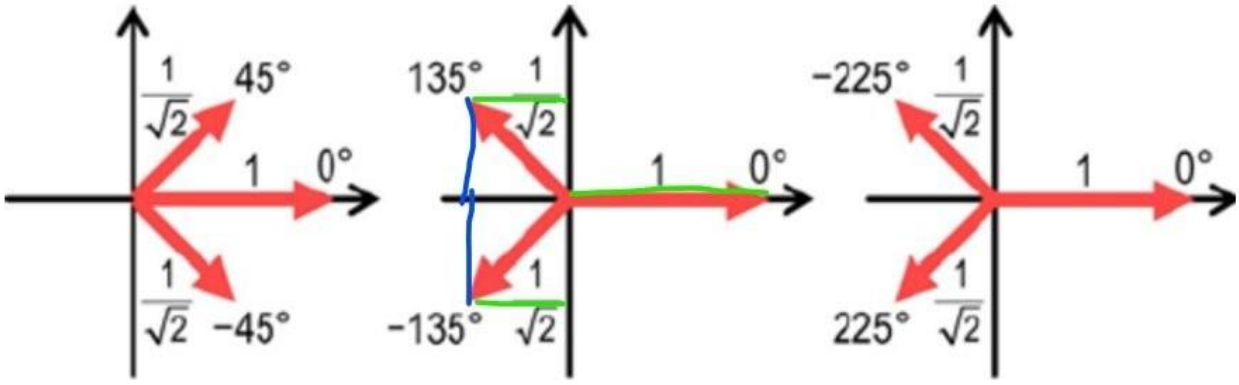

Figure 32

The real and imaginary components of the frequency and subharmonic measurements after the phase shift. All the imaginary components cancel out, as well as some of the real components. [C]

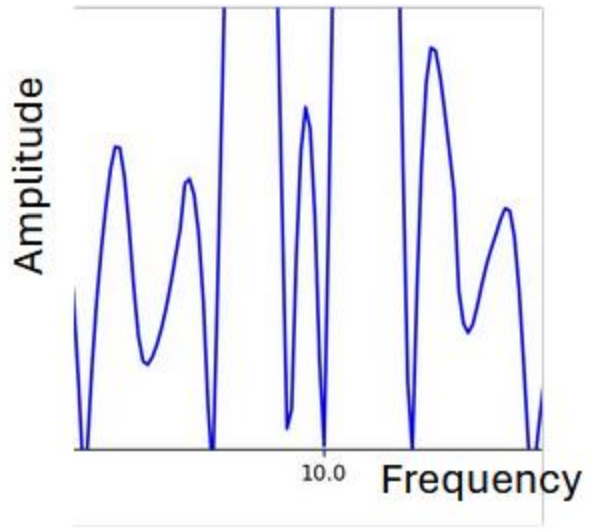

Figure 34

The third subharmonic of a 30Hz frequency, located at 10Hz is completely suppressed. The artifacts around it are not.

