

信息科学与工程学院

《计算方法》 实验报告

**系 别 计算机科学与工程系**

**专 业 计算机科学与技术**

**年 级 2020级**

**姓 名 刘子言**

**指导教师 郭卫斌**

**2021-2022 学年 第 2 学期**

**实验一 插值方法**

学号：20002462 姓名：刘子言

**一. 实验目的**

（1）熟悉数值插值方法的基本思想，解决某些实际插值问题，加深对数值插值方法的理解。

（2）熟悉Matlab编程环境，利用Matlab实现具体的插值算法，并进行可视化。

**二. 实验要求**

用Matlab软件实现Lagrange插值、分段线性插值、Hermite插值、Aitken逐步插值算法，并用实例在计算机上计算和作图。

1. **实验内容**

**3-1：已知正弦积分的数据表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***x*** | **0.3** | **0.4** | **0.5** | **0.6** | **0.7** |
| ***y*** | **0.29850** | **0.39646** | **0.49311** | **0.58813** | **0.68122** |

**构造适合该数据表的一次、二次和三次Lagrange插值公式，计算*x*=0.358, 0.462, 0.514, 0.635时*f*(*x*)的值，比较不同次数的插值公式的计算结果。**

**1、设计思想**

Lagrange插值方法要求插值函数p(x)与所逼近的函数f(x)在一系列的节点上去相同的函数值，实际上是用简单的曲线代替复杂的曲线来得到最终结果。根据节点数，可以分为两点插值、三点插值和多点插值公式，Lagrange插值公式的形式如下：

 （1）

Lagrange插值的算法设计：给定数据表以及插值节点x，根据公式（1），求得插值结果y。

Lagrange公式（1）具有累乘累加的嵌套结构，在逻辑上表现为二重循环，内循环（j循环）累乘求得系数，然后通过外循环（i循环）累加得出插值结果。

**2、对应程序**

题目3-1：Lagrange插值的函数文件名：“liuziyan\_3\_1.m”，代码如下：

function [y0, N] = liuziyan\_3\_1(X, Y, x0)

%Lagrange插值算法函数

%X,Y是已知的插值点坐标，N是Language插值函数的权系数

%x0是插值点，y0是Lagrenge多项式在x0处的值

m = length(X);

N = zeros(m, 1);

y0 = 0;

for i = 1:m

N(i) = 1;

for j = 1:m

if j~=i

N(i) = N(i)\*(x0-X(j))/(X(i)-X(j));%迭乘

end

end

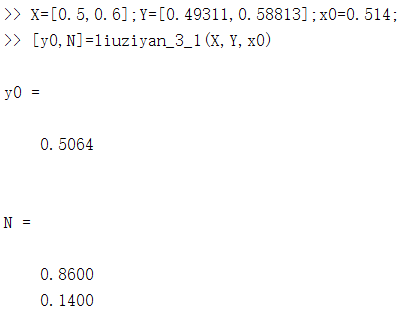
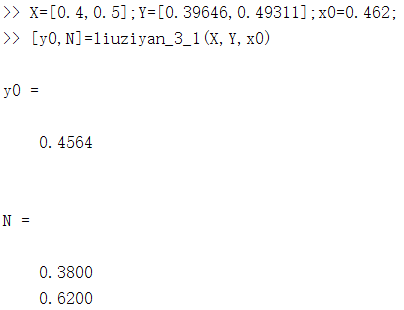
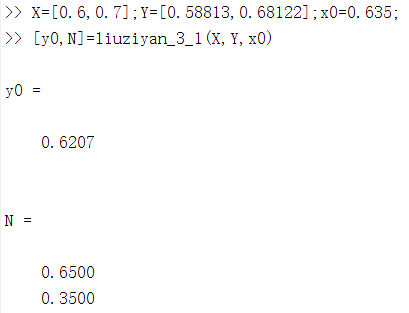
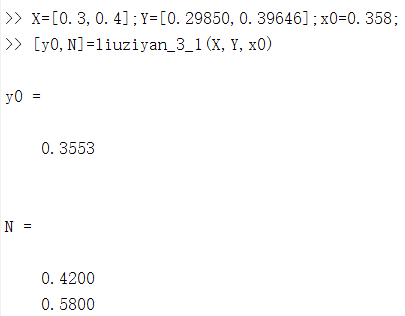
y0 = y0+Y(i)\*N(i);%迭加

