Interpolating Polynomials

MAS 3114 Spring 2023, Prof. Curtin

By

Kanye Smith

Contents

1 Abstract

Throughout this project, we will explore the idea of using a set of linear equations to estimate data values between given data points.

2 Motivation

2.1 Main Problem

In this project, we will examine how the general curve of a polynomial function attempts to explain the behavior of data, as well as understand how to find the simplest form of the respective function. The text[6] provides us with the following data plot summation.

Seven data points are plotted on an (x,y) coordinate plane, ranging from [0,5] on the x-axis and [-1,1] on the y-axis. Let the red dots depict the data points, and let the blue curve represent the interpolation polynomial.

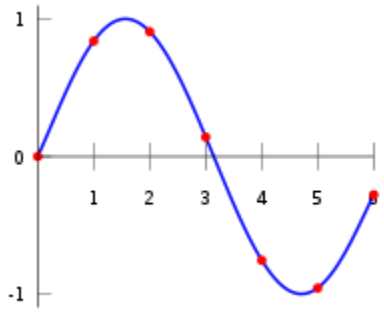


Figure 1: Interpolation Polynomial

By observing the overall curve of the graph, it is noticeable that there are gaps between the red data points. As such, it is impossible to know the exact coordinates within a gap. Nevertheless, we are able to estimate the points between the gaps in order to obtain a better understanding of the function behavior. The function $p(t)=a_0+a_1t+a_2t^2 + \ldots + a_nt^{n-1}$ represents the polynomial between a given interval, where the a_n coefficients are the points satisfied by the curve. As seen in the text[5], one method of obtaining the interpolating polynomial is by solving a system of linear equations through matrix reduction.

2.2 History and General Overview

The very idea of polynomial interpolation dates back as far as 1655[3] when mathematician and clergymen J. Wallis introduced “interpolation”, a term translated to acquiring a set of data as a

result of interpreting an existing set of data[3]. Prior to and throughout Wallis' exploration of interpolation, he used extensive research to built upon the concept of Cavalieri's method, a tool commonly associated with the calculation of three-dimensional volume and cross-sectional area. The text[1] reduced the extensive explanation of the method to a more simplified formulaic mindset, reading Volume= bh, where b represents the cross-sectional area and h represents the height. As such, by acknowledging the cause-effect relationship between variables b and h in regards to impacting the overall volume, Wallis' concept of interpolation, more specifically polynomial interpolation, is able to effectively establish a correlation between different "events" or criteria related to a data set.

3 Mathematical Description of Solutions

3.1 Problems 1 and 2

In the first problem, we are given four data points: (1,29), (-1,-35), (2, 31), and (-3,-19), which yield the individual "xval" and "b" matrices

```
xval =

     1
    -1
     2
    -3
```

```
b =

    29
   -35
    31
   -19
```

As we attempt to find a reducible coefficient matrix to get the a_n coefficients of the interpolated polynomial form, we consider the variable B to equal *[ones(4,1) xval xval.^2 xval.^3]*

(In which the Matlab syntax ones(4,1) produces a 4-by-1 matrix)

Thus, after using MATLAB to compute the coefficient matrix B, we receive

```
B =

     1     1     1     1
     1    -1     1    -1
     1     2     4     8
     1    -3     9   -27
```

When using Gaussian Elimination (or row reduction), the given vector "*b*" acts as the constants on the right side of the coefficient matrix "*B*" in order to prove that there indeed exists a unique solution to the polynomial passing through the four points. Utilizing the variable *M* to represent the entirety of the augmented matrix with the form M = [B b], MATLAB sets up the reduced M-matrix to equal:

M =

```
     1     0     0     0    -1
     0     1     0     0    36
     0     0     1     0    -2
     0     0     0     1    -4
```

Recall the two criteria for a system to have a unique solution:

1. The linear system is consistent.
2. There are not any free variables.

Since the reduced row echelon form (RREF) proves that each coefficient column contains a pivot associated with a constant, the augmented matrix can be deemed as having a consistent and unique solution.

$$p(t) = a_0 + a_1 t + a_2 t^2 + \ldots + a_{n-1} t^{n-1}$$

Moreover, considering the interpolated polynomial form, the first constant under RREF(A) takes the place of the coefficient a_0, making the other three constant values (36, -2, -4) take the respective places of a_1, a_2, a_3. As such, due to the coefficient a_3 being multiplied by t^3, the interpolating polynomial is that of the third degree.