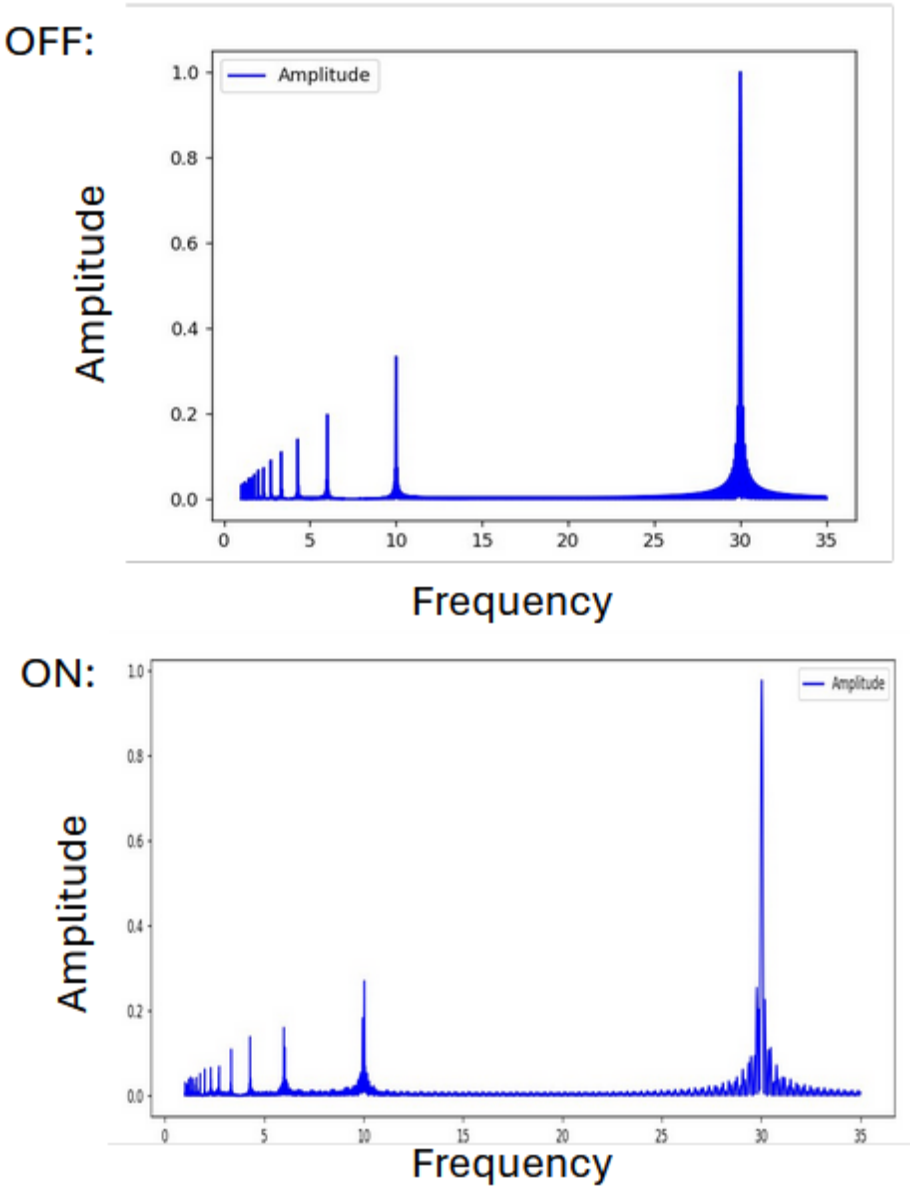

Figure 35

Comparison of amplitude measurements with the phase shift turned off and on. The ratios of the subharmonic amplitudes to the main frequency are smaller.

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import scipy as sp
import math

def harmrej(bucket_arr):
    shift_idx=math.floor(len(bucket_arr)*shift_rat)
```

```
    h1=bucket_arr[0:shift_idx]
    h2=bucket_arr[shift_idx:(len(bucket_arr)-shift_idx)]
    h3=bucket_arr[(len(bucket_arr)-shift_idx):len(bucket_arr)]
    h1=np.roll(h1,-(bucket_samples//4))
    h3=np.roll(h3,(bucket_samples//4))
    return [*h1, *h2, *h3]

def offset_noise(daxis,offset,noise):
    daxis=daxis+offset
    daxis=[d+((np.random.rand()-0.5)*2*noise) for d in daxis]
    return daxis

korea=0
phase=4
bucket_samples=32
correct=0
for c in range(0,bucket_samples,1):
    correct=correct+np.sin(((2*c+1)/(2*bucket_samples))*np.pi)
correct=2*correct
duty_cycle=0.5
if korea==1:
    phase=4
markers=0
semilog=0
supharm=0
bucket_plot=1
plt_freq=10
transform_depth=0
offset=0
noise=0
Min_freq=1
Max_freq=35
T_min=1/(Max_freq*2)
T_max=1/(Min_freq*2)
Num_T=10000
Max_time=10
shift_rat=1/(2+math.sqrt(2))
if Min_freq < (1/Max_time):
    print("CORRECTING MIN_FREQ. lower than 1/maxtime:")
    print("Old value of Min_freq:")
    print(Min_freq)
    Min_freq=1/Max_time
    print("New value of Min_freq:")
    print(Min_freq)
```