end

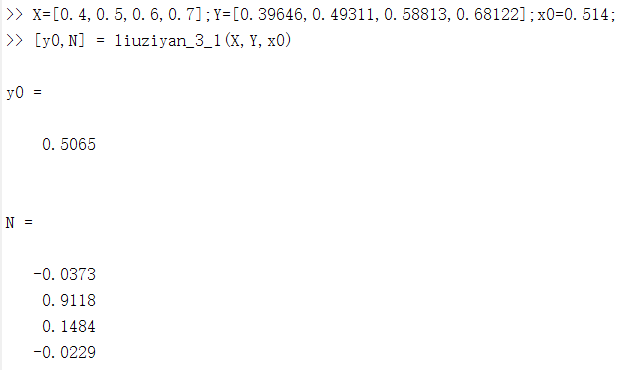
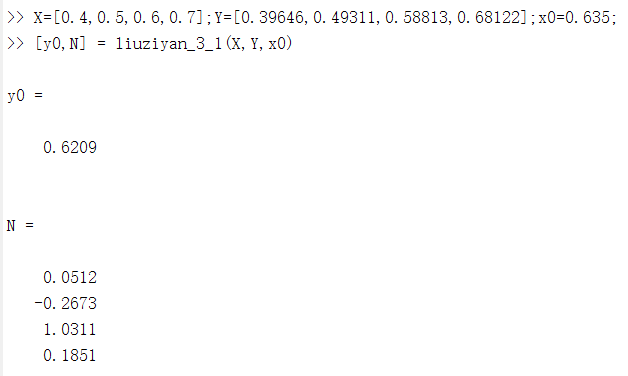
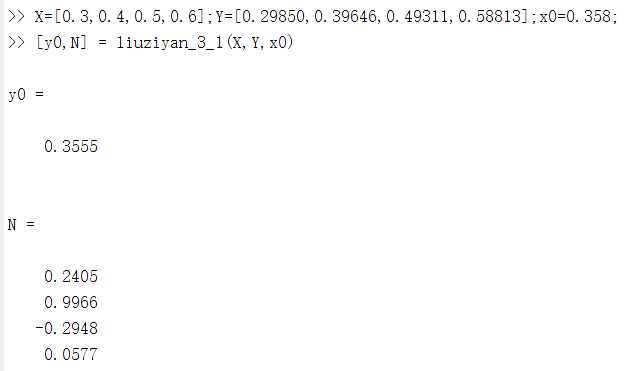
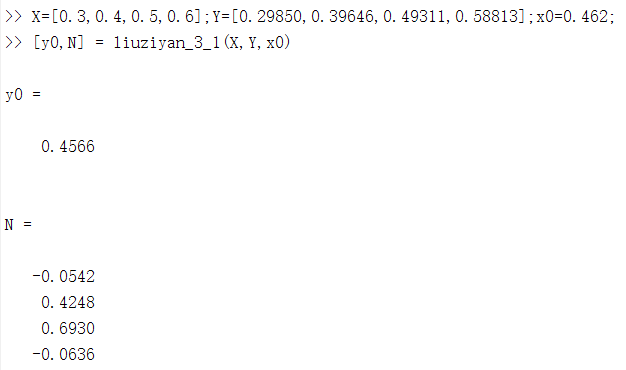
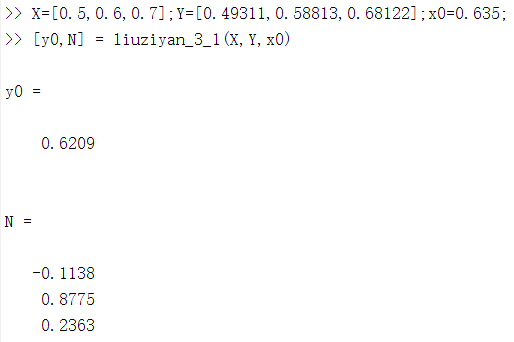
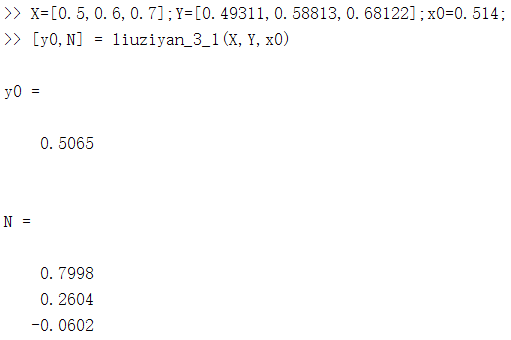
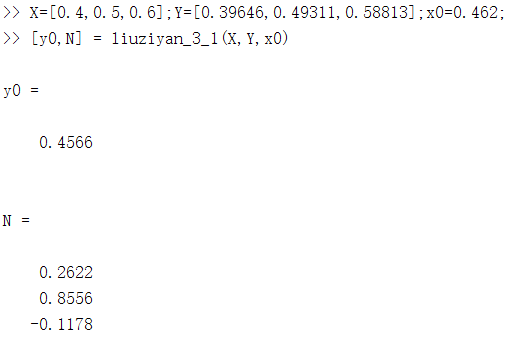
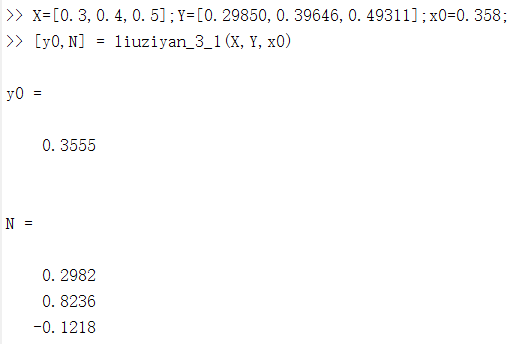
**3、实验结果**（下列截图中的函数名为我的名字liuziyan+题号）

