Journal of Engineering and Science Research 2 (4): 12-19, 2018
e-ISSN: 2289-7127
© RMP Publications, 2018
DOI: 10.26666/rmp.jesr.2018.4.3

Polynomial Interpolation in Matlab

Siti Hawa Binti Aziz and Zuliana Bt Abdul Mutalib
Mathematics, Science & Computer Department, Politeknik Ungku Omar,
Jalan Raja Musa Mahadi, 31400 Ipoh, Perak, Malaysia.

Abstract: The problem of constructing such a continuous function is called data fitting. Many times, data given only at discrete points. With interpolation, we seek a function that allows us to approximate *f(x)* such that functional values between the original data set values may be determined. The process of finding such a polynomial is called interpolation and one of the most important approaches used are Lagrange interpolating formula. In this study, researcher determining the polynomial interpolation by using Lagrange interpolating formula. Then, a mathematical modelling was built by using MATLAB programming to determine the polynomial interpolation for a given points using the Lagrange method. The result of the study showed that the manual calculating and the MATLAB mathematical modelling will give the same answer for evaluated *x* and graph.

Key words: *Data fitting, Polynomial, Interpolation, Lagrange interpolating formula, MATLAB*

INTRODUCTION

The problem of constructing a continuously defined function from a given discrete data is unavoidable whenever one wishes to manipulate the data in a way that requires information not included explicitly in the data. Interpolation, by polynomials or other functions, is a rather old method in applied mathematics. More generally, the process of reconstructing a curve, surface, or any other geometric object from certain known data is called interpolation, a word that is derived from the Latin word interpolare which means "to refurbish" or "to patch" [1]. Gasca mention that it is already indicated by the fact that, apparently, the word 'interpolation' itself has been introduced by J. Wallis as early as 1655 [2]. Compared to this, polynomial interpolation in several variables is a relatively new topic and probably only started in the second-half of the last century with work in. In view of its increasing relevance, it is only natural that the subject of interpolation is receiving more and more attention these days. Sir Edmund Whittaker, a professor of Numerical Mathematics at the University of Edinburgh from 1913 to 1923, said "the most common form of interpolation occurs when we seek data from a table which does not have the exact values we want" [3]. Many problems concerning the applications of neural networks, such as in pattern recognition and systems control, can be converted into the ones of approximating multivariate functions by the superposition of activation functions of the neural networks, for which an extensive study on approximation by neural networks has been carried out in a huge literature [4].

There are four types of interpolation such as piecewise constant interpolation, linear interpolation, polynomial interpolation and spline interpolation. In some sense, polynomials are the simplest type of interpolates to work with, as their definition only involves a finite number of additions, subtractions, and multiplications. The fact that polynomial interpolants can suffer from Runge's phenomenon (see Figure 1) has given them a slightly bad reputation. In general, it is not wise to use high-degree interpolating polynomial, and equal-spaced interpolation points to

Corresponding Author: Siti Hawa Aziz, Mathematics, Science & Computer Department, Politeknik Ungku Omar, Jalan Raja Musa Mahadi, 31400 Ipoh, Perak, Malaysia., 0195599442

approximate a function on an interval [a,b] unless the interval is sufficiently small. The Figure 1 is a well-known example of the difficulty of high-degree polynomial interpolation using equally-spaced points, and it is known as Runge's example. The simplicity of the polynomial, however, makes them perfectly suitable to be used as the building blocks of other interpolating functions with better behavior.