```
f_dt = (Max_freq-Min_freq)/(Num_T-1)
amp_val=[]
tx_val=[]

for n in range(0,Num_T,1):
    samp_freq=Min_freq+(n*f_dt)
    t=1/(samp_freq*2)
    tx_val.append(t)
    T_step=t
    T_sub=T_step/bucket_samples
    S=(int(np.ceil(Max_time*bucket_samples/T_step)))
    time = np.linspace(0, S*T_sub, S+1)
    time2 = 2.0*np.pi*time
    depth = 1*np.sin(30*time2)

    if transform_depth==1:
        depth=offset_noise(depth,offset,noise)

    length = len(depth)
    if korea==0:
        duty=int(2*duty_cycle*bucket_samples)
        pattern = np.roll([1] * duty + [0] * ((2*bucket_samples)-
duty),bucket_samples)
    else:
        pattern = np.array([0, 0, 0, 0, 1, 0, 0, 0])
    repeats = length // len(pattern)
    sequence = np.tile(pattern, repeats)
    c1=sequence

    if korea==0:
        pattern=np.roll(pattern,bucket_samples)
    else:
        pattern = np.array([1, 0, 0, 0, 0, 0, 0, 0])
    sequence = np.tile(pattern, repeats)
    c2=sequence

    if korea==0:
        pattern=np.roll(pattern,-bucket_samples//2)
    else:
        pattern = np.array([0, 0, 0, 0, 0, 0, 1, 0])
    sequence = np.tile(pattern, repeats)
    c3=sequence
```

```
    if korea==0:
        pattern=np.roll(pattern,bucket_samples)
    else:
        pattern = np.array([0, 0, 1, 0, 0, 0, 0, 0])
    sequence = np.tile(pattern, repeats)
    c4=sequence

    if supharm==1:
        c1=harmrej(c1)
        c2=harmrej(c2)
        c3=harmrej(c3)
        c4=harmrej(c4)

    depth=depth[0:len(c1)]

    q1=np.multiply(depth,c1)
    q2=np.multiply(depth,c2)
    q3=np.multiply(depth,c3)
    q4=np.multiply(depth,c4)

    q1_sum=np.sum(q1)
    q2_sum=np.sum(q2)
    q3_sum=np.sum(q3)
    q4_sum=np.sum(q4)

    if phase==4:
        amp=math.sqrt(math.pow((q1_sum-q2_sum),2)+math.pow((q4_sum-q3_sum),2))
    elif phase==2:
        amp=math.sqrt(math.pow((q1_sum),2)+math.pow((q3_sum),2))
    else:
        amp=np.abs(q1_sum)

    time_plot=time[0:len(c1)]
    if samp_freq>=plt_freq and bucket_plot==1:
        print("FREQUENCY:")
        print(samp_freq)
        print("depth_cut:")
        print(depth)
        print("q1:")
        print(q1)
        print("q2:")
        print(q2)
        print("q3:")
```

