暨南大学本科实验报告专用纸

课程名称 数值计算实验 成绩评定

实验项目名称 Interpolation problem 指导教师 Liangda Fang

实验项目编号 02 实验项目类型 综合性 实验地点 N116

学生姓名 甄洛生、吴凯一 学号 2018054625、2018050821

学院 信息科学技术学院 系 计算机系 专业 计算机科学技术专业

实验时间 2020 年 10 月 29 日下午 10:30~ 12:10

I. Problems

Let $f(x) = e^{-2x}$ and the interval to be $[-1, 1]$.

1. Write a program generating the Newton's divided difference formula;
2. Use the program to generate a degree n polynomial with evenly spaced points and Chebyshev points for n = 10, 20 and 40;
3. Plot the polynomials for the above types (see Figure 3.8);
4. By sampling at a 0.05 step, create the empirical interpolation errors for each type, and plot a comparison (see Figure 3.11).

II. Algorithm summary

● About interpolation

Before we begin, I would like to talk about the problem of interpolation.

To put it simply, it is to construct a function so that it passes through a given sample point. In this way, **continuous functional relations** can be obtained through **finite discrete data**. We assume that these sample points are different from each other.

There are many methods to construct such functions, such as the simplest linear interpolation, or **Lagrange interpolation**, **Newton's divided difference interpolation**, **Hermite interpolation**, **spline interpolation** and so on.

This experimental report mainly discusses the application of Newton difference quotient interpolation method.

● Newton's divided difference

First of all, we give the formula of Newton's divided difference interpolation polynomial:

$$f(x) = c_0 + c_1(x - x_1) + \cdots + c_{n-1}(x - x_1)(x - x_2) \ldots (x - x_{n-1}) \qquad (1-1)$$

Well, **the crux** is how to find the **coefficients** in polynomials? Now, let's talk about what the divided difference is:

Degree of **1** divided difference:

$$f[x_i, x_j] = \frac{y_i - y_j}{x_i - x_j}, \quad i \neq j \qquad (1-2)$$

Degree of **2** divided difference:

$$f[x_i, x_j, x_k] = \frac{f[x_i, x_j] - f[x_j, x_k]}{x_i - x_k}, \quad i \neq k \tag{1-3}$$

Degree of **n** divided difference:

$$f[x_1, x_2, \dots, x_n] = \frac{f[x_1, x_2, \dots, x_{n-1}] - f[x_2, x_3, \dots, x_n]}{x_1 - x_n} \tag{1-4}$$

Let's look at a simple example：

Example

$$\begin{array}{lll} x_1 & | & f[x_1] \\ & | & f[x_1 x_2] = \dfrac{f[x_2] - f[x_1]}{x_2 - x_1} \\ x_2 & | & f[x_2] \qquad\qquad f[x_1 x_2 x_3] = \dfrac{f[x_2 x_3] - f[x_1 x_2]}{x_3 - x_1} \\ & | & f[x_2 x_3] = \dfrac{f[x_3] - f[x_2]}{x_3 - x_2} \\ x_3 & | & f[x_3]. \end{array}$$

All we need to do is:

$$\begin{cases} c_i = y_1, i = 0 \\ c_i = f[x_1, \dots, x_{i+1}], i = 1,2, \dots, n-1 \end{cases}$$

And we get the Newton's divided difference polynomial:

$$\begin{aligned} f(x) = y_1 + f[x_1, x_2](x - x_1) + \cdots \\ + f[x_1, \dots, x_{i+1}](x - x_1)(x - x_2) \dots (x - x_{n-1}) \end{aligned} \tag{1-5}$$

Newton's divided difference formula looks so weird, right? How did Newton derive it? Let me (i.e. genius) lead you walk to his mind.

First, we want to make $f(x_1) = y1$, so:

$$f(x) = y_1$$

Now we want $f(x_2) = y_2$, so:

$$f(x) = y_1 + b_1(x - x_1)$$

This can satisfy $f(x_1) = y_1$, and we just let $b_1 = \frac{y_2 - y_1}{x_2 - x_1}$, it holds. Now we want $f(x_3) = y_3$, so:

$$f(x) = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + b_2(x - x_1)(x - x_2)$$

Like above, $f(x_1) = y_1$ *and* $f(x_2) = y_2$ can be satisfied. Just let:

$$b_2 = \frac{\frac{f(x_3) - f(x_2)}{x_3 - x_2} - \frac{f(x_2) - f(x_1)}{x_2 - x_1}}{x_3 - x_1}$$

Then, $f(x_3) = y_3$ is satisfied! So, Newton observed the form of b_i, and so-called divided difference derived! What a genius!

