EECS639 Final Project

Devin Setiawan, Isaac Papineau, Ethan Dirkes

I. Summary of Contributions

Devin Setiawan: Led the implementation of interpolation methods for Vandermonde, Newton, Lagrange polynomials, and cubic splines (natural, complete, and not-a-knot) in Part A. Also developed testing scripts for comparing these methods on various datasets in Part B, and applied the interpolation techniques to the real-world application dataset in Part D.

Isaac Papineau: Focused on Part C, implementing cubic Bézier curves for parametric interpolation. Conducted extensive testing on various parametric curves, including ellipses, hypotrochoids, wavy, butterfly, and rose curves, using different sampling techniques and parameter ranges to ensure accurate representation.

Ethan Dirkes: Managed dataset preprocessing and analysis for Part D. This included identifying and preparing the real-world application dataset, defining the scientific questions to be answered, and supporting the application of interpolation methods from Parts A and C to derive meaningful insights.

Devin Setiawan

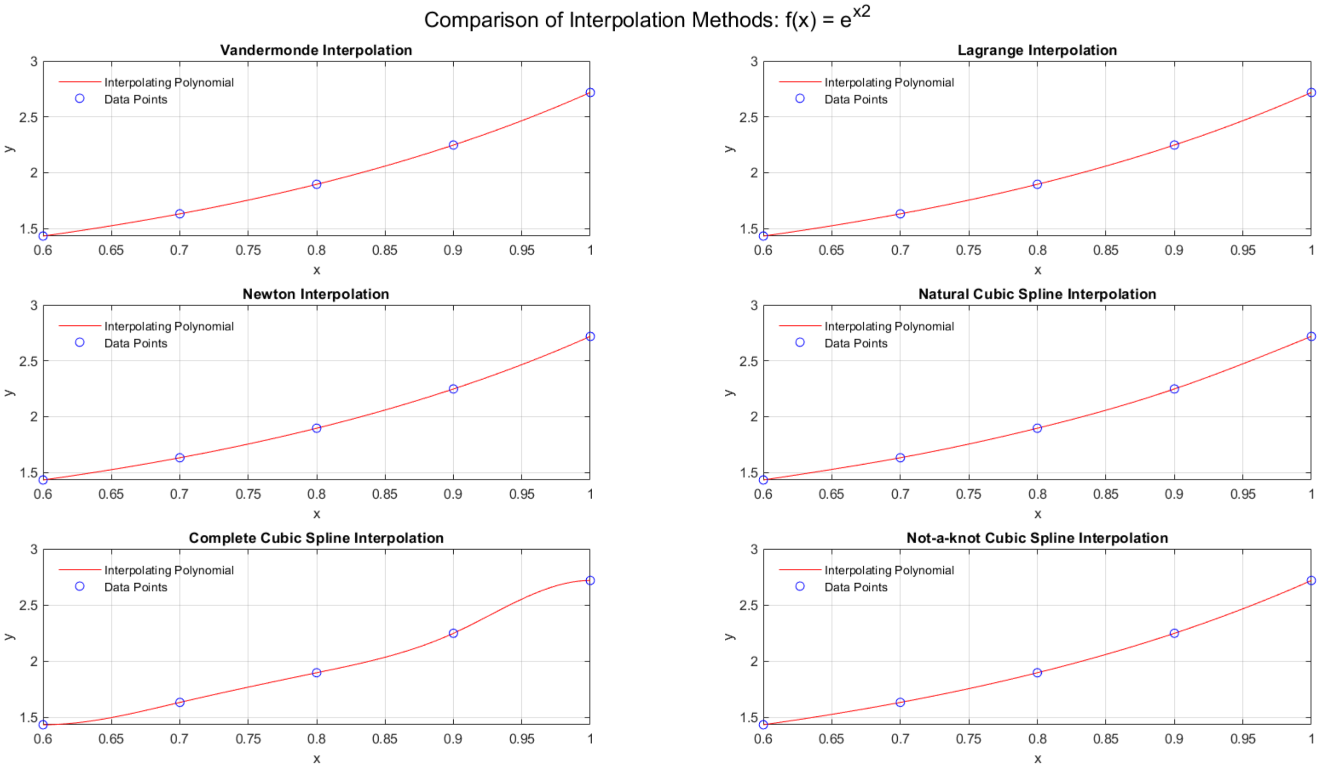
Isaac Papineau

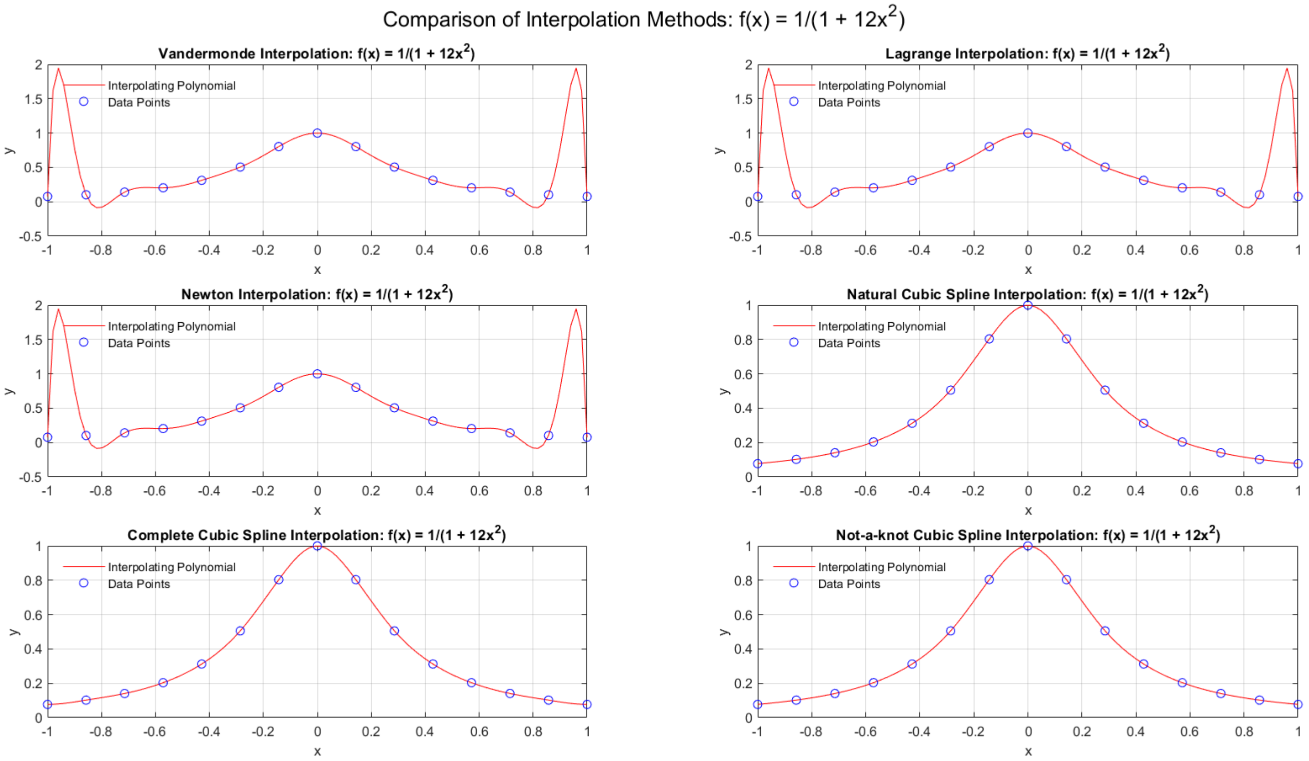
Isaac Papineau

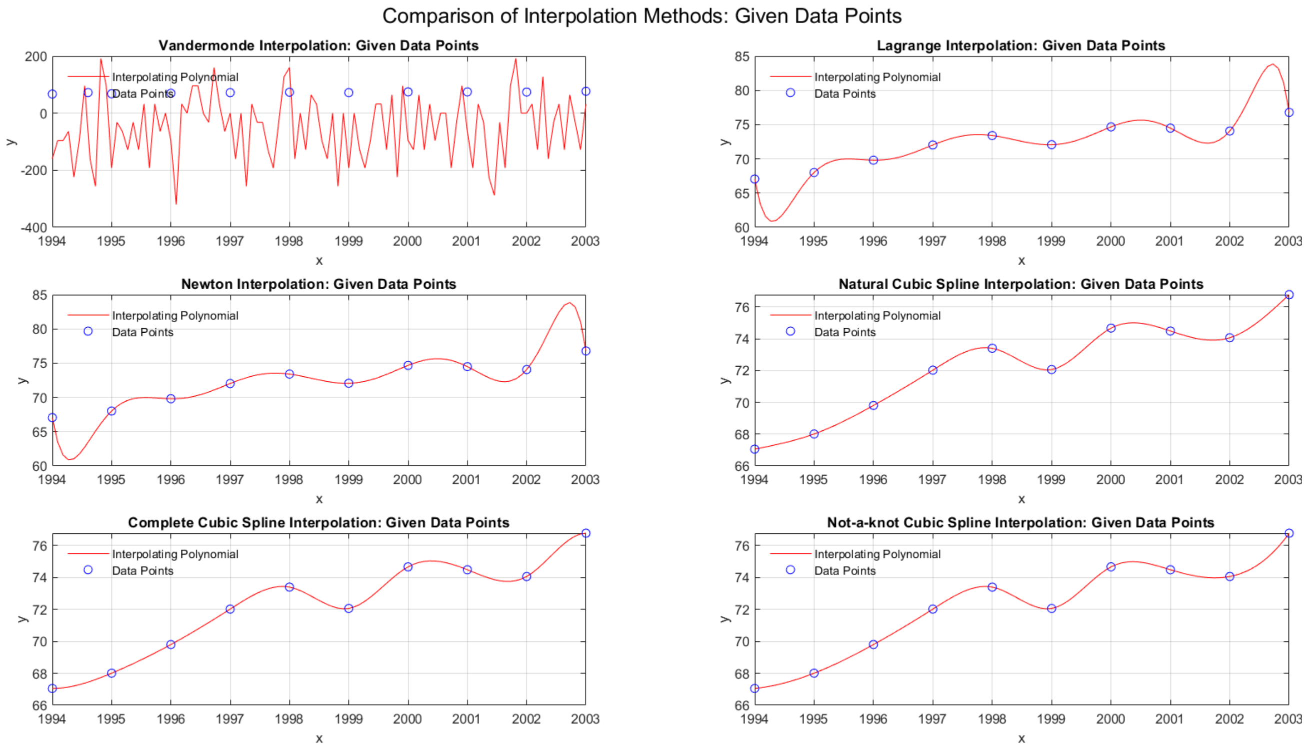
Ethan Dirkes

Ethan Dirkes

II. Interpolation Graphs







a. Conditioning

	Vandermonde	**Natural**	**Complete**	**Not-a-knot**
f(x) = e^(x^2)	70583.0886	71654.449	109297.6649	69973.3461
f(x) = 1/(1 + 12x^2)	1104808.5294	22234.6394	31613.0188	24205.9898
Given points	9.594780569360444e+43	3.285208073729305e+21	2.299031468621381e+21	2.070323250089768e+24

The conditioning of the interpolation problem varies significantly across methods and problems, as evidenced by the condition numbers**. For f(x)=e^(x^2)**, the condition numbers for the Vandermonde matrix and the splines are comparable, with the Vandermonde and not-a-knot splines being slightly better conditioned than the natural and complete splines. However, **for f(x)=1/(1+12x^2)**, the Vandermonde method has a significantly higher condition number, indicating severe sensitivity to small changes in the data, whereas the spline methods, particularly the natural and not-a-knot splines are much better conditioned, suggesting greater numerical stability. **For the third problem**, the Vandermonde method is catastrophically ill-conditioned, with an astronomical condition number of 10^43, making it unsuitable for practical use. The splines, though still ill-conditioned for this dataset, have condition numbers that are many orders of magnitude smaller, indicating they are far more stable. Overall, spline methods demonstrate much better conditioning and are preferable for ensuring numerical stability, especially for challenging datasets.

