**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

实验名称 Computing Problems 指导教师 Liangda Fang

试验编号 0806011602 类型 验证性

实验地点 N117 学生姓名 董春彤 学号 2017053110

实验地点 N117 学生姓名 邝庆璇 学号 2016051598

学院 信息科学技术学院 系 计算机 专业 计算机科学与技术

实验时间 2019 年 10 月 18 日上午

**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**

**● Newton’s divided difference formula**

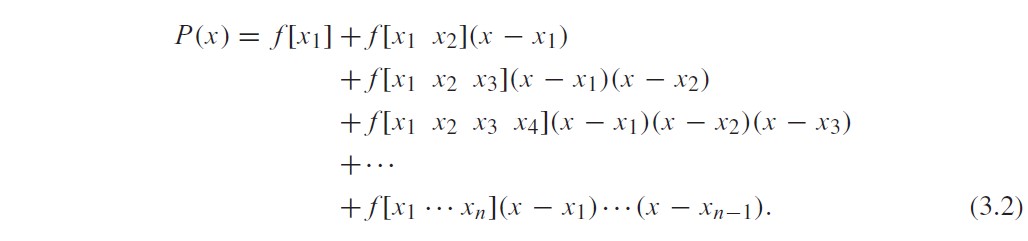
**Main Theorem of Polynomial Interpolation.**

Let (x1,y1), . . . , (xn,yn) be n points in the plane with distinct xi . Then there exists one and only one polynomial P of degree n − 1 or less that satisfies P(xi ) = yi for i = 1,...,n.

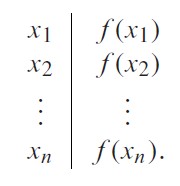
Newton’s divided differences give a particularly simple way to write the interpolating polynomial. Given n data points, the result will be a polynomial of degree at most n − 1.

The idea of divided differences is fairly simple, but some notation needs to be mastered first.Assume that the data points come from a function f (x), so that our goal is to interpolate (x1,f (x1)), . . . , (xn,f (xn)).

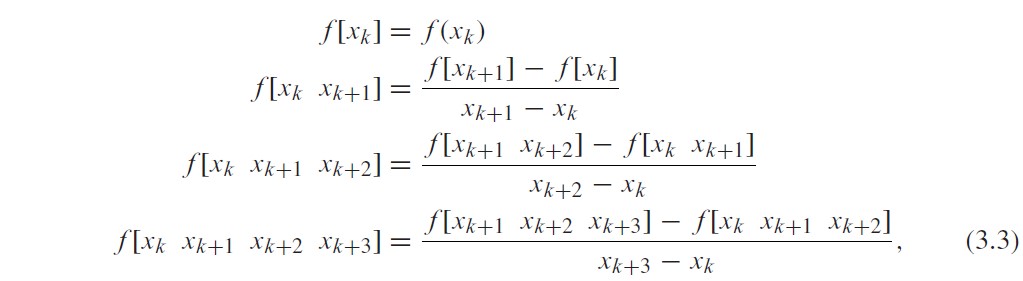
Denote by f [x1 . . . xn] the coefficient of the xn−1 term in the (unique) polynomial that interpolates (x1,f (x1)), . . . , (xn,f (xn)). Using this definition, the following somewhat remarkable alternative formula for the interpolating polynomial holds, called **the Newton’s divided difference formula:**