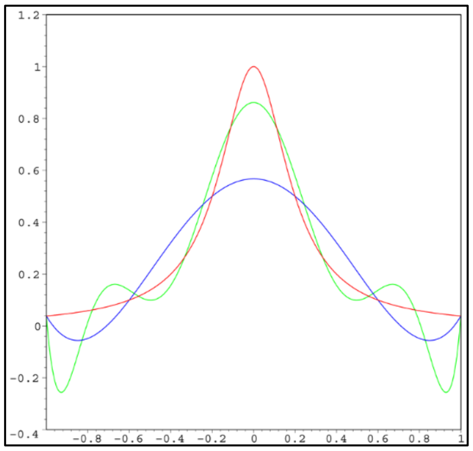


Figure 1 The red curve is the Runge function.The blue curve is a 5th-order interpolating polynomial (using six equally spaced interpolating points).The green curve is a 9th-order interpolating polynomial (using ten equally spaced interpolating points). At the interpolating points, the error between the function and the interpolating polynomial is (by definition) zero. Between the interpolating points (especially in the region close to the endpoints 1 and −1), the error between the function and the interpolating polynomial gets worse for higher-order polynomials.

The two most important approaches in interpolation are Newton's interpolating formula and Lagrange's interpolating formula. In this study, researcher will show only the Lagrange approaches which can be introduced and developed at the precalculus level in the context of fitting polynomials to data.

For this study, researcher use the MATLAB 7.8.0 (R2009a) which doesn't have a Lagrange function for polynomial interpolation. This version has many coding for polynomial but doesn't have specific functions for Lagrange. As the method of Lagrange polynomials is not suited towards numeric computation, it is not implemented in MATLAB. So, to solve this problem, researcher design a command depend on manual calculation of Lagrange Interpolation. It used the basic built-in command such as *loop* and *plot*; and compatible it with another command. Lagrange interpolating polynomials are implemented in the MATLAB as LARGRANGEPOLY and this command only have in certain version of MATLAB.

LITERATURE REVIEW

According to Meijering, the problem of interpolation by finite or divided differences had been studied at the beginning of the 20th century by astronomers, mathematicians, statisticians, and actuaries and most of the now well-known variants of Newton's original formulae had been worked out [3]. There are many researches that have been done about the polynomial interpolation. A new approach to multivariate Lagrange interpolation by polynomials via finite differences has been given and leading to, algorithms for the practical computation of interpolating polynomials [5]. These algorithms cover both aspects of classical univariate polynomial interpolation for example in computation of the Lagrange fundamental polynomials as well as a Newton method. There are many researches that have been done about the polynomial interpolation. Polynomial interpolation has been used to solve many problems in Mathematics. T.Yabe and T.Aoki from Japan have developed a new universal solver for hyperbolic equations by using a cubic-polynomial interpolation [6].

Lane concerned with a practical method for fitting an ordered set of data in space with a free-form curve, with no specific function or parameterization given for the data. Problems such as this arise routinely in a variety of disciplines from the Arts to Engineering and Science. The techniques presented in the thesis is for data in 2 plane (2 dimensions), but can be adapted to many dimensions [7]. He also has implement algorithms in MATLAB to further explore the feasibility of an automated routine which will examine an ordered set of data and, with possible user interaction, produce a fitted curve within specified conditions and tolerances. In considering the problem, he seeks to fit a G1 cubic Bezier curve to the ordered set of data using least squares approximation.

In other research, Saeur focuses on speed and robustness of the both algorithms which is Lagrange and Newton method [5]. From his research, he found that the smoothness of the interpolated function can hardly affect the Lagrange method. Wang also has done a study on interpolation which is cubic spline interpolation [8]. Besides studying the cubic spline interpolation and its applications in numerical analysis such as representing functions by approximating polynomial and data correlation, he also did cubic splines interpolation with simulations in MATLAB.

According to Vikstrom, there are several benefits of starting the algorithm development process in MATLAB and then ending it in C. MATLAB offers a wide selection of functions, automatic memory handling of variables and other interesting features that allows the engineer to focus on the function of the algorithm instead of the practical implementation. MATLAB also simplifies the algorithm testing process with its ability to easily produce plots and reports. However, the process to translate the MATLAB code to a language which is more suitable for hardware implementation such as C is time consuming and error prone [9].

