***2021***



**数值分析 ·实验报告·**

j0242087[1]

|  |  |
| --- | --- |
| 学 院： | 电气与电子工程学院 |
| 班 级： | 硕2007班 |
| 学 号： | M202071661 |
| 姓 名： | 江易星 |
| 完成日期： | 2021-05-16 |

目录

[实验4.1复化求积公式计算定积分 2](#_Toc72224504)

[实验目的 2](#_Toc72224505)

[实验题目 2](#_Toc72224506)

[实验要求 2](#_Toc72224507)

[第一问 2](#_Toc72224508)

[第二问 9](#_Toc72224509)

[第三问 12](#_Toc72224510)

[实验4.2比较复化Simpson和变步长Simpson 13](#_Toc72224511)

[实验目的: 13](#_Toc72224512)

[实验题目： 13](#_Toc72224513)

[实验要求 13](#_Toc72224514)

[第一问 13](#_Toc72224515)

[第二问 17](#_Toc72224516)

[实验2.1 多项式插值的震荡现象 18](#_Toc72224517)

[问题提出 18](#_Toc72224518)

[实验内容 18](#_Toc72224519)

[实验要求 18](#_Toc72224520)

[第一问 18](#_Toc72224521)

[第二问 27](#_Toc72224522)

# 实验4.1复化求积公式计算定积分

### 实验目的

复化求积公式计算定积分

### 实验题目

数值计算下列各式右端定积分的近似值：

### 实验要求

1. 若用复化梯形公式，复化Simpson公式和复化Guass-Legendre I型公式做计算，要求绝对误差限为,分别利用它们对每种算法做出步长的事前估计。
2. 分别用复化梯形公式，复化Simpson公式和复化Guass-Legendre I型公式公式计算。
3. 将计算结果与精确解做比较，并比较各种算法的计算量。