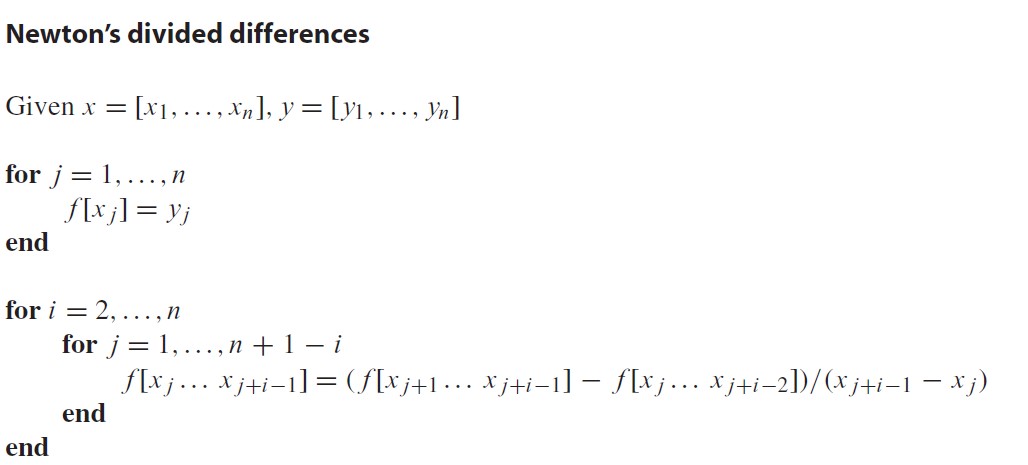
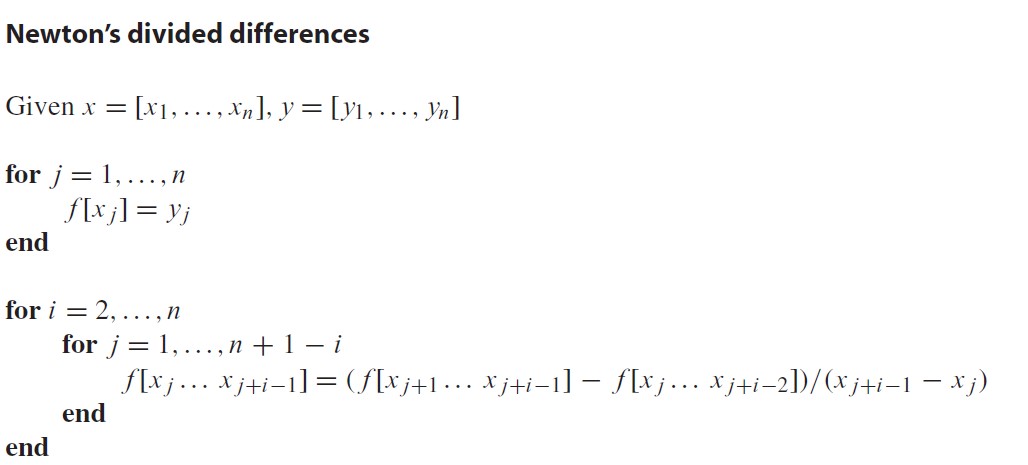
Moreover, the coefficients f [x1 . . . xk] from the above definition can be recursively calculated as follows. List the data points in a table:



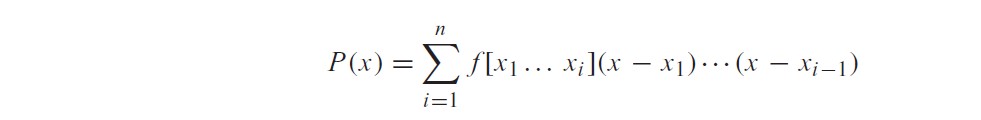
Now define the divided differences, which are the real numbers



and so on. Both important facts, that (1) the unique polynomial interpolating (x1,f (x1)), . . . , (xn,f (xn)) is given by (3.2) and (2) the coefficients can be calculated as (3.3), are not immediately obvious, and proofs will be provided in Section 3.2.2. Notice that the divided difference formula gives the interpolating polynomial as a nested polynomial. It is automatically ready to be evaluated in an efficient way.

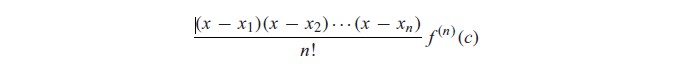
 

The interpolating polynomial is



## ● Chebyshev interpolation

The motivation for Chebyshev interpolation is to improve control of the maximum value of the interpolation error



on the interpolation interval. Let’s fix the interval to be [−1,1] for now.

The choice of real numbers −1 ≤ x1, . . . , xn ≤ 1 that makes the value of



and the minimum value is 1/2n−1. In fact, the minimum is achieved by



where Tn(x) denotes the degree n Chebyshev polynomial.

We conclude from the theorem that interpolation error can be minimized if the n interpolation base points in [−1,1] are chosen to be the roots of the degree n Chebyshev interpolating polynomial Tn(x). These roots are



where “odd’’ stands for the odd numbers from 1 to 2n − 1. Then we are guaranteed that the absolute value of (3.9) is less than 1/2n−1 for all x in [−1,1].

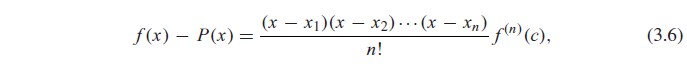
Choosing the Chebyshev roots as the base points for interpolation distributes the interpolation error as evenly as possible across the interval [−1,1]. We will call the interpolating polynomial that uses the Chebyshev roots as base points the Chebyshev interpolating polynomial.