```
        print(q3)
        print("q4:")
        print(q4)
        print("q1_sum:")
        print(q1_sum)
        print("q2_sum:")
        print(q2_sum)
        print("q1_sum-q2_sum:")
        print(q1_sum-q2_sum)
        print("q3_sum:")
        print(q3_sum)
        print("q4_sum:")
        print(q4_sum)
        print("amplitude (with freq correction)")
        print(amp/(samp_freq*Max_time*correct))
        print("MATH TEST:")
        print(math.sqrt(math.pow((q1_sum-q2_sum),2)+math.pow((q4_sum-q3_sum),2)))
        plt.plot(time_plot, depth, label = "Depth_cut", color="black")
        plt.plot(time_plot, c1, 'o', label = "C1", marker=4, mec = 'red', mfc =
'red')
        plt.plot(time_plot, c2, 'o', label = "C2", marker=5, mec = 'green', mfc =
'green')
        plt.plot(time_plot, c3, 'o', label = "C3", marker=6, mec = 'orange', mfc
= 'orange')
        plt.plot(time_plot, c4, 'o', label = "C4", marker=7, mec = 'blue', mfc =
'blue')
        plt.legend()
        plt.show()
        bucket_plot=0

    amp_val.append(amp)

tx_val=[1/(t * 2) for t in tx_val]
print(tx_val)
amp_val=np.divide(amp_val,tx_val)
amp_val=np.divide(amp_val,Max_time)
amp_val=np.divide(amp_val,correct)



if markers==0 and semilog==0:
    plt.plot(tx_val, amp_val, label = "Amplitude", color="blue")
elif markers==1 and semilog==0:
```

```
    plt.plot(tx_val, amp_val, label = "Amplitude", color="blue", marker='x', mec
= 'red', mfc = 'red')
elif markers==0 and semilog==1:
    plt.semilogx(tx_val, amp_val, label = "Amplitude", color="blue")
else:
    plt.semilogx(tx_val, amp_val, label = "Amplitude", color="blue", marker='x',
mec = 'red', mfc = 'red')