Therefore, we get that the interpolating polynomial that passes through the four given points is $p(t)= -1 + 36t-2t^2-4t^3$.

We further explore the idea of extracting an interpolating polynomial from given data points in the second problem. However, we are tasked with finding a fourth degree polynomial that passes through (-1, 0.5), (0,1), (1,2), (2,4), and (3,8).

Taking a similar approach to the first problem, the variable B represents *[ones(5,1) xval xval.^2 xval.^3 xval.^4],* where *ones(5,1)* constructs a 5x1 matrix,
xval equals the vector,

```
    -1
     0
     1
     2
     3
```

and b equals the vector

```
    0.5000
    1.0000
    2.0000
    4.0000
    8.0000
```

ma

Thus, assuming that the variable M equals the augmented matrix [B b], MATLAB computes that the reduced row echelon form of matrix M is

```
    1.0000         0         0         0         0    1.0000
         0    1.0000         0         0         0    0.7083
         0         0    1.0000         0         0    0.2292
         0         0         0    1.0000         0    0.0417
         0         0         0         0    1.0000    0.0208
```

Adapting the solution to the interpolating polynomial form, we get that

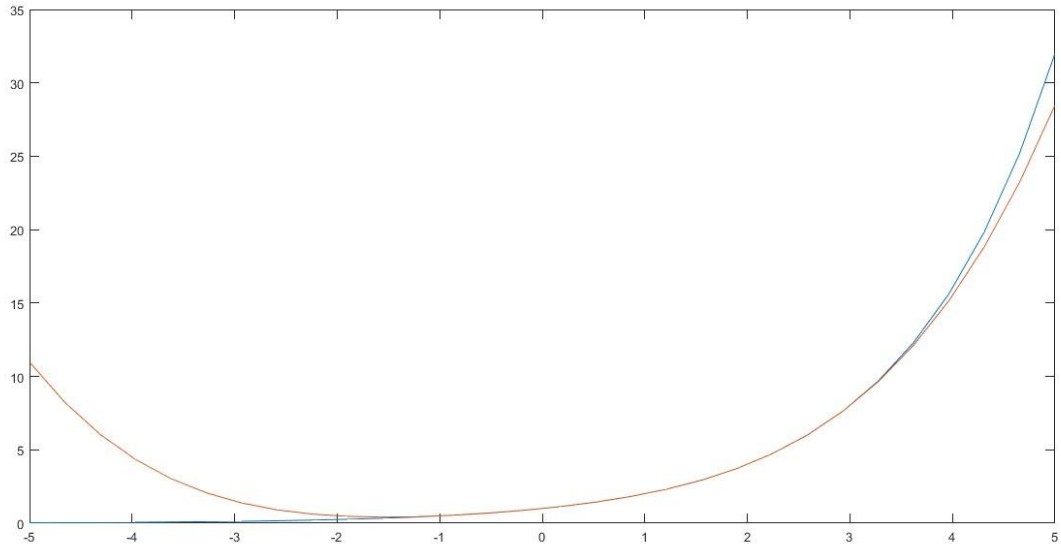
$$p(t)= 1 + .7083t + .2292t^2 + .0417t^3 + .0208t^4$$

While the first problem took a basic approach to understanding how to use data points to acquire a polynomial function, the second problem decides to expand the concept more-so into a 'line of best fit' or polynomial fitting mindset. With this mindset, rather than strictly estimating the values between given data points, we also track how the behavior of a calculated function relates to the behavior of a "base" function, much like how *line of best fit* strives to find a correlation between dependent/independent variables and uses and general line to assess the error of scattered points.

In this problem, we are using MATLAB to graph $p(t)= 1 + .7083t +.2292t^2 + .0417t^3 + .0208t^4$ on the same graph as $y(t)=2^t$. Thus, with the following commands, and translating the function p(t) to be equal to *z,*

```
>> t=linspace(-5,5,30);
  y=2.^t; z=1+.7083*t+.2292*t.^2+.0417*t.^3+.0208*t.^4;
  plot (t, y, t, z)
```

We graph 30 points between -5 and 5, and receive the graph:



As seen in the graph, the base function $y(t)=2^t$ takes the shape of a typical function while the interpolating polynomial takes the shape of a biquadratic equation. Strictly based on the graph placements, it appears that between the t-values -1 and 3, the two functions are laying directly on top of each other. However, to prove this theory, we can insert t=-1 into both function.