3-1实验的运行结果截图如下：

1. 一次插值



1. 二次插值
2. 三次插值



**3-2：仿照附录C中“文件1.2 逐步插值”程序（Neville算法）编写相应的Aitken逐步插值算法的程序，根据实验题目3-1中所给数据，分别利用上述两种算法求正弦积分*f*(*x*)在*x*=0.358, 0.462, 0.514, 0.635处的值，比较两种算法的计算结果，并与3-1中的计算结果进行比较。**

**1、设计思想**

多点Lagrange插值可以划归为两点插值的重复，令每一步增添一个新的节点，直到遍历所有的节点为止，就得到了Aitken逐步插值算法。通过Aitken逐步插值算法可以生成三角形的逐步插值表。Neville算法与Aitken算法原理类似，也可以生成三角形的逐步插值表，它们都是将高阶插值逐步归结为线性插值最简单、最基本的重复。

Neville逐步插值算法逐列生成新的值，且每次运算得到的新值为前一列相邻元素及其上一元素间的运算。Aitken逐步插值算法也是逐列生成新的值，每次运算得到的新值为前一列元素与列首元素间的运算，因此只需将Neville插值算法的公式做适当修改即可，运算公式如下，其中对i=1,2,...计算，k=0,1,2,...,i-1：

 （2）

Aitken逐步插值的算法设计：依据公式（2）逐行生成插值表；检查计算误差，对于给定精度，当时计算终止，并输出作为插值结果；自然停机，当时输出作为插值结果。

逐步插值公式（2）在具体实现过程中可以将系数合并化简，在逻辑上也表现为二重循环，内循环（j循环）每一列中单个y的值，外循环（i循环）表示逐列生成，i每加1就进行下一列y的计算。