b. Accuracy of Interpolation

For f(x) = e^(x^2), the Vandermonde, Newton, and Lagrange interpolants are identical and capture the curve's properties accurately. The natural and not-a-knot cubic splines also provide accurate results, but the complete cubic spline inaccurately introduces concave-down curvature near the endpoints. **For f(x) = 1/(1 + 12x^2)**, where we expect a smooth bell-shaped curve, the Vandermonde, Newton, and Lagrange interpolants exhibit significant oscillations, particularly at the endpoints, failing to capture the bell shape accurately. In contrast, all spline interpolants produce smooth, well-shaped curves, with splines being the more accurate choice. **For the given data points**, the Vandermonde interpolation fails entirely, while Newton and Lagrange interpolants, although successful, exhibit fluctuations that undermine their accuracy. Spline interpolants are smoother and better suited to this data, with the natural and not-a-knot splines being the most accurate. However, the complete cubic spline again displays a concavity change at the endpoints, making it less reliable.

c. Efficiency

Vandermonde interpolation involves solving a dense system of linear equations, which has a computational cost of O(n^3). This method is the least efficient method. **Newton interpolation** is computationally efficient as our implementation uses divided differences to compute the coefficients in O(n^2) time. Once the coefficients are computed, evaluating the polynomial at a point is fast and stable. **Lagrange interpolation**, although easy to compute coefficients, is less efficient than Newton's method because each evaluation requires recomputing the basis polynomials. This makes it less practical for large datasets. For the **spline interpolations**, constructing the splines involves solving a tridiagonal system, which has a computational cost of O(n). Once the spline is constructed, evaluating it at any point is efficient, as it only requires identifying the correct interval and evaluating a cubic polynomial.

