实验一：插值算法的实现

序号：14　姓名：岳宇轩　专业：19 慧与　学号：19020011038

实验目的：

通过编程实践，熟练掌握拉格朗日插值算法和牛顿插值算法，对比二者的不同，并观察不同节点数量，不同拟合函数对于结果的影响。

实验步骤：

a. 实现拉格朗日插值；
b. 验证随着插值结点的增多插值曲线的变化情况。
c. 实现牛顿插值，显示插商结果；
d. 比较拉格朗日插值与牛顿插值的插值结果是否相同。

实验原理：

（1）拉格朗日插值：

构造基函数

$$l_i(x) = \frac{(x-x_0)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_0)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)} = \prod_{j=0\, j\neq i}^{n} \frac{x-x_j}{x_i-x_j}, i = 0, 1, \ldots, n$$

基函数满足如下条件：

$$l_i(x)=\begin{cases}0 & j\neq i\\ 1 & j=i\end{cases}$$

则 n 次拉格朗日插值多项式为：

$$p_n(x)=\sum_{i=0}^{n}y_i l_i(x)=\sum_{i=0}^{n}y_i(\prod_{j=0\,j\neq i}^{n}\frac{x-x_j}{x_i-x_j})$$

插值余项为：

$$R_n(x)=f(x)-p_n(x)=\frac{f^{(n+1)}(\epsilon)}{(n+1)!}\sum_{i=0}^{n}(x-x_i),\epsilon\in(a,b)$$

（2）牛顿插值：

差商求解：

$$f[x_0,x_1,\cdots,x_k]=\sum_{j=0}^{k}\frac{f(x_j)}{(x_j-x_0)\cdots(x_j-x_{j-1})(x_j-x_{j+1})\cdots(x_j-x_n)}$$

牛顿插值公式：

$$f(x)=f(x_0)+f[x_0,x_1](x-x_0)$$

$$+f[x_0,x_1,x_2](x-x_0)(x-x_1)+\cdots\ +f[x_0,x_1,\cdots,x_n](x-x_0)(x-x_1)\cdots(x-x_{n-1})$$

$$+f[x,x_0,x_1,\cdots,x_n](x-x_0)(x-x_1)\cdots(x-x_n)$$

实验过程：

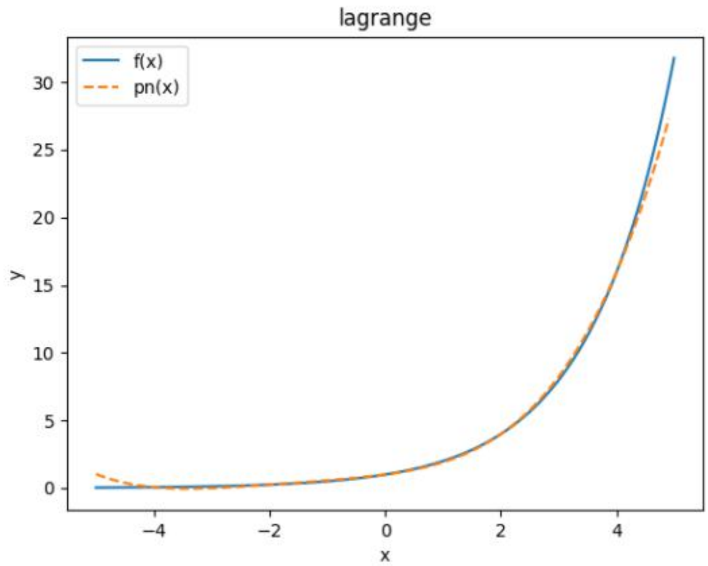
1. 实现拉格朗日插值，验证拉格朗日插值对不同函数的效果

①指数函数：$f(x) = 2^x$

在区间[-5, 5]上每隔 0.01 截取一个点绘制 f(x)曲线；

取 f(x)在 x=-4, x=-2, x=0, x=2, x=4 处的点为插值节点，计算在[-5, 5]区间内的 100 个点的插值结果，每个点之间间隔为 0.1，绘制 pn(x)曲线；

最终结果如下所示：



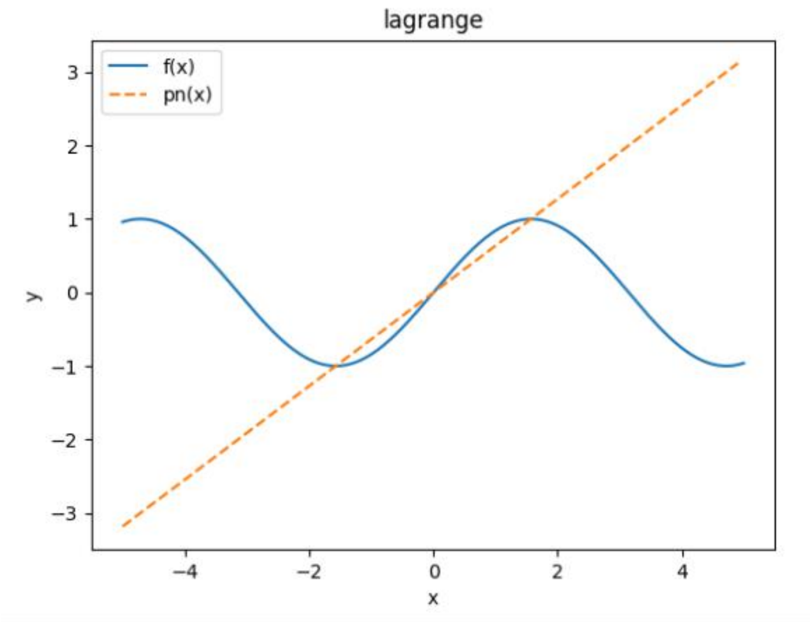
通过图像可以看出，在区间[-4, 4]内效果较好，在[-5, -4]和[4, 5]区间内出现明显偏差，这也印证了老师上课时讲到的对于相同的插值公式，内插比外推精度高。