A research has been done on the practical implementation of the two methods which is Newton and Lagrange to compute the solution of polynomial interpolation problems. Wilhelm Werner in his journal has shown that the Lagrangian form of the interpolating polynomial can be calculated with the same number of arithmetic operations as the Newtonian form [10]. For a small degrees of the interpolating polynomial$(n \ll 20)$, the new interpolation algorithm introduced in this paper as satisfactory as Newton interpolation does. In critical situations, however, where interpolation polynomials of very high degree must be evaluated, both algorithms require a special arrangement of the interpolating points to avoid numerical instabilities [10].

.

Farea has used the MATLAB to solve a polynomial with degree 5 in her thesis paper [11]. It has shown that by using MATLAB, we can calculate the roots of eight degree polynomial which cannot be calculated manually since there is no radical formula. An obvious fact, however, is that the MATLAB routines are easier to handle and therefore are much more suitable for the casual user who is mainly interested in interactively experimenting with polynomial interpolation without having to write and compile a program [2]. On the other hand, the C++ routines in MPI are much more trimmed for efficiency, but in order to apply them clearly a certain background in programming is needed to use classes in an appropriate way.

MOTIVATION OF THE STUDY

1. This study explores the usage of Lagrange in solving univariate polynomial fitting.
2. Developing mathematical modeling using MATLAB software which can insert input and get the polynomial fitting and a graph instantly. So, by using this mathematical modeling, students will understand about the polynomial interpolation.
3. MATLAB programming is a very simple and user friendly software compare to C and FORTRAN. The flexibility of MATLAB GUI to create applets was the reason to this choice. So, by using MATLAB in this study, researcher can explore furthermore about polynomial such as solving the polynomial with a given point by solving a matrix and Lagrange.
4. In this study, researcher does not use the built-in function in MATLAB for Lagrange or Polynomial, but the researcher design the command from the manual calculation. The polynomial with points given can be shown in a graph which has been designed by researchers.
5. This application will help the students to have a clear view about what is a polynomial interpolation, how to solve it by using Matrix method and Lagrange; and the graph of the polynomial interpolation.

LAGRANGE INTERPOLATING POLYNOMIAL DEGREE 1

The manual calculation to find Lagrange interpolating polynomial if two points are given is shown below:

x	2	-3
f(x)	5	8

Since the points given is 2 points, so we need to calculate $L_0(x)$ and $L_1(x)$.

$$L_0(x) = \frac{(x-x_1)}{(x_0-x_1)} = \frac{(x-(-3))}{2-(-3)} = \frac{x+3}{5}$$

$$L_1(x) = \frac{(x-x_0)}{(x_1-x_0)} = \frac{(x-2)}{-3-2} = \frac{x-2}{-5}$$

We know the equation for $P(x)$ is,

$$P(x) = f_0 L_0(x) + f_1 L_1(x)$$

Substitute the value of $L_0(x)$, $L_1(x)$, f_0 and f_1

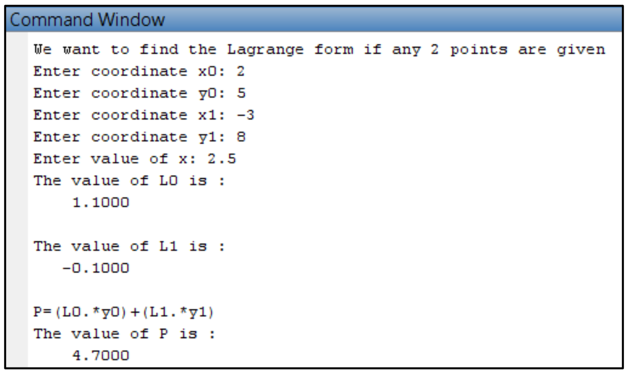
$$P(x) = (5)\left(\frac{x+3}{5}\right) + (8)\left(\frac{x-2}{-5}\right)$$

Let *x=2.5*. Substitute the value of x in the equation of $P(x)$.

$$P(x) = (5)\left(\frac{2.5+3}{5}\right) + (8)\left(\frac{2.5-2}{-5}\right) = 4.700$$

Then, we get the answer for $P(x)$ is 4.700.