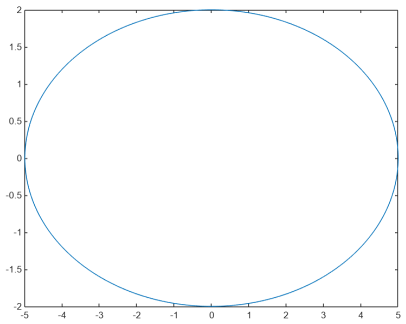
III. Parametric Curves

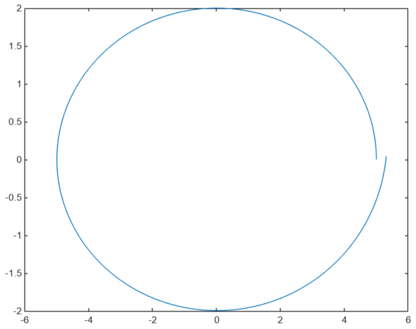
Bezier

```
function
[a0,b0,a1,b1,a2,b2,a3,b3]=Bezier_Curve(n,X,Y,XPlus,YPlus,XMinus,YMinus)
for i=1: n
a0(i)= X(i);
b0(i)= Y(i);
a1(i)= 3.*(XPlus(i)-X(i));
b1(i)= 3.*(YPlus(i)-Y(i));
a2(i)= 3.*(X(i)+XMinus(i+1)-2.*XPlus(i));
b2(i)= 3.*(Y(i)+YMinus(i+1)-2.*YPlus(i));
a3(i)= X(i+1)-X(i)+3.*XPlus(i)-3.*XMinus(i+1);
b3(i)= Y(i+1)-Y(i)+3.*YPlus(i)-3.*YMinus(i+1);
end
end
```

Ellipse

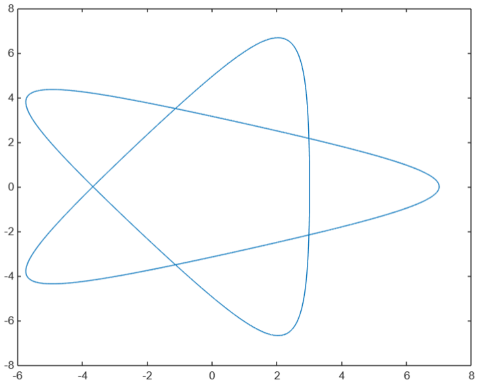
```
t= 0:pi/200:2*pi;
x=5.*cos(3.*t);
y=2.*sin(3.*t);
plot (x,y)
n=141;
xplus= 5.*cos(3.*t)+0.001;
xminus= 5.*cos(3.*t)-0.001;
yplus= 2.*sin(3.*t)+0.001;
yminus= 2.*sin(3.*t)-0.001;
[a0,b0,a1,b1,a2,b2,a3,b3]=Bezier_Curve(n,x,y,xplus,yplus,xminus,yminus);
figure()
for i=1: n
fx(i)=a0(i)+(a1(i).*t(i))+(a2(i).*t(i).^2)+(a3(i).*t(i).^3);
fy(i)=b0(i)+(b1(i).*t(i))+(b2(i).*t(i).^2)+(b3(i).*t(i).^3);
T(i)=t(i);
end
plot(fx,fy)
```

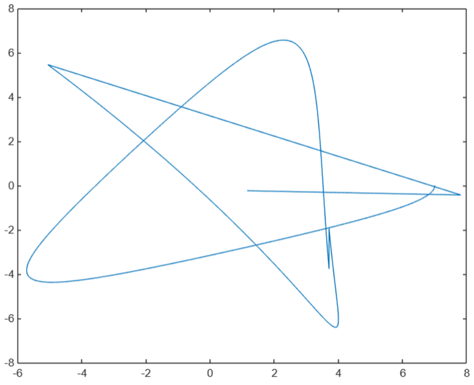




Hypotrochoid

```
t= 0:pi/10000:8*pi;
x= (2.*cos(t))+(5.*cos(2.*t./3));
y= (2.*sin(t))-(5.*sin(2.*t./3));
plot (x,y)
n=47861;
xplus= (2.*cos(t))+(5.*cos(2.*t./3))+0.0001;
xminus= (2.*cos(t))+(5.*cos(2.*t./3))-0.0001;
yplus= (2.*sin(t))-(5.*sin(2.*t./3))-0.0001;
yminus= (2.*sin(t))-(5.*sin(2.*t./3))+0.0001;
for i=33606:n
xplus(i)= (2.*cos(t(i)))+(5.*cos(2.*t(i)./3))+0.0001;
xminus(i)= (2.*cos(t(i)))+(5.*cos(2.*t(i)./3))-0.0001;
yplus(i)= (2.*sin(t(i)))-(5.*sin(2.*t(i)./3))+0.0002;
yminus(i)= (2.*sin(t(i)))-(5.*sin(2.*t(i)./3))-0.0002;
end
for i=47860:n
xplus(i)= (2.*cos(t(i)))+(5.*cos(2.*t(i)./3))+0.0008;
xminus(i)= (2.*cos(t(i)))+(5.*cos(2.*t(i)./3))-0.0008;
yplus(i)= (2.*sin(t(i)))-(5.*sin(2.*t(i)./3))-0.00012;
yminus(i)= (2.*sin(t(i)))-(5.*sin(2.*t(i)./3))+0.00012;
end
for i=60000:n
xplus(i)= (2.*cos(t(i)))+(5.*cos(2.*t(i)./3))-0.0003;
xminus(i)= (2.*cos(t(i)))+(5.*cos(2.*t(i)./3))+0.0003;
yplus(i)= (2.*sin(t(i)))-(5.*sin(2.*t(i)./3))-0.001;
yminus(i)= (2.*sin(t(i)))-(5.*sin(2.*t(i)./3))+0.001;
end
[a0,b0,a1,b1,a2,b2,a3,b3]=Bezier_Curve(n,x,y,xplus,yplus,xminus,yminus);
figure()
for i=1:n
fx(i)=a0(i)+(a1(i).*t(i))+(a2(i).*t(i).^2)+(a3(i).*t(i).^3);
fy(i)=b0(i)+(b1(i).*t(i))+(b2(i).*t(i).^2)+(b3(i).*t(i).^3);
end
plot(fx,fy)
```





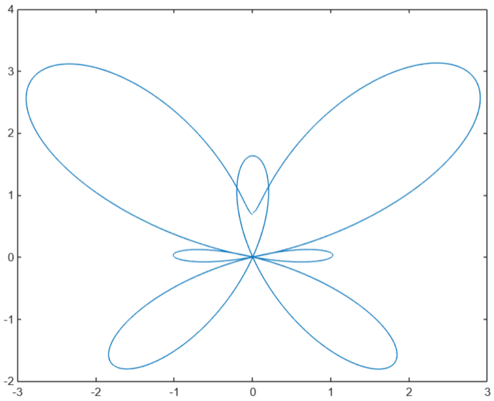
Wavy

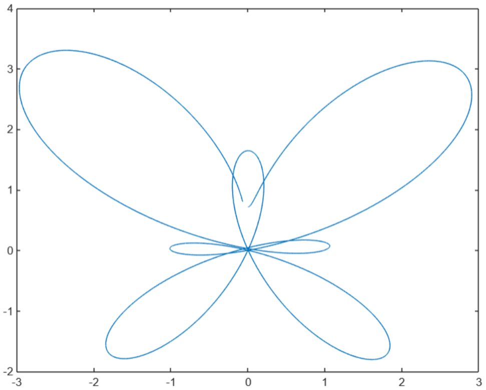
```
t= 0:pi/20:80*pi;
x= (20.*t).*cos(0.2.*t);
```

```
y= 10.*sin(t);
plot (x,y)
n=1600;
xplus= ((20.*t).*cos(0.2.*t));
xminus= ((20.*t).*cos(0.2.*t));
yplus= (10.*sin(t));
yminus= (10.*sin(t));
[a0,b0,a1,b1,a2,b2,a3,b3]=Bezier_Curve(n,x,y,xplus,yplus,xminus,yminus);
figure()
for i=1: n
fx(i)=a0(i)+(a1(i).*t(i))+(a2(i).*t(i).^2)+(a3(i).*t(i).^3);
fy(i)=b0(i)+(b1(i).*t(i))+(b2(i).*t(i).^2)+(b3(i).*t(i).^3);
end
plot(fx,fy)
```