②三角函数：

在区间[-5, 5]上每隔 0.01 截取一个点绘制 f(x)曲线；

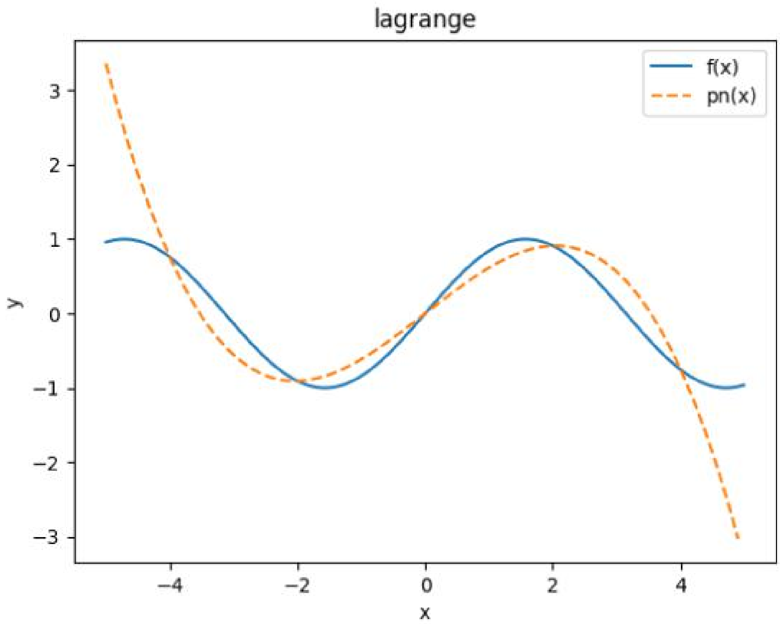
取 f(x)在 x=-π/2, x=0, x=π/2 处的点为插值节点，计算在[-5, 5]区间内的 100 个点的插值结果，每个点之间间隔为 0.1，绘制 pn(x)曲线；

最终结果如下所示：



可以看到，插值多项式并没有很好的拟合 sinx 曲线，原因是：选取的插值节点在低次多项式上。因此，在使用拉格朗日插值时要注意避免此问题。

选取 f(x)在 x=-4, x=-2, x=0, x=2, x=4 处的点为插值节点后结果如下：



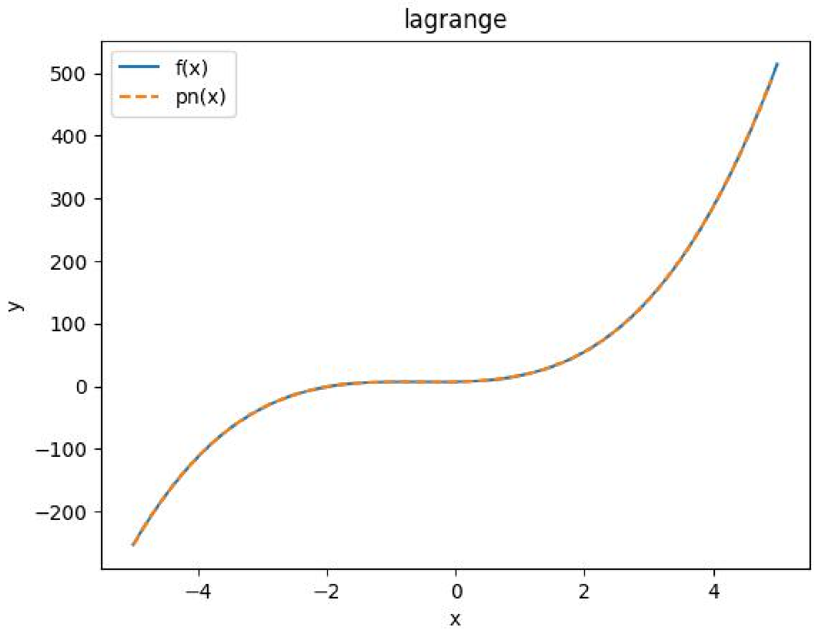
可以看到效果一般，我认为原因是插值节点过少的原因导致的。

③多项式函数：$f(x) = 3x^3 + 5x^2 + 2x + 7$

在区间[-5, 5]上每隔 0.01 截取一个点绘制 f(x)曲线；

取 f(x)在 x=-2, x=-1, x= 0, x=1, x=2 处的点为插值节点，计算在[-5, 5]区间内的 100 个点的插值结果，每个点之间间隔为 0.1，绘制 pn(x)曲线；