● Runge phenomenon

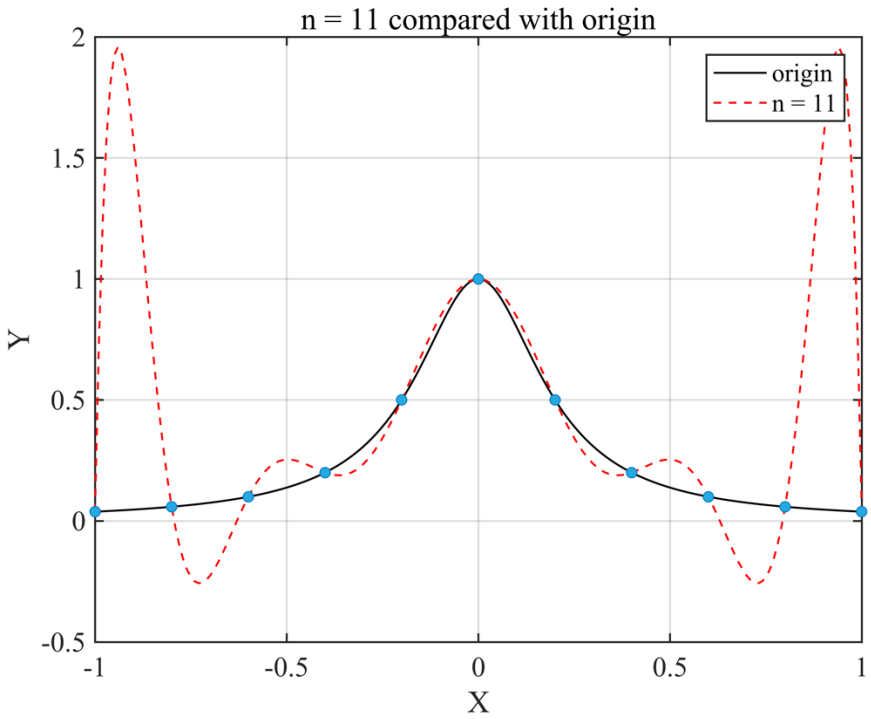
Is it true that the more sample points we have, the higher the order of the interpolation polynomial, the smaller the interpolation error? The answer is no, which is confirmed by the

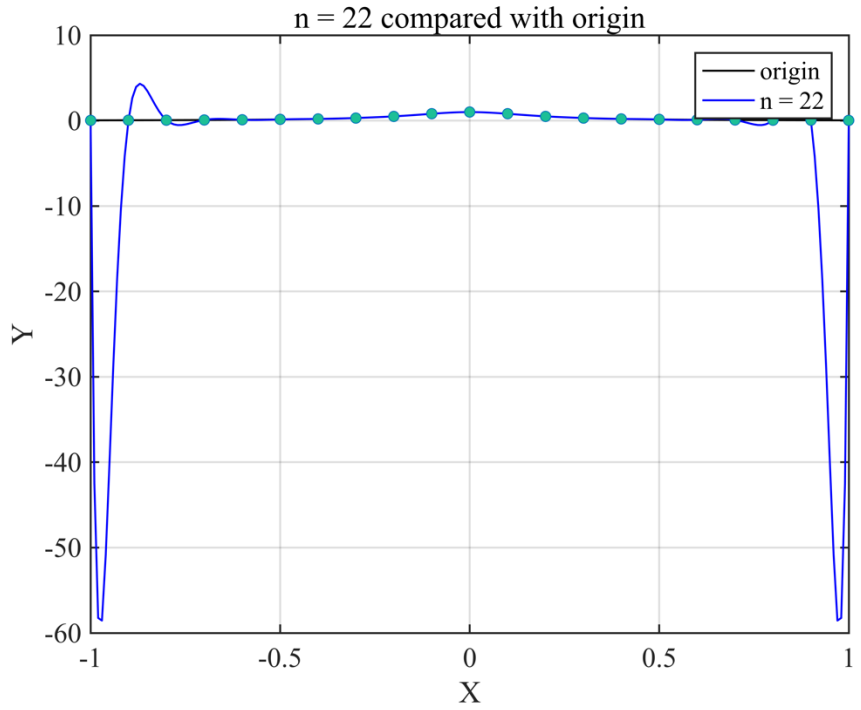
Runge phenomenon.