This gives $y(-1)=2^{-1}=0.5$ and p(-1)=1+.7083(-1)+.2292(-1)^2+.0417(-1)^3+.0208(-1)^4= 0.5. Therefore, the calculations show that the amount of "error" between t=-1 and 3 is 0. Nevertheless, this trend doesn't stick for the entirety of the graph because when looking above the interval, the polynomial's data values are less than that of y(t), and t-values below the interval show an increase among values of p(t) while y(t) gets closer to 0 without necessarily touching it.

3.2 Problem 3

We are now working with a new data set, and are trying to assess the roots of each derived polynomial. The table below provides the time (in seconds) that it takes each car to reach each velocity (30,60,90) from 0 mph.

Honda CR - VEX	Time	0	3.1	10.3	30.1
	Velocity	0	30	60	90
Jeep Cherokee SE	Time	0	3.2	12	38.2
	Velocity	0	30	60	90
Kia Sportage	Time	0	4.2	12.8	38.7
	Velocity	0	30	60	90
Subaru Forester L	Time	0	2.8	9.5	22.7
	Velocity	0	30	60	90
Toyota RAV4	Time	0	3.0	10.2	31.7
	Velocity	0	30	60	90

Table 2: Elapsed Time (seconds) and Velocities (m.p.h.) for various models of automobiles

Converting the given data into vectors, we get:

```
velocity =

     0
    30
    60
    90
```

```
kia =

         0
    4.2000
   12.8000
   38.7000
```

```
honda =

         0
    3.1000
   10.3000
   30.1000
```

```
subaru =

         0
    2.8000
    9.5000
   22.7000
```

```
jeep =

         0
    3.2000
   12.0000
   38.2000
```

```
toyota =

         0
    3.0000
   10.2000
   31.7000
```

Starting with Honda, we assume that the variable A equals *[ones(4,1) honda honda.^2 honda.^3 velocity],* and reduce the matrix in MATLAB:

```
    1.0000         0         0         0         0
         0    1.0000         0         0   11.7994
         0         0    1.0000         0   -0.7295
         0         0         0    1.0000    0.0145
```

Afterwards, we can use the roots command to find the real solution for the Honda 's acceleration time as it travels from 0 to 50 mph:

```
>> roots([0.0145 -0.7295 11.7994 -50])

ans =

  21.8879 + 6.9725i
  21.8879 - 6.9725i
   6.5346 + 0.0000i
```

Thus we find that the Honda's acceleration time for a velocity of 0-50 mph is approximately 6.53 seconds. sa

Using the same template, we can do the same MATLAB computation for the other 4 cars.

Jeep:

```
    1.0000         0         0         0         0
         0    1.0000         0         0   11.4006
         0         0    1.0000         0   -0.6692
         0         0         0    1.0000    0.0113
```

```
>> roots([0.0113 -0.6692 11.4006 -50])

ans =

   32.0494
   20.4062
    6.7656
```

Kia:

```
    1.0000         0         0         0         0
         0    1.0000         0         0    8.6448
         0         0    1.0000         0   -0.3813
         0         0         0    1.0000    0.0056
```

```
>> roots([0.0056 -0.3813 8.6448 -50])

ans =

  29.7002 +12.0620i
  29.7002 -12.0620i
   8.6888 + 0.0000i
```

Subaru:

```
    1.0000         0         0         0         0
         0    1.0000         0         0   13.1919
         0         0    1.0000         0   -0.9522
         0         0         0    1.0000    0.0240
```

```
>> roots([0.0240 -0.9522 13.1919 -50])

ans =

  16.8439 + 8.0157i
  16.8439 - 8.0157i
   5.9871 + 0.0000i
```

Toyota:

```
    1.0000         0         0         0         0
         0    1.0000         0         0   12.1745
         0         0    1.0000         0   -0.7698
         0         0         0    1.0000    0.0150
```

```
>> roots([0.0150 -0.7698 12.1745 -50])

ans =

  22.4978 + 4.5731i
  22.4978 - 4.5731i
   6.3243 + 0.0000i
```

Now, we are able to interpret each car's 0-50 mph acceleration time as being approximately:

Honda CR - VEX	Time(sec)	0	6.53
	Velocity(mph)	0	50
Jeep Cherokee SE	Time(sec)	0	6.77

	Velocity(mph)	0	50
Kia Sportage	Time(sec)	0	8.69
	Velocity(mph)	0	50
Subaru Forester L	Time(sec)	0	5.99
	Velocity(mph)	0	50
Toyota RAV4	Time(sec)	0	6.32
	Velocity(mph)	0	50