最终结果如下所示：



可以看到插值多项式的结果与原函数完全拟合，造成这种结果的原因是：

对于次数≤n 的多项式 f(x)，其 n 次差值多项式就是它自身

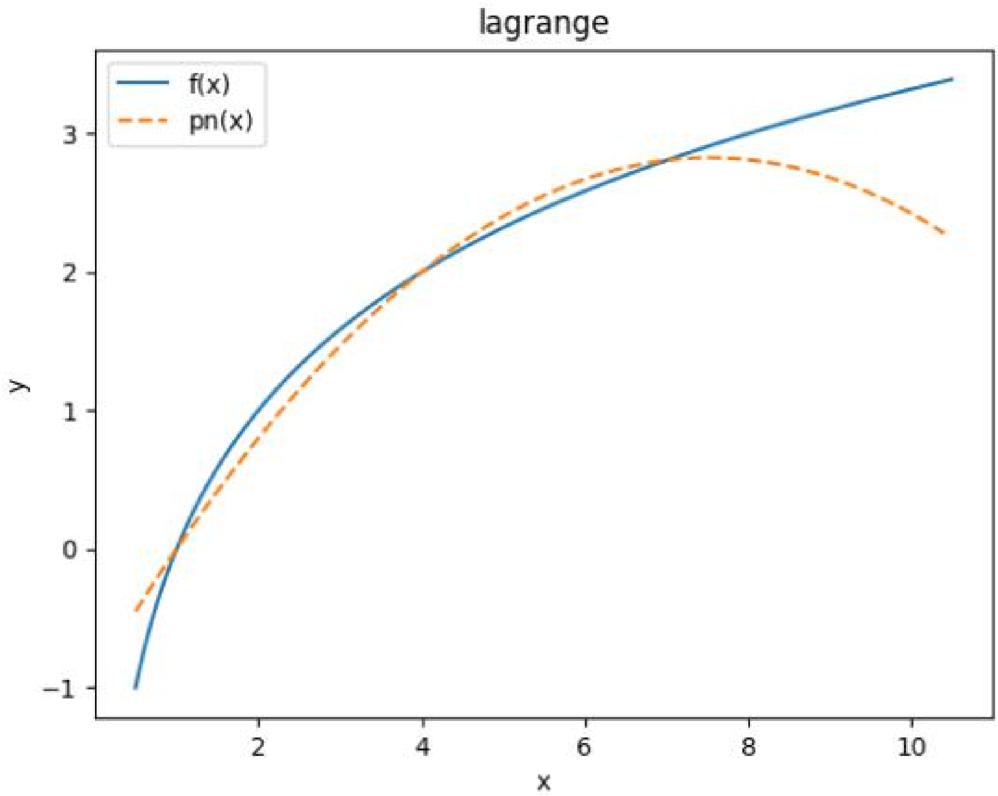
2. 随插值节点增多，插值曲线的变化

原函数$f(x) = log_2 x$，选取的结点个数与其对应的拟合结果如下

①选取三个节点

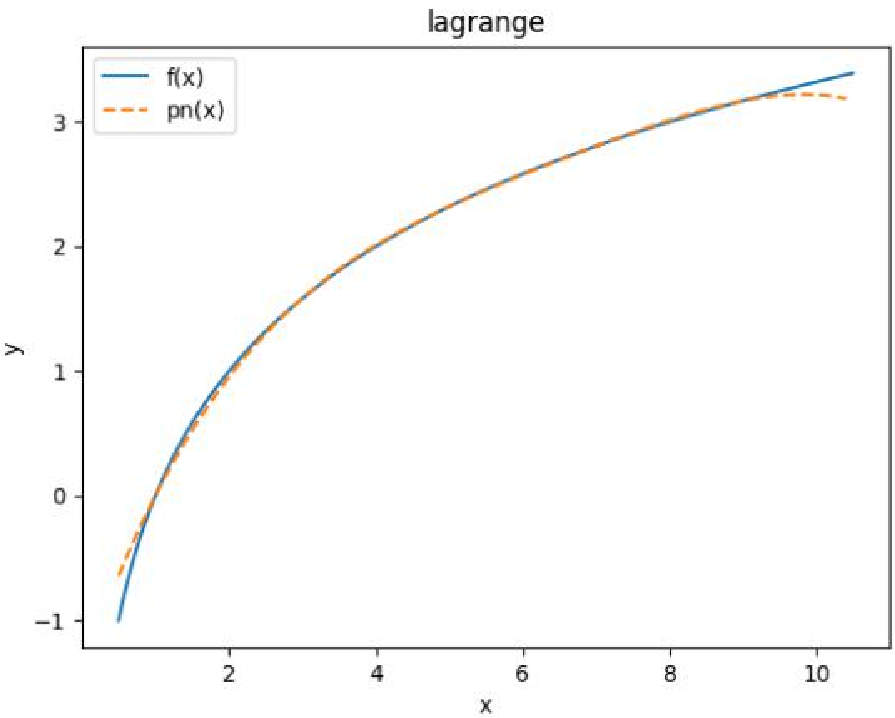
x	1	4	7
y	0	2	2.807354922

拟合图像如下：



②选取五个节点

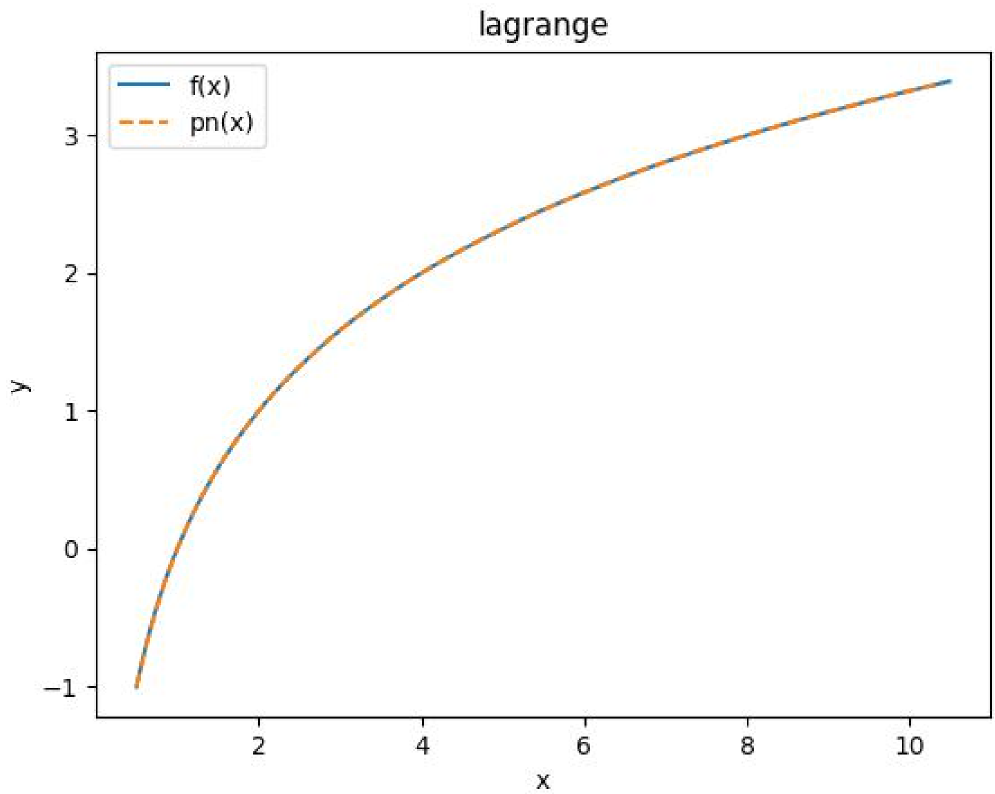
x	1	3	5	7	9
y	0	1.584963	2.321928	2.807355	3.169925