## ● Interpolation error formula

Assume that we start with a function y = f (x) and take data points from it to build an interpolating polynomial P(x), as we did with f (x) = sin x in Example 3.7. The interpolation error at x is f (x) −

P(x), the difference between the original function that provided the data points and the interpolating polynomial, evaluated at x. The interpolation error is the vertical distance between the curves in Figure 3.3. The next theorem gives a formula for the interpolation error that is usually impossible to evaluate exactly, but often can at least lead to an error bound.

Assume that P(x) is the (degree n − 1 or less) interpolating polynomial fitting the n points (x1,y1), . . . , (xn,yn). The interpolation error is



where c lies between the smallest and largest of the numbers x,x1, . . . , xn.

**III. Source Code**

**1.** main.m

%a degree n polynomial with evenly spaced points

n = 10;

x\_sample\_even = [-1:2/n:1]

y\_sample\_even = exp(-2\*x\_sample\_even);

c\_even = newtdd(x\_sample\_even,y\_sample\_even,n+1);

x\_test = -1:.05:1;

y\_test\_even= nest(n,c\_even,x\_test,x\_sample\_even);

%generate a degree n-1 polynomial with Chebyshev points

n = 11;

x\_sample\_cheb = cos((1:2:2\*n-1)\*pi/(2\*n));

Y\_sample\_cheb = exp(-2\*x\_sample\_cheb);

c\_cheb = newtdd(x\_sample\_cheb, y\_sample\_cheb, n);

% x\_test has been defined above

y\_test\_cheb = nest(n-1, c\_cheb , x\_test, x\_sample\_cheb);

% polynomials -- even points

figure(1)

plot(x\_sample\_even, y\_sample\_even, 'o',x\_test, y\_test\_even)

% polynomials -- Chebyshev points

figure(2)

plot(x\_sample\_cheb,y\_sample\_cheb,'\*',x\_test,y\_test\_cheb,'--')

% empirical interpolation errors -- even points

h\_even = y\_test\_even - y\_sample\_even ;

% empirical interpolation errors -- Chebyshev points

h\_cheb = y\_test\_cheb - y\_sample\_cheb;

plot(x\_test,h\_even) hold on

plot(x\_test,h\_cheb,'--')

**2.** nest.m

% Compute the coefficients

function y = nest(d,c,x,b)

if nargin < 4

b = zeros(d, 1);

end

y = c\_chebyshev(d+1);

for i = d:-1:1

y = y.\*(x-b(i)) + c\_chebyshev(i);

end

**3.** newtdd.m

% Newton Divided Difference Interpolation Method

function c\_chebyshev = newtdd(x,y,n)

% Obtain Newton Triangle

for j = 1:n

v(j,1) = y(j); % f[xj]

end

for i = 2:n

for j = 1:n+1-i

v(j,i) = (v(j+1,i-1)-v(j,i-1))/(x(j+i-1)-x(j));

end

End

% Obtain the coefficients

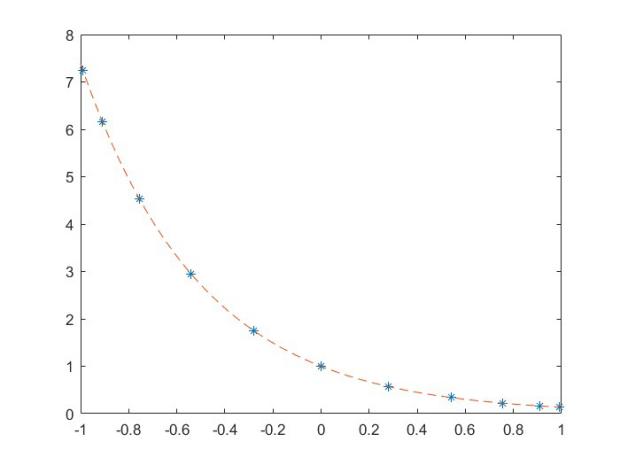
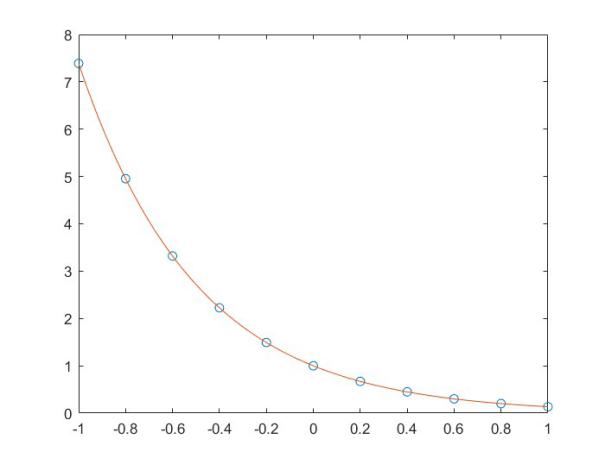
for i = 1:n