Butterfly

```
t= 0:pi/10000:2*pi;
x= sin(t).*((exp(cos(t)))-(2.*cos(4.*t))-((sin(t./12)).^5));
y= cos(t).*((exp(cos(t)))-(2.*cos(4.*t))-((sin(t./12)).^5));
plot (x,y)
n=20000;
xplus= (sin(t).*((exp(cos(t)))-(2.*cos(4.*t))-((sin(t./12)).^5)))+0.00001;
xminus= (sin(t).*((exp(cos(t)))-(2.*cos(4.*t))-((sin(t./12)).^5)))-0.00001;
yplus= (cos(t).*((exp(cos(t)))-(2.*cos(4.*t))-((sin(t./12)).^5)))+0.0001;
yminus= (cos(t).*((exp(cos(t)))-(2.*cos(4.*t))-((sin(t./12)).^5)))-0.0001;
[a0,b0,a1,b1,a2,b2,a3,b3]=Bezier_Curve(n,x,y,xplus,yplus,xminus,yminus);
figure()
for i=1: n
fx(i)=a0(i)+(a1(i).*t(i))+(a2(i).*t(i).^2)+(a3(i).*t(i).^3);
fy(i)=b0(i)+(b1(i).*t(i))+(b2(i).*t(i).^2)+(b3(i).*t(i).^3);
end
plot(fx,fy)
```

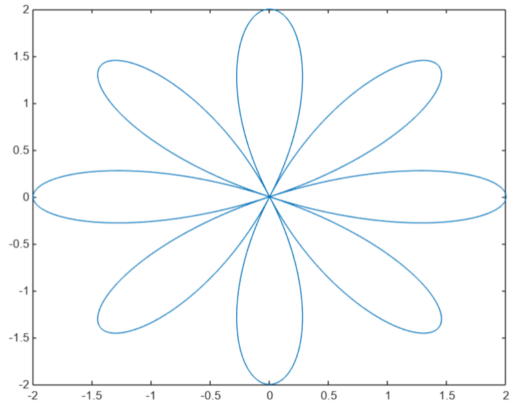


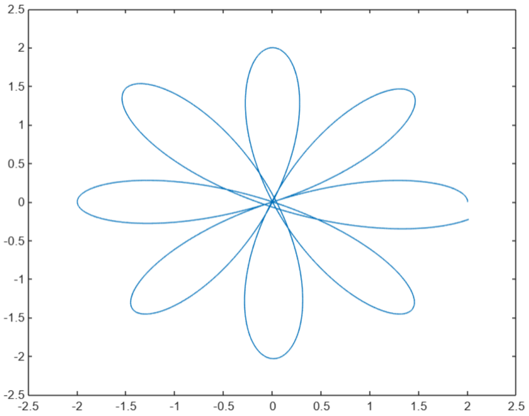


Rose

```
t= 0:pi/10010:2*pi;
a=2;
k=4;
```

```
x= a.*cos(k.*t).*cos(t);
y= a.*cos(k.*t).*sin(t);
plot (x,y)
n=20020;
xplus= (a.*cos(k.*t).*cos(t))+0.00001;
xminus= (a.*cos(k.*t).*cos(t))-0.00001;
yplus= (a.*cos(k.*t).*sin(t))+0.00001;
yminus= (a.*cos(k.*t).*sin(t))-0.00001;
[a0,b0,a1,b1,a2,b2,a3,b3]=Bezier_Curve(n,x,y,xplus,yplus,xminus,yminus);
figure()
for i=1: n
fx(i)=a0(i)+(a1(i).*t(i))+(a2(i).*t(i).^2)+(a3(i).*t(i).^3);
fy(i)=b0(i)+(b1(i).*t(i))+(b2(i).*t(i).^2)+(b3(i).*t(i).^3);
end
plot(fx,fy)
```





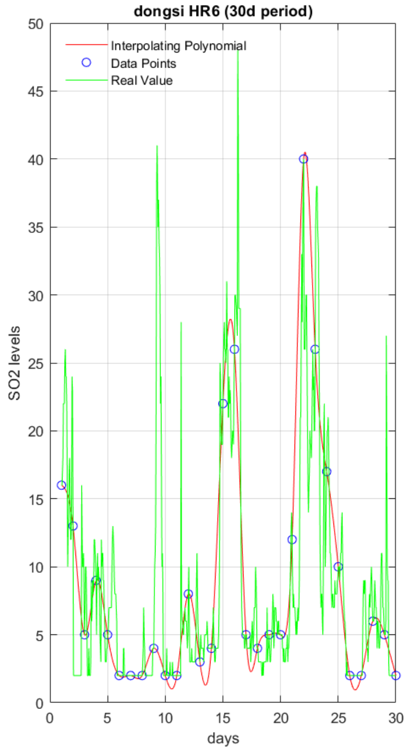
IV. Real-World Application

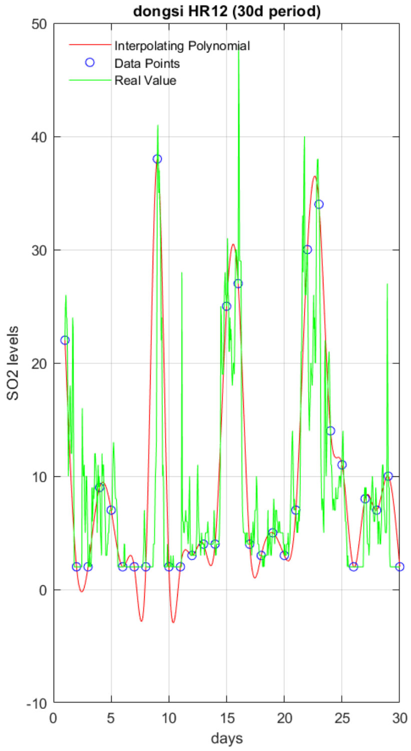
In this project, we analyze the Beijing Multi-Site Air Quality dataset from the UCI Machine Learning Repository to investigate interpolation methods for air quality data. Specifically, we focus on data from 3 selected sites over a period of time, targeting a particular air pollutant. Our research explores:

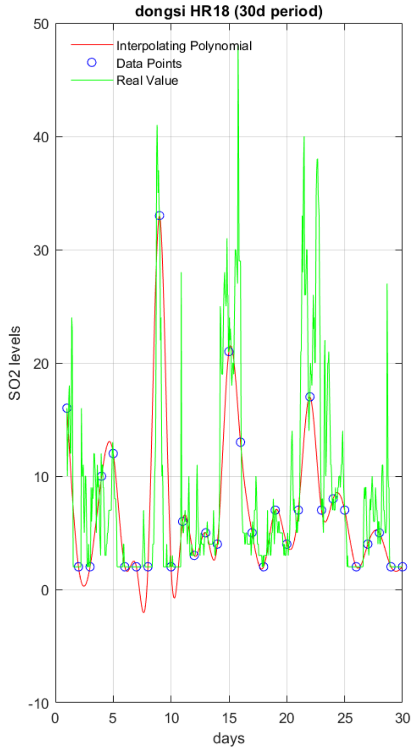
1. **Optimal Data Points for Interpolation:** Determining whether interpolation performs better with data sampled at the start, middle, or end of a day, week, month, or year. For the short term, we interpolate a 30-day period of month 6 in 2015. For the medium term, we interpolate a 12-week period over months 5, 6, and 7 in 2015. For the long term, we interpolate a 4-year period from 2013 to 2016.
2. **Site-Specific Differences:** Comparing the interpolated results across different sites and visualizing the variations.
3. **Real-World Implications:** Examining the practical significance of the observed differences in interpolation outcomes, particularly in understanding and addressing air quality trends.