③选取 12 个节点

x	0.5	1	2	3	4	5	6	7	8	9	10	10.5
y	-1	0	1	1.584963	2	2.321928	2.584963	2.807355	3	3.169925	3.321928	3.392317

做出图像如下



根据上面三张图可以看出，随着插值节点的增加，插值多项式越接近原函数，效果越好

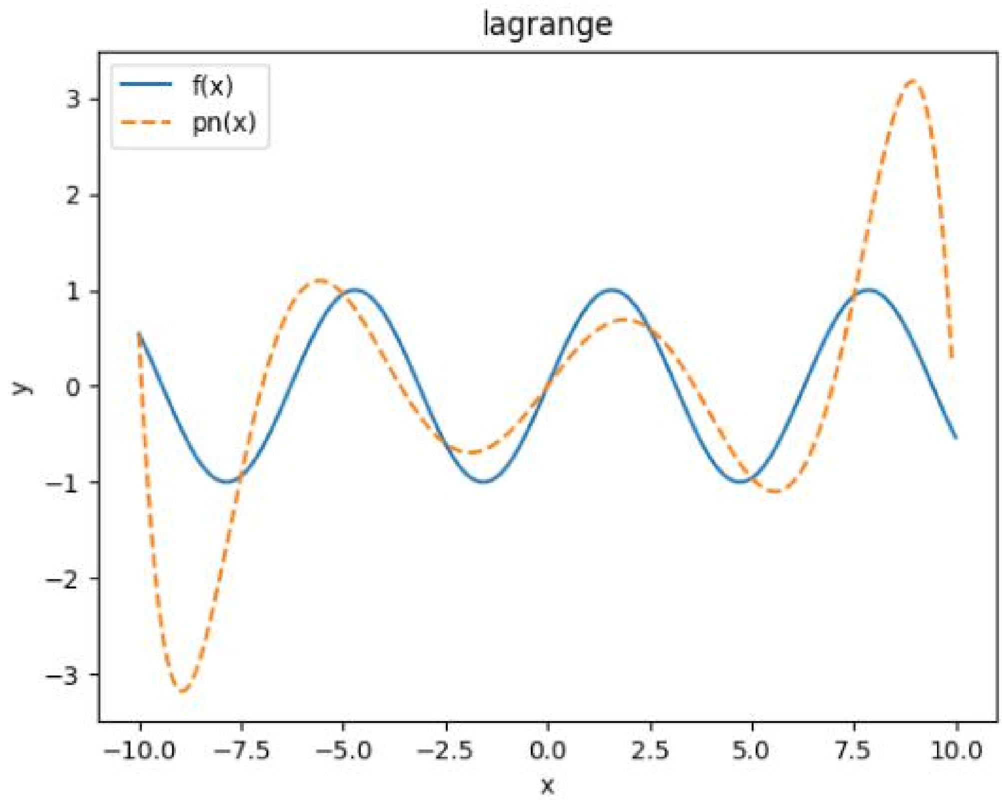
3. 探究当 n>=8 时，随着插值节点增加，某一点的相对误差变化情况。

以 f(x)=sinx，-10<=x<=10 函数为例进行探究

①n=8 时 x 取值如下

x	-10	-7.5	-5	-2.5	0	2.5	5	7.5	10

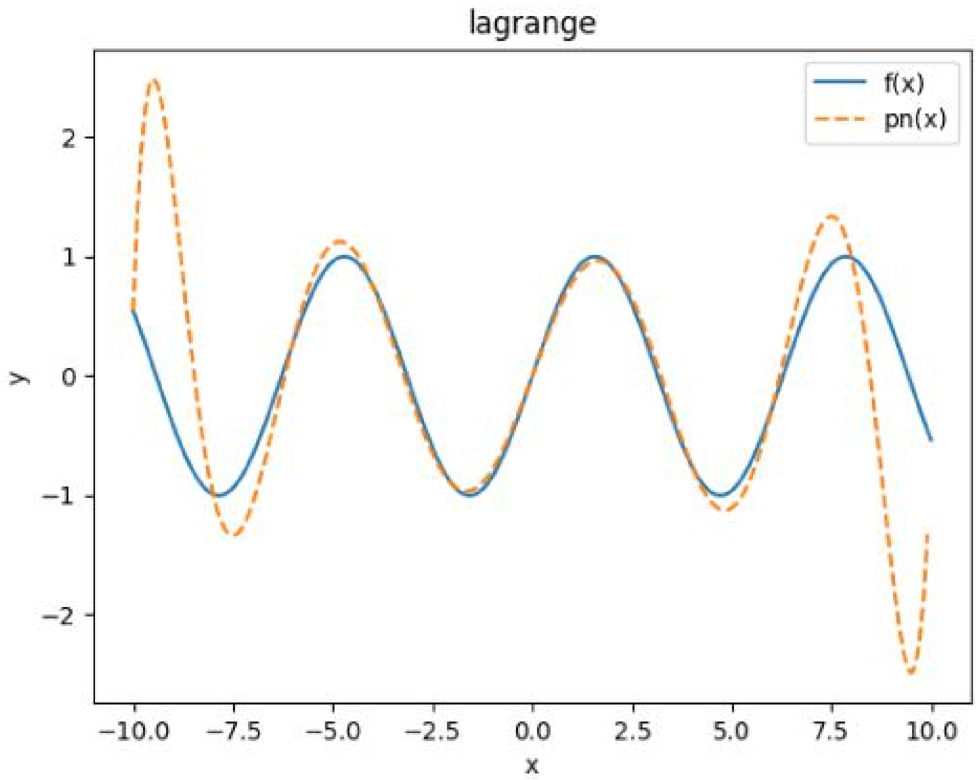
插值积多项式结果如图所示：



②n=10

x	-10	-8	-6	-4	-2	0	2	4	6	8	10

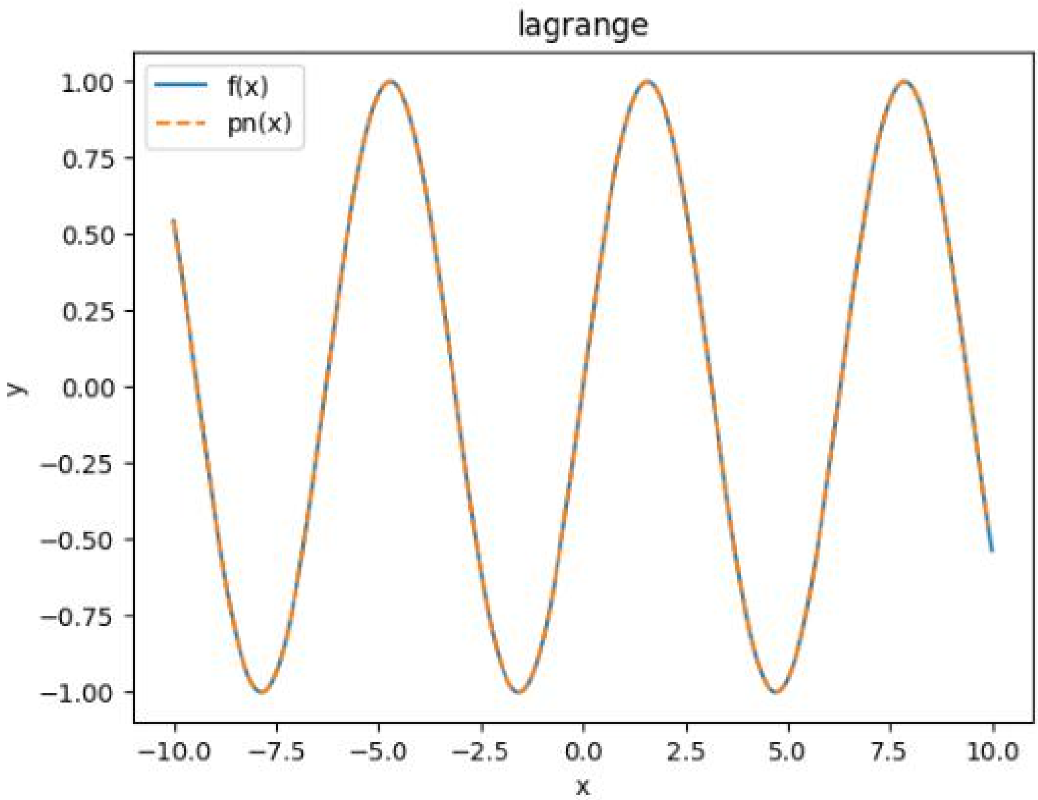
插值多项式结果如图所示：



②n=20

