Qeios

Pulse amplitude measurement using low sampling ADC and interpolation technique

Arpit Patel, Bhumi Patel

Arpit Patel is with the Physical Research Laboratory, Ahmedabad 380009, Gujarat, India (corresponding author phone: 07926314927; e-mail: arpitp@prl.res.in).
Bhumi Patel is with the Electronics and Communication department, L J University, Ahmedabad 382210, Gujarat, India (e-mail: bhumi.patel@ljinstitutes.edu.in).

ABSTRACT
Pulse amplitude measurement for the analog signal is now widely used in applications like spectroscopy, IR / laser receiver, RF signal receiver, etc. Accurate Amplitude measurement requires a high-speed analog to digital converter which consumes high power. In this paper we are aiming to detect analog pulse amplitude with low sampling ADC and interpolation method. The comparative study of seven interpolating methods namely Near interpolation, Linear interpolation, Cubic interpolation, LaGrange's interpolation, Newton Raphson interpolation, Whittaker Shannon interpolation and Neville's algorithm were conducted for measurement of analog pulse peak height. The interpolation method is selected based on the relative error, mean square error and mean absolute deviation. The performance of each method and the overall system is discussed in this paper.

Keywords:
Interpolation, Relative error, Mean-square error, ADC, Mean absolute deviation, FPGA

1 INTRODUCTION

Many applications require measurements using an analog-to-digital converter (ADC) to convert the analog input information in to the digital form. Such applications will have resolution requirements based on the dynamic range of the signal, the minimum measurable change in a parameter, and the signal-to-noise ratio (SNR). Many systems use a better resolution off-chip ADC as a result. However, the cost of the ADC increases with the level of desired accuracy. By creating hardware to quantize the analog signal amplitude into the digital signal with a longer code-word length, better ADC accuracy can be attained. Word lengths in practical ADCs are limited. Higher conversion accuracy is attained by calculating additional samples in order to efficiently strike a balance between system cost and accuracy. Analyzing the input signal is necessary in order to improve the required ADC resolution. Digital interpolation methods can be used to increase the resolution of digital data. Through an FPGA, the digital signal can be processed. This technique, which processes samples and increases the 12-bit ADC conversion's accuracy by extra bits, is examined in this study. As the work's application, peak height detection is used. The function generator's gaussian pulse is delivered to the ADC for sampling before the data is sent to the FPGA for interpolation. The interpolation will be implemented by FPGA, which will use the interpolated data to determine the maximum sample value. To obtain data from the FPGA and display it on the computer screen with the necessary graphs and numbers, a national instrument brand called NI DAQ is employed.

2 INTERPOLATION TECHNIQUES

With the help of a few precise sample data points, interpolation can be used to estimate the unknown values of a nonlinear or linear function. Theoretically, more data points are needed for better fitting accuracy. However, it is not practical to evaluate ideas based on big data sets due to the constraints of the difficulties of hardware computing. The initial objective of the problem at hand is to sample the input pulse into uniformly spaced data points. Any of the following interpolation techniques can be used to determine the analog pulse's unknown peak value.

2.1 Nearest neighbor Interpolation

The simplest interpolation technique is nearest neighbor interpolation. The value of the interpolated function f at coordinate X for given the samples F(k) as follows:

$$\boldsymbol{f(x) = F\left(\lfloor x + \tfrac{1}{2} \rfloor\right)} \tag{1}$$

The closest sample point to x is simply chosen. The greatest integer value that is less than or equal to x in this case is called the floor function [x]. This results in an interpolated function that looks like a staircase. The real axis as a whole is where the function f(x) is defined, although it is neither continuous (since it contains jumps) nor differentiable. Consider the function f(x)=sin(x) and its interpolated nearest neighbor version in Figure 1 as an illustration.