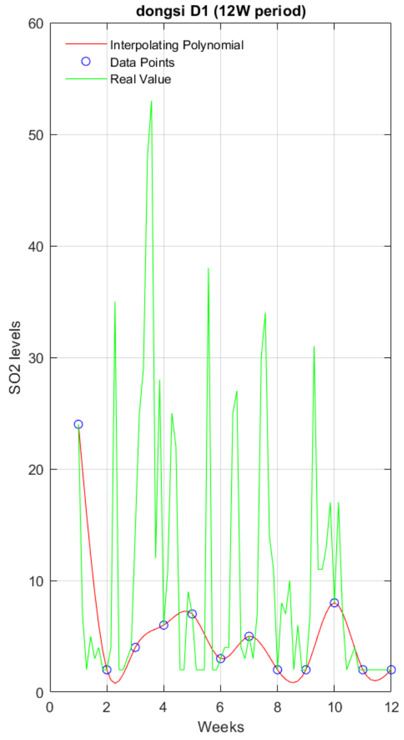
We selected Dongsi, Huairou, and Wanliu regions for this study. For each region, there are 9 total graphs grouped by their interpolation duration (e.g. short-term, medium-term, long-term). We select 3 different points to be used for constructing our interpolant for each interpolation duration. This study provides insights into the effectiveness of interpolation techniques and the variability of air quality across multiple urban sites.