Then, the M-file for Lagrange interpolating polynomial was built. The results after we run the M-file are shown in Figure 2 and Figure 3. Figure 2 show the command window in MATLAB after run the M-file. As we can see, we can insert the points and the x value. Then it will calculate value of L_0, L_1 and P(x). Figure 3 show the graph generated which is just a straight line and the points inserted is shown in the graph.



```
Command Window
We want to find the Lagrange form if any 2 points are given
Enter coordinate x0: 2
Enter coordinate y0: 5
Enter coordinate x1: -3
Enter coordinate y1: 8
Enter value of x: 2.5
The value of L0 is :
    1.1000

The value of L1 is :
   -0.1000

P=(L0.*y0)+(L1.*y1)
The value of P is :
    4.7000
```

Figure 2 The command window in MATLAB for Lagrange Interpolating Polynomial Degree 1

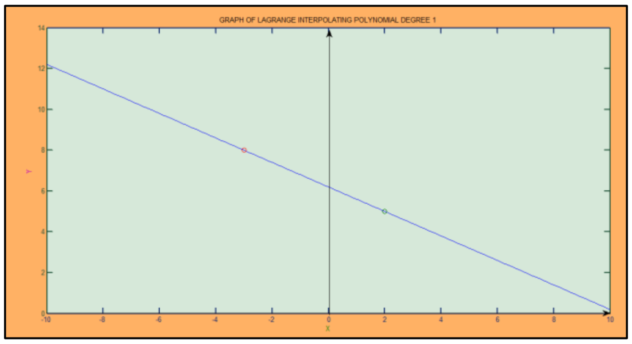


Figure 3 The graph generated in MATLAB for Lagrange Interpolating Polynomial Degree 1

LAGRANGE INTERPOLATING POLYNOMIAL DEGREE 2

The manual calculation to find Lagrange interpolating polynomial if three points are given is shown below:

x	1	6	-2
f(x)	8	4	5

Since the points given are 3 points, so we need to calculate $L_0(x)$, $L_1(x)$ and $L_2(x)$.

$$L_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{(x-6)(x-(-2))}{(1-6)(1-(-2))} = \frac{(x-6)(x+2)}{-15}$$

$$L_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = \frac{(x-1)(x-(-2))}{(6-1)(6-(-2))} = \frac{(x-1)(x+2)}{40}$$

$$L_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{(x-1)(x-6))}{(-2-1)(-2-6)} = \frac{(x-1)(x-6)}{24}$$

The equation for $P(x)$ is as below

$$P(x) = f_0 L_0(x) + f_1 L_1(x) + f_2 L_2(x).$$

Substitute the value of $L_0(x)$, $L_1(x)$, $L_2(x)$, f_0, f_1 and f_2.

$$P(x) = (8)\left(\frac{(x-6)(x+2)}{-15}\right) + (4)\left(\frac{(x-1)(x+2)}{40}\right) + (5)\left(\frac{(x-1)(x-6)}{24}\right)$$

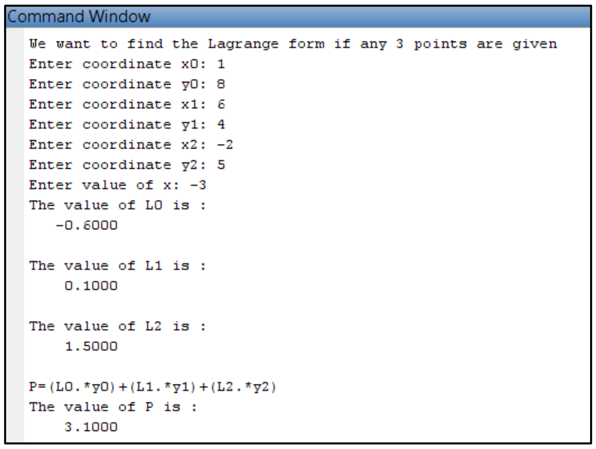
Let *x=-3*. Substitute the value of x in the equation of *P(x)*

$$P(x) = (8)\left(\frac{(-3-6)(-3+2)}{-15}\right) + (4)\left(\frac{(-3-1)(-3+2)}{40}\right) + (5)\left(\frac{(-3-1)(-3-6)}{24}\right)$$

$$P(x) = 3.100$$

Then, we get the answer for $P(x)$ is 3.100.