### 第一问

对于区间[a,b]上的复化梯形求积公式的余项为: 其中

复化simpson求积公式的余项为 其中

复化Guass-Legendre I型公式的余项为其中

所以要求绝对误差限

积分区间

#### 复化梯形公式：

的图像如图1所示，可以明显看到在区间[2,3]上单调递减，求，图像如图2所示，也可以看到在[2,3]上的值为负数，由此可以进一步确定在[2,3]是单调递减的。

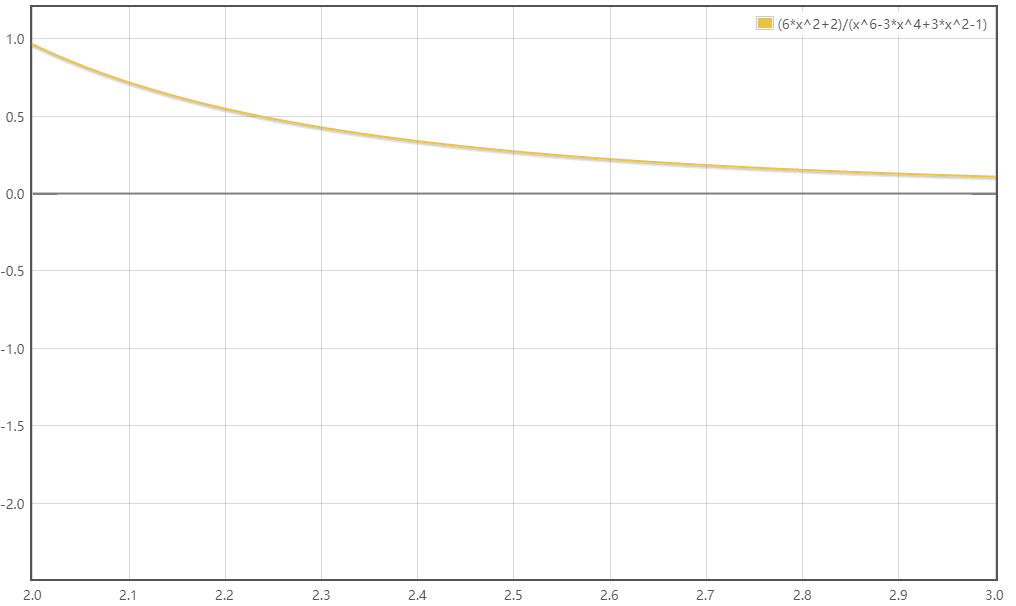


图1:的图像

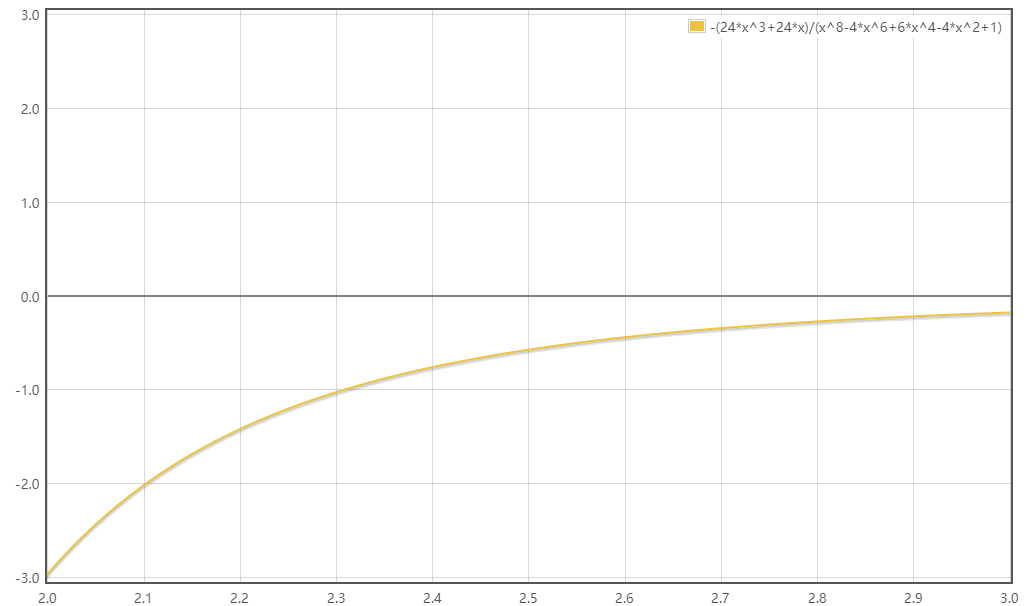


图2:的图像

由此可以得到在[2,3]上的最大值为

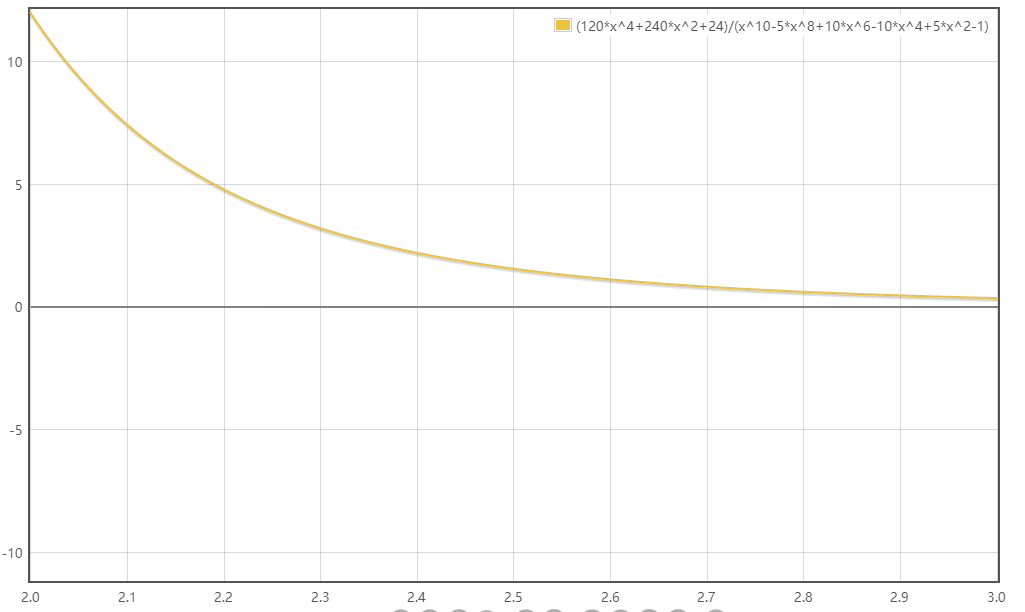
所以代入复化梯形求积公式的余项公式得到：

解得:

n取整数为：

#### 复化Simpson以及复化Guass-Legendre I型求积公式

在[2,3]上单调递减



所以在[2,3]上的最大值为

带入复化Simpson余项公式得到:

解得：

取整：

带入复化Gauss-Legendre I型求积公式的余项得到:

解得：

取整：

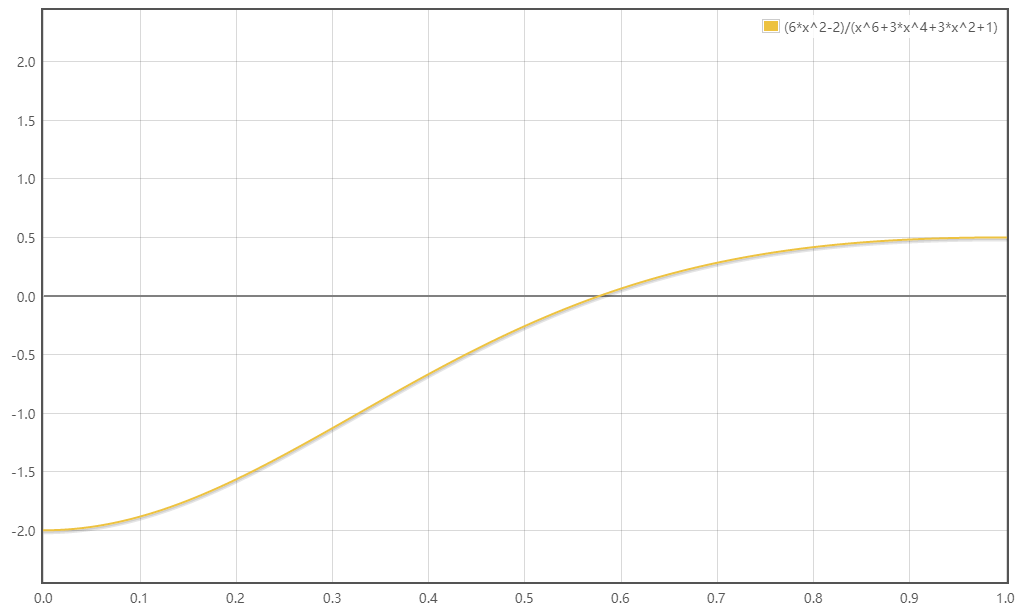
综上。要使得绝对误差限为，复化梯形公式的步长复化Simpson公式的步长要满足 ，复化Guass-Legendre I型公式的步长要满足

2.