**2、对应程序**

（1）题目3-2：Neville插值的函数文件名：“liuziyan\_3\_2\_Neville.m”，代码如下：

function y0 = liuziyan\_3\_2\_Neville(X, Y, x0)

%Neville逐步插值算法，逐列生成新的值，且每个新值为前一列相邻两元素间的运算

%X,Y是已知插值点的坐标点，x0是插值点，y0是多项式在x0处的值

m = length(X);

P = zeros(m, 1);

P1 = zeros(m, 1);

P = Y;

for i=1:m

P1 = P;

k = 1;

for j = i+1:m

k = k+1;

P(j) = P1(j-1)+(P1(j)-P1(j-1))\*(x0-X(k-1))/(X(j)-X(k-1));

end

if abs(P(m)-P(m-1)) < 10^-6;

y0 = P(m);

return;

end

end

y0 = P(m);

（2）题目3-2：Aitken插值的函数文件名：“liuziyan\_3\_2\_Aitken.m”，代码如下：

function y0 = liuziyan\_3\_2\_Aitken(X, Y, x0)

%Aitken逐步插值算法，逐列生成新的值，每个新值为前一列元素与列首元素间的运算

%把Neville算法代码改写为Aitken算法

%关键在于将j-1,k-1改成每列第一个元素对应的值i,k

%X,Y是已知插值点的坐标点，x0是插值点，y0是多项式在x0处的值

m = length(X);

P = zeros(m, 1);

P1 = zeros(m, 1);

P = Y;

for i=1:m

P1 = P;

k = 1;

for j = i+1:m

P(j) = P1(i)+(P1(j)-P1(i))\*(x0-X(k))/(X(j)-X(k));

end

k = k+1;

if abs(P(m)-P(m-1)) < 10^-6;

y0 = P(m);

return;

end

end

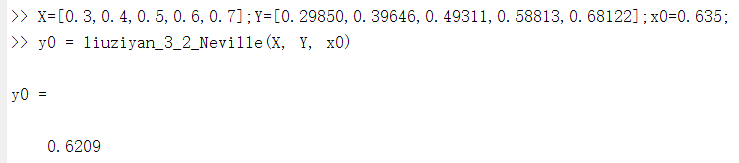
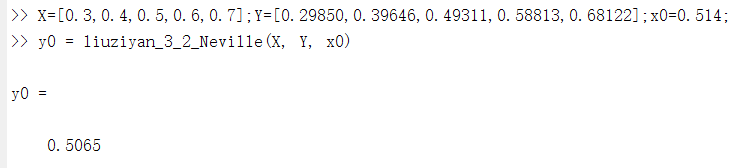
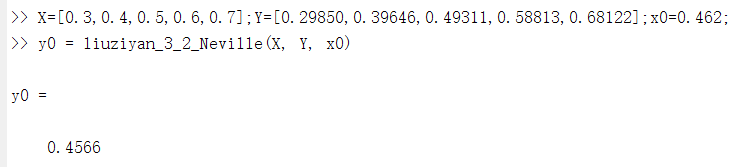
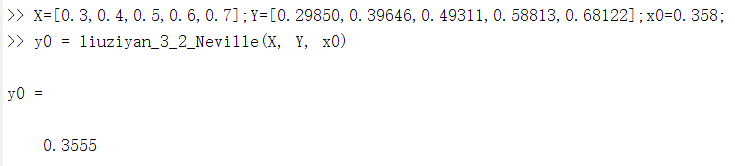
y0 = P(m);