https://doi.org/10.32388/UFBEFX

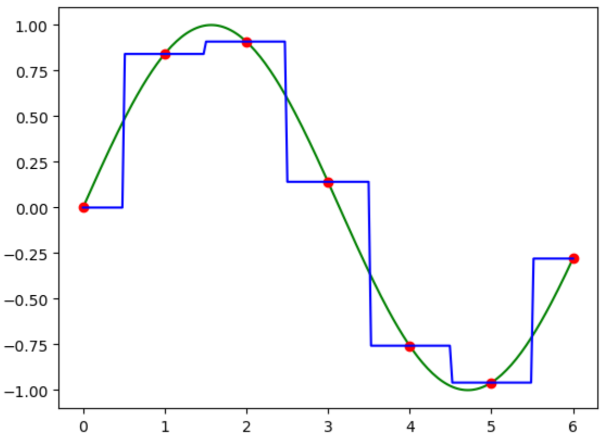


Figure 1. In green the continuous function f(x) = sin(x), in red the sampled function and in blue the nearest interpolated function.

2.2 Linear Interpolation

The simplest technique for determining the value of a function between any two known values is the linear interpolation formula. A technique for fitting curves with linear polynomials is the linear interpolation formula. In essence, the interpolation method uses the collection of values to find new values for any function. The linear interpolation formula is used to determine the unknown values in the table. In this section, let's study more about the linear interpolation formula. Data forecasting, data prediction, mathematical and scientific applications, market research, etc. all employ the linear interpolation algorithm. The unknown values in the table can be found using the linear interpolation formula. The linear interpolation formula is as follows:

$$\boldsymbol{y = y_1 + (x - x_1)\frac{(y_2-y_1)}{(x_2-x_1)}} \tag{2}$$

where,

- x_1 and y_1 are the first coordinates
- x_2 and y_2 are the second coordinates
- x is the point to perform the interpolation
- y is the interpolated value

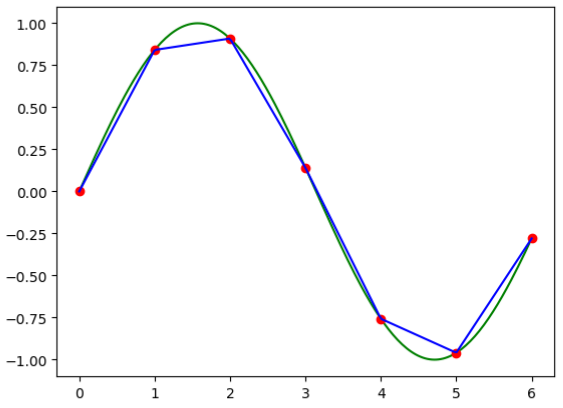


Figure 2. In green the continuous function f(x)= sin (x), in red the sampled function and in blue the linear interpolated function.

Despite being a straightforward interpolation technique, linear interpolation is frequently employed in practice. It is quick and easy to put into practice.

2.3 Lagrange's interpolation

The Lagrange interpolation formula can be used to locate a polynomial known as a Lagrange polynomial that assumes certain values at every location. Lagrange's interpolation is an approximation to f(x) using a Nth degree polynomial. There is a single polynomial P with real coefficients that, given n different real values X1, X2,..., Xn and n real values Y1, Y2,..., Yn (not necessarily distinct), fulfils $P(X_i)=Y_i$ for I ∈ 1, 2,..., n and has deg(P) < n. The following is the Lagrange interpolation formula for various orders of polynomials:

$$\boldsymbol{y = f(x) = \frac{(x-x_1)(x-x_2)\ldots(x-x_n)}{(x_0-x_1)(x_0-x_2)\ldots(x_0-x_n)}y_0 + \frac{(x-x_0)(x-x_2)\ldots(x-x_n)}{(x_1-x_0)(x_1-x_2)\ldots(x_1-x_n)}y_1 + \cdots + \frac{(x-x_0)(x-x_1)\ldots(x-x_{n-1})}{(x_n-x_0)(x_n-x_1)\ldots(x_n-x_{n-1})}y_n} \tag{3}$$

