Josh Miranda

Erik K. Grimmelmann, Ph.D

CSc 30100 – Scientific Programming

21 May 2021

Lagrange and Newton Polynomial Interpolation

The objective of this report is to further expand on topics that were discussed in class. These topics are the Lagrange and Newton's polynomial interpolation. The structure of this report will first discuss the history behind these two methods and how they were developed. Then discuss the real world applications and how they are used. Next the actual theorems themselves along with formulas. After a basic foundation is created, then there will be example problems which will be explained. For these problems they will be solved with python using the created algorithm of the Lagrange and Newton's methods for polynomial interpolation. Once the results are shown they will be compared to each other. Closing off the report will be an analysis of a dataset for the historical annual gold price of the United States.

Historical Context of Lagrange Formula

The Lagrange method was popularized by Joseph-Louis Lgrange who was a mathematician and astronomer. This method was published by him independently in 1795 but other known mathematicians in the field such as Isaac Newton, Edward Warring, and Leonhard Euler have created it earlier. It is said that Warring was the first to create this formula using a variation of Newton's divided differences which was 16 years earlier than Lagrange. However, Lagrange was unaware of Warring's work and published it regardless, giving him the credit for the formula. Another mathematician Carl Friedrich Gauss found a connection between the two and later proved that they were both equivalent. This in turn created another name for the formula called the Warring-Lagrange formula.

Historical Context of Newton's Formula

The Newton's polynomial interpolation formula is created by one of the most historical figures in math and sciences, Isaac Newton. This formula is also called the divided differences formula which is later built upon to create new theorems such as the Warring-Lagrange formula. One of the earlier general forms of interpolation was discovered in 1670 by James Gregory, a Scottish mathematician. Many variations of interpolation were created by other mathematicians at the time. However, Gregory and Newton created a variation that was similar to each other and it was later known as the Gregory-Newton formula. This was a popular formula at the time that was published in many books and newspapers. As more and more variations were developed with this formula Newton then invented the concept of divided differences.

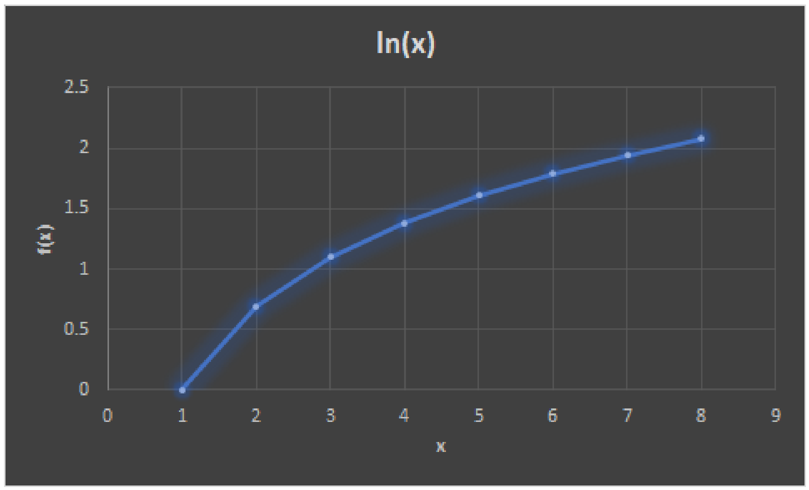
Real World Applications of Interpolation

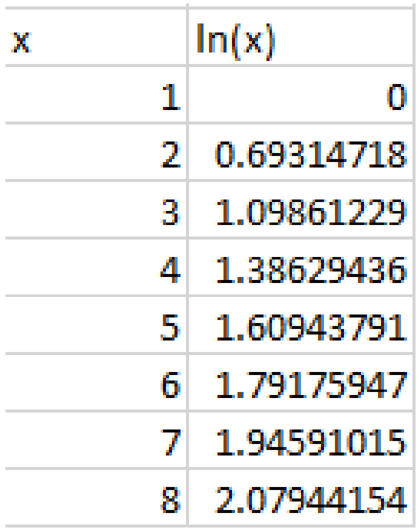
The definition of interpolation in math is to estimate points in a function through the use of already known values like a dataset. Applications of using interpolation can be found in 3D modeling, crypto mining and estimating data points over a time period. An interesting application that uses the Lagrange and Newton's formula is Shamir's Secret Sharing Scheme. It is a type of multi factor authentication that secures a secret. This secret is distributed in multiple parts and once all the parts are connected then the secret is able to be read. The similarities can be seen as for Lagrange and Newton a set of data points is inputted into the formula and that formula is able to fit in a polynomial best fitted for the data points. Once the polynomial is found it is able to estimate future or missing data points.