**3、实验结果**

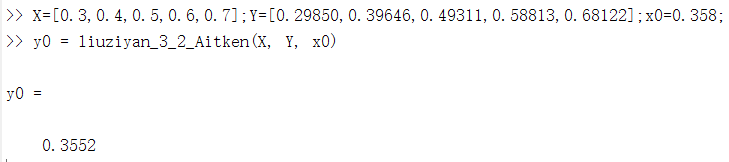
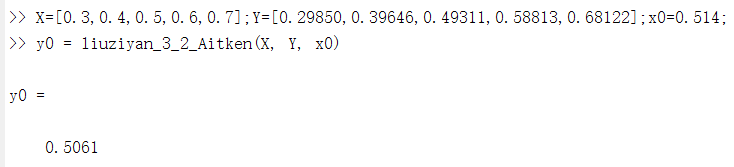
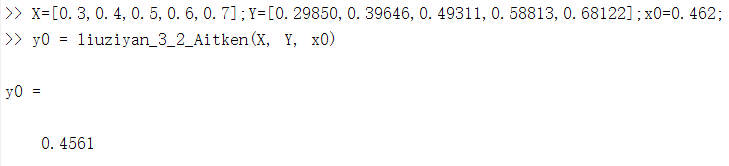
3-2实验的运行结果截图如下：

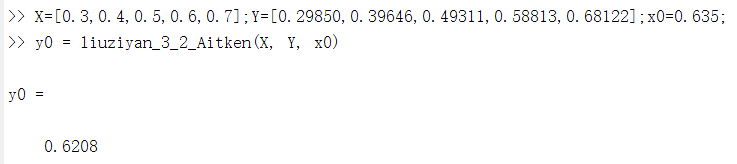
（下列截图中的函数名是我的名字liuziyan+题号+插值算法名称）

1. Neville逐步插值算法



1. Aitken逐步插值算法



综上所述，结合实验3-1，并比较3-2中的两种插值算法的计算结果，可以得到：

在精确性上，对Lagrange插值算法，插值次数越高，计算结果越精确；逐步差值算法中，相同节点数的情况下，Neville算法的计算结果比Aitken更加精确，因为Neville算法采用的是上一列相邻两个y值之间运算得到新的结果，能够更大效率地运用各节点的值，更快达到精度要求；Neville逐步插值算法的结果准确性与Lagrange三次插值得到的结果相似；

在优越性上，逐步插值算法比Lagrange算法更优越，因为增加节点时并不需要全部重新计算。

**3-3：对于函数，在利用Lagrange插值方法进行插值时，随着插值次数的增大，会出现如下图中所示的Runge现象：**

**要求：**

**（1）利用Lagrange插值方法验证Runge现象；**

**（2）将区间[-5,5]分为*n*等份（*n*=5,10,20），做的Lagrange分段线性插值函数*L*5(*x*)、*L*10(*x*)、*L*20(*x*)，考察上述三种插值在*x*=-4.8、4.8处的误差，并分析。**

**1、设计思想**

（1）验证Runge现象

对于代数插值，插值多项式的次数随着节点个数的增加而升高，然而高次插值的逼近效果往往是不理想的，次数越高，会带来舍入误差的增大，引起计算失真。

本实验我们采用Lagrange插值方法验证Runge现象，Lagrange插值算法依旧按照实验3-1设计的算法来进行实现，利用输入的阶数n计算步长h，x自取1000个点，调用Lagrange插值算法函数计算每个x对应的y值，在主函数中绘制出插值函数的图像。

（2）Lagrange分段线性插值函数

分段线性插值，就是将被插值函数逐段多项式（一次）化，对于函数f(x)，如果在每个子段上用线性插值，即用连接相邻节点的折线逼近所考察的曲线，就能保证一定的逼近效果。

具体过程是先将区间[-5,5]作一分划，将等分的份数作为参数n（n=5,10,20）输入，并在每个子区间上利用实验3-1设计的Lagrange插值算法构造插值多项式（阶数为2），最后再将每个子段上的插值多项式装配拼接在一起，作为整个区间[-5,5]上的插值函数。再求这三个插值函数在x=-4.8、4.8处的值，最后分析误差。

**2、对应程序**

（1）题目3-3：Lagrange插值验证Runge现象的函数文件名：“liuziyan\_3\_3\_Runge.m”，代码如下：

function y0 = liuziyan\_3\_3\_Runge(n)