plt.legend()
plt.show()
```

Figure 36

The code above will generate a depth signal, measure the frequencies inside, and plot the results using the method described in this paper. The parameters control its behavior.

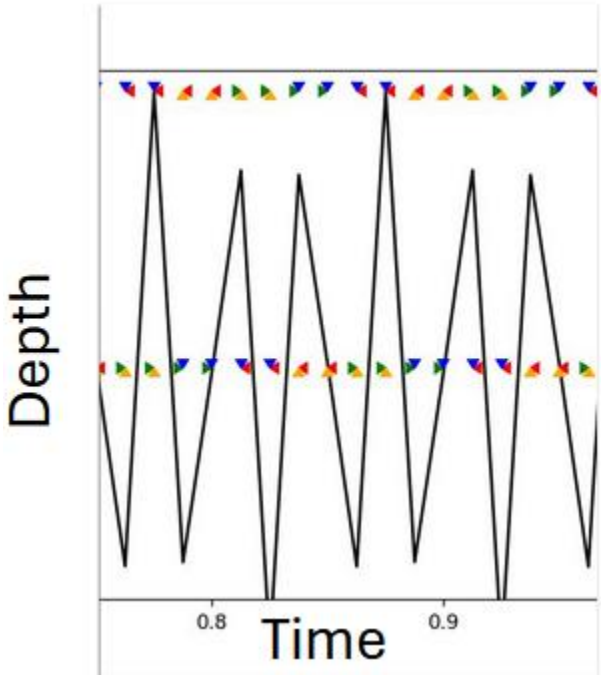

Figure 37

Four discrete square waves plotted as colored markers. This is useful for troubleshooting and observing the phase shift of the square waves.