Then, we convert the calculation into MATLAB command and the results after we run the M-file are shown in Figure 4 and Figure 5. Figure 4 show the command window in MATLAB for the M-file. As we can see, we can insert the points and the x value. Then it will calculate value of L_0, L_1, L_2 and $P(x)$. Figure 5 show the graph generated which is either maximum or minimum graph and the points inserted is shown in the graph.



```
Command Window
  We want to find the Lagrange form if any 3 points are given
  Enter coordinate x0: 1
  Enter coordinate y0: 8
  Enter coordinate x1: 6
  Enter coordinate y1: 4
  Enter coordinate x2: -2
  Enter coordinate y2: 5
  Enter value of x: -3
  The value of L0 is :
     -0.6000

  The value of L1 is :
      0.1000

  The value of L2 is :
      1.5000

  P=(L0.*y0)+(L1.*y1)+(L2.*y2)
  The value of P is :
      3.1000
```

Figure 4 The command window in MATLAB for Lagrange Interpolating Polynomial Degree 2

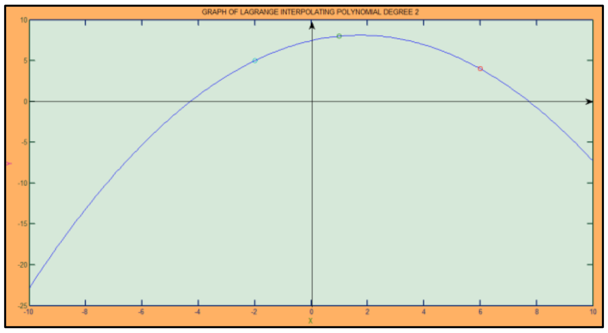


Figure 5 The graph generated in MATLAB for Lagrange Interpolating Polynomial Degree 2

LAGRANGE INTERPOLATING POLYNOMIAL DEGREE 3

The manual calculation to find Lagrange interpolating polynomial if four points are given is shown below :

x	1	0	3	-2
f(x)	2	2	5	-3

Since the points given are 4 points, so we need to calculate $L_0(x)$, $L_1(x)$, $L_2(x)$ and $L_3(x)$.

$$L_0(x) = \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} = \frac{(x-0)(x-3)(x-(-2))}{(1-0)(1-3)(1-(-2))} = \frac{x(x-3)(x+2)}{-6}$$

$$L_1(x) = \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} = \frac{(x-1)(x-3)(x-(-2))}{(0-1)(0-3)(0-(-2))} = \frac{(x-1)(x-3)(x+2)}{6}$$

$$L_2(x) = \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} = \frac{(x-1)(x-0)(x-(-2))}{(3-1)(3-0)(3-(-2))} = \frac{(x-1)(x)(x+2)}{30}$$

$$L_3(x) = \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} = \frac{(x-1)(x-0)(x-3)}{(-2-1)(-2-0)(-2-3)} = \frac{(x-1)(x)(x-3)}{-30}$$

The equation for $P(x)$ is as below

$$P(x) = f_0L_0(x) + f_1L_1(x) + f_2L_2(x) + f_3L_3(x)$$

Substitute the value of $L_0(x)$, $L_1(x)$, $L_2(x)$, $L_3(x)$ f_0, f_1, f_2 and f_3.

$$P(x) = (2)\left(\frac{(x)(x-3)(x+2)}{-6}\right) + (2)\left(\frac{(x-1)(x-3)(x+2)}{6}\right) + (5)\left(\frac{(x-1)(x)(x+2)}{30}\right) + (-3)\left(\frac{(x-1)(x)(x-3)}{-30}\right)$$