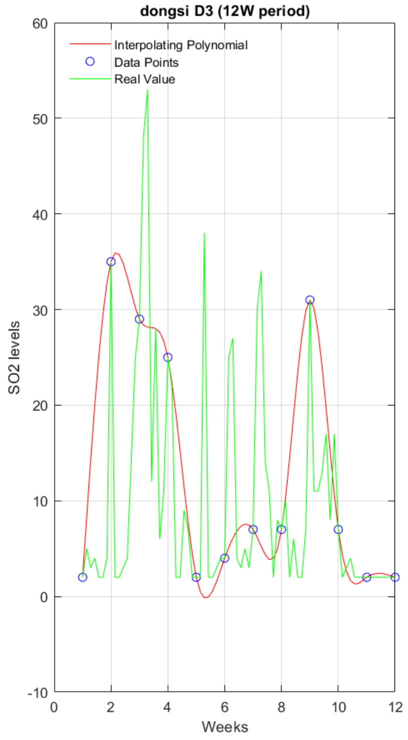
a. Dongsi Region

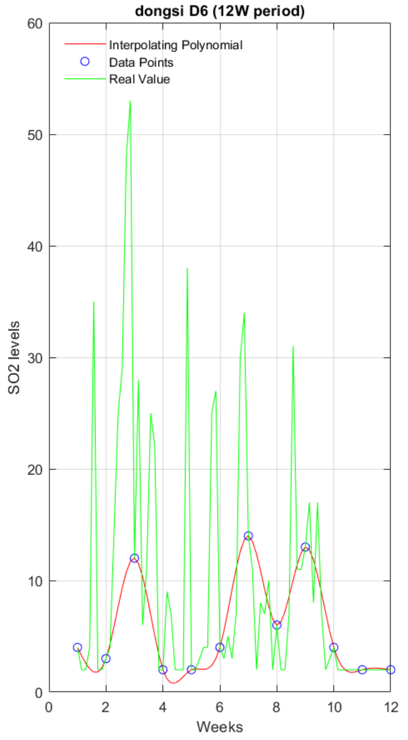


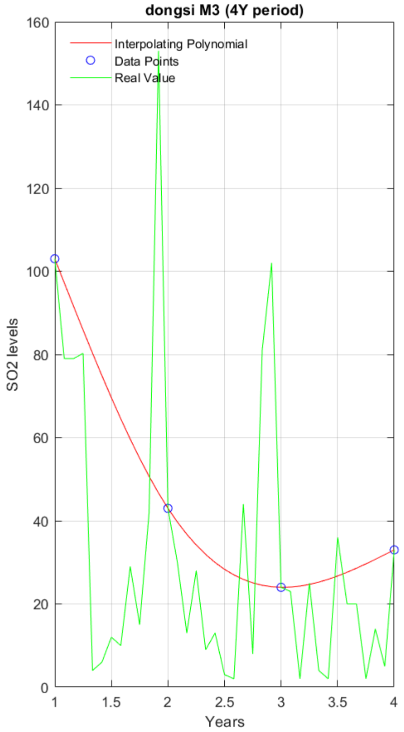


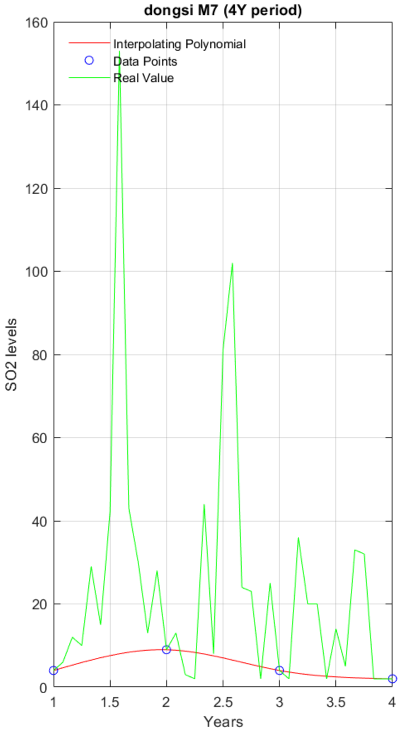


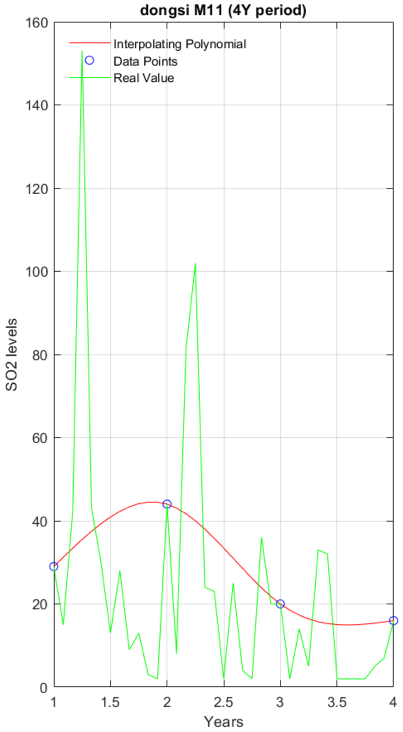












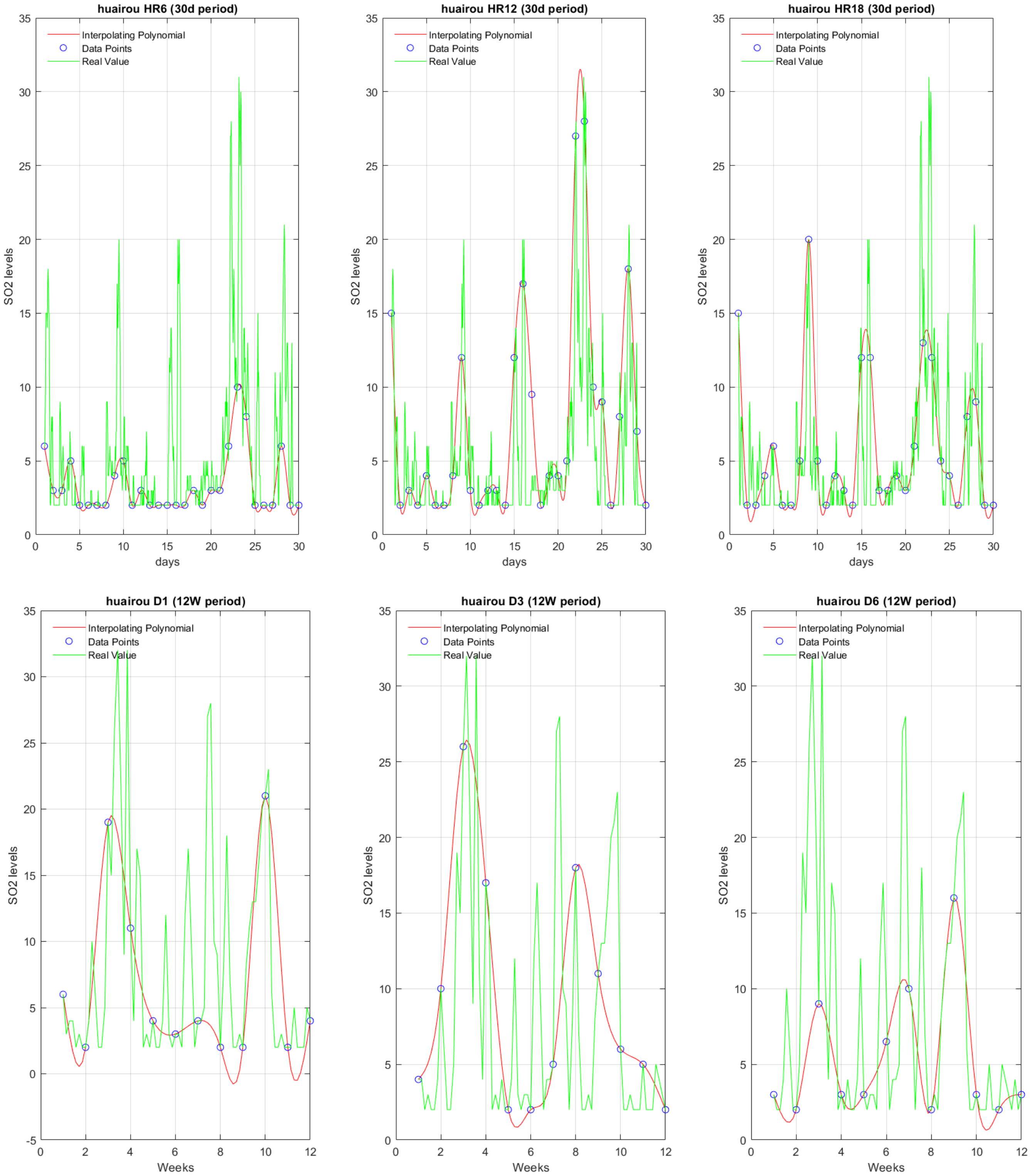
- *Which interpolation best follows the real value trend for the different durations?*

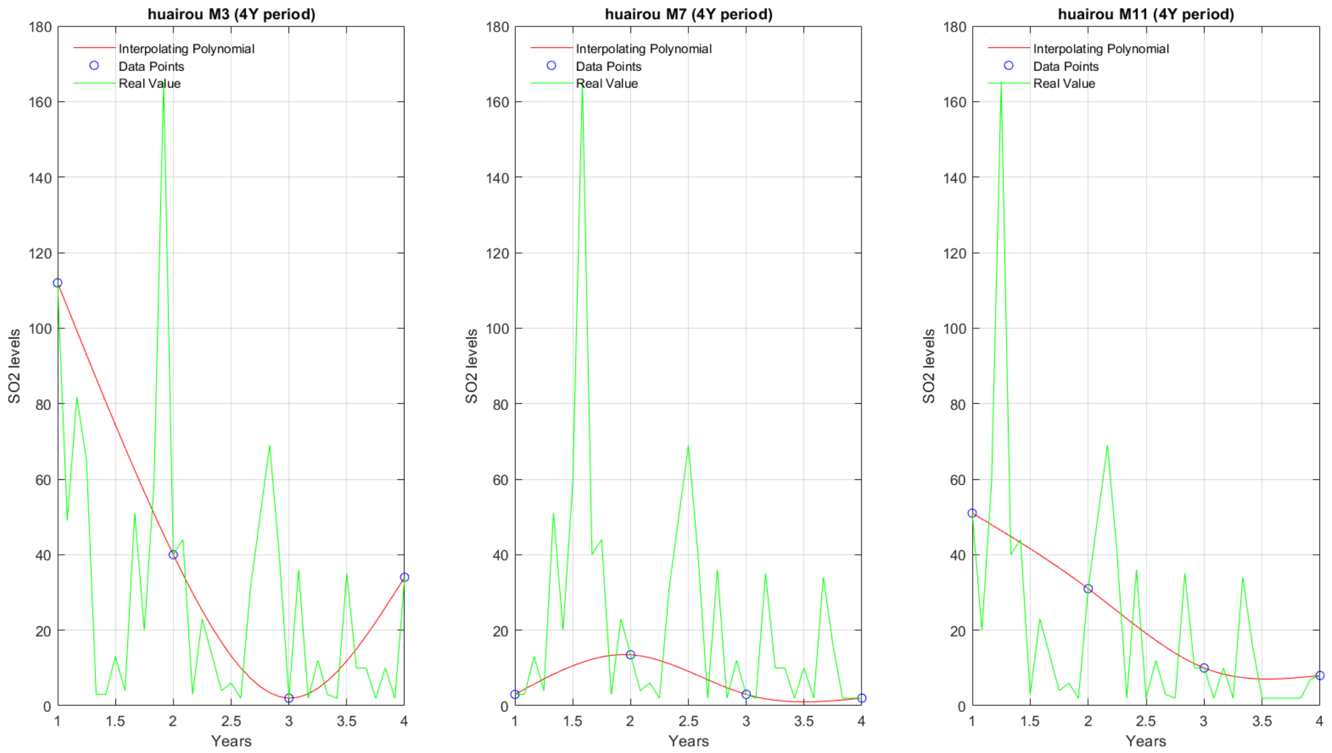
For the short-term duration, the interpolation for measurements taken in the middle of the day (hour 12) best follow the real data values. The interpolation that best follows the medium-duration trend is the interpolation taking measurements from the middle of the week (day 3 of each week). This interpolation has peaks that better approach the actual peaks of the data, while still reaching the lows from the actual data. The trend of the interpolation from the end of the week also matches the real values well, however it does not have peaks close in value to the peaks in the actual data. For the long-term duration, the interpolation from the beginning of each year (month 3 of each year) best matches the general trend from the real values.