c\_chebyshev(i) = v(1,i); % Read along top of triangle

end

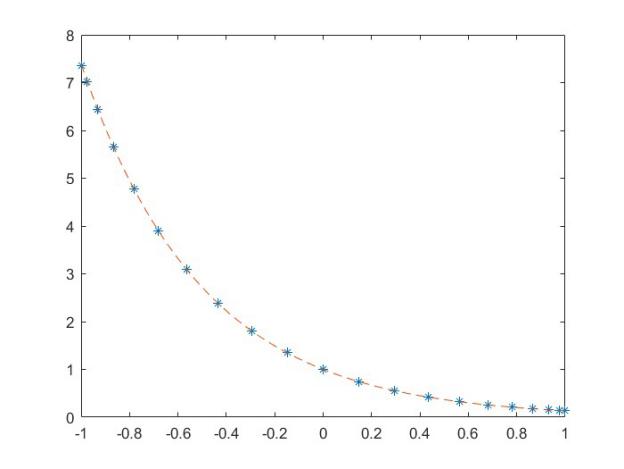
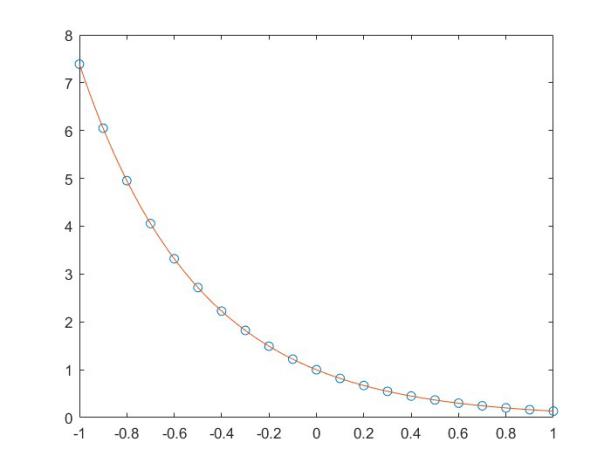
**IV. Result & Analysis**

generate a degree 10 polynomial with evenly spaced points and Chebyshev points

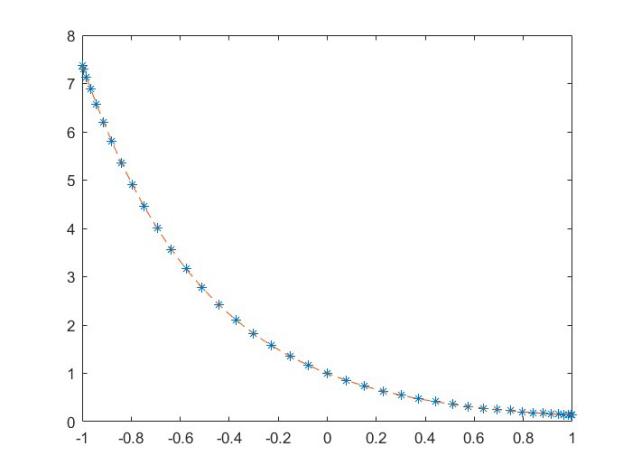
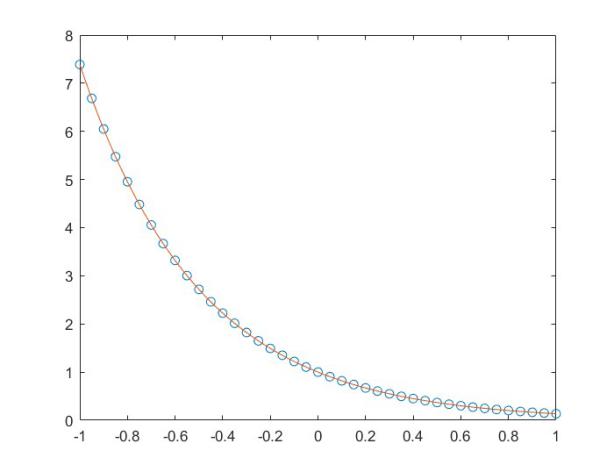