Although Lagrange and Newton's formulas are similar there are advantages and disadvantages to them. Lagrange is more complex and static as more data points can not be added to the set. This means that in order to add more data points recalculation will be needed. As opposed to Newton's formula, it is more flexible allowing it to add more points and reuse old data without recalculating. Newton's method has the ability to reuse old data and insert more data points. This makes Newton's method more approachable due to it being able to adapt.

Lagrange Demonstration

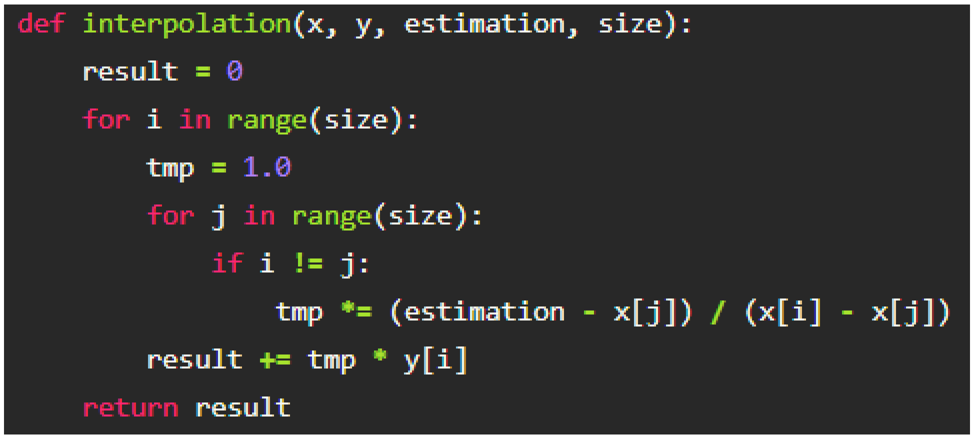
The demonstration that will be discussed in this report is to use the lagrange method and predict the values of the natural log function. To compare the results we will have the natural logarithm plotted along with a table with the exact points.





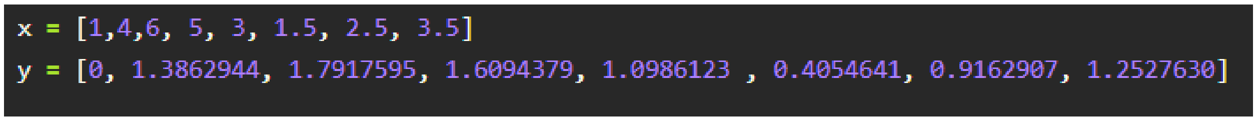
x	ln(x)
1	0
2	0.69314718
3	1.09861229
4	1.38629436
5	1.60943791
6	1.79175947
7	1.94591015
8	2.07944154

The code provided below is the function for the lagrange interpolation. The parameters that are imputed are the data points for x and y, the point that is estimated and the size of the list. The result value is initialized at 0, then there is a nested for loop that goes through the list. If i is not equal to j then it will compute the individual terms in the lagrange formula and finally go through summation. This will keep going until it loops through all the terms in the given data set.