Given a function:

$$runge(x) = \frac{1}{1+25x^2}, x \in [-1,1] \qquad (2-1)$$

Using 11 and 22 sample points with **equidistant intervals** and apply them to Newton's divided difference. The interpolation polynomial curve is compared with the original function curve as follows:





It can be seen that with the increase of the number of sample points, the interpolation

polynomial **does not converge** to the original function, and there is a large error at both ends of the interval, which is the so-called Runge phenomenon.

The main reason is that the analytic area of the interpolated function is too small. For a more detailed proof, see https://www.zhihu.com/question/39329749/answer/149497977.

But does this contradict Weierstrass's theorem?

In fact, it is not, because the Weierstrass theorem says that there is a polynomial of sufficient high order to approximate, and does not require equaling at the sample point.

● Chebyshev points

Above we mentioned the concept and causes of the Runge phenomenon, so how should we select the sample points to **lighten** the Runge phenomenon?

The most intuitive method is making the **maximum error** small enough. The interpolation error, which is in the form of the following:

$$err(x) = f(x) - P_n(x) = \frac{\prod_{i=1}^{n}(x - x_i)}{n!} f^{(n)}(c) \qquad (3-1)$$

You may be curious about how the interpolation error is derived. Now I will give you the derivation process below:

$$\begin{aligned}
&If\ we\ insert\ one\ more\ point\ called\ x \\
&P_{n+1}(x) = P_n(x) + f[x_1, x_2, \dots, x_n, x](x - x_1) \dots (x - x_n) \\
&Now\ we\ have \\
&f(x) = P_{n+1}(x) \\
&Assume\ h(x) = f(x) - P_{n+1}(x) = 0 \\
&\therefore f(x) - P_n(x) = f[x_1, x_2, \dots, x_n, x](x - x_1) \dots (x - x_n)
\end{aligned}$$

Now we want to prove $f[x_1, x_2, \dots, x_n, x] = \frac{f^{(n)}(c)}{n!}$.

$$\begin{aligned}
&\because h(x_1) = h(x_2) = \cdots = h(x_n) = h(x) = 0 \\
&\therefore \exists a_i, h'(a_i) = 0, i = 1,2, \dots, n \\
&\therefore \exists b_i, h''(b_i) = 0, i = 1,2, \dots, n-1 \\
&Finally, it\ has\ h^{(n)}(c) = 0, c \in [\min\{x_1, \dots, x\}, \max\{x_1, \dots, x\}] \\
&\because h^{(n)}(x) = f^{(n)}(x) - n!\, f[x_1, x_2, \dots, x_n, x] \\
&\therefore f[x_1, x_2, \dots, x_n, x] = \frac{f^{(n)}(c)}{n!}
\end{aligned}$$

In order to simplify the problem, we do not consider the derivative part.
In other words, **the crux** is:

$$\min\{\max\{|(x - x_1)(x - x_2) \dots (x - x_n)|\}\} = ? \qquad (3-2)$$

It may be a little **abstract** to understand it based on the formula alone. Therefore, let me tell you what happen in the formula.

Our goal is to find a specific set of coefficients $(x_1, x_2, \dots, x_n)$, and we get this monic polynomial called $P_n(x)$, and **the extreme value** of this polynomial is the **smallest** of **all other polynomials** with other coefficients.

Next, let's talk about Chebyshev polynomials. His general formula is as follows:

$$T_n(x) = \cos(n * \cos^{-1} x) \qquad (3-3)$$

You may notice that this is a **trigonometric function** at all, but in fact, it can be proved to be a **polynomial** as bellow:

$Let\ y\ =\ arccosx, i.e., cosy\ =\ x.$

$T_{n+1}(x) = \cos(n\ +\ 1)\,y\ = \cos(ny\ +\ y) =\ cosnycosy\ -\ sinnysiny$

$T_{n-1}(x) = \cos(n\ -\ 1)\,y\ = \cos(ny\ -\ y) =\ cosnycosy\ +\ sinnysiny$

$T_{n+1}(x) +\ T_{n-1}(x) =\ 2\cos ny\cos y =\ 2xTn(x)$

$T_{n+1}(x) =\ 2xT_n(x) -\ T_{n-1}(x)$

$\because T_0(x) = 1, T_1(x) = x$

$\therefore T_2(x) = 2x * x - 1 = 2x^2 - 1$

$\therefore T_3(x) = 2x(2x^2 - 1) - x = 4x^3 - 3x$

Interestingly, Chebyshev polynomial have the following **properties**:

1. $deg(T_n) = n$
2. The leading coefficient of $T_n(x)$ is $2^{n-1}\ for\ n \geq 1$

Now we will use these two properties to prove that the minimum extreme value of a monic polynomial **can only be** $\frac{1}{2^{n-1}}$:

Counterproof method:

$Q_n(x)\ is\ a\ monic\ polynomial, and\ |Q_n(x)| < \frac{1}{2^{n-1}}, x \in [-1,1]$

$Assume\ f(x) = Q_n(x) - \frac{T_n(x)}{2^{n-1}}$

$when\ x = \cos\frac{2k\pi}{n}, k \in \mathbb{Z}\ and\ 2k \in [0,n], f(x) < 0$

$when\ x = \cos\frac{(2k+1)\pi}{n}, k \in \mathbb{Z}\ and\ 2k+1 \in [0,n], f(x) > 0$

$By\ intermediate\ value\ theorem, f(x)\ has\ at\ least\ n\ roots.$

$But\ \deg(f) \leq n - 1$

$\therefore f(x)\ has\ at\ most\ n-1\ roots.$

$\therefore \exists x_0, |Q_n(x_0)| \geq \frac{1}{2^{n-1}}$

Noted in $(3-2)$, our stuff is also a **monic polynomial** with degree of n. And obviously:

$$\exists x, |(x - x_1)(x - x_2)\ldots(x - x_n)| \geq \frac{1}{2^{n-1}}$$

That is to say:

$$\min\{\max\{|(x - x_1)(x - x_2)\ldots(x - x_n)|\}\} = \frac{1}{2^{n-1}} \qquad (3-4)$$

Now we care about what the value of $(x_1, x_2, \ldots, x_n)$ is, the equation holds. Noted that when $x_i = \cos\frac{(2i-1)\pi}{2n}, i = 1,2,..,n,\ T_n(x) = 0$. That is to say:

$$\prod_{i=1}^{n}(x - x_i) = \frac{T_n(x)}{2^{n-1}} \qquad (3-5)$$

$\because |T_n(x)| \leq 1$

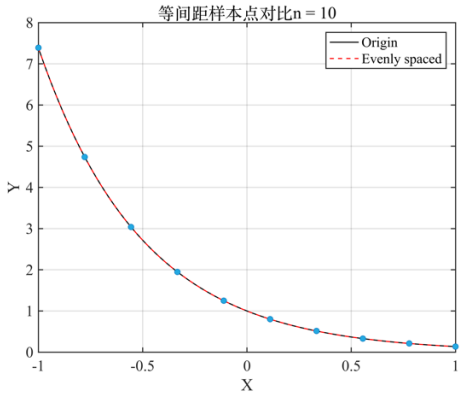
$\therefore \max\left(\left|\frac{T_n(x)}{2^{n-1}}\right|\right) = \frac{1}{2^{n-1}}$

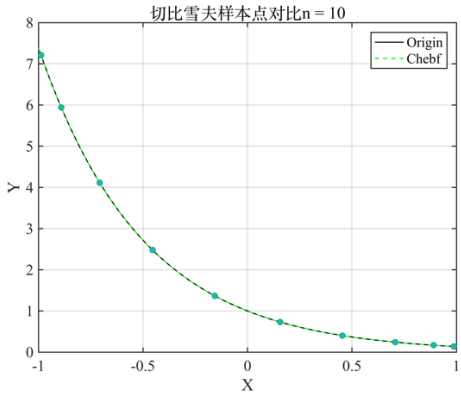
That is exactly what we want! We get it, we make it, right? Now we just select

Chebyshev points to interpolate, and it can definitely lighten the Runge Phenomenon!

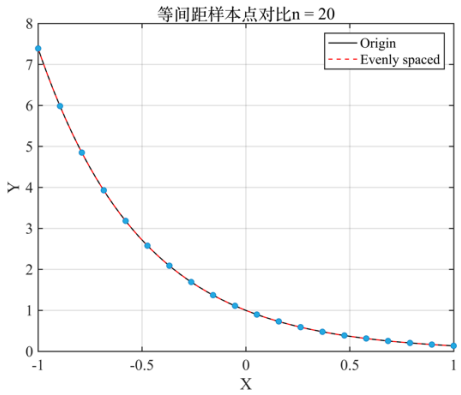
III. Result analysis

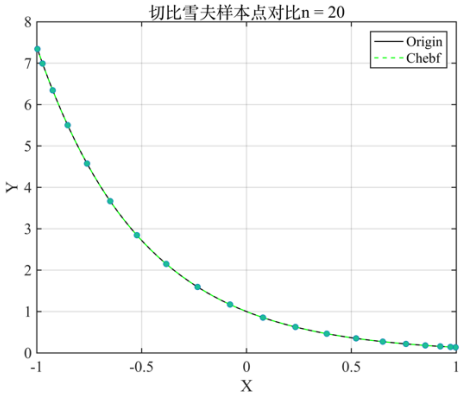
When n = 10, step 3 result as bellow:



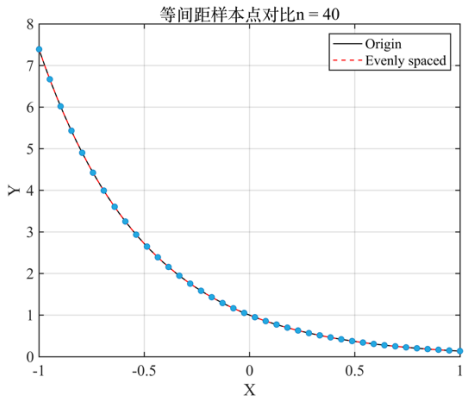


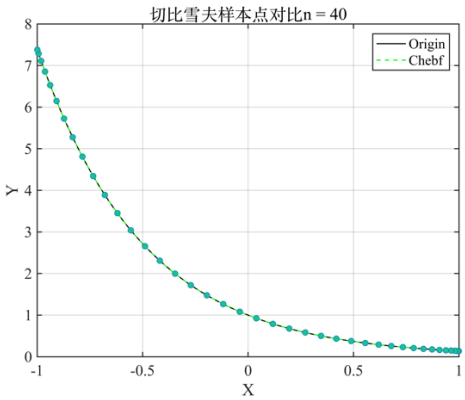
When n = 20, step 3 result as bellow:





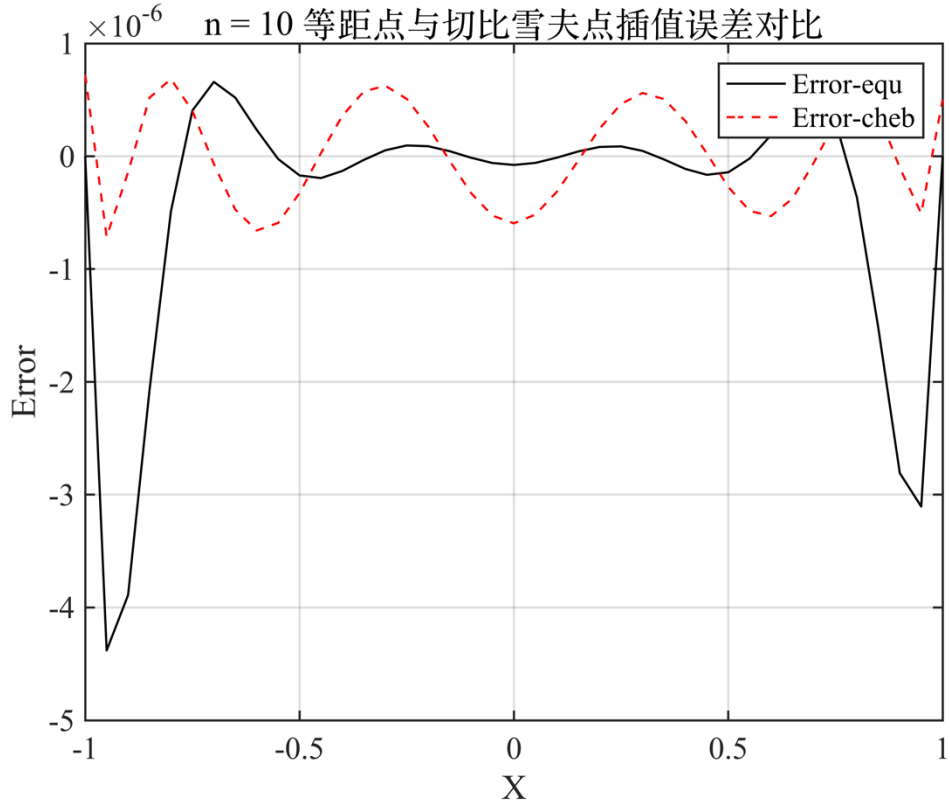
When n = 40, step 3 result as bellow:



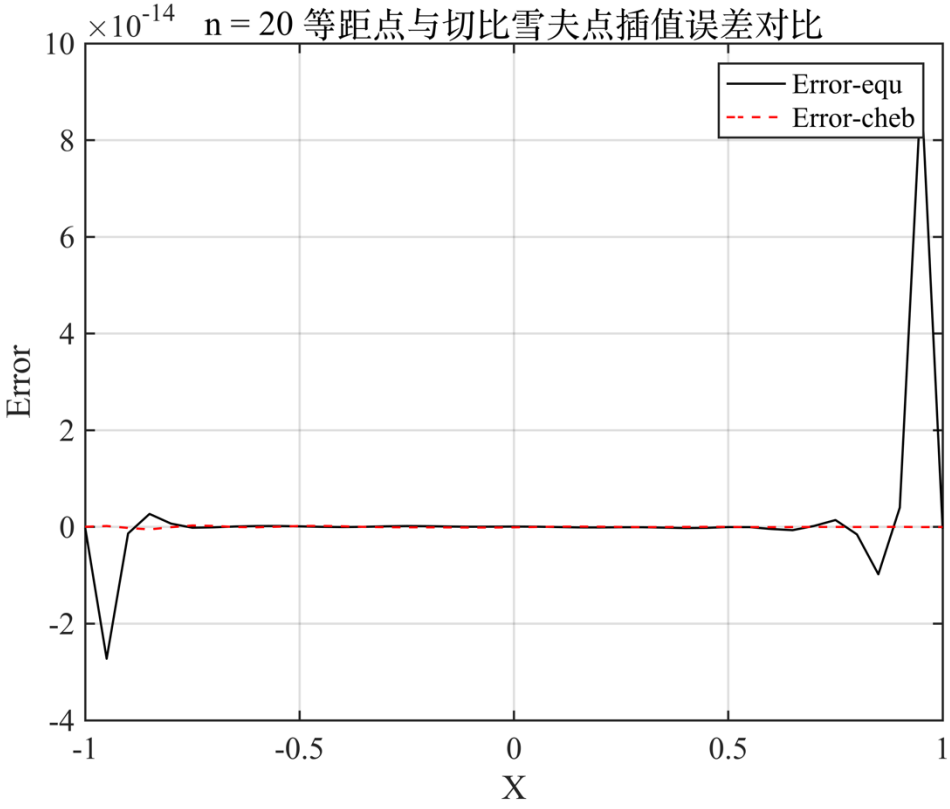


We can find that for e^{-2x}, the Runge phenomenon does **not** appear, and with the increase of the number of sample points, the error with the original function is negligible. That is to say, no Runge phenomenon.

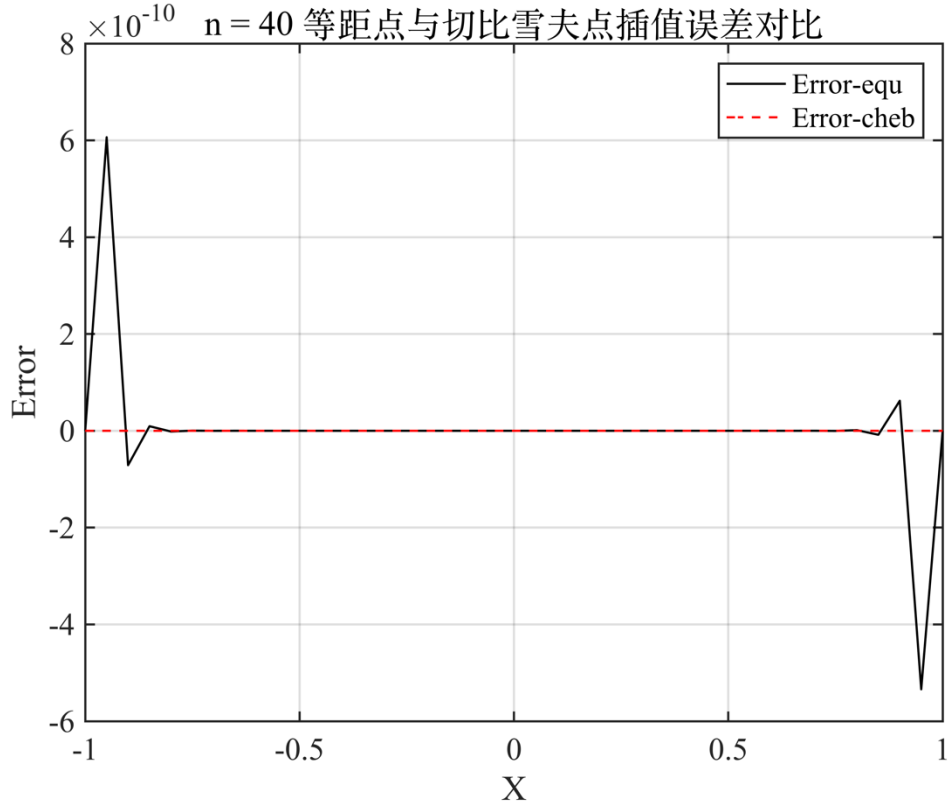
When n = 10, step 4 result as bellow:



When n = 20, step 4 result as bellow:



When n = 40, step 4 result as bellow:



We can find that the error of interpolation using evenly spaced sample points will **increase sharply** on both sides of the interval.