with evenly spaced points with Chebyshev points

generate a degree 20 polynomial with evenly spaced points and Chebyshev points

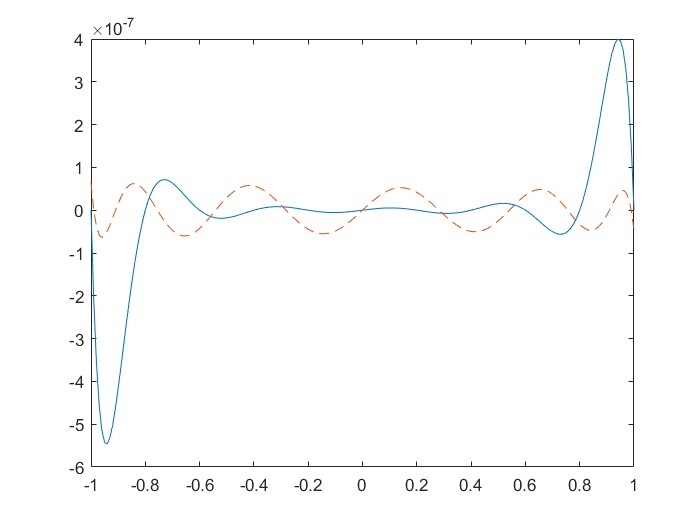


with evenly spaced points with Chebyshev points generate a degree 40 polynomial with evenly spaced points and Chebyshev points

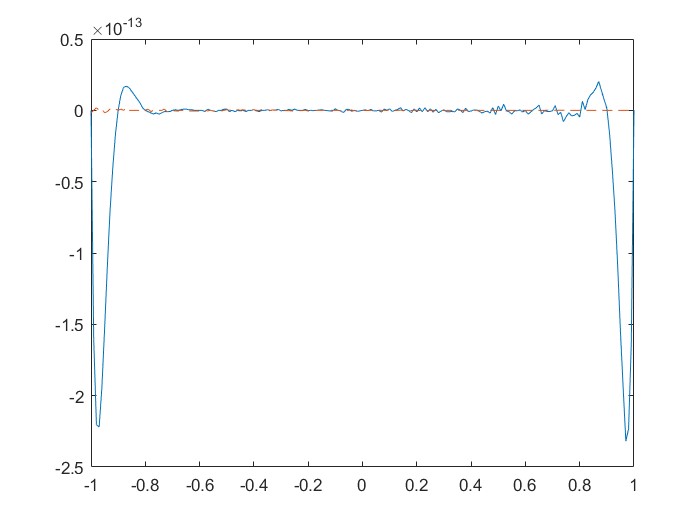


with evenly spaced points with Chebyshev points

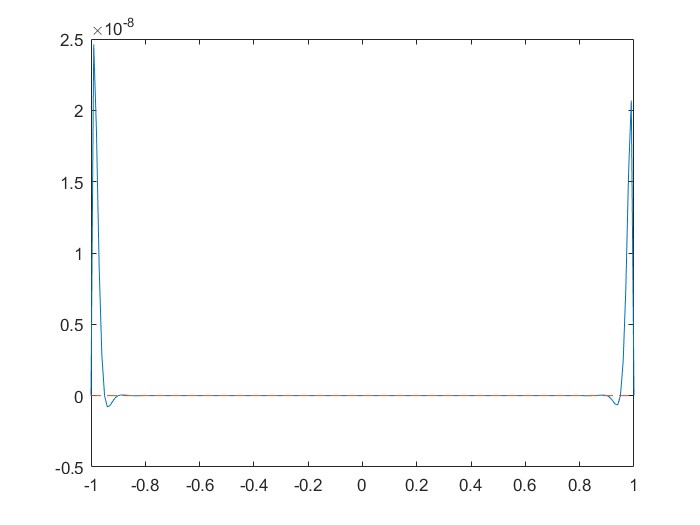
interpolation error for degree 10 interpolating polynomial with evenly spaced base points (bule solid curve) and Chebyshev base points (red dashed curve).



interpolation error for degree 20 interpolating polynomial with evenly spaced base points (bule solid curve) and Chebyshev base points (red dashed curve).



interpolation error for degree 40 interpolating polynomial with evenly spaced base points (bule solid curve) and Chebyshev base points (red dashed curve).



# Ⅴ、Experimental summary

In this experiment, nest.m and newttd.m are used to realize the interpolation polynomial by Newton’s divided method. Moreover, the roots of chebyshev are selected as the basis points of interpolation to solve the interpolation polynomial, so as to improve the control of the maximum interpolation error. And Compared the evenly spaced points and Chebyshev points as the error of interpolation points respectively.