# Numerical Analysis Programming Report 2

Chen Wanqi 3220102895 *

*Information and Computer Science 2201, Zhejiang University*

October 28, 2024

## Problem A

The task is to write a program for Newton's interpolation polynomial, the specific implementation can be seen in ./src/interpolation.h and ./src/interpolation.cpp, and the design document can be found in ./DesignDoc.md.

## Problem B

Given the function

$$f(x) = \frac{1}{1+x^2},$$

and the rule for selecting interpolation points

$$x_i = -5 + 10\frac{i}{n}, \quad i = 0, 1, 2, ..., n; \quad (n = 2, 4, 6, 8)$$

test the program of Newton's interpolation polynomial from Problem A.

1. First, define the target function $f(x) = \frac{1}{1+x^2}$.

2. For each $n$, select the corresponding interpolation points according to the formula $x_i = -5+10\frac{i}{n}$, $i = 0, 1, 2, ..., n$, store them in the array x_vals, and calculate the corresponding function values, stored in the array f_vals.

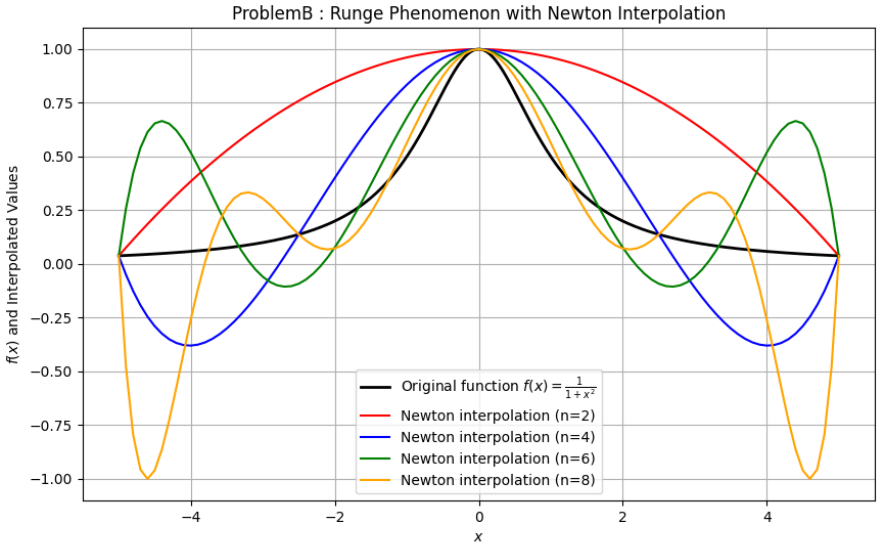
3. Call the initialization function of the NewtonInterpolator class to construct an instance, call the interpolate function to obtain the interpolation polynomial, and use the print function of the Polynomial class to output the polynomial expression.

4. To plot the image of the interpolation polynomial and compare it with the original function, use the evaluate function to calculate the values of the interpolation polynomial within the domain, store them in the file ./data/ProblemB.dat, and then use ./plot/plot_ProblemB.py to draw the image.

The plotted image is as follows:

It can be seen that the larger the $n$, the better the fitting effect of the interpolation polynomial with the original function.

*Electronic address: 3220102895@zju.edu.cn




# Problem C

In this problem, we performed Chebyshev interpolation on the given function

$$f(x) = \frac{1}{1 + 25x^2}$$

to improve the stability and fitting accuracy of the interpolation. We chose the zeros of the Chebyshev polynomial as the interpolation nodes, avoiding the Runge phenomenon. The selected interpolation points satisfy

$$x_i = \cos\left(\frac{(2i+1)\pi}{2n}\right), \quad i = 0, 1, 2, \ldots, n-1,$$

where $n = 5, 10, 15, 20$.