- What is the short/medium/long-term trend? (If no trend, what's the major peak and valleys or other structure present captured by the interpolation?)

The short-term trend shows that the peaks occur very briefly before dropping to lower values that last for a few days. The medium-term trend shows a general decline over the course of weeks, with peaks becoming gradually smaller and smaller. The long-term trend shows a general decrease in SO_2 pollutants over the span of a few years.

b. Huairou Region





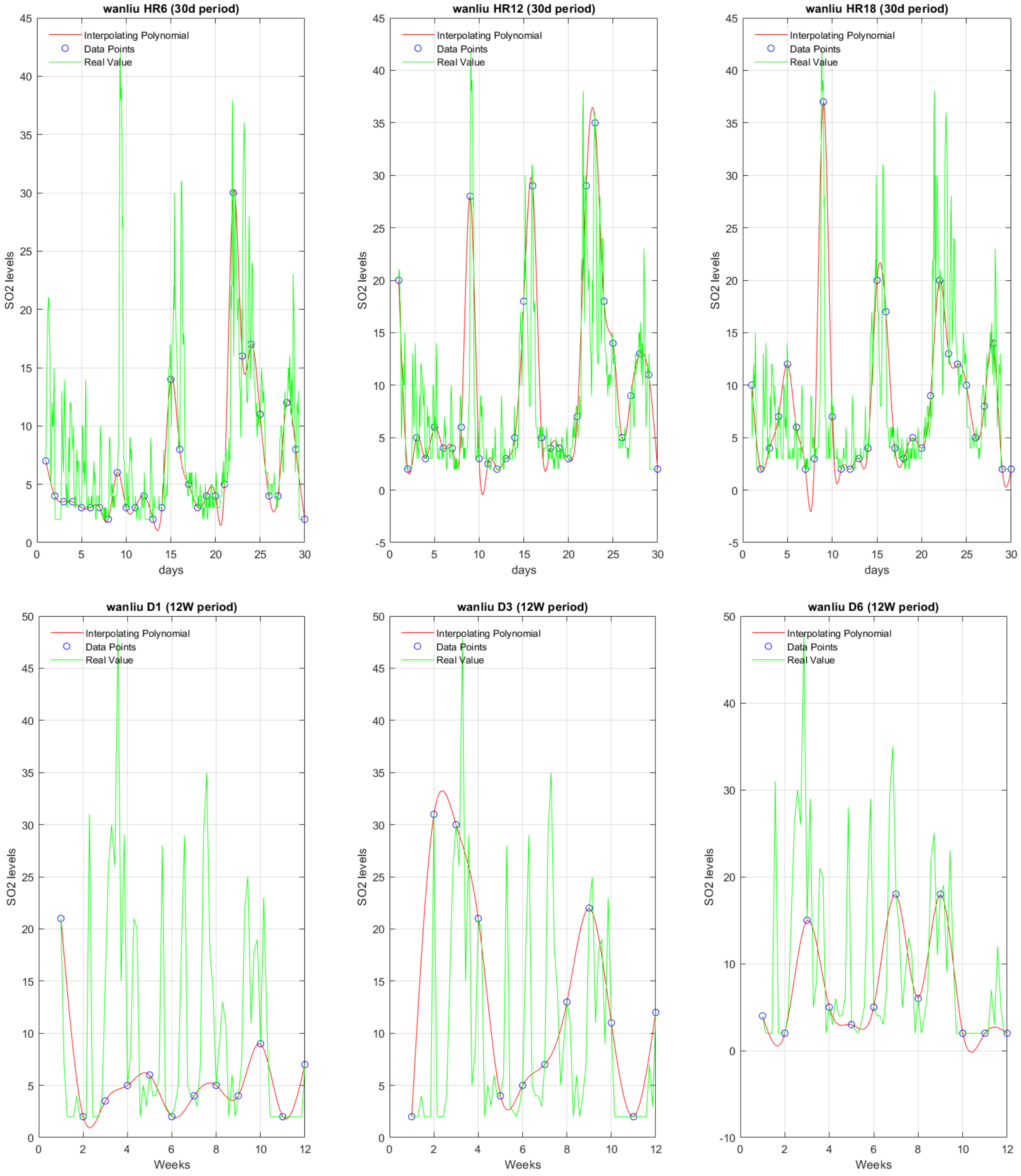
- Which interpolation best follows the real value trend for the different durations?

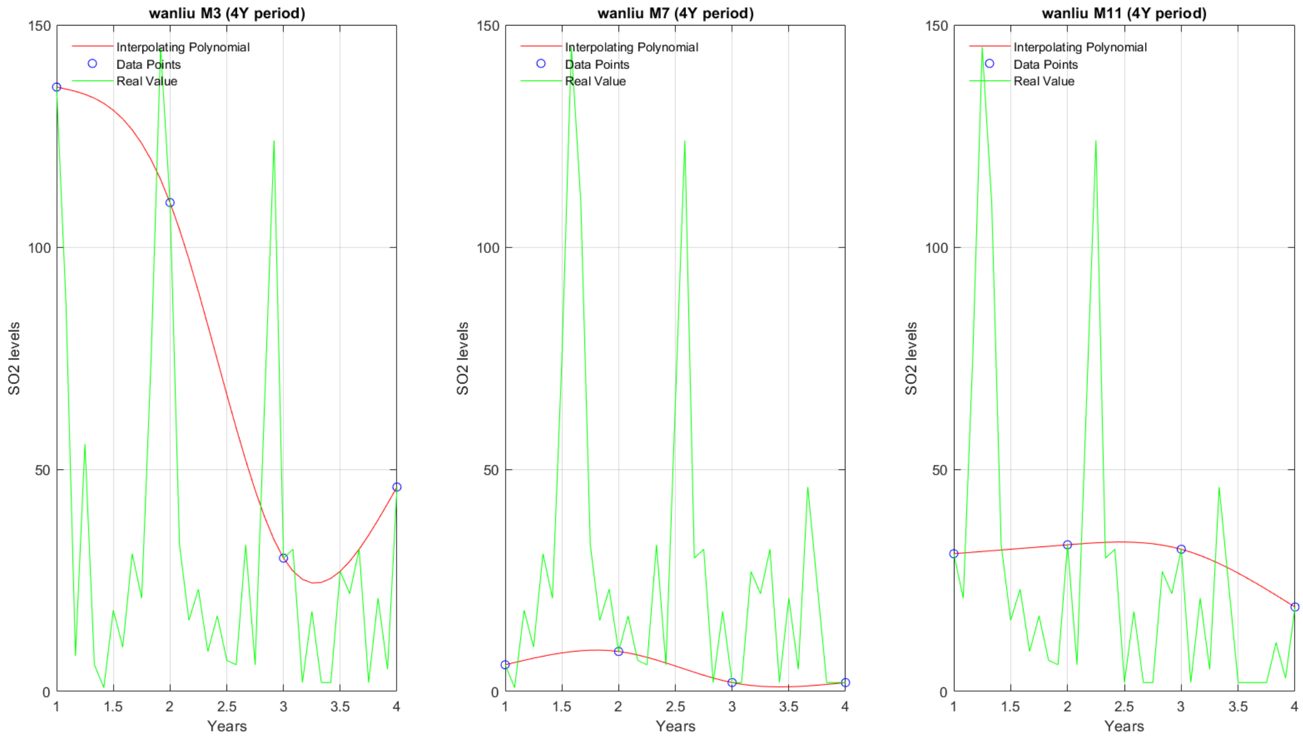
The interpolation taken in the middle of the day (hour 12 of each day) best follows the real value trend for the short-term duration. For the medium-term duration, the interpolation for the data points taken during the middle of the week (day 3 of each week) best follows the real value trend. The interpolation using data from the end of the year (month 11 of each year) best follows the real value trend for the long-term duration.