2.4 Newton Raphson Interpolation

The Newton-Raphson approach, commonly referred to as Newton's method, is a rapid way to approximate the real-valued root function f(x)=0. It makes use of the idea that a straight line parallel to a continuous, differentiable function can serve as a rough approximation. A continuous, differentiable function f(x) needs to have a root, and you know the root you're seeking for is close to the point x = x_0. Then, according to Newton's approach, a more accurate approximation for the root is

$$\boldsymbol{x_1 = x_0 - \frac{f(x_0)}{f\prime(x_0)}} \tag{4}$$

This process may be repeated as many times as necessary to get the desired accuracy. In general, for any x-value x_n, the next value is given by

$$\boldsymbol{x_{(n+1)} = x_n - \frac{f(x_n)}{f\prime(x_n)}} \tag{5}$$

2.5 Cubic interpolation

Only one sample (the closest) is used in nearest neighbor interpolation to determine the interpolated value [1]. The two nearest sample locations are examined in linear interpolation (one on the left and one on the right). We examine two datapoints on the left and two on the right for cubic interpolation. We fit a cubic polynomial over the range between x=k and x=k+1 to interpolate [2].

k ≤ x ≤ k+1:

$f(x) = a(x)^3 + b(x)^2 + c(x) + d$ to the 4-sample points *k−1, k, k+1* and *k+2.*

For the 4 points we have:

$F(k-1) = a(-1)^3 + b(-1)^2 + c(-1) + d$

$F(k) = d$

$$F(k+1) = a(1)^3 + b(1)^2 + c(1) + d$$

$$\boldsymbol{F(k+2) = a(2)^3 + b(2)^2 + c(2) + d} \tag{6}$$

Solving these equations for *a, b, c* and *d* we get:

$$a = \frac{1}{6}\left(-F(k-1) + 3\,F(k) - 3F(k+1) + F(k+1)\right)$$

$$b = \frac{1}{2}\left(F(k-1) - 2F(k) + F(k+1)\right)$$

$$c = \frac{1}{6}\left(-2F(k-1) - 3F(k) + 6F(k+1) - F(k+2)\right)$$

$$d = F(k) \tag{7}$$

2.6 Whittaker–Shannon interpolation

The Whittaker–Shannon interpolation formula or sinc interpolation is a method to construct a continuous-time bandlimited function from a sequence of real numbers. The process is similar to that of the Lagrange polynomial interpolation. The key to polynomial interpolation is to identify a set of component polynomials, each of which traverses a certain set of provided points. The component polynomials must also equal zero whenever there is a different point in order to be able to combine them all into a single polynomial that traverses the entire set of points. This makes sure that adding the polynomials together will not cause an interference between them at the given set of points. In Whittaker-Shannon interpolation, the components are the sinc function instead of polynomials. Sinc functions are periodic functions that are as follows:

$$\boldsymbol{sinc(x)} = \begin{cases} \boldsymbol{1,\ x = 0} \\ \frac{\sin(x)}{x},\ \boldsymbol{Otherwise} \end{cases} \tag{8}$$

2.7 Neville's Algorithm for interpolation

Neville's algorithm is an interpolation algorithm which proceeds by first fitting a polynomial of degree 0 through the point (x_k, y_k) for k=1, ..., n, i.e., $P_k(x)=y_k$. Then, in a subsequent iteration, Pi and P(i+1) are merged to fit between pairs of points, resulting in P_{12}, P_{23}, and so forth. Up until the desired outcome is attained, the method is repeated, creating a "pyramid" of approximations.

$$\begin{array}{lllll}
x_1: y_1 = P_1 & & & & \\
 & P_{12} & & & \\
x_2: y_2 = P_2 & & P_{123} & & \\
 & P_{23} & & P_{1234} & \\
x_3: y_3 = P_3 & & P_{234} & & \\
 & P_{34} & & & \\
x_4: y_4 = P_4 & & & &
\end{array}$$

The final result is

$$\boldsymbol{P_{i(i+1)\ldots.(i+m)}} = \frac{(x - x_{i+m})P_{i(i+1)\ldots.(i+m-1)}}{(x_i - x_{i+m})} + \frac{(x_i - x)P_{(i+1)(i+2)\ldots.(i+m)}}{(x_i - x_{i+m})} \tag{9}$$

3 IMPLEMENTATION OF INTERPOLATION METHODS

In the present work, all the above-mentioned methods of interpolation in simulation as well as in experimentation were implemented in FPGA and LabVIEW environment. The initial finding of this research was that as the number of interpolation points increased, fitting accuracy likewise increased, reducing the error [3]. The 3-point interpolation is employed for comparison.