1. First, define the target function $f(x) = \frac{1}{1+25x^2}$.

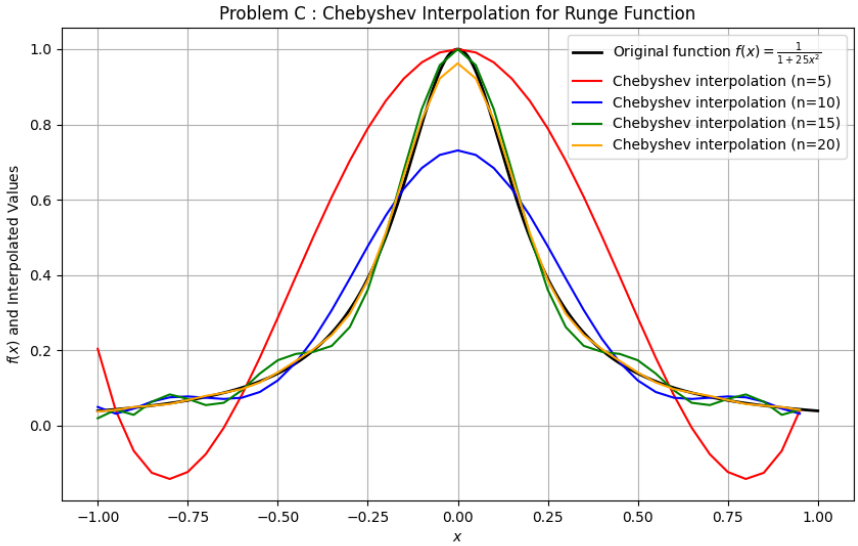
2. Use the formula $x_i = \cos\left(\frac{(2i+1)\pi}{2n}\right)$ to generate Chebyshev interpolation points, store the results in the x_vals array. Calculate the function values corresponding to each interpolation point and store them in the f_vals array.

3. Construct an instance of NewtonInterpolator for each $n$, and generate the corresponding interpolation polynomial Polynomial by calling the interpolate method.

4. Use the evaluate function to calculate the values of the interpolation polynomial on the interval $[-1, 1]$, and store the results in the file ./data/ProblemC.txt for subsequent use with ./plot/plot_ProblemC.py to draw the image.

The figure below shows the fitting effect of the interpolation polynomial and the original function:

By observing the image, it can be found that as $n$ increases, the interpolation polynomial gradually gets closer to the original function, especially the fitting effect near the interval endpoints is significantly improved. This shows that Chebyshev interpolation performs well in reducing edge errors and enhancing overall fitting effects.




# Problem D

The problem requires predicting the position and velocity of a car at $t = 10$ seconds based on the observation data shown in the table below using Hermite interpolation polynomial:

| Time (s) | Displacement (ft) | Velocity (ft/s) |
|---|---|---|
| 0 | 0 | 75 |
| 3 | 225 | 77 |
| 5 | 383 | 80 |
| 8 | 623 | 74 |
| 13 | 993 | 72 |

1. First, input the observation data required for Hermite interpolation, stored in the time, displacement, velocity three arrays.

2. Call the initialization function of the HermiteInterpolator class to construct an instance, and finally call the interpolate function to obtain the Hermite interpolation polynomial (the relationship between distance and time), use the derivative function of the Polynomial class to calculate the derivative to get the relationship between velocity and time.

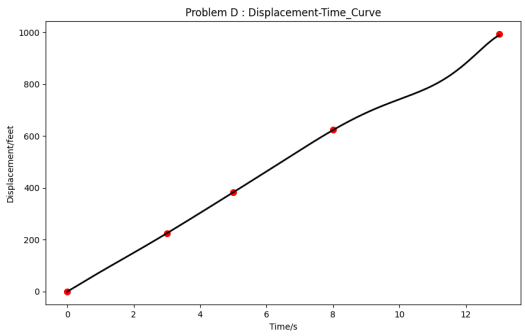
3. Output the velocity and displacement polynomials to the ./data/ProblemD.txt file for use with ./plot/plot_ProblemD.py to draw the image.

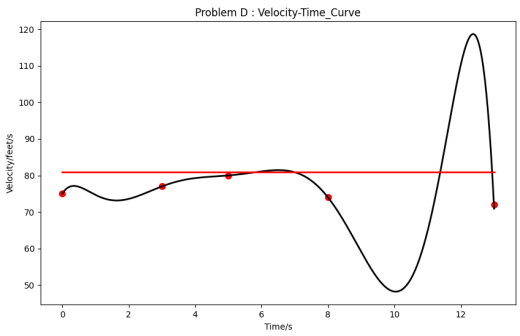
4. Call the evaluate function to calculate the position and velocity of the Hermite interpolation polynomial at $t = 10$ seconds, and output the results.

(a) According to the output results at $t = 10\,s$, Displacement= 742.503 feet, Velocity= 48.3817 feet/second.

(b) According to the plotted image, it can be seen that the car was speeding during this period.

The figure below shows the fitting effect of the Hermite interpolation polynomial and the original data:



# Problem E

The problem requires predicting the change of weight over time for two samples based on the observation data shown in the table below using Newton's interpolation polynomial:

| Day | Sample 1 Weight (g) | Sample 2 Weight (g) |
|---|---|---|
| 0 | 6.67 | 6.67 |
| 6 | 17.3 | 16.1 |
| 10 | 42.7 | 18.9 |
| 13 | 37.3 | 15.0 |
| 17 | 30.1 | 10.6 |
| 20 | 29.3 | 9.44 |
| 28 | 28.7 | 8.89 |

1. First, input the observation data required for Newton's interpolation, stored in the days, sp1_weights, sp2_weights three arrays.