x 取-10，-9，-8, ……， 9， 10

插值多项式结果如图所示：



对于 x=3.5，求其在 n=8, n=10 和 n=20 时的相对误差分别为

1.200602045445019

0.1791292286421309

3.4776680258036976e-07

显而易见，误差是越来越小的。

由此得出结论，当 n 越大时，对于某一点其相对误差越小

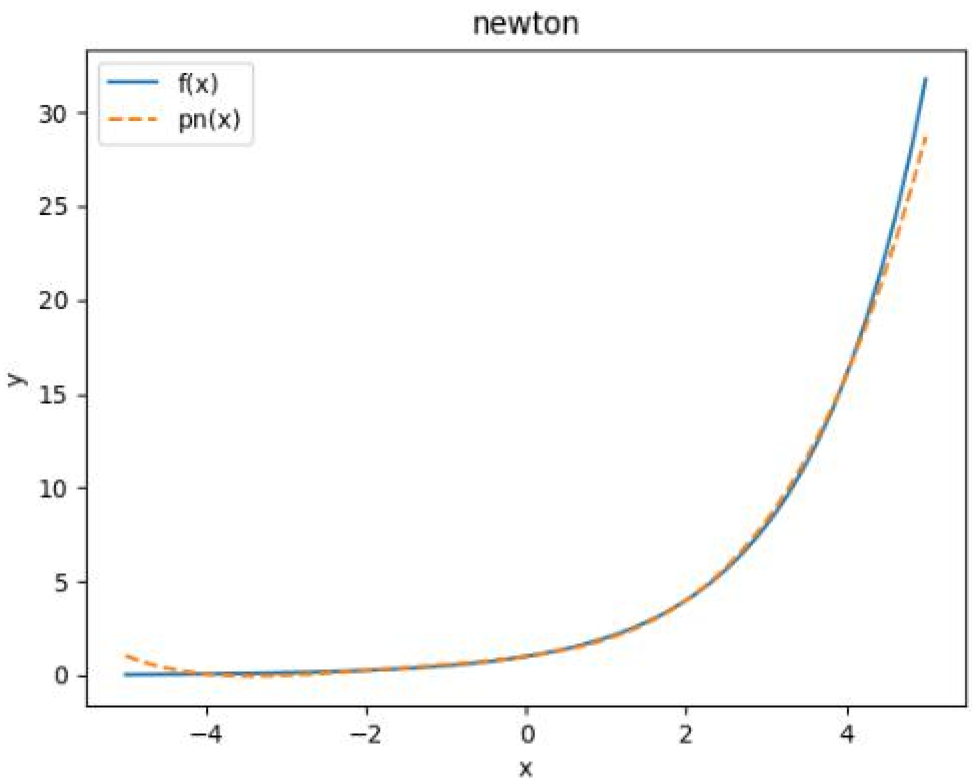
3.实现牛顿插值，显示插商结果；

①指数函数：$f(x) = 2^x$

在区间[-5, 5]上每隔 0.01 截取一个点绘制 f(x)曲线；

取 f(x)在 x=-4, x=-2, x=0, x=2, x=4 处的点为插值节点，计算在[-5, 5]区间内的 100 个点的插值结果，每个点之间间隔为 0.1，绘制 pn(x)曲线；

最终结果如下所示：



差商结果

f[x0,x1]=0.09375

f[x0,x1,x2]=0.0703125

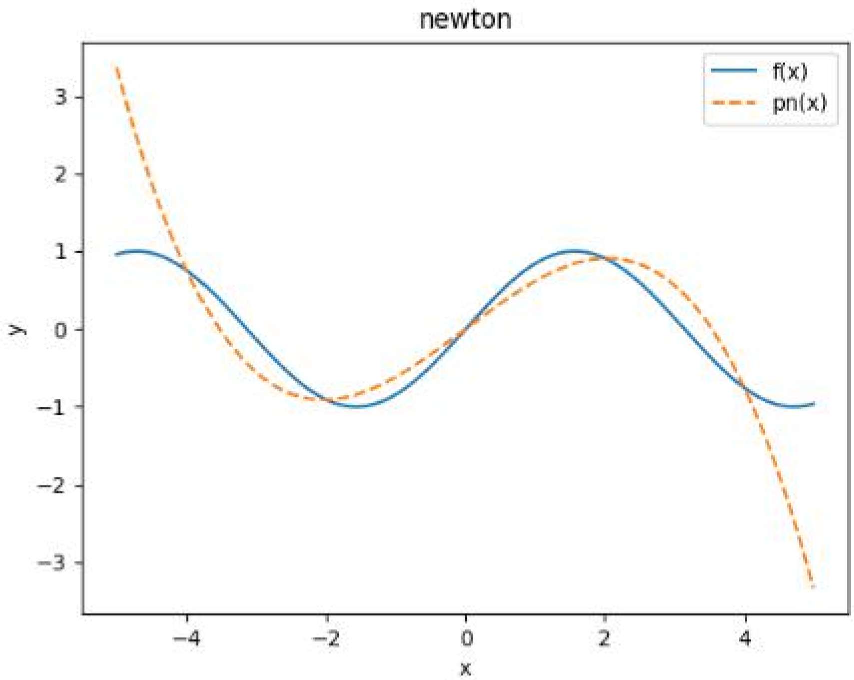