%插值节点选择取值范围上的等距节点，只输入阶数n，每条线x取1000个点

for i=1:1001

x(i)=-5+(i-1)\*0.01;

y(i)=lagrange(x(i),n);

end

plot(x,y)

function y1=lagrange(x,n) %以下为插值多项式计算

h = 10/n;

y1 = 0;

N = 1;

for i=1:(n+1)

for j=1:(n+1)

if j==i

N = 1\*N;

else

N = N\*(x-(-5+(j-1)\*h))/((i-j)\*h);%迭乘

end

end

y1 = y1+N\*(1/(1+(-5+(i-1)\*h)^2));%迭加

N = 1;

end

1. 题目3-3：Lagrange分段线性插值函数文件名：“liuziyan\_3\_3\_Piece\_Lagrange.m”，代码如下：

function y0 = liuziyan\_3\_3\_Piece\_Lagrange(x0,n)

%Lagrange分段线性插值函数，n是分段的段数，每段均线性插值

for i=1:1001

x(i) = -5+(i-1)\*0.01;

y(i) = lagrange(x(i),n);

if x0 == x(i)

y0 = y(i);

end

end

plot(x,y)

hold on

grid on %添加网格线

xlabel('x轴');

ylabel('y轴');

function y1=lagrange(x,n) %以下为插值多项式计算

H = 10/n;%每段的长度

h = H/1;%每段中每步的长度

y1 = 0;

N = 1;

k = floor((x-(-5))/H)+1;%判断当前的x属于第几段

for i=1:2

for j=1:2

if j==i

N = 1\*N;

else

N = N\*(x-(-5+(k-1)\*H+(j-1)\*h))/((i-j)\*h);%迭乘

end

end

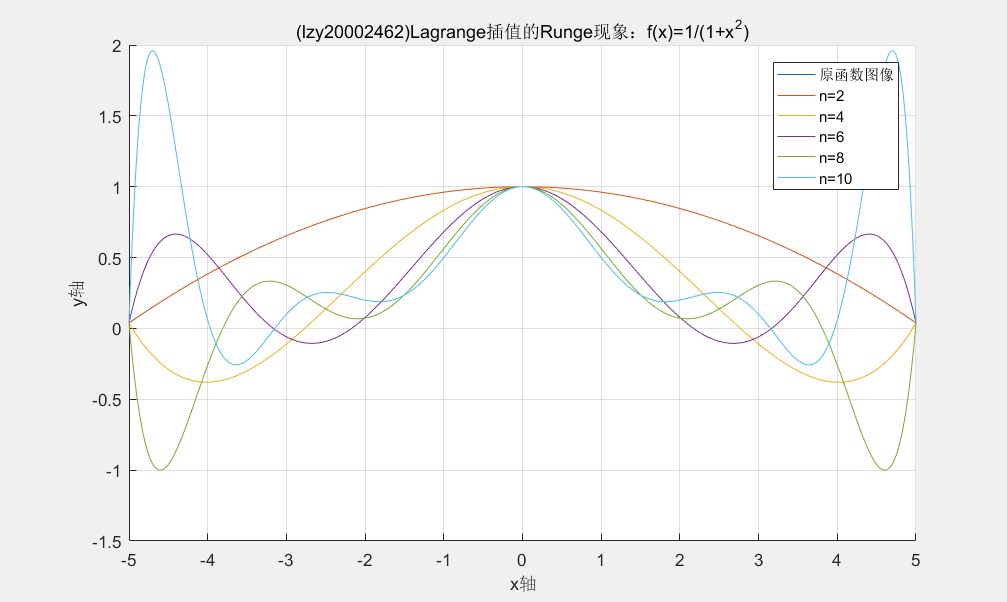
y1 = y1+N\*(1/(1+(-5+(k-1)\*H+(i-1)\*h)^2));%迭加

N = 1;