On the other hand, the **error fluctuation** of interpolation using Chebyshev points is not obvious, and the interpolation error is smaller than that of evenly spaced sample points.

What's more, noted that when n is changed from 20 to 40, the maximum error after interpolation using evenly spaced sample points increases by **10000 times**!

IV. Experimental summary

In this experiment, we use the **Newton's divided difference** method to find the interpolation polynomial $P_n(x)$.

We take **evenly spaced** sample points and **Chebyshev** points respectively, and compare the effects of the two kinds of points on the **interpolation error**.

Let me know that it is best to take the Chebyshev points as the sample points, so as to ensure that the error of the interpolation polynomial is as small as possible. Taking evenly spaced sample points for some functions will cause the **Runge phenomenon**, resulting in a very large interpolation error on both sides of the interval. However, in this experiment, even if using the evenly spaced sample points to interpolation, there is no Runge phenomenon in e^{-2x}.

More importantly, it is **not** that the more the number of sample points, the better the interpolation effect, as reflected in changing from 20 to 40!

V. Appendix: Source code

main.m

```
%% 数值分析第二次实验 插值
% 作者：天才少年甄洛生
% 抄袭狗必死!!

%% 前情提要：
% 原函数 fx 为 e^(-2x)
% 定义域为[-1,1]
% 样本点数分别为 10 20 40
% 使用等距间隔采样和切比雪夫采样进行牛顿差商插值
clear;clc;
syms x
f(x) = exp(-2*x);
%% n = ?
n = input('Enter nums of sample points: ');
equ_x = linspace(-1,1,n);
chebx = cos((2*[1:n]-1)*pi/(2*n));
[equ_f, chebf] = step3(equ_x, chebx, f);
step4(equ_f,chebf,f,['n = ', num2str(n), ' 等距点与切比雪夫点插值误差对比'])
```

ndd.m

```
%% 数值分析第二次实验 插值
% 作者：天才少年甄洛生
% 抄袭狗必死!!

function f = ndd(sam_x,sam_y)
% ndd - Description
% ndd 为 Newton's divided difference 的缩写。
% 顾名思义，生成牛顿差商插值多项式
% x：样本点 x
% y：样本点 y
% f：插值多项式
% Syntax: f = ndd(x,y)
    n = length(sam_x);          % 获取样本点数
    table = zeros(n,n);
    table(:,1) = sam_y;         % 初始化差商表

    % 计算差商表
    for gap = [1: n-1]
        for count = [1: n-gap]
```

```
            table(count, gap+1) = (table(count+1,gap) - table(count,gap))/(sam_x(count+gap) - sam_x(count));
        end
    end
    % 形成系数表
    c = table(1,:);
    % 形成(x-x_i) x_matrix
    syms x
    x_mat = sym(ones(n,1));
    for iter1 = [2:n]
        x_mat(iter1:n,1) = x_mat(iter1:n,1) * (x - sam_x(iter1-1));
    end
    % 构成牛顿差商插值多项式
    f(x) = c * x_mat;
    % 化简，返回
    f = simplify(f);
end
```

runge.m

```
%% 数值分析第二次实验 插值
% 作者：天才少年甄洛生
% 抄袭狗必死!!

%% 龙格现象素材生成脚本
clear,clc;
sam_x1 = linspace(-1,1,11);
sam_y1 = 1 ./ (1 + 25 * sam_x1.^2);

sam_x2 = linspace(-1,1,21);
sam_y2 = 1 ./ (1 + 25 * sam_x2.^2);

poly_f11 = ndd(sam_x1,sam_y1);
poly_f22 = ndd(sam_x2,sam_y2);

x = [-1:0.01:1];
y1 = 1 ./ (1 + 25 * x.^2);
y2 = poly_f11(x);
y3 = poly_f22(x);

% 开始画图 origin with n = 11
figure(1)
plot(x, y1, 'k-', 'linewidth', 1.1)
hold on
grid on
```

```
plot(x, y2, 'r--', 'linewidth', 1.1)
plot(sam_x1, sam_y1, 'o', 'markerfacecolor', [36, 169, 225]/255)

% 坐标轴边框线宽 1.1, 坐标轴字体与大小为 Times New Roman 和 16
set(gca, 'linewidth', 1.1, 'fontsize', 16, 'fontname', 'times')
xlabel('X')
ylabel('Y')
% legend 还可以使用 Location 参数设置图标位置
legend('origin', 'n = 11')
title('n = 11 compared with origin')
hold off

% 开始画图 origin with n = 22
figure(2)
plot(x, y1, 'k-', 'linewidth', 1.1)
hold on
grid on
plot(x, y3, 'b-', 'linewidth', 1.1)
plot(sam_x2, sam_y2, 'o', 'markerfacecolor', [29, 191, 151]/255)

% 坐标轴边框线宽 1.1, 坐标轴字体与大小为 Times New Roman 和 16
set(gca, 'linewidth', 1.1, 'fontsize', 16, 'fontname', 'times')
xlabel('X')
ylabel('Y')
% legend 还可以使用 Location 参数设置图标位置
legend('origin', 'n = 22')
title('n = 22 compared with origin')
hold off
```