3.3 Problems 4 and 5

Question 4: As we transition to the 4th question, it is noted that our newly acquired data is to be kept for usage in this new context. Rather than finding the acceleration times for each car, we are now looking to find the approximate distance needed for each car to accelerate from 0 to 90 mph. Therefore, we are using the given data from the previous problem:

Vehicle	0 mph-90 mph
Honda CR - VEX	30.1
Jeep Cherokee SE	38.2
Kia Sportage	38.7
Subaru Forester L	22.7
Toyota RAV4	31.7

In order to calculate the distances for each car, we use the given integral

$\int_0^T v(t)/3600\,dt$ With T being each car's 0-90 mph time.

Since the actual velocity function *v(t)* is not known, we will use the reduced row echelon forms from question 3, and convert the constants in the augmented matrix into the interpolating polynomials

Honda: p(t) = 11.7994t-.7295t^2+.0145t^3
Jeep: p(t) = 11.4006t-.6692t^2+.0113t^3
Kia: p(t) = 8.6448t-.3813t^2+.0056t^3
Subaru: p(t) = 13.1919t-.9522t^2+.0240t^3
Toyota: p(t) = 12.1745t-.7698t^2+.0150t^3

Therefore, using the MATLAB commands fun and q, where fun represents the intepolating polynomial, and q = integral(fun,xmin,xmax), we get the following q values for each car:

Honda:

```
>> fun = @(t) (11.7994*t-.7295*t.^2+.0145*t.^3)/(3600);
>> q= integral(fun,0,30.1)

q =

    0.4693
```

Jeep:

```
>> fun = @(t) (11.4006*t-.6692*t.^2+.0113*t.^3)/(3600);
>> q= integral(fun,0,38.2)

q =

    0.5276
```

Kia:

```
>> fun = @(t) (8.6448*t-.3813*t.^2+.0113*t.^3)/(3600);
>> q= integral(fun,0,38.7)

q =

    1.5121
```

Subaru:

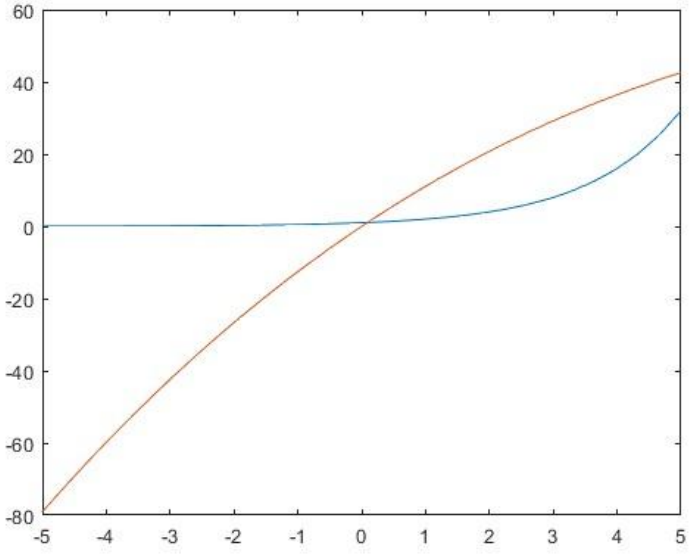
```
>> fun = @(t) (13.1919*t-.9522*t.^2+.0240*t.^3)/(3600);
>> q= integral(fun,0,22.7)

q =

    0.3554
```

Toyota:

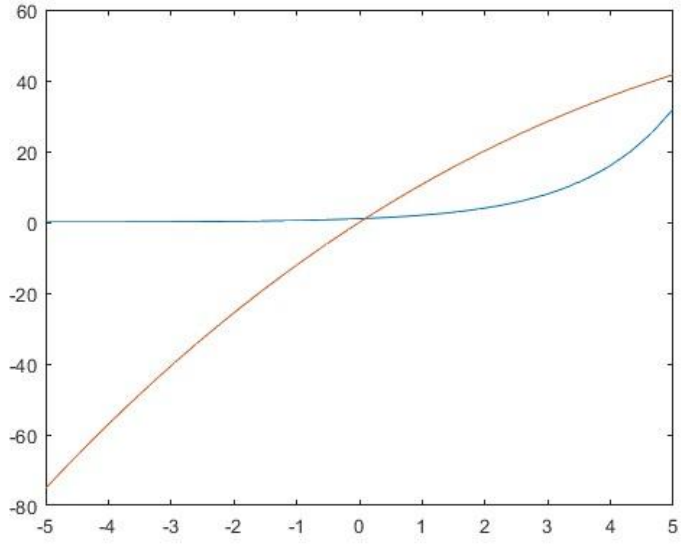
```
>> fun = @(t) (12.1745*t-.7698*t.^2+.0150*t.^3)/(3600);
>> q= integral(fun,0,31.7)

q =

    0.4805
```