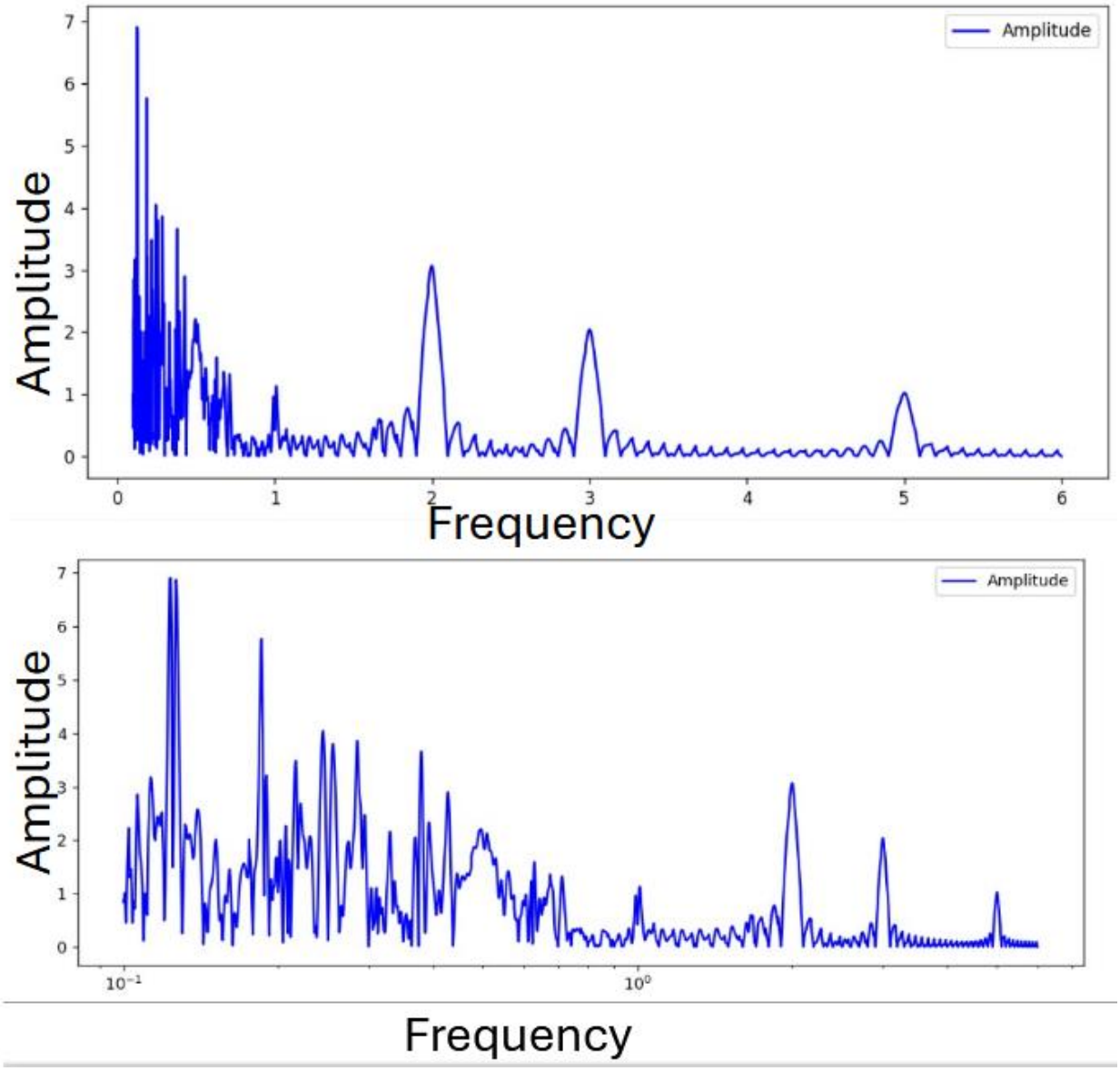

Figure 38

The top uses normal scaling, while the bottom magnifies small frequencies using semi log scaling, enabling easier viewing of the subharmonics.

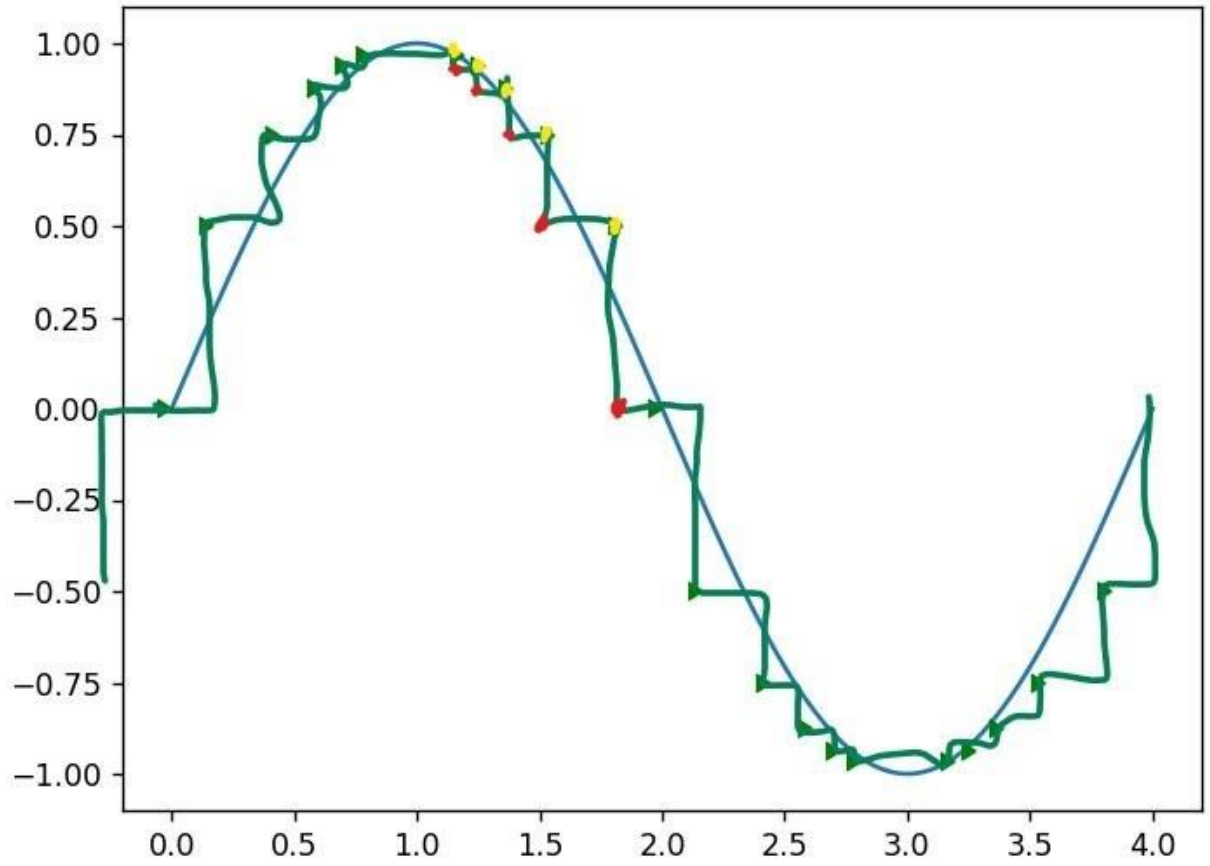

Figure 39

A hypothetical example of a sine wave approximation made using bitwise