step3.m

```
function [equ_f, chebf] = step3(equ_x, chebx, f)
    % 插值实验第三步，绘制插值多项式的图
    % equ_x = 等间距样本点
    % chebx = 切比雪夫样本点
    % f = 原函数 使用符号表达式

    % 统计样本点个数
    n = length(equ_x);
    % 计算对应样本点的 y 值
    equ_y = f(equ_x);
    cheby = f(chebx);
    % 生成插值函数
    equ_f = ndd(equ_x, equ_y);
    chebf = ndd(chebx, cheby);
```

```
15.
16.     x = [-1:0.01:1];
17.     y1 = f(x);
18.     y2 = equ_f(x);
19.     y3 = chebf(x);
20.
21.     % 原函数与等间距插值比较
22.     figure()
23.     plot(x, y1, 'k-', 'linewidth', 1.1)
24.     hold on
25.     grid on
26.     plot(x, y2, 'r--', 'linewidth', 1.1)
27.     plot(equ_x, equ_y, 'o', 'markerfacecolor', [36, 169, 225]/255)
28.
29.     % 坐标轴边框线宽 1.1, 坐标轴字体与大小为 Times New Roman 和 16
30.     set(gca, 'linewidth', 1.1, 'fontsize', 16, 'fontname', 'times')
31.     xlabel('X')
32.     ylabel('Y')
33.     % legend 还可以使用 Location 参数设置图标位置
34.     legend('Origin', 'Evenly spaced')
35.     title(['等间距样本点对比','n = ', num2str(n)])
36.     hold off
37.
38.     % 原函数和切比雪夫插值多项式比较
39.     figure()
40.     plot(x, y1, 'k-', 'linewidth', 1.1)
41.     hold on
42.     grid on
43.     plot(x, y3, 'g--', 'linewidth', 1.1)
44.     plot(chebx, cheby, 'o', 'markerfacecolor', [29, 191, 151]/255)
45.
46.     % 坐标轴边框线宽 1.1, 坐标轴字体与大小为 Times New Roman 和 16
47.     set(gca, 'linewidth', 1.1, 'fontsize', 16, 'fontname', 'times')
48.     xlabel('X')
49.     ylabel('Y')
50.     % legend 还可以使用 Location 参数设置图标位置
51.     legend('Origin', 'Chebf')
52.     title(['切比雪夫样本点对比','n = ', num2str(n)])
53.     hold off
54.
55. end
```

step4.m

```
1.  function [] = step4(equ_f, chebf, f, name)
```

```
    % 插值实验第三步，绘制插值多项式的图
    % equ_f = 等间距插值多项式
    % chebf = 切比雪夫插值多项式
    % f = 原函数 使用符号表达式

    % 以 0.05 间隔从[-1, 1]采样点进行误差计算
    x = -1:0.05:1;
    % 计算对应多项式的 y 值
    y = f(x);
    equ_y = equ_f(x);
    cheby = chebf(x);
    % 计算误差
    error_equ = y - equ_y;
    error_cheb = y - cheby;
    % 画图
    figure()
    plot(x, error_equ, 'k-', 'linewidth', 1.1)
    hold on
    grid on
    plot(x, error_cheb, 'r--', 'linewidth', 1.1)
    % 坐标轴边框线宽 1.1, 坐标轴字体与大小为 Times New Roman 和 16
    set(gca, 'linewidth', 1.1, 'fontsize', 16, 'fontname', 'times')
    xlabel('X')
    ylabel('Error')
    % legend 还可以使用 Location 参数设置图标位置
    legend('Error-equ', 'Error-cheb')
    title(name)
    hold off
end
```