3.1 FPGA Implementation

Due to its adaptability and programmability, FPGAs have many uses in electronics engineering. FPGA can do addition, subtraction, multiplication, and division operations in real-time mode [4]. Fixed point operations are used by the FPGA in the current task, so it is necessary to determine the best bit size to use for each operation in order to prevent overflows that could result in inaccurate measurement. Fixed point operations, which allow for customized bit sizes of data for various mathematical operations, speed up execution, but the output may become saturated if the bit size is not chosen properly or the high and low values are not taken into account. Furthermore, if the resolution is less than required, the output accuracy may suffer and may give erratic values.

In the present work, National Instruments (NI) USB6343 and A3PE1500 FPGA with 1.5 Million gates are used. The code for FPGA implementation should be designed to use minimum resources as resources are limited. Also, the timing constraints need to be considered. The base clock is designed to work at 40MHz. Any mathematical operation that cannot be performed within the time frame of this clock speed has to be treated under a special case of single cycle timed loops with a reduced clock speed. In the above-mentioned algorithms, a reduced clock rate was used wherever necessary. The resource utilization summary for various algorithms on FPGA is given in upcoming sections.

3.2 Experimental Setup & Validation

The experimental setting that was utilized to evaluate the effectiveness of the piecewise interpolation algorithms that were implemented is described in depth in this section. Figure 3 displays the block schematic for the experimental setup. The same number of samples and interpolation order were taken into consideration to give a consistent platform for comparison when assessing the effectiveness of different interpolating methods based on fitting accuracy.

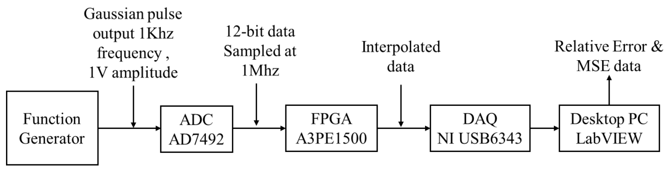

Gaussian pulse output 1Khz frequency , 1V amplitude
12-bit data Sampled at 1Mhz
Interpolated data
Relative Error & MSE data
Function Generator
ADC AD7492
FPGA A3PE1500
DAQ NI USB6343
Desktop PC LabVIEW

Figure 3. Experimental scheme for RE and MSE measurement

The gaussian pulse shown in Figure 4 is generated using the arbitrary waveform generator facility in the function generator. The Gaussian pulse form the function generator is sampled using 12-bit analog to digital converter AD7492 with

2.5V reference and 1 MHz sampling rate. The digital data is given to FPGA for interpolation. The required logic to generate control signal, acquire data from ADC and interpolation technique is implemented in FPGA. Three-point interpolation is carried out for each technique. Interpolated data from the FPGA is given to computer using National instrument (NI) make data acquisition module. In computer the LabVIEW code is written to find the peak (maximum) value from the sample data and the errors are estimated from the sample data. Each interpolation technique discussed in earlier section is implemented in FPGA separately. The errors are estimated using LabVIEW code and data is stored in computer for post analysis.

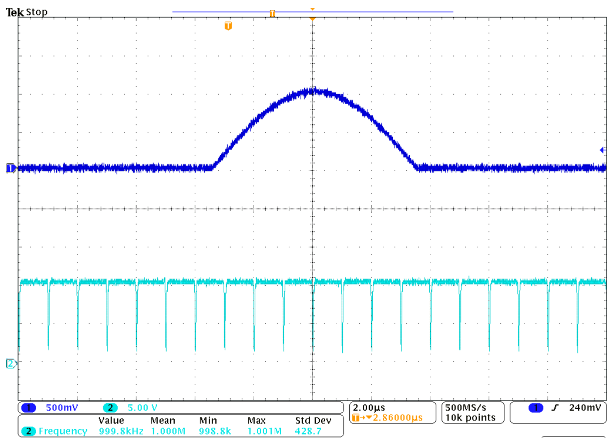


Figure 4. Waveform obtained using an oscilloscope showing gaussian pulse from function generator output (dark blue) and convert start pulse for ADC from FPGA (light blue).