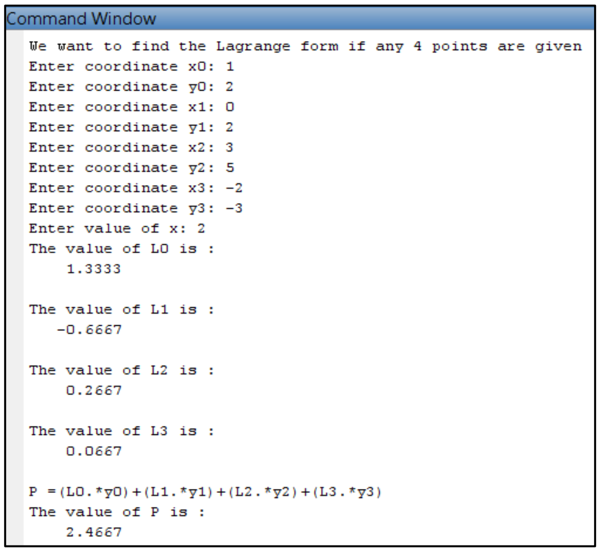
Let x=2, substitute the value of x in the equation of $P(x)$.

$$P(x) = (2)\left(\frac{(2)(2-3)(2+2)}{-6}\right) + (2)\left(\frac{(2-1)(2-3)(2+2)}{6}\right) + (5)\left(\frac{(2-1)(2)(2+2)}{30}\right) + (-3)\left(\frac{(2-1)(2)(2-3)}{-30}\right)$$

$$P(x) = \frac{37}{15} = 2.4667$$

Then, we get the final answer is 2.4667.

Then, we convert the calculation into MATLAB command. The results after we run the M-file are shown in Figure 6 and Figure 7. Figure 6 show the command window in MATLAB for the M-file. As we can see, we can insert the points and the x value. Then it will calculate value of L_0, L_1, L_2, L_3 and $P(x)$. Figure 7 show the graph generated and the points inserted is shown in the graph.



```
We want to find the Lagrange form if any 4 points are given
Enter coordinate x0: 1
Enter coordinate y0: 2
Enter coordinate x1: 0
Enter coordinate y1: 2
Enter coordinate x2: 3
Enter coordinate y2: 5
Enter coordinate x3: -2
Enter coordinate y3: -3
Enter value of x: 2
The value of L0 is :
    1.3333

The value of L1 is :
   -0.6667

The value of L2 is :
    0.2667

The value of L3 is :
    0.0667

P =(L0.*y0)+(L1.*y1)+(L2.*y2)+(L3.*y3)
The value of P is :
    2.4667
```

Figure 6 Command window in MATLAB for Lagrange Interpolating Polynomial Degree 3

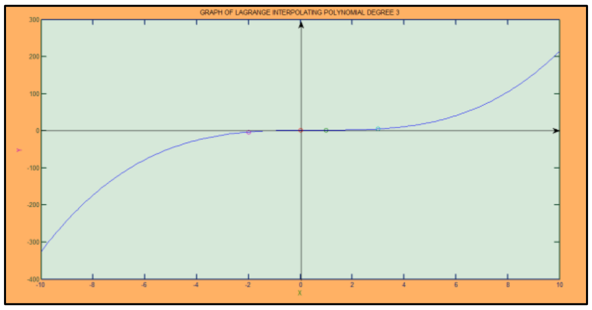
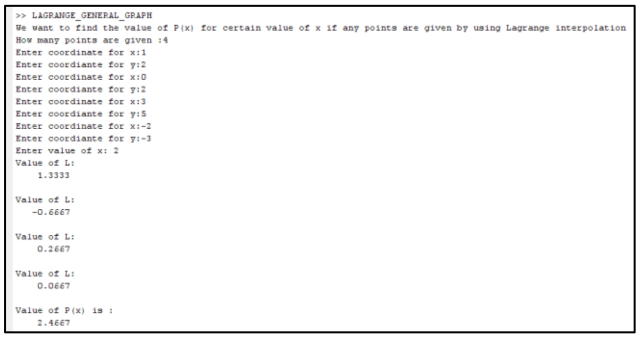