2. Perform Newton's interpolation to obtain the polynomials for the change in weight over time for the two samples, and output them to the file ./data/ProblemE.txt, use ./plot/plot_ProblemE.py to draw the image.

3. Call the evaluate function to calculate the weight of the two samples at $t = 43$ days, and output the results.

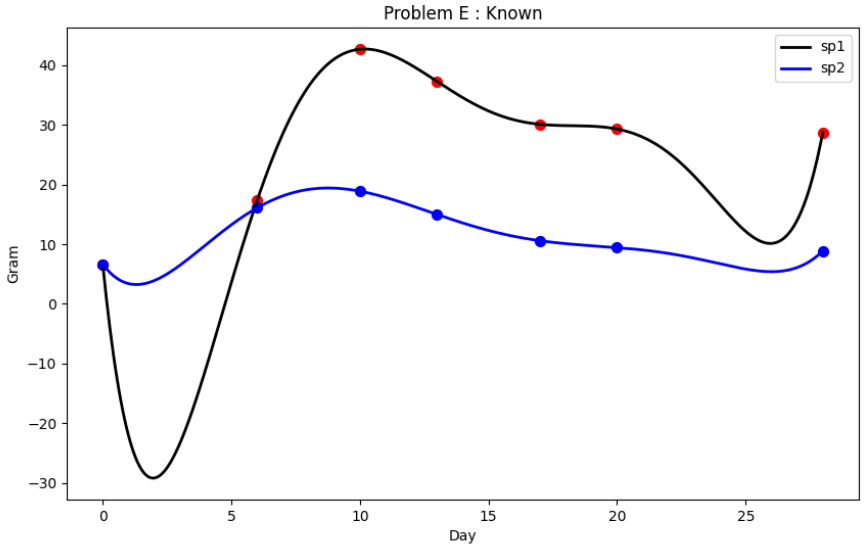
(a) From the output results, it can be seen:

```
Sample 1 Weight-Time Polynomial:
0.00x^6 -0.00x^5 +0.13x^4 -2.12x^3 +16.29x^2 -43.01x^1 +6.67x^0
Sample 2 Weight-Time Polynomial:
0.00x^6 -0.00x^5 +0.03x^4 -0.42x^3 +2.98x^2 -5.85x^1 +6.67x^0
```

(b) From the output results, it can be seen:

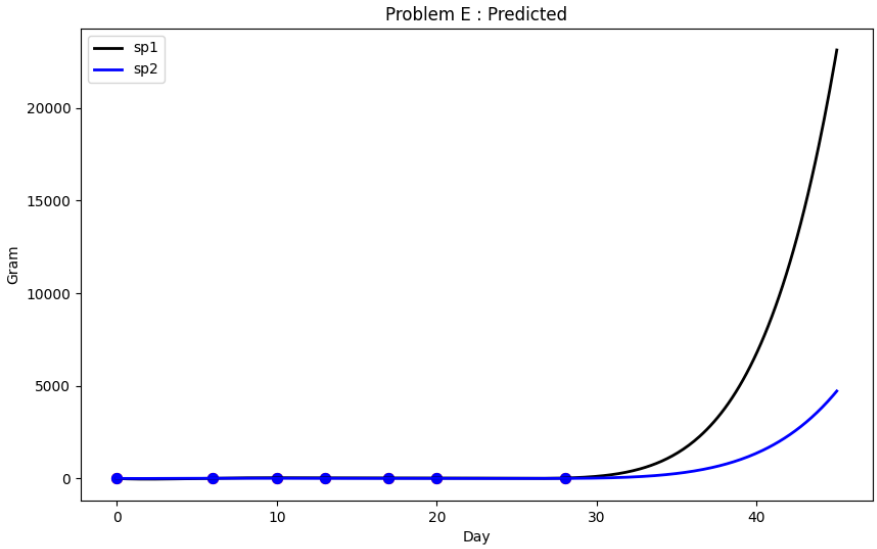
```
Predicted Weight of Sample 1 at \( t = 43.00 \): 14640.26 grams
Predicted Weight of Sample 2 at \( t = 43.00 \): 2981.48 grams
```

The figure below shows the change in weight over time for the two samples:




It can be seen that the fitting effect is good.

The figure below is the prediction of the change in weight over time for the two samples to 43 days:




It can be seen that after more than 30 days, the weight increase rate of the two samples has an abnormal increase, indicating that Newton's interpolation has poor predictive effect outside the given interpolation point range.

# Problem F

This problem gives the expression of the heart-shaped curve

$$x^2 + \left(\frac{3}{2}y - \sqrt{|x|}\right)^2 = 3$$

and requires fitting the heart-shaped curve with Bezier curves.