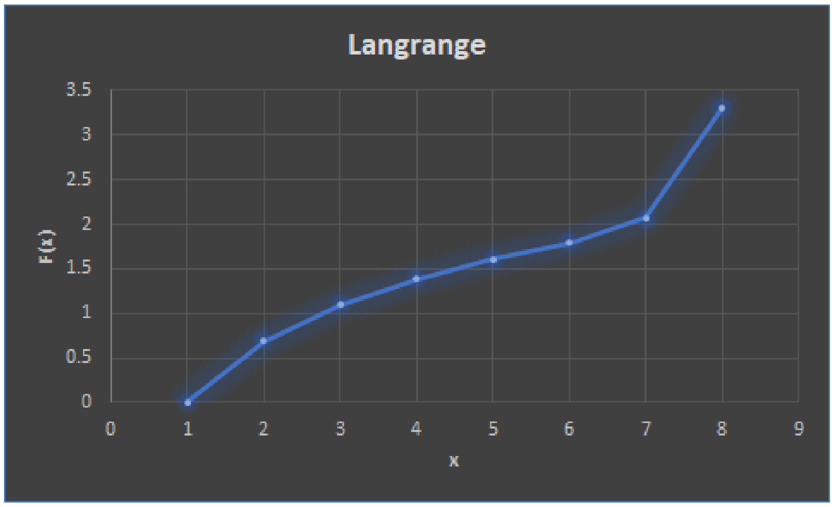


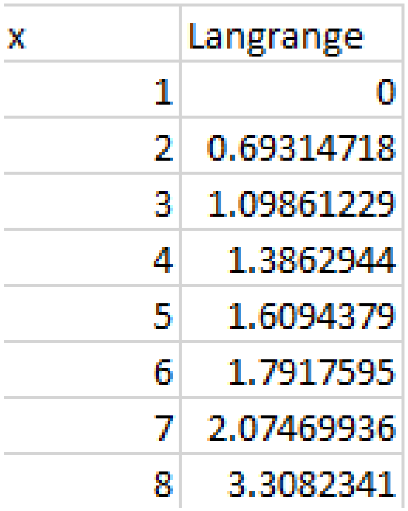
```
def interpolation(x, y, estimation, size):
    result = 0
    for i in range(size):
        tmp = 1.0
        for j in range(size):
            if i != j:
                tmp *= (estimation - x[j]) / (x[i] - x[j])
        result += tmp * y[i]
    return result
```

The data set that will be inputted are preset data points of exact values of the natural log function.



```
x = [1,4,6, 5, 3, 1.5, 2.5, 3.5]
y = [0, 1.3862944, 1.7917595, 1.6094379, 1.0986123 , 0.4054641, 0.9162907, 1.2527630]
```



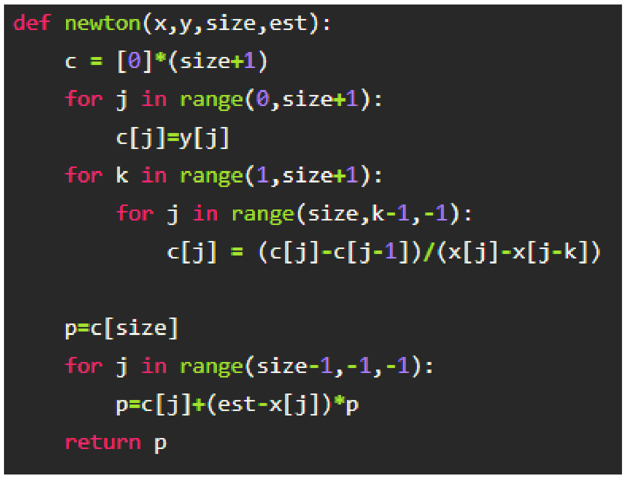


x	Langrange
1	0
2	0.69314718
3	1.09861229
4	1.3862944
5	1.6094379
6	1.7917595
7	2.07469936
8	3.3082341

The data points that were estimated are points 4 - 8. As seen in the graph the beginning looks similar to the exact natural log function. However, the line between points 6 - 7 there is a slight change with the graph looking a bit steeper. Then on points 7 - 8 there is a huge change where the graph goes up. Additional points were run through the application and produced more incorrect estimations. As seen in the percent error column for points 9 - 10 the error percentages are really high, almost reaching 1000% error. From the data shown it can be concluded points that are within the data set or are one to two points after producing lower percent errors.

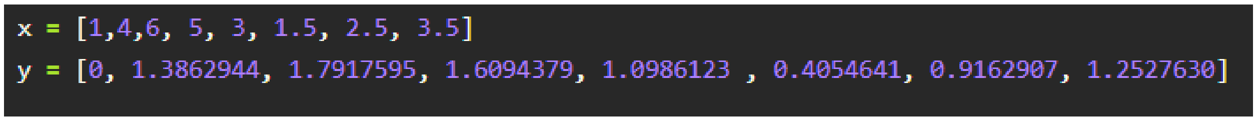
Newton Demonstration

The demonstration for Newton's formula will be the same as the Lagrange formula so that if there is a change it can be seen. The same log function will be seen so refer above for the graph and table.