- What is the short/medium/long-term trend? (If no trend, what's the major peak and valleys or other structure present captured by the interpolation)

The short-term trend reveals high peaks that occur about once every week, with low valleys appearing on the days in between the peaks. The medium-term duration trend shows a general decline in SO_2 pollution over the span of a few weeks as the peaks get lower over time. The long-term trend shows that, over a few years, the amount of SO_2 pollution is on a gradual decline.

c. Wanliu Region





- Which interpolation best follows the real value trend for the different durations?

The interpolation that best follows the real value trend for the short-term duration is the interpolation for the measurements taken in the middle of the day (hour 12 of each day). For the medium-term duration, the best interpolation is from the data points in the middle of the week (day 3 of each week). For the long-term duration, the best interpolation comes from the beginning of the year (month 3 of each year).

- What is the short/medium/long-term trend? (If no trend, what's the major peak and valleys or other structure present captured by the interpolation)

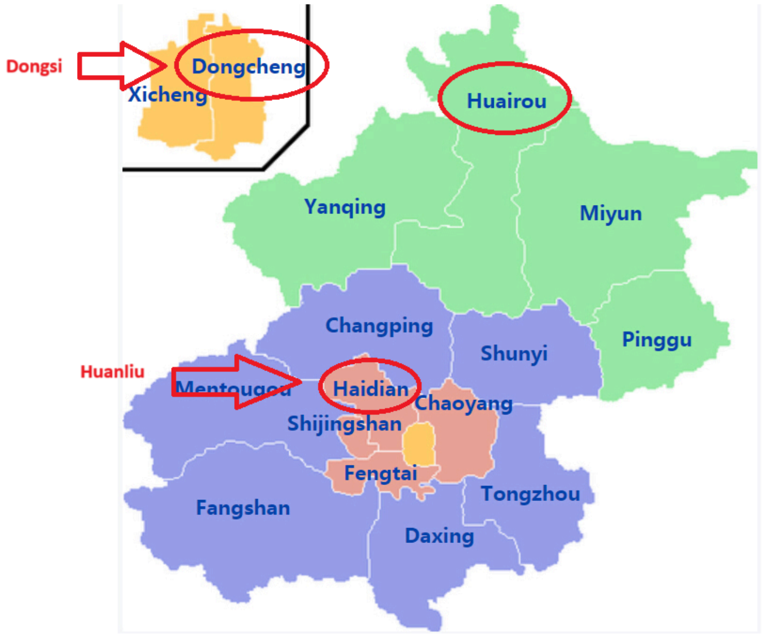
The short-term trend follows a similar pattern to the other two sites in terms of where the peaks and valleys occur. The medium-term trend shows a decrease in SO_2 pollution as the weeks pass, with generally smaller peaks as time progresses. The long-term trend shows a decline in SO_2 pollution over the years, with an increase at the end as the data begins to form another peak.

d. Cross-site Comparisons

- What are the main differences that can be seen between sites?

The interpolation of SO_2 levels across the Dongsi, Huairou, and Wanliu regions in Beijing over various time periods reveals both similarities and differences. Across 30-day, 12-week, and 4-year periods, the regions show consistent peaks and valleys in SO_2 levels occurring at the same times, indicating shared temporal trends. This trend is also confirmed by our higher time frame interpolation where all regions show the same long term trend. However, the magnitude of these peaks varies by region, with Dongsi exhibiting the highest peak at 157 $\mu g/m^3$, followed by Huairou at 152 $\mu g/m^3$, and Wanliu with the lowest peak at 145 $\mu g/m^3$ over the 4-year period. Additionally, differences are evident in the trends following the initial yearly peak: while Huairou bottoms out after 1.5 years, Dongsi and Wanliu maintain relatively higher SO_2 levels before bottoming out at the 2-year mark. This is also confirmed by our interpolation where Huairou has the steepest decline when comparing the beginning of the year interpolation. These variations highlight regional disparities in SO_2 concentration magnitudes and decline patterns.

- Why do we see these differences?



The differences in SO_2 levels between Dongsi, Huairou, and Wanliu can be attributed to a combination of urbanization, local emission sources, geography, and meteorological factors:

1. **Urbanization and Local Emissions**:
    - Dongsi, being in central Beijing, experiences high levels of traffic, industrial emissions, and energy consumption, all of which contribute to elevated SO_2 levels.
    - Wanliu, while still urban, is less central and likely has fewer sources of heavy pollution compared to Dongsi.
    - Huairou, in contrast, is more suburban or rural, with fewer industrial sources and lower traffic density, leading to generally lower SO_2 concentrations.
2. **Geography and Land Use**:
    - Huairou's location in a less developed area with more vegetation and open spaces may facilitate natural pollutant absorption and better air quality overall.
    - Urban areas like Dongsi and Wanliu have more built-up environments that trap pollutants and limit natural dispersion.
3. **Meteorological Influences**:
    - Wind patterns, temperature inversions, and humidity can vary significantly between these locations. Huairou may benefit from more favorable wind conditions that disperse pollutants more effectively.
    - Urban heat islands in Dongsi and Wanliu could lead to stable air layers that trap SO_2 and slow its decline.
4. **Topographical Effects**:
    - Differences in elevation or proximity to mountains (Huairou being closer to the Yan Mountains) might influence pollutant accumulation and dispersion. Elevated areas can experience better airflow, enhancing pollutant dispersal.

These factors collectively explain the higher SO_2 peaks in urban Dongsi, the slightly lower peaks in Wanliu, and the more rapid post-peak decline in rural Huairou.

- What real-world implication or insight can be obtained?

These findings highlight the importance of tailoring air quality management strategies to the specific characteristics of each region. Urban areas like Dongsi may require stricter regulations on industrial emissions and vehicle pollution, whereas efforts in Huairou might focus on preserving green spaces to maintain its lower pollution levels. Understanding these patterns can guide policymakers in designing targeted interventions to improve air quality across diverse regions.

V. Summary

From this project, we gained significant insights into the effectiveness and characteristics of various interpolation techniques. We observed how different methods vary in accuracy, efficiency, and suitability for different datasets. Additionally, our focus on the choice of interpolation points highlighted the impact of data point selection. We learned that the distribution and spacing of points can lead to stark differences in the resulting interpolants, influencing both the accuracy of the approximation and the ability to capture essential trends in the data. This understanding is important for selecting appropriate methods and data configurations in practical applications.

In our dataset, we specifically noted that choosing points in the middle of the day or week captured the highs and lows of the data more effectively, while selecting points at the start of the day or week tended to emphasize the lower values. This observation showed us the importance of point selection to accurately represent and interpret the underlying data trends.

VI. Code Availability

The code for this project is available on: https://github.com/DevinRS/EECS639_Final