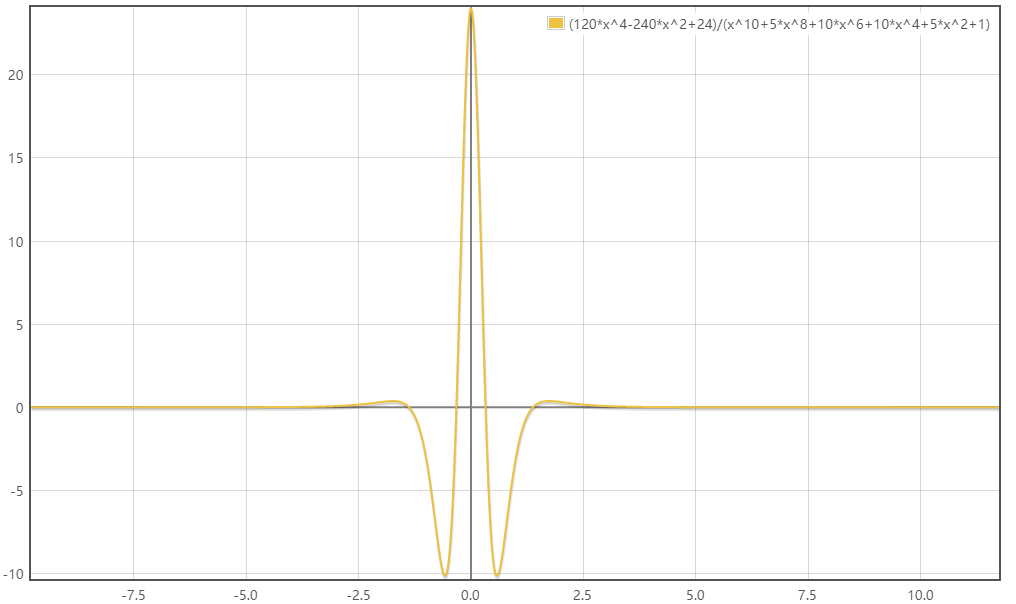
积分区间[0,1]

在区间[0,1]上的图像如图所示：。

所以在[0,1]上的绝对值的最大值为



其函数图像在[0,1]上并不是单调的，如下图所示



在[0,1]上的绝对值的最大值为

代入复化梯形求积公式余项：

解得n并取整数得到

代入复化Simpson公式的余项:

解得n并取整数得到:

代入复化Guass-Legendre I型求积公式的余项得到；

解得n并取整数得到：

综上:对于积分区间[0,1]，想要把绝对误差控制在，复化梯形求积公式的步长要满足复化Simpson公式的步长要满足,复化Guass-Legendre I型求积公式的步长要满足

3.

积分区间[0,1]

在区间[0,1]上单调递增且为正，所以在[0,1]上的绝对值最大值为

在区间[0,1]上单调递增且为正，所以在[0,1]上的绝对值的最大值为

带入复化梯形求积公式的余项得到：

解得n并取整得到：

带入复化Simpson求积公式的余项得到：

解得n并取整得到：

带入复化Guass-Legendre I型求积公式的余项得到：

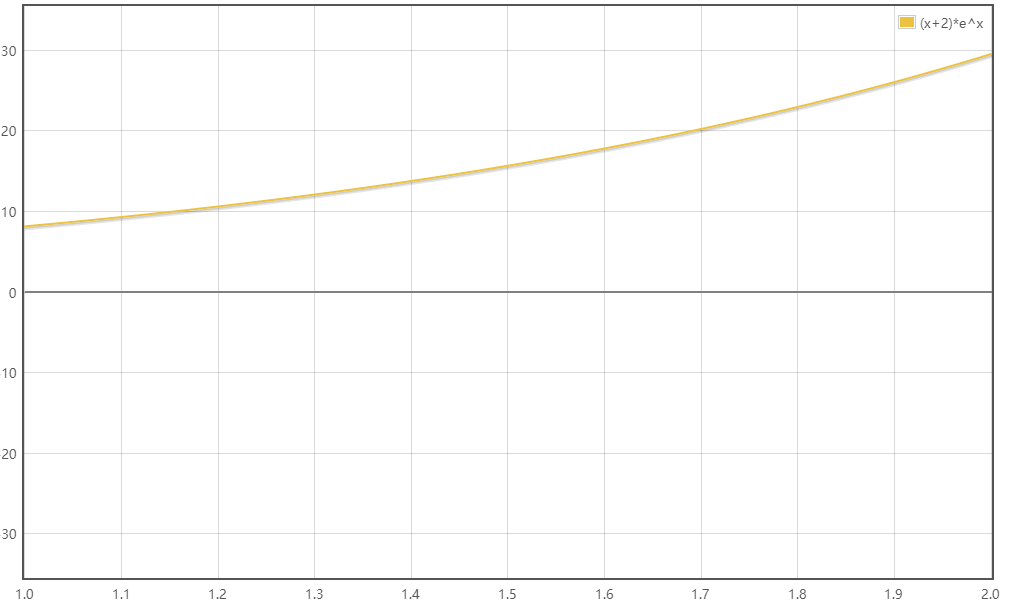
解得n并取整得到：

综上所述，对于，想要把绝对误差控制在以内，复化梯形求积公式的步长要满足复化Simpson公式的步长要满足,复化Guass-Legendre I型求积公式的步长要满足.