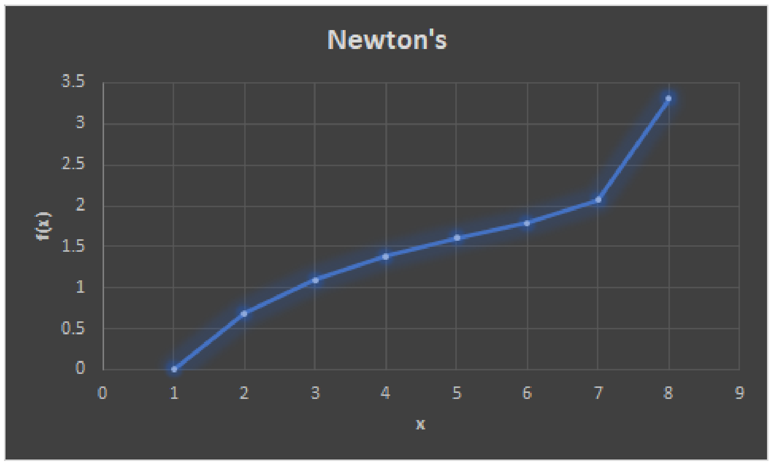


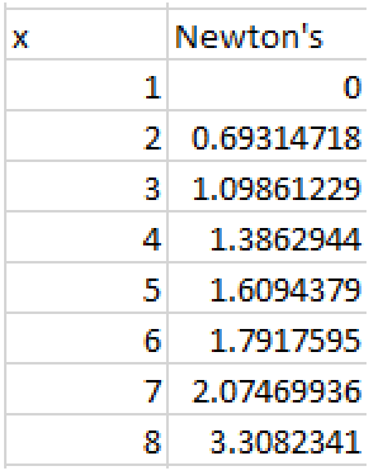
```python
def newton(x,y,size,est):
    c = [0]*(size+1)
    for j in range(0,size+1):
        c[j]=y[j]
    for k in range(1,size+1):
        for j in range(size,k-1,-1):
            c[j] = (c[j]-c[j-1])/(x[j]-x[j-k])

    p=c[size]
    for j in range(size-1,-1,-1):
        p=c[j]+(est-x[j])*p
    return p
```

The code shown is Newton's function which takes in the inputs for the x and y list, along with the size and estimation point. A requirement of Newton's formula is to have the data size minus one,this can be seen in the source code where size = len(x) - 1. The variable c is initialized then the for loop will go through all the values in the array, then the y list will be copied to the variable c. In the next for loop it will find the coefficients by using the formula for divided differences. In the final portion it will sum up all the calculations giving us the estimated point at the end.



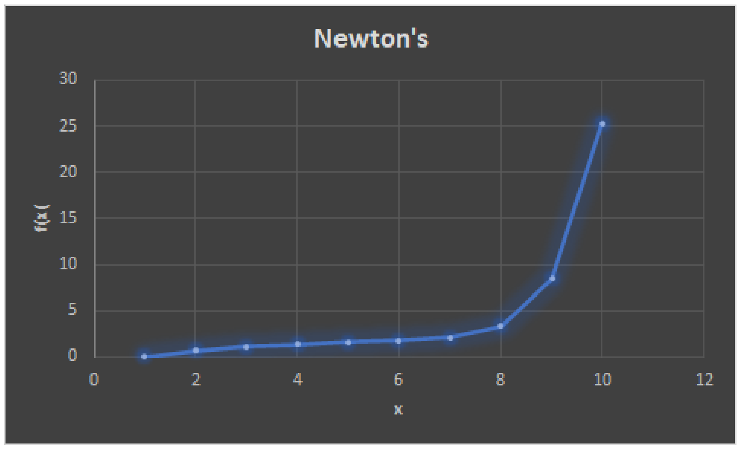
```
x = [1,4,6, 5, 3, 1.5, 2.5, 3.5]
y = [0, 1.3862944, 1.7917595, 1.6094379, 1.0986123 , 0.4054641, 0.9162907, 1.2527630]
```





x	Newton's
1	0
2	0.69314718
3	1.09861229
4	1.3862944
5	1.6094379
6	1.7917595
7	2.07469936
8	3.3082341

The dataset used will be the same as the lagrange dataset so that it will be easier to compare results. In looking at the graph it is identical to the lagrange method in which the points in the beginning of the table are accurate to the natural log function. However, upon looking at points 6 - 7 there is a slight difference where it can be seen going steeper. Then points 7 - 8 there is drastic change where it makes almost a 45 degree angle.