In the final question, we are reverting back to our mindset in problem 2 by attempting to find a correlation between dependent and independent variables. Typically used in mathematical areas such as statistics, the “goodness of fit” method is an effective tool when using sample data to establish conclusions about an entire population. However, in the context of the project, it is to be used to understand how variables work together to impact a plot of data. Regarding this particular problem, we look to plot the interpolating polynomial functions against the original base function y=2^t in order to identify a potential relationship between elapsed time and vehicular velocity. As such, we dedicate each MATLAB graph to each car as follows:

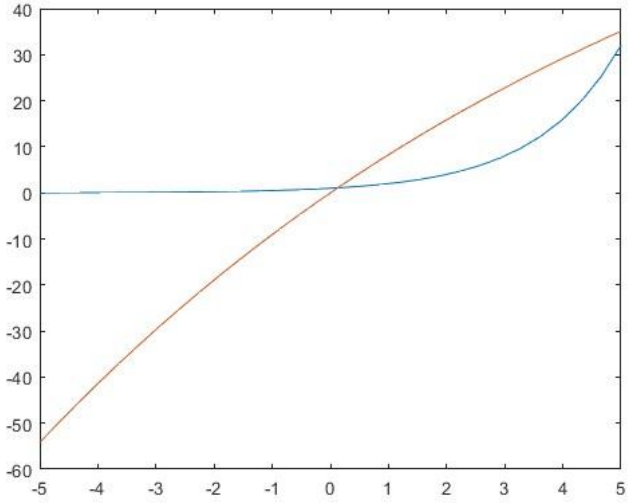
Honda:



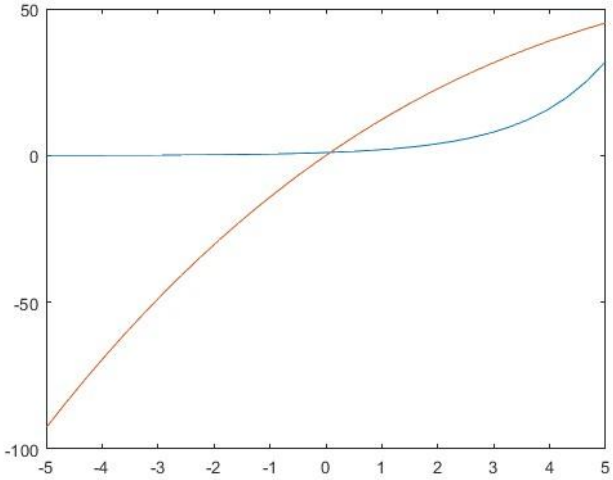
Jeep:



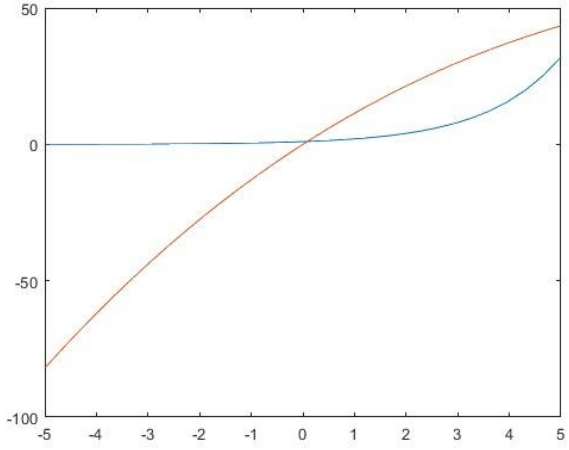
Kia:



Subaru:



Toyota:



When comparing the plots of each respective polynomial function against the base function y=2^t, it was quite difficult to acknowledge a clear difference between the graphs that could point out a correlation between elapsed time and velocity. Nevertheless, concerning the latter half of each graph as the t values approached 5, it was somewhat apparent that the functions with greater a_3 coefficients had greater p values when insert 5 for t within each p(t) function. Comparing the graphs to the 0-90 mph table used in Section 3, it was interesting that the graphs with lower p(5) values also had lower 0-90mph times. Moreover, the integration data from Problem 4 also proved that it took less distance for the "faster" vehicles to accelerate fromm 0-90mph. Therefore, given the correlation established amongst tables, graphs, and MATLAB outputs, it is reasonable to assume that there polynomial relationship between the two variables, time and velocity.