Figure 7 The graph generated in MATLAB for Lagrange Interpolating Polynomial Degree 3

Since the above model is limited for a certain points, so, we build another mathematical modeling which is more general and suitable for any points. The results after we run the M-file are shown in Figure 8 and Figure 9. Figure 8 show the command window in MATLAB for the M-file. As we can see, we can insert the k points and the x value. Then it will calculate value of $L_1 \ldots \ldots\ L_k$ and $P(x)$. Figure 9 show the graph generated and the point inserted is shown in the graph.



```
>> LAGRANGE_GENERAL_GRAPH
We want to find the value of P(x) for certain value of x if any points are given by using Lagrange interpolation
How many points are given :4
Enter coordinate for x:1
Enter coordiante for y:2
Enter coordinate for x:0
Enter coordiante for y:2
Enter coordinate for x:3
Enter coordiante for y:5
Enter coordinate for x:-2
Enter coordiante for y:-3
Enter value of x: 2
Value of L:
    1.3333

Value of L:
   -0.6667

Value of L:
    0.2667

Value of L:
    0.0667

Value of P(x) is :
    2.4667
```

Figure 8 Command window in MATLAB for General Lagrange Interpolation

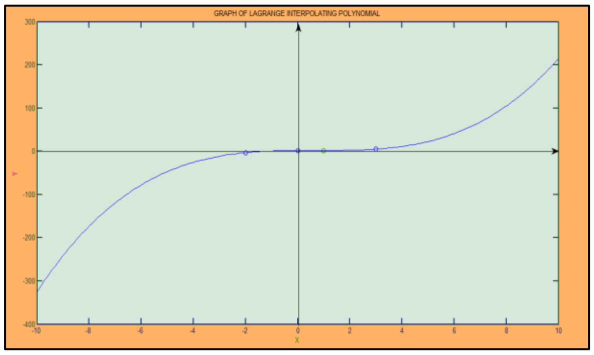


Figure 9 Graph generated in MATLAB for General Lagrange Interpolation

DISCUSSION

In this study, we show the MATLAB coding and the result for a method to interpolate data which is Lagrange Interpolation polynomial. Before we write the command in MATLAB, we need to do a manual calculation so that we can define the general function of Lagrange for each number of given points. The manual calculation by using Lagrange is shown. Then, we convert each calculation into MATLAB coding.

In the MATLAB coding, we didn't use the specific built-in Lagrange command, but we compatible the manual calculation into a simple built-in command in MATLAB. The most important command that we are using for each degree of polynomial are *input* where we can insert the points and value of *x* that we want to interpolate and *for loops* which allow a group of commands to be repeated for a fixed, predetermined number of times. *for loops* command is important as we want to plot the function into a graph.

In above section, it shows the step by step how to solve the Lagrange from degree 1 until degree 3 by manual calculation and MATLAB coding. Equally, we can determine, both will give the same result for *y* or *P*. The Lagrange method is more complicated as we require to work out the value of L_n and substitute in the equation:

$$P(x) = f_0 L_0(x) + f_1 L_1(x) + f_2 L_2(x) + f_3 L_3(x) + ...$$

So, by using this application, we can calculate the value of L_n for a given points. It also show the given points on the graph with the 'o' mark.

As we can see from Figure 2, Figure 4 and Figure 6, every model only suits for a certain point. This is complicated as we want to save a time and get the answer quickly. So, to solve this problem, we build another mathematical modeling which can suit any number of points. It is shown in Figure 8. By using this model, we can solve Lagrange Interpolation Polynomial problems for any points. This model uses a more complicated command in MATLAB. We are still using the *for loops* command, but this time it was used more than three times in the command. Besides that, we also used the *if-else-end* command. This is because, the sequences of the commands must be conditionally evaluated on the basis of a relational test. The *command* between *if* and *end* statements are evaluated if all elements in *expression* are True (nonzero). The model also generate the graph for each Lagrange polynomial interpolation and hold the points inserted on the graph. So, this model will give a clear view for a students about the interpolation

CONCLUSION AND RECOMMENDATION

This mathematical modeling is more user friendly than the built-in command in MATLAB.We modified the command in MATLAB and make it more user friendly which is user or students just need to insert an input and it will automatically calculate the equation and show the graph. If we compare with the built-in command in MATLAB to plot a polynomial graph for two points, it is complicated as we need to rewrite the command every time we want to insert an input.