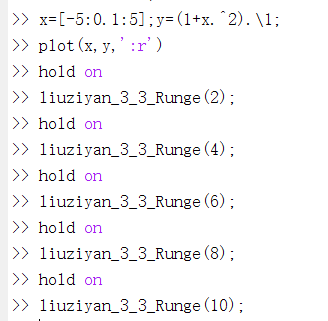
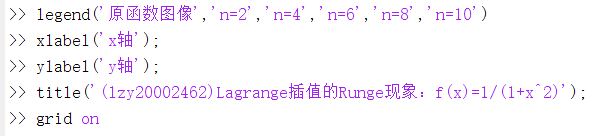
end

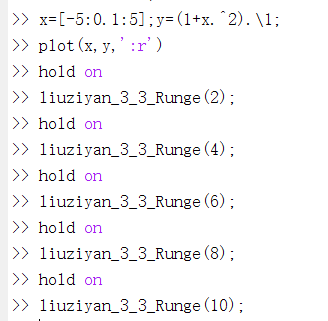
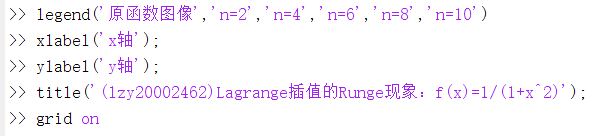
**3、实验结果**（下列截图中的出现的函数名是我的名字liuziyan+题号+插值算法名称）

3-2实验的运行结果截图如下：

1. Lagrange插值验证Runge现象，绘制出的Runge现象图像如下：

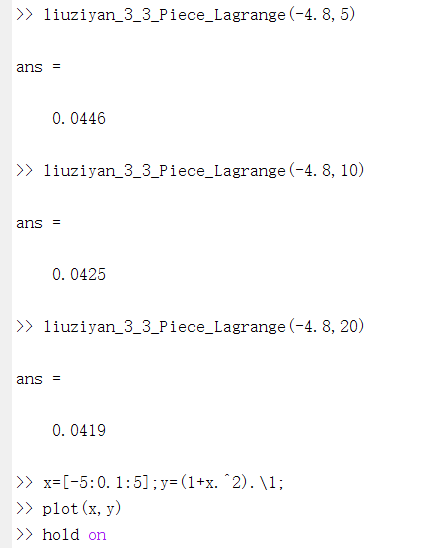
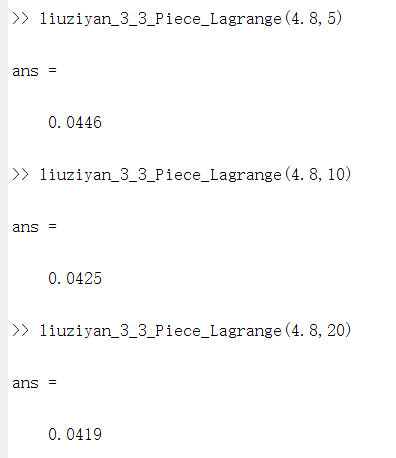
命令窗口输入的代码截图如下：