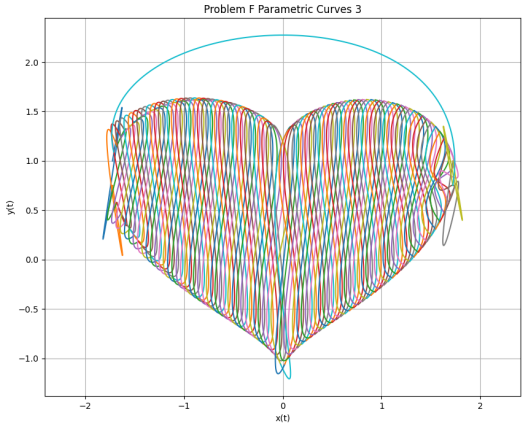
1. First, define the heart-shaped curve function heart_function, and the tangent vector function heart_tangent.

2. Set the value of $m$, for each $m$, within the domain $(-\sqrt{3}, \sqrt{3})$, call find_points to uniformly select $m+1$ points, uniform point selection is beneficial to the fitting effect of all points of the fitted Bezier curve.

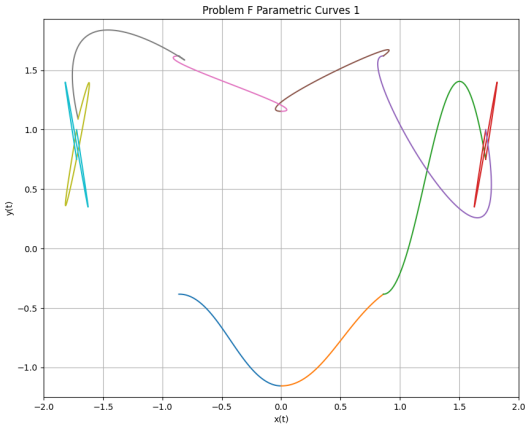
3. Use the convert_to_control_points function to convert $m+1$ points into $m$ groups of Bezier curve control points, each group contains 4 points.

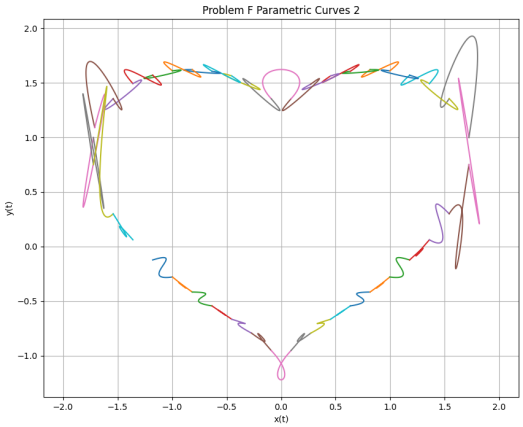
4. Construct the Bezier curve in turn with the control points of the $m$ groups, call the fileOut function to output the results. Use ./plot/plot_ProblemF.py to draw the image.

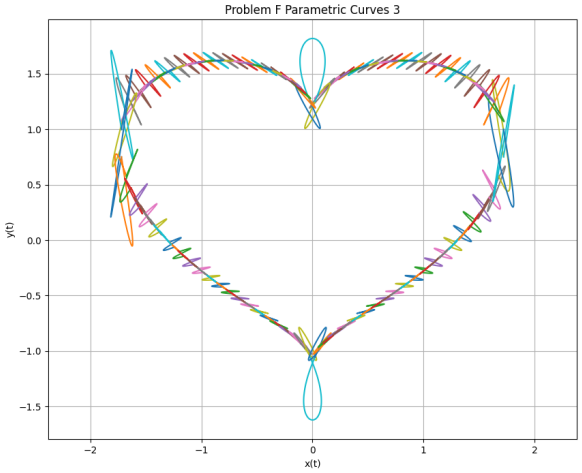
It is noted here that if the selected points are not sorted in a clockwise or counterclockwise order, the drawn image will appear as shown in the following figure:



Therefore, we calculated the polar angle of the selected points in the find_points function and sorted the points in ascending order of the polar angle, resulting in the following fitting effect: fitting effects when $m = 10, 40$, and 160.




It can be seen that as $m$ increases, the Bezier curve gradually gets closer to the heart-shaped curve.

However, there are still some problems, such as at $x = 0$ and at $x = -\sqrt{3}$ and $x = \sqrt{3}$, due to non-differentiability, the fitting effect is poor, so the appropriate selection of the find_points interval is very important.

# References


- handoutsNumPDEs
- ChatGPT, *AI Language Model*, OpenAI Platform, 2024.