The equation for relative error is listed below (11) where the peak-actual is ideal 12-bit data corresponds to 1V and peak-measured is peak value measured after interpolation which is taken from LabVIEW code. The fitting accuracy is accessed by MSE (12) and Mean Absolute Deviation MAD (13) (Sonowal & Bhuyan, 2013). The relative error is calculated for thousand pulse data.

$$\boldsymbol{Peak_{actual}\ for\ 1V\ amplitude\ pulse} = \frac{\mathbf{4096\ \ (12-bit)}}{\mathbf{2500\ \ (2.5V\ ref)}}\ \boldsymbol{x}\ \mathbf{1000}$$

$$= \mathbf{1638.4}$$

(1)

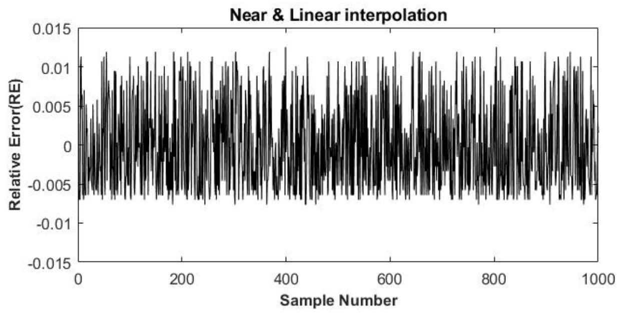
$$\boldsymbol{Relative\ error} = \frac{\boldsymbol{Peak}_{\boldsymbol{actual}} - \boldsymbol{Peak}_{\boldsymbol{Measured}}}{\boldsymbol{Peak}_{\boldsymbol{actual}}}$$

(2)

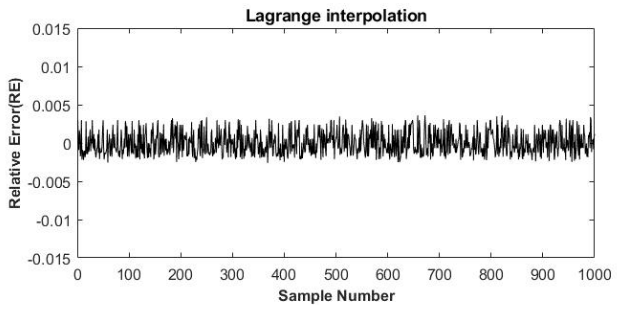
$$\boldsymbol{MSE} = \frac{\sum_{1}^{n}(\boldsymbol{Relative\ Error})^2}{\boldsymbol{n}} \qquad (3)$$

$$\boldsymbol{MAD} = \frac{\sum_{1}^{n}|\boldsymbol{Relative\ Error}|}{\boldsymbol{n}} \qquad (4)$$

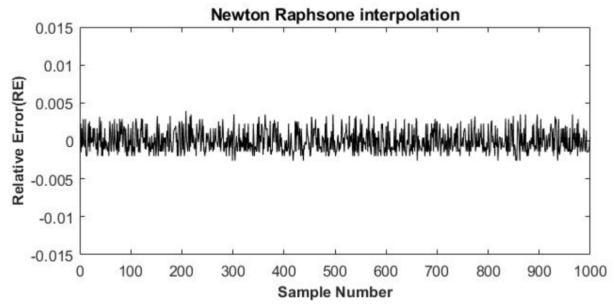
The mean square error and mean absolute deviation are calculated using relative error as shown in equation (12) and (13) [3]. where, the value of n is 1000. The relative error (RE) is calculated for thousand samples for various interpolation techniques and shown in Figure 5.