f[x0,x1,x2,x3]=0.035156249999999

f[x0,x1,x2,x3,x4]=0.01318359375

②三角函数：

在区间[-5, 5]上每隔 0.01 截取一个点绘制 f(x)曲线；

选取 f(x)在 x=-4, x=-2, x=0, x=2, x=4 处的点为插值节点后结果如下：



插商结果：

f[x0,x1]=-0.833049961066805

f[x0,x1,x2]=0.3219246686199114

f[x0,x1,x2,x3]=-0.05365411143665191

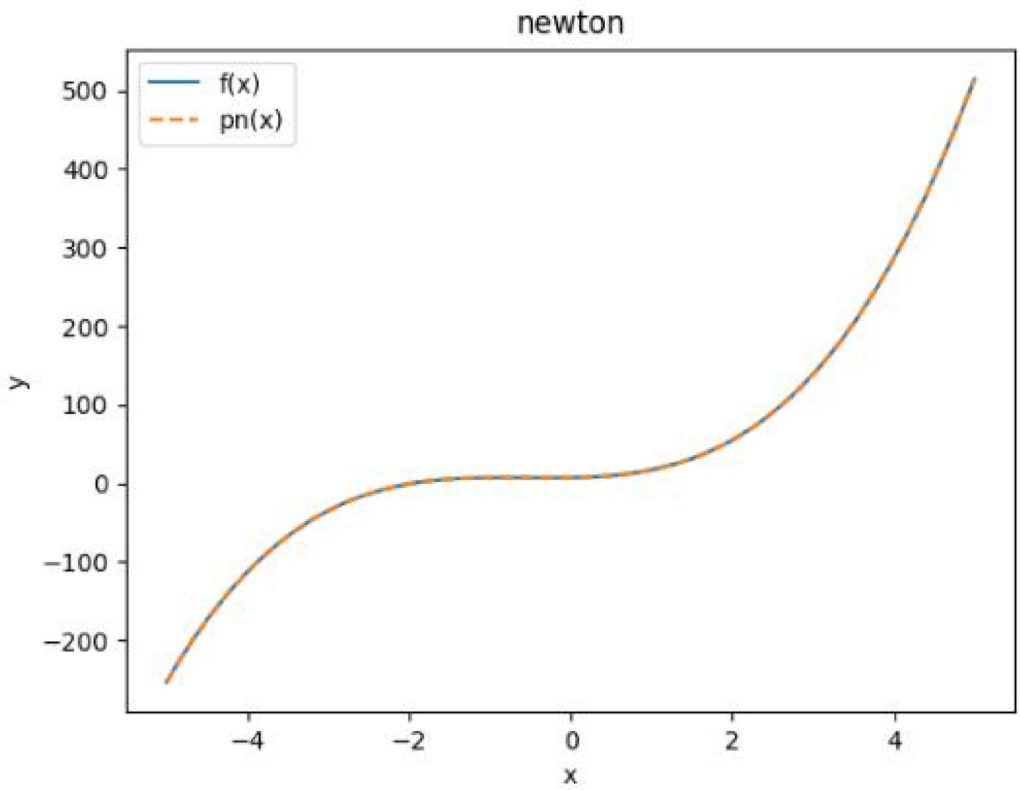
f[x0,x1,x2,x3,x4]=0

③多项式函数：$f(x)=3x^3+5x^2+2x+7$

在区间[-5, 5]上每隔 0.01 截取一个点绘制 f(x)曲线；

取 f(x)在 x=-2, x=-1, x= 0, x=1, x=2 处的点为插值节点，计算在[-5, 5]区间内的 100 个点的插值结果，每个点之间间隔为 0.1，绘制 pn(x)曲线；