4 Discussions and Conclusions

The polynomial interpolation method has proven to be an effective method when attempting to estimate non-plotted data points.

As seen in Problem 2 where the graph of the calculated function was compared to the expected function plot, although the project articulates the importance of "best fit", it is also necessary to take note of the following suggestions. If given the opportunity to choose the data points used to construct the interpolating polynomial, more points would in turn make a higher-degree polynomial. Compared to lower degree polynomials which retain a more "flattened" and basic curve, a higher degree would improve the results of the data plot estimation. To illustrate this concept, the text provides the following imagery:

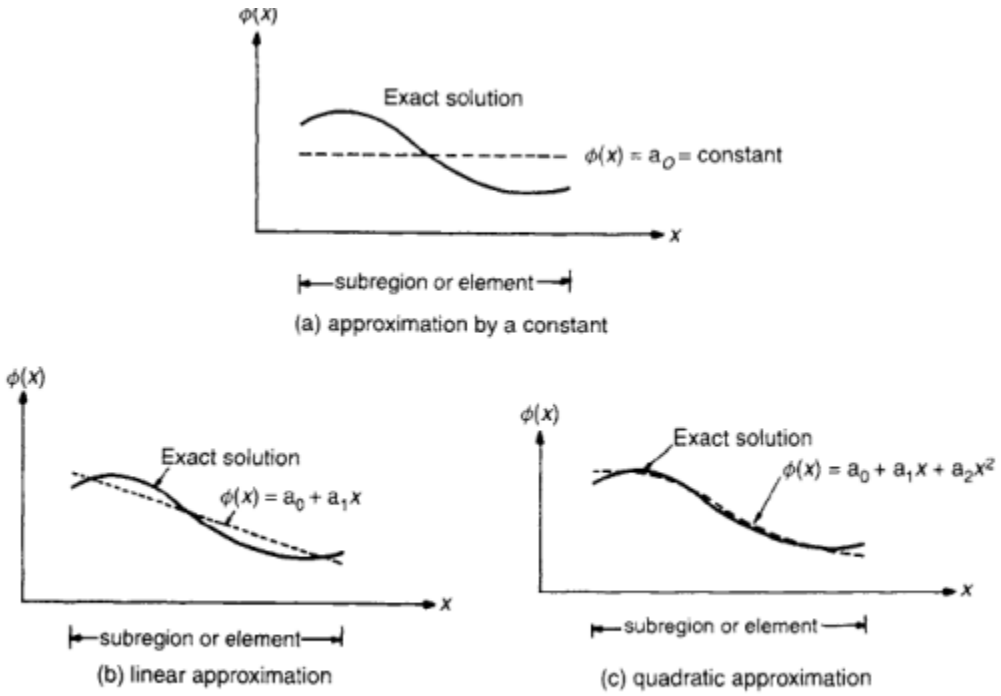


Figure: Polynomial Approximation in One Dimension.

Moreover, given the context of a particular problem, there are multiple iterations of polynomial interpolation that can be utilized for plot estimation. Considering Lagrange Interpolating Polynomials, this method is used more-so when estimating values on curves with coordinates. In contrast to the approach within Problem 2, where we were asked to interpret the difference of data points between t= -1 and t=3, the Lagrange Interpolation formula would allow for the approximation of more obscure data points across a multiple-axis coordinate plane.

$$P(x) = \sum_{j=0}^{n} y_j \left(\prod_{i=0, i \neq j}^{n} \frac{X - x_i}{x_j - x_i} \right)$$

5 Nomenclature

Variable/Unit	Meaning/Unit
ones	Creates a matlab matrix given the mxn values within the parentheses.
linspace	MATLAB command that returns an n amount of evenly spaced data points between a chosen interval
p(t)	Represents the interpolating polynomial function
roots	MATLAB command that returns the roots of a polynomial
fun	MATLAB variable that allows the user to input a desired function along with the variables within the function
velocity	mph
time	seconds

6 References

[1]Admin. (2020, April 15). *Lagrange interpolation Calculator - free online calculator*. BYJUS. Retrieved February 19, 2023, from https://byjus.com/lagrange-interpolation-calculator/

[2]Cavalieri's principle. Mathwords. (n.d.). Retrieved February 19, 2023, from https://www.mathwords.com/c/cavalieris_principle.htm#:~:text=Formula%3A,the%20height%20of%20the%20solid.