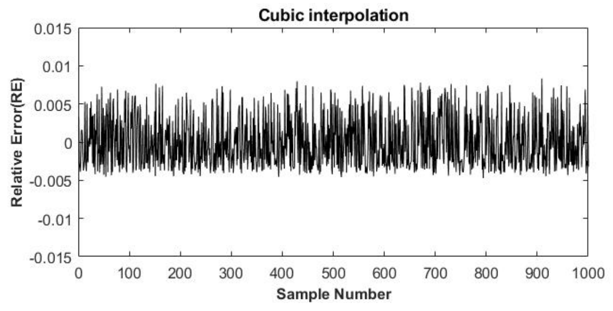
(a)



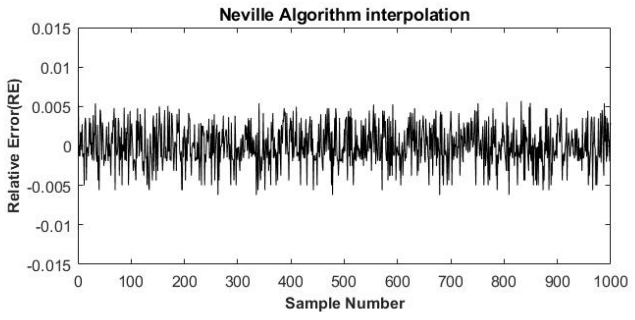
(b)



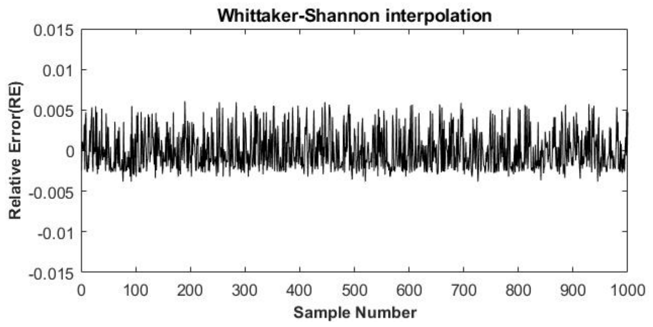
(c)



(d)



(e)



(f)

Figure 5. Relative error (MSE) values for different interpolation method for 1000 sample data.

From Figure 5 it is observed that the relative error is highest for near and linear interpolation and it is less for Lagrange's and Newton raphson interpolation. The relative error for near and linear interpolation is same because in these technieues the derived intermidiate points are always less than the actual points in case of peak measurement. Because RE is same, the value of MSE and MAD are also same for Near and Linear interpolation. The RE value for cubic interpolation is more than the Neville's and Whittaker shannon interpolation technique.

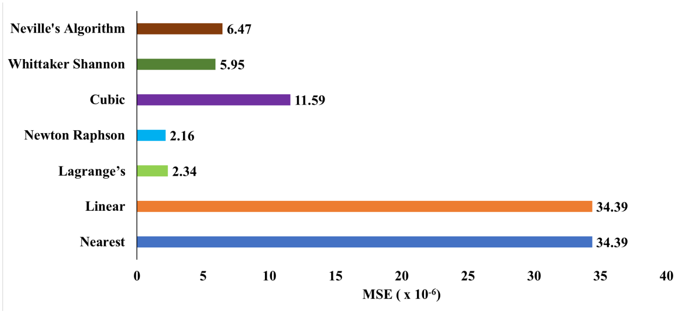


Figure 6. Mean square error (MSE) values for different interpolation method for 1000 sample data

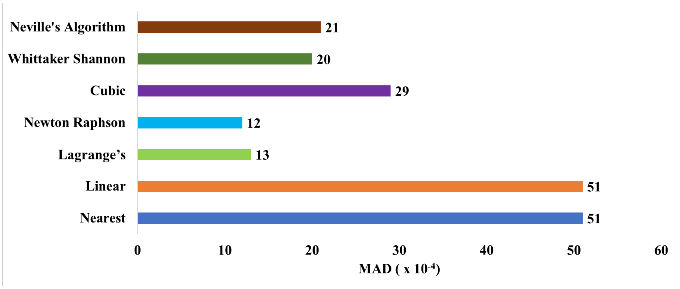


Figure 7. Mean absolute deviation (MAD) values for different interpolation method for 1000 sample data

From the Figure 6 and Figure 7 data, it is observed that the MSE and MAD is lowest for the Lagrange's and Newton Raphson methods. MSE is 2.34 x 10^{-6} and MAD is 13 x 10^{-4} for Lagrange's interpolation method. MSE is 2.16 x 10^{-6} and MAD is 12 x 10^{-4} for Newton Raphson interpolation method which are very close to Lagrange's interpolation method.