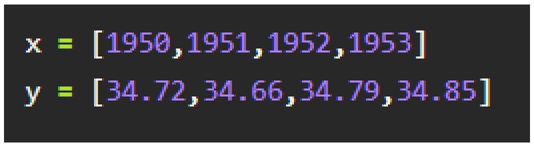


x	Newton's	% Error
1	0	0
2	0.69314718	0
3	1.09861229	0
4	1.3862944	2.8E-06
5	1.6094379	7.73E-07
6	1.7917595	1.72E-06
7	2.07469936	6.618456
8	3.3082341	59.09243
9	8.47101852	285.5327
10	25.3053318	998.9966

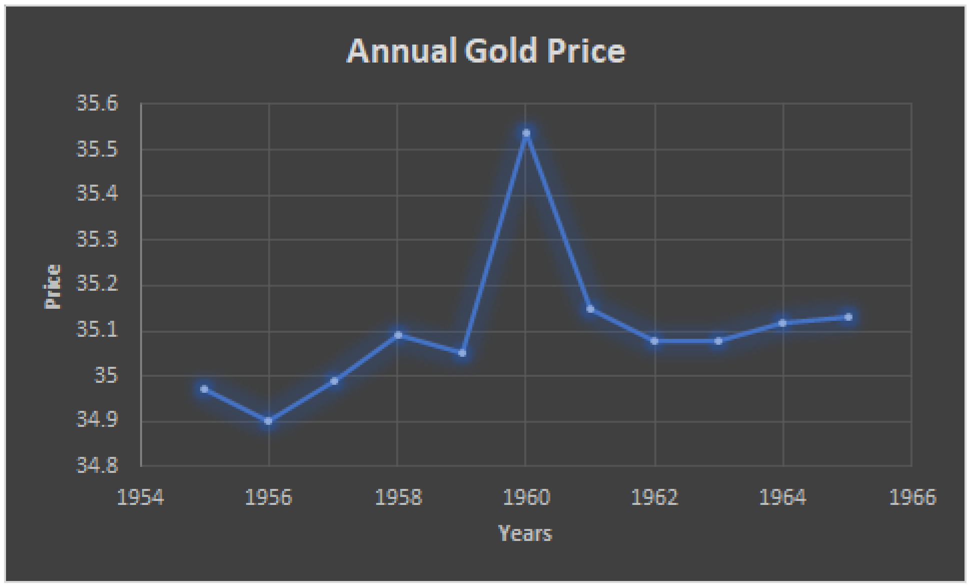
More data points were added to see if they would improve results of the graph. However, it continued the upward trend making the graph even steeper. Similar to the lagrange demonstration a percent error column was added by using the formula % error = $\left|\frac{VA - VE}{VE}\right| * 100$. Where 'VA' is equal to actual value and 'VE' is equal to expected value. In terms of the demonstration, 'VA' is the point estimated using the program and 'VE' is the actual value of the log function. The percent error for point 9 and 10 tell us that estimating big jumps using the program will produce a large error.

Dataset Analysis

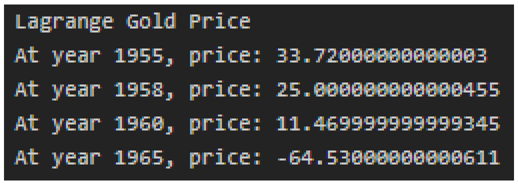
The dataset that will be analyzed using the Lagrange and Newton's polynomial interpolation is Annual Gold Price History in the United States. For this demonstration we will feed the program selected points from the data set so that it will estimate the given point. The x values entered will be the year and the y values will be the price.



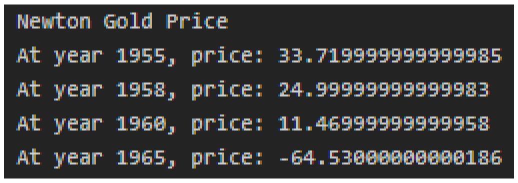
```
x = [1950,1951,1952,1953]
y = [34.72,34.66,34.79,34.85]
```



Plotted above is the actual graph of the dataset, so that we are able to compare our results accurately. The years that will be estimated are 1955, 1958, 1960 and 1965.



```
Lagrange Gold Price
At year 1955, price: 33.72000000000003
At year 1958, price: 25.000000000000455
At year 1960, price: 11.469999999999345
At year 1965, price: -64.53000000000611
```