最终结果如下所示：



插商结果：

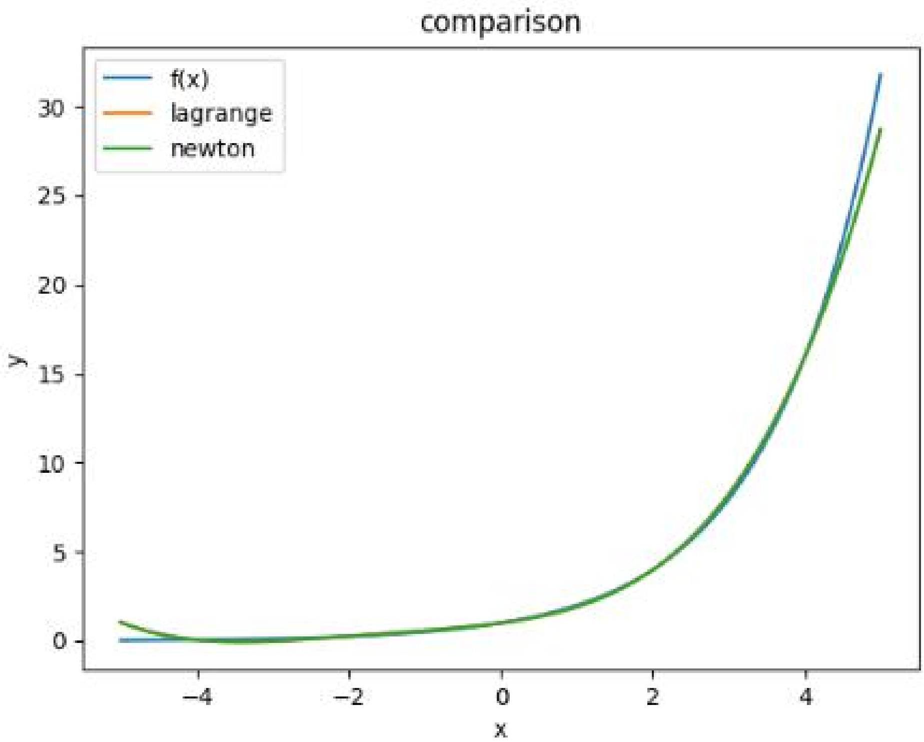
f[x0,x1]=56

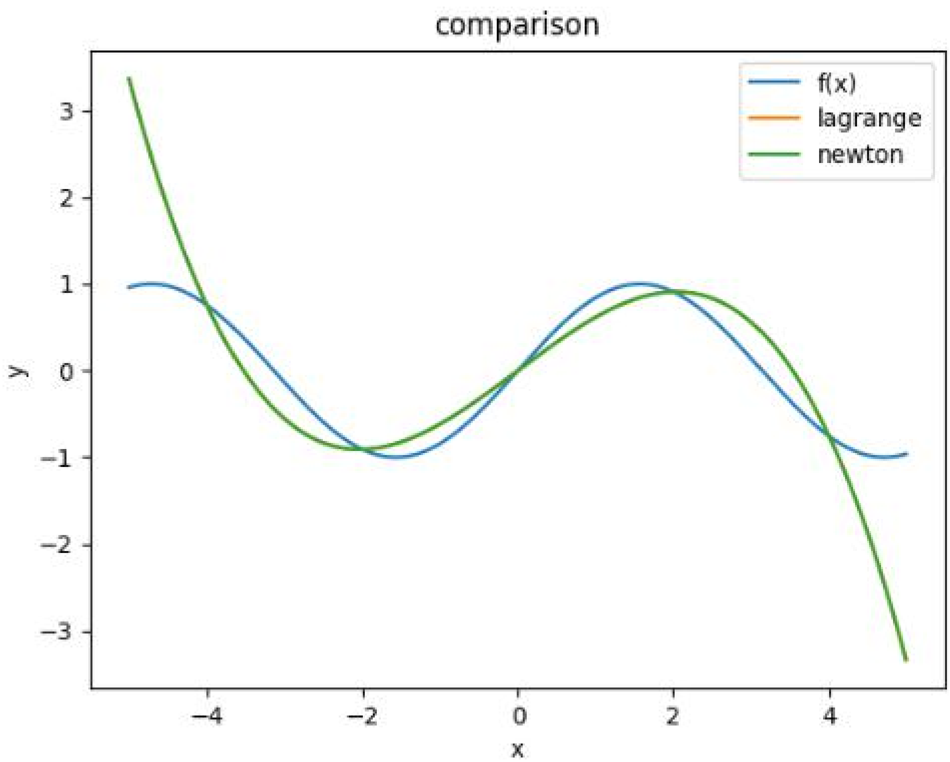
f[x0,x1,x2]=-13

f[x0,x1,x2,x3]=3

f[x0,x1,x2,x3,x4]=0

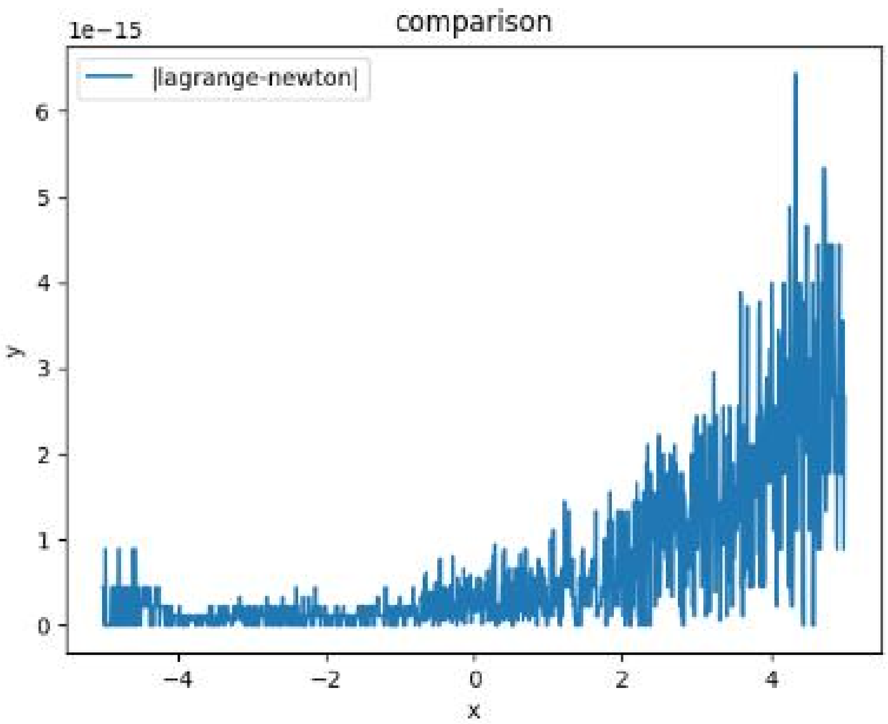
4.比较拉格朗日插值与牛顿插值的插值结果是否相同。





比较对 sinx 进行拉格朗日插值和牛顿插值的结果，发现两者的差别很小，肉眼很难区别，曲线几乎重合。下面分别展示拉格朗日

插值和牛顿插值的|f(x)-pn(x)|。



可以看到，拉格朗日插值的结果与牛顿插值的结果相差很小，在 1e-15 量级，但是牛顿插值具有继承性，适用于新增节点的情况。在不新增节点的情况下，两者插值结果相差不大。

实验心得

在这次的实验过程中，利用 python 编写了拉格朗日插值算法和牛顿插值算法的有关代码，增加了编程能力，也进一步理解了这两种算法。

体会到了两种算法在不同条件下的不同结果。实验结果表明，在插值拟合时，应该尽量选取尽可能多的点，使拟合的

结果更精确;在选择算法时一定要考虑现实情况，如果结点个数不确定时，可以选择牛顿插值算法，这样可以有效利用计算结果，简化计算过程，节省时间。

我也亲身体会了，对于相同的插值公式，内插比外推精度高；选取的插值节点在低次多项式上会导致较大误差；对于次数≤n 的多项式 f(x)，其 n 次差值多项式就是它自身；随着插值节点的增加，插值多项式越接近原函数，效果越好；拉格朗日插值与牛顿插值结果相差不大。

Appendix

```
import math

import numpy as np

import matplotlib.pyplot as plt


#  插值节点数

n = 4

x = []

y = []



#  拉格朗日插值

def lagrange(q):
```

```
    temp = 1
    ans = 0
    for i in range(n + 1):
        for j in range(n + 1):
            if j != i:
                temp *= (q - x[j]) / (x[i] - x[j])
        ans += temp * y[i]
        temp = 1
    return ans


# 差商计算
def difference(k):
    temp = 1
    ans = 0
    for i in range(k + 1):
        for j in range(k + 1):
            if j != i:
                temp *= 1.0 / (x[i] - x[j])
