**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 Problems 指导教师 Liangda Fang

实验项目编号 02 实验项目类型 综合型 实验地点 N116

学生姓名 蒋云翔 学号 2022102330

学院 国际学院 系 计算机系 专业 计算机科学与技术

实验时间2024年 11 月 8 日上午10:30～ 12:10

# Ⅰ、Problem

Let 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.

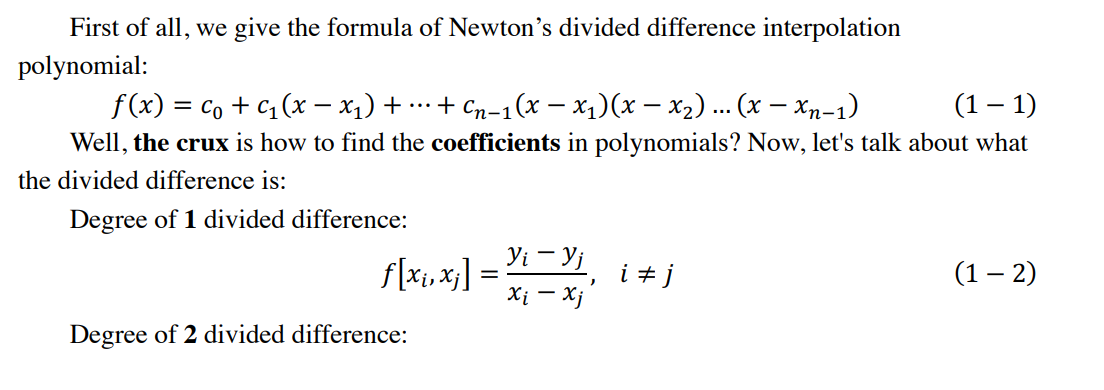
There are many methods to construct such functions, such as the simplest linear

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**



## Appendix : Source code

1. **main.m**

|  |
| --- |
| %% tips：  % Original function f(x) is e^(-2x)  % Domain is [-1,1]  % Sample points are 10 20 40  % Newton difference quotient interpolation is carried out  % using isometric interval sampling and Chebyshev sampling  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']) |

1. **ndd.m**

|  |
| --- |
| function f = ndd(sam\_x,sam\_y)  % ndd - Description  % ndd stands for Newton's divided difference.  % Generate Newton difference quotient interpolation polynomial  % x：Sample point x  % y：Sample point y  % f：Interpolation polynomial  % Syntax: f = ndd(x,y)  n = length(sam\_x); % Get sample points  table = zeros(n,n);  table(:,1) = sam\_y; % Initializes the difference quotient table  % Calculate the difference quotient table  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  % Table of formation coefficient  c = table(1,:);  % Form (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  % Form Newton difference quotient interpolation polynomial  f(x) = c \* x\_mat;  % Simplify, return  f = simplify(f);  end |

1. **runge.m**

|  |
| --- |
| %% Runge phenomenon material generating script  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);  % Start drawing 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)  % Axis border line width 1.1, The axis font and size are Times New Roman and 16  set(gca, 'linewidth', 1.1, 'fontsize', 16, 'fontname', 'times')  xlabel('X')  ylabel('Y')  % legend can also use Location parameter Setting icon position  legend('origin', 'n = 11')  title('n = 11 compared with origin')  hold off  % Start drawing 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)  % Axis border line width 1.1, The axis font and size are Times New Roman and 16  set(gca, 'linewidth', 1.1, 'fontsize', 16, 'fontname', 'times')  xlabel('X')  ylabel('Y')  % legend can also use Location parameter Setting icon position  legend('origin', 'n = 22')  title('n = 22 compared with origin')  hold off |

1. **step3.m**

|  |
| --- |
| function [equ\_f, chebf] = step3(equ\_x, chebx, f)  % The third step of the interpolation experiment is  % to draw the graph of the interpolation polynomial  % equ\_x = Equally spaced sample points  % chebx = Chebyshev sample point  % f = The original function uses symbolic expressions  % Count the number of sample points  n = length(equ\_x);  % Calculate the y value of the corresponding sample point  equ\_y = f(equ\_x);  cheby = f(chebx);  % Generated interpolation function  equ\_f = ndd(equ\_x, equ\_y);  chebf = ndd(chebx, cheby);  x = -1:0.01:1;  y1 = f(x);  y2 = equ\_f(x);  y3 = chebf(x);  % The original function is compared with equal spacing interpolation  figure()  plot(x, y1, 'k-', 'linewidth', 1.1)  hold on  grid on  plot(x, y2, 'r--', 'linewidth', 1.1)  plot(equ\_x, equ\_y, 'o', 'markerfacecolor', [36, 169, 225]/255)  % The axis border line width is 1.1, and the axis font and size are Times New Roman and 16  set(gca, 'linewidth', 1.1, 'fontsize', 16, 'fontname', 'times')  xlabel('X')  ylabel('Y')  % legend can also use the Location parameter to set the icon position  legend('Origin', 'Evenly spaced')  title(['等间距样本点对比','n = ', num2str(n)])  hold off  % Comparison of primitive function and Chebyshev interpolation polynomial  figure()  plot(x, y1, 'k-', 'linewidth', 1.1)  hold on  grid on  plot(x, y3, 'g--', 'linewidth', 1.1)  plot(chebx, cheby, 'o', 'markerfacecolor', [29, 191, 151]/255)  % The axis border line width is 1.1, and the axis font and size are Times New Roman and 16  set(gca, 'linewidth', 1.1, 'fontsize', 16, 'fontname', 'times')  xlabel('X')  ylabel('Y')  % legend can also use the Location parameter to set the icon position  legend('Origin', 'Chebf')  title(['切比雪夫样本点对比','n = ', num2str(n)])  hold off  end |

1. **step4.m**

|  |
| --- |
| function [] = step4(equ\_f, chebf, f, name)  % The fourth step of the interpolation experiment is  % to draw the graph of the interpolation polynomial  % equ\_f = Equally spaced interpolating polynomial  % chebf = Chebyshev interpolation polynomial  % f = The original function uses symbolic expressions  % The error was calculated from the [-1, 1] sampling point at 0.05 intervals  x = -1:0.05:1;  % Calculate the y value of the corresponding polynomial  y = f(x);  equ\_y = equ\_f(x);  cheby = chebf(x);  % Calculation error  error\_equ = y - equ\_y;  error\_cheb = y - cheby;  % draw  figure()  plot(x, error\_equ, 'k-', 'linewidth', 1.1)  hold on  grid on  plot(x, error\_cheb, 'r--', 'linewidth', 1.1)  % The axis border line width is 1.1, and the axis font and size are Times New Roman and 16  set(gca, 'linewidth', 1.1, 'fontsize', 16, 'fontname', 'times')  xlabel('X')  ylabel('Error')  % legend can also use the Location parameter to set the icon position  legend('Error-equ', 'Error-cheb')  title(name)  hold off  end |