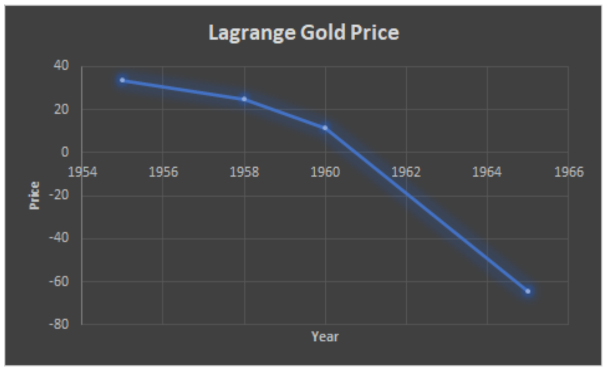


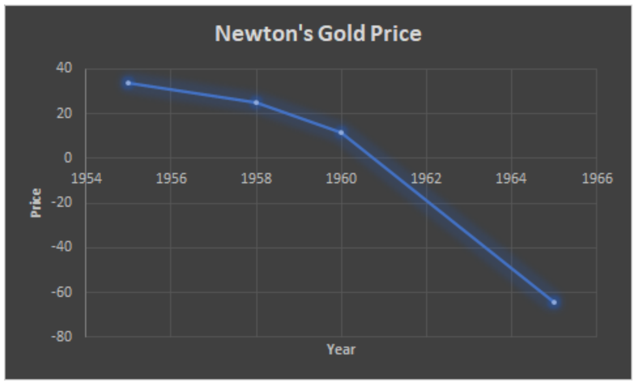
```
Newton Gold Price
At year 1955, price: 33.719999999999985
At year 1958, price: 24.99999999999983
At year 1960, price: 11.46999999999958
At year 1965, price: -64.53000000000186
```

Year	Lagrange Price	% Error
1955	33.72	3.57449
1958	25	28.7546
1960	11.47	67.7265
1965	-64.53	283.689
2000	-4831.53	1877
2010	-8529.78	712.107

Year	Newton Price	% Error
1955	33.72	3.57449
1958	25	28.7546
1960	11.47	67.7265
1965	-64.53	283.689
2000	-4831.53	1877
2010	-8529.78	712.107

When estimating the year for 1955 the percent error is low with an error of 3.5%. However, as we increase in years the percent error also increases.





Looking at the graphs above if you analyze it looking at parts it is pretty accurate. Since there are no points in between it will be a linear line summarizing the basic trends. Adding more points will help the graph curve giving accurate results. Looking at the graph from 1954 to 1956 it shows a downward trend which is also shown in the exact graph for the dataset of gold prices. For 1958 and 1960 it also shows a downward trend, but looking at the percent error it is 30 to 60 percent. Although the percent error is high it still gives a basic understanding of how the graph moves. Then looking at a big jump from 1960 to 1965 it shows the price going negative with a percent error of around 200 percent. This does not reflect the exact annual gold price graph because the price should only drop by a couple dollars not 60. Additional data points were added like 2000 and 2010 where there is a huge gap in years. This was unsuccessful, producing a percent error of 700% to 1000%.

Comparing Lagrange and Newton

After conducting many tests on the Lagrange and Newton polynomial interpolation from my results they are the same. The only differences are the methods applied to creating Lagrange and Newtons. We can say that better results are produced when the points approximated are close to the given dataset. When estimating bigger jumps then that will produce higher percentages of errors.

Works Cited

“Lagrange's Interpolation.” *GeeksforGeeks*, 18 Feb. 2020, www.geeksforgeeks.org/lagranges-interpolation/.

Lecture 2-1: Lagrange Interpolating Polynomials, dmpeli.math.mcmaster.ca/Matlab/Math4Q3/NumMethods/Lecture2-1.html.

Lecture 2-2: Newton Divided Difference Polynomials, dmpeli.math.mcmaster.ca/Matlab/Math4Q3/NumMethods/Lecture2-2.html.

“Newton's Divided Difference Interpolation Formula.” *GeeksforGeeks*, 13 Aug. 2019, www.geeksforgeeks.org/newtons-divided-difference-interpolation-formula/.

Tunguz, Bojan. “Gold Prices.” *Kaggle*, 30 Mar. 2021, www.kaggle.com/tunguz/gold-prices?select=annual_csv.csv.