[3]Gasca, M., & Sauer, T. (2000, September 25). *On the history of multivariate polynomial interpolation*. Journal of Computational and Applied Mathematics. Retrieved February 19, 2023, from https://www.sciencedirect.com/science/article/pii/S0377042700003538#:~:text=Introduction,is%20claimed%20in%20%5B13%5D.

[4]John Wallis - Biography. Maths History. (n.d.). Retrieved February 19, 2023, from https://mathshistory.st-andrews.ac.uk/Biographies/Wallis/

[5]Linear Algebra and its Applications, 6th edition. David Lay, Judi McDonals, Steven Lay, Pearson (2020).

[6]Wikimedia Foundation. (2023, February 10). *Polynomial interpolation*. Wikipedia. Retrieved February 19, 2023, from https://en.wikipedia.org/wiki/Polynomial_interpolation#:~:text=In%20numerical%20analysis%2C%20polynomial%20interpolation,the%20points%20of%20the%20dataset.&text=Two%20common%20explicit%20formulas%20for,Lagrange%20polynomials%20and%20Newton%20polynomials.

7 Appendix

Problem 1:

```
A =

     1     1     1     1
     1    -1     1    -1
     1     2     4     8
     1    -3     9   -27


xval =

     1
    -1
     2
    -3


b =

    29
   -35
    31
   -19
```

```
To produce the coefficient matrix, try
the command B=[ones(4,1) xval xval.^2 xval.^3]
>> B=[ones(4,1) xval xval.^2 xval.^3]

B =

     1     1     1     1
     1    -1     1    -1
     1     2     4     8
     1    -3     9   -27
```

```
>> M= [B b];
>> rref(M)

ans =

     1     0     0     0    -1
     0     1     0     0    36
     0     0     1     0    -2
     0     0     0     1    -4
```

Problem 2

```
A =

     1    -1     1    -1     1
     1     0     0     0     0
     1     1     1     1     1
     1     2     4     8    16
     1     3     9    27    81


b =

    0.5000
    1.0000
    2.0000
    4.0000
    8.0000

>> B=A

B =

     1    -1     1    -1     1
     1     0     0     0     0
     1     1     1     1     1
     1     2     4     8    16
     1     3     9    27    81

>> M=[B b];
>> rref(M)

ans =

    1.0000         0         0         0         0    1.0000
         0    1.0000         0         0         0    0.7083
         0         0    1.0000         0         0    0.2292
         0         0         0    1.0000         0    0.0417
         0         0         0         0    1.0000    0.0208

>> t=linspace(-5,5,30);
  y=2.^t; z=1+.7083*t+.2292*t.^2+.0417*t.^3+.0208*t.^4;
  plot (t, y, t, z)
```