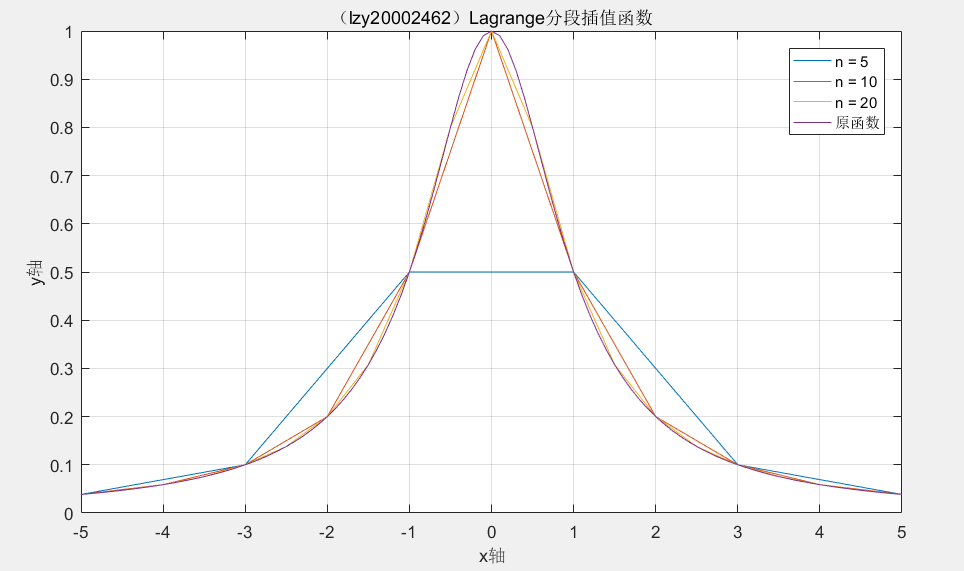
从图中可以看出，随着节点的加密采用高次插值，虽然插值函数会在更多的点上与所逼近的函数取相同的值，但整体上不一定能改善逼近的效果。事实上，当n增大时，所得的插值函数在两端会发生激烈的振荡，这就是Runge现象。Runge现象说明，在大范围内使用高次插值，逼近的效果往往是不理想的。



1. Lagrange分段线性插值函数（n=5,10,20）的图像如下：

*L*5(*x*)、*L*10(*x*)、*L*20(*x*)分别在-4.8,4.8处的值如下：



经过计算，，相对误差：

，，



由此可见，分段插值函数的分段数越多，插值函数的曲线越趋近平滑，误差越小，精度越高。Lagrange分段插值函数能够有效避免Runge现象造成的计算失真。

1. **实验体会**

对实验过程进行总结，分析比较各插值算法的效率和精度差异，指出每种算法的设计要点及应注意的事项，以及自己通过实验所获得的对插值方法的理解。

通过以上三个实验，我有以下几点总结和收获：

1. 对于Lagrange插值方法：

·它的计算公式是一个累乘累加的二重算式，结构紧凑，其中各个节点地位对等，形式上很对称，符合数学公式的美学；

·但是这个插值方法也有缺点，在实际运用时，如果临时需要增添一个节点，则其所有的系数都需要重新计算，造成了计算量上的浪费；

·在计算精度上，一次（线性）插值即用直线来代替原曲线，往往精度不足，而用更高次数（二次、三次）的简单曲线来代替原曲线，往往能够得到更高精度的结果。

1. 对两种逐步插值方法：

·Aitken和Neville逐步差值方法，在设计算法的过程中，我们选择逐列生成的方式，每增加一列，插值节点数减少一个。如果将插值节点数定义为插值问题的规模，那么逐步插值算法的加工过程就是个规模逐次减1的规模缩减过程；

·这两个逐步差值算法中，相同节点数的情况下，Neville算法的计算结果比Aitken更易精确，Neville算法采用的是上一列相邻两个y值之间运算得到新的结果，能更充分运用各节点的值，结果更准确；Neville逐步插值算法得到的结果值与Lagrange三次插值得到的结果相似；

·与Lagrange插值方法对比，Neville逐步插值比Lagrange插值更优越，对Lagrange插值来说，临时增加一个节点需要全部重新计算，而Neville逐步插值仅需在原计算基础上做些许修改即可，更便于实际运用。

1. 对于Runge现象和分段插值的运用：

·对于代数插值，插值多项式的次数随着节点个数的增加而升高，然而高次插值的逼近效果往往是不理想的，次数越高，同时也会带来舍入误差的增大，引起计算失真，这就造成了Runge现象。

·利用分段线性插值，就是将被插值函数逐段多项式（一次）化，对于函数f(x)，如果在每个子段上用线性插值，即用连接相邻节点的折线逼近所考察的曲线，就能保证一定的逼近效果。

·通过计算结果我们也能得到，Lagrange分段插值函数能够有效避免Runge现象造成的计算失真，并且分段插值函数的分段数n越大，插值函数的曲线就越趋近平滑，产生的误差越小。