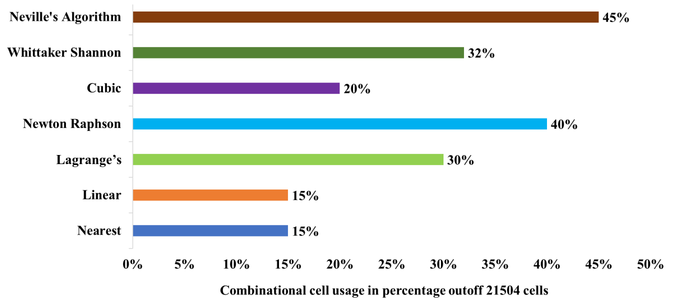


Figure 8: Combinational cell utilization of A3PE1500 FPGA for different interpolation algorithm

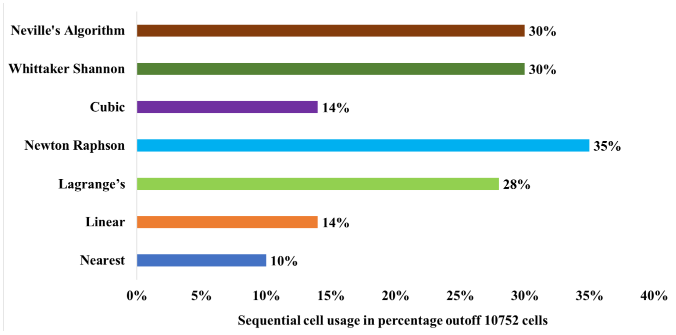


Figure 9: Sequential cell utilization of A3PE1500 FPGA for different interpolation algorithm

In terms of FPGA utilization as shown in Figure 8 and Figure 9, Neville's Algorithm and newton Raphson interpolation requires highest number of logic cells and the nearest interpolation method require lowest number of logic cells because the function is less complex. The hardware utilization is more for newton Raphson method compared to Lagrange's method.

4 CONCLUSION

This paper compares seven interpolation techniques for measuring the peak height of analogue pulses: Near, Linear, Cubic, LaGrange's, Newton Raphson, Whittaker Shannon, and Neville's. Performance compression has been done based on hardware resource usage, Mean Square Error (MSE), and Mean Absolute Deviation (MAD). MSE on FPGA for Near, Linear, Cubic, LaGrange's, Newton Raphson, Whittaker Shannon, and Neville's interpolation methods were determined to be 34.3 x 10^{-6}, 34.3 x 10^{-6}, 11.59 x 10^{-6}, 2.34 x 10^{-6}, 2.16 x 10^{-6}, 5.95 x 10^{-6}, and 6.47 x 10^{-6} per thousand equal amplitude gaussian samples investigated. It was observed that the nearest interpolation uses minimum hardware, Neville's Algorithm and newton Raphson interpolation uses maximum resources and Lagrange's method uses moderate hardware resources.

ACKNOWLEDGMENT

We acknowledge the financial support by the Indian Space Research Organization, Department of Space, Government of India. We also thank Dr. Y B acharya for discussions on the interpolation techniques. This work is a part of the Ph.D. work of Arpit Patel. Director PRL, Head of Planetary Science Division, PRL, and Dean DDU, Nadiad are gratefully acknowledged for constant encouragement during the work.

REFERENCES

[1] Bartels, R., Beatty, J., & et al. (1998). Hermite and Cubic Spline Interpolation. An Introduction to Splines for Use in Computer Graphics and Geometric Modelling, 9-17.

[2] H. Press, W., P. Flannery, B., A. Teukolsky, S., & T. Vetterling, W. (1992). Cubic Spline Interpolation. Numerical Recipes in FORTRAN: The Art of Scientific Computing, 107-110.

[3] Sonowal, D., & Bhuyan, M. (2013). Linearizing Thermistor Characteristics by Piecewise Linear Interpolation in Real Time FPGA. Inter-national Conference on Advances in Computing Communications and Informatics (ICACCI) IEEE, 1976-1980.

[4] Nenova, Z., & Nenov, T. (2009). Linearization circuit of the thermistor connection. IEEE Transactions on Instrumentation and Measurement, 58(2), 441-449.