Problem 3:

```
velocity =

     0
    30
    60
    90


honda =

         0
    3.1000
   10.3000
   30.1000


jeep =

         0
    3.2000
   12.0000
   38.2000



kia =

         0
    4.2000
   12.8000
   38.7000


subaru =

         0
    2.8000
    9.5000
   22.7000


toyota =

         0
    3.0000
   10.2000
   31.7000
```

```
>> A = [ones(4,1) honda honda.^2 honda.^3 velocity]

A =

   1.0e+04 *

    0.0001         0         0         0         0
    0.0001    0.0003    0.0010    0.0030    0.0030
    0.0001    0.0010    0.0106    0.1093    0.0060
    0.0001    0.0030    0.0906    2.7271    0.0090

>> rref(A)

ans =

    1.0000         0         0         0         0
         0    1.0000         0         0   11.7994
         0         0    1.0000         0   -0.7295
         0         0         0    1.0000    0.0145

>> roots([0.0145 -0.7295 11.7994 -50])

ans =

  21.8879 + 6.9725i
  21.8879 - 6.9725i
   6.5346 + 0.0000i
```

```
>> A = [ones(4,1) jeep jeep.^2 jeep.^3 velocity]

A =

   1.0e+04 *

    0.0001         0         0         0         0
    0.0001    0.0003    0.0010    0.0033    0.0030
    0.0001    0.0012    0.0144    0.1728    0.0060
    0.0001    0.0038    0.1459    5.5743    0.0090

>> rref(A)

ans =

    1.0000         0         0         0         0
         0    1.0000         0         0   11.4006
         0         0    1.0000         0   -0.6692
         0         0         0    1.0000    0.0113

>> roots([0.0113 -0.6692 11.4006 -50])

ans =

   32.0494
   20.4062
    6.7656
```

```
>> A = [ones(4,1) kia kia.^2 kia.^3 velocity]

A =

   1.0e+04 *

    0.0001         0         0         0         0
    0.0001    0.0004    0.0018    0.0074    0.0030
    0.0001    0.0013    0.0164    0.2097    0.0060
    0.0001    0.0039    0.1498    5.7961    0.0090

>> rref(A)

ans =

    1.0000         0         0         0         0
         0    1.0000         0         0    8.6448
         0         0    1.0000         0   -0.3813
         0         0         0    1.0000    0.0056

>> roots([0.0056 -0.3813 8.6448 -50])

ans =

  29.7002 +12.0620i
  29.7002 -12.0620i
   8.6888 + 0.0000i
```

```
>> A = [ones(4,1) subaru subaru.^2 subaru.^3 velocity]

A =

   1.0e+04 *

    0.0001         0         0         0         0
    0.0001    0.0003    0.0008    0.0022    0.0030
    0.0001    0.0009    0.0090    0.0857    0.0060
    0.0001    0.0023    0.0515    1.1697    0.0090

>> rref(A)

ans =

    1.0000         0         0         0         0
         0    1.0000         0         0   13.1919
         0         0    1.0000         0   -0.9522
         0         0         0    1.0000    0.0240

>> roots([0.0240 -0.9522 13.1919 -50])

ans =

  16.8439 + 8.0157i
  16.8439 - 8.0157i
   5.9871 + 0.0000i
```

```
>> A = [ones(4,1) toyota toyota.^2 toyota.^3 velocity]

A =

   1.0e+04 *

    0.0001         0         0         0         0
    0.0001    0.0003    0.0009    0.0027    0.0030
    0.0001    0.0010    0.0104    0.1061    0.0060
    0.0001    0.0032    0.1005    3.1855    0.0090

>> rref(A)

ans =

    1.0000         0         0         0         0
         0    1.0000         0         0   12.1745
         0         0    1.0000         0   -0.7698
         0         0         0    1.0000    0.0150

>> roots([0.0150 -0.7698 12.1745 -50])

ans =

  22.4978 + 4.5731i
  22.4978 - 4.5731i
   6.3243 + 0.0000i
```

Problem 4:

```
>> fun = @(t) (12.1745*t-.7698*t.^2+.0150*t.^3)/(3600);
>> q= integral(fun,0,31.7)

q =

    0.4805

>> fun = @(t) (11.7994*t-.7295*t.^2+.0145*t.^3)/(3600);
>> q= integral(fun,0,30.1)

q =

    0.4693

>> fun = @(t) (11.4006*t-.6692*t.^2+.0113*t.^3)/(3600);
>> q= integral(fun,0,38.2)

q =

    0.5276

>> fun = @(t) (8.6448*t-.3813*t.^2+.0113*t.^3)/(3600);
>> q= integral(fun,0,38.7)

q =

    1.5121
```

```
>> fun = @(t) (13.1919*t-.9522*t.^2+.0240*t.^3)/(3600);
>> q= integral(fun,0,22.7)

q =

    0.3554
```

Problem 5:

```
>> t=linspace(-5,5,30);
  y=2.^t; z=11.7994*t-.7295*t.^2+.0145*t.^3;
  plot (t, y, t, z)
>> t=linspace(-5,5,30);
  y=2.^t; z=11.4006*t-.6692*t.^2+.0113*t.^3;
  plot (t, y, t, z)
>> t=linspace(-5,5,30);
  y=2.^t; z=8.6448*t-.3813*t.^2+.0113*t.^3;
  plot (t, y, t, z)
>> t=linspace(-5,5,30);
  y=2.^t; z=13.1919*t-.9522*t.^2+.0240*t.^3;
  plot (t, y, t, z)
>> t=linspace(-5,5,30);
  y=2.^t; z=12.1745*t-.7698*t.^2+.0150*t.^3;
  plot (t, y, t, z)
>> t=linspace(-5,5,30);
  y=2.^t; z=12.1745*t-.7698*t.^2+.0150*t.^3;
  plot (t, y, t, z)
```