Same goes to Lagrange interpolation polynomial, if we calculate by using manual calculation, it will take a time and it was difficult to plot the graph instantly. So, by using the MATLAB model, it is more easier to determine the answer and time consuming. For all model, user can insert any points repeatedly without need to rewrite the command. However, there is some constraint when using all the model which is the value of *x* cannot be more than 10 or less than -10. This is because; we only limit the value of *x* in plotting graph from -10 till 10. So, this limitation can be solved by designing a model with wider limit of *x*.

The simplicity of Lagrange interpolating formulas makes it widely used results in all areas where there are underlying calculations based on data. Within Mathematics, the formulas provide a foundation for the development of methods in numerical integration and differentiation, approximation theory, and the numerical solution of differential equations. Consequently, they become very important results in the interpolation theory of numerical analysis. However, these ideas can also be valuable additions to a modern course in algebra or precalculus.

Moreover, curve fitting has become an important topic in modern algebra and precalculus classes, though it is usually approached almost exclusively from the perspective of regression. Polynomial regression is usually limited by the available technology such as graphing calculators up to fourth degree and Excel up to sixth degree.

At least, precalculus is a place where the Newton and Lagrange formulas can be investigated by setting a sequence of what-if questions when we discuss polynomial curve fitting. It is also a place to foster deep learning of Mathematics by using a number of topics together during the investigation.

There are some limitation or constraint by using Lagrange method. While the Lagrange polynomials are easy to compute, they are difficult to work with. Furthermore, if new interpolation points are added, all of the Lagrange polynomials must be recomputed. So, we hope, in the future, there is a mathematical modeling by using Newton methods which is more stable than Lagrange. Besides that, we may design this modeling using other popular programming such as Maple and etc.

ACKNOWLEDGMENTS

The authors acknowledge the supervision and support from Dr Ahmad Lutfi Amri Ramli from University of Science Malaysia (USM), Penang, Malaysia.

REFERENCES

[1]Skeat, W.W. 1911. A concise etymological dictionary of the English languange, Forgotten Books.

[2]Gasca, M., & Sauer, T. 2000. Polynomial interpolation in several variables. *Advances in Computational Mathematics*.**12**(4), 377-410.

[3]Meijering, E. 2002. A Chronology of Interpolation: From Ancient Astronomy to Modern Signal and Image Processing. In: *Proceedings of the IEEE*. vol. 90, no. 3, pp. 319-42. March 2002.

[4]Li, X. 2002. Interpolation by Ridge Polynomials and its application in neural networks. Journal of Computational and Applied Mathematics.144, 197-209.

[5] Saeur, T. & Xu, Y. 1995. On Multivariate Lagrange Interpolation. *Mathematics of Computation*. **64** (211) , 1147-1170.

[6] Yabe, T. & Aoki, T. 1991. A universal solver for hyperbolic-equations by cubic-polynomial interpolation. I. one-dimensional solver, Comput.Phys.Cpmmun,66(1991),219-232.

[7]Lane, E.J. 1995. Fitting Data Using Piecewise G1 Cubic Bezier Curves.Naval Postgraduate School Monterey.

[8]Wang, K. 2013. A study of cubic spline interpolation. *Rivier Academic Journal*.**9**.(2).

[9]Vikstrom, A. 2009. A study of automatic translation of MATLAB code to C code using software from the MathWorks. (Master), Lulea University of Technology.

[10]Werner, W. Polynomial Interpolation : Lagrange versus Newton. *Mathematics of computation*.**43**(167), 205-217

[11]Hussain, F.A. (1994),*Solving polynomial equations from 2000 B.C. through 20TH Century*. (Master), Oregon State University