doubling. This demodulation wave will result in lower amplitudes for the subharmonics.

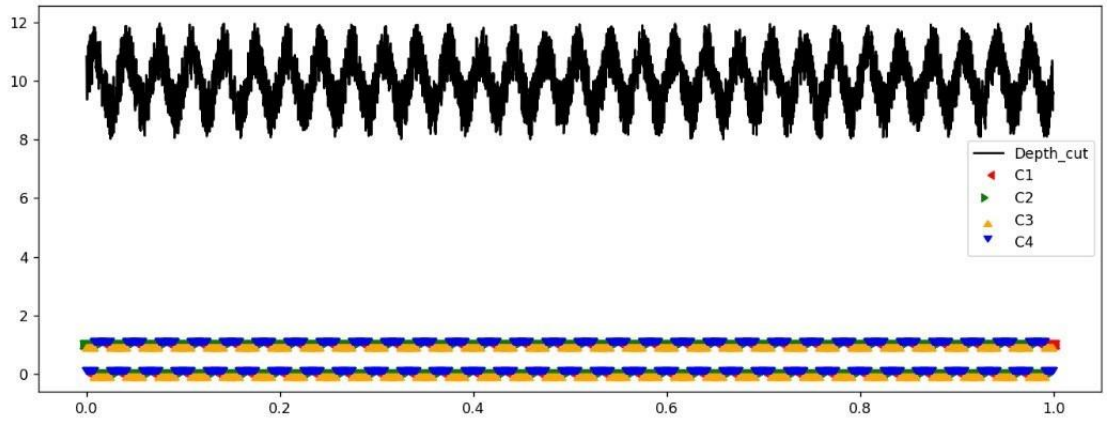

Figure 40

Depth data with noise added to the signal to make it more realistic.

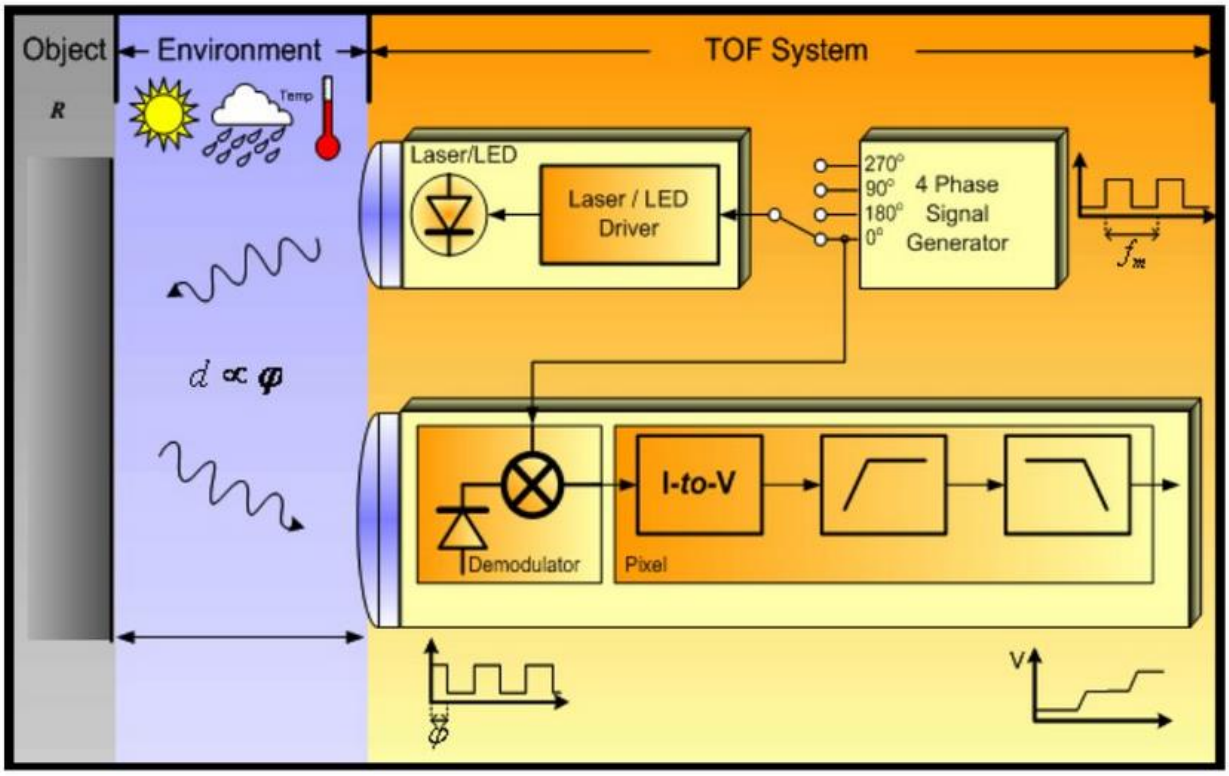

Figure 41

An example of a TOF sensor inside of an electrical circuit. The signal generator and mixer (X) are visible. [B]