        ans += y[i] * temp
        temp = 1
    return ans
```

```python
# 牛顿插值
def newton(q):
    temp = 1
    ans = y[0]
    for i in range(n):
        temp *= (q - x[i])
        ans += difference(i + 1) * temp
    return ans


# f(x)
def fx(x):
    return math.sin(x)


# 构建插值节点集合
for i in range(n + 1):
    x.append(float(input()))
    y.append(fx(x[i]))
```

```
# 构建绘图 f(x)集合
x0 = np.arange(-5, 5, 0.01)
y0 = []
for t in x0:
    y0.append(fx(t))

# 构建 lagrange 插值序列集合
x1 = np.arange(-5, 5, 0.01)
y1 = []
for t in x1:
    y1.append(lagrange(t))

# 构建 newton 插值序列集合
x2 = np.arange(-5, 5, 0.01)
y2 = []
for t in x2:
    y2.append(newton(t))

# 构建 lagrange 误差集合
x3 = np.arange(-5, 5, 0.01)
y3 = []
for i in range(len(x1)):
```

```
        y3.append(abs(y0[i] - y1[i]))

# 构建 newton 误差集合
x4 = np.arange(-5, 5, 0.01)
y4 = []
for i in range(len(x1)):
        y4.append(abs(y0[i] - y2[i]))

# 相差结果集合
x5 = np.arange(-5, 5, 0.05)
y5 = []
for i in range(len(x5)):
        y5.append(abs(y3[i] - y4[i]))

# 绘图
# plt.plot(x0,y0,label="f(x)")
# plt.plot(x1,y1,label="lagrange")
# plt.plot(x2,y2,label="newton")
# plt.plot(x3,y3,label="lagrange")
# plt.plot(x4,y4,label="newton")
plt.plot(x5,y5,label="|lagrange-newton|")
plt.xlabel("x")
```

```
plt.ylabel("y")
plt.title('comparison')
plt.legend()
plt.show()
```