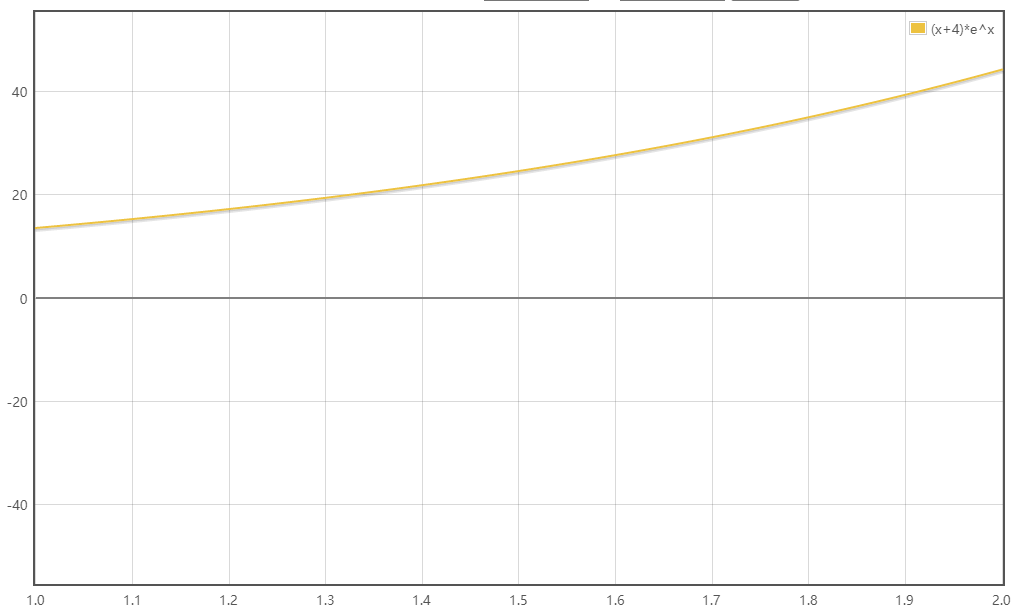
4.

积分区间[1,2]

在[1,2]上单调递增且为正，所以在[1,2]上的绝对值的最大值为



在[1,2]上单调递增且为正，所以在[1,2]上的绝对值的最大值为



带入复化梯形求积公式的余项得到：

解得n并取整得到：

带入复化Simpson求积公式的余项得到：

解得n并取整得到：

带入复化Guass-Legendre I型求积公式的余项得到：

解得n并取整得到：

综上所述，对于，想要把绝对误差控制在以内，复化梯形求积公式的步长要满足复化Simpson公式的步长要满足,复化Guass-Legendre I型求积公式的步长要满足..

### 第二问

利用复化梯形求积公式、复化Simpson求积公式、复化Guass-Legendre I 型公式计算数值积分的python代码如下所示：

import math

def Trapzoid(f,a,b):

return (b-a)\*(f(a)+f(b))/2

def Simpson(f,a,b):

return (b-a)\*(f(a)+4\*f((a+b)/2)+f(b))/6

def GuassLegendre(f,a,b):

t1 = (a+b)/2-(b-a)\*(1/math.sqrt(3))/2

t2 = (a+b)/2+(b-a)\*(1/math.sqrt(3))/2

return (b-a)\*(f(t1)+f(t2))/2

# 复化梯形求积公式

def ComplexTrapzoid(f,a,b,n):

h=(b-a)/n

sum = 0

for k in range(n):

sum += Trapzoid(f,a+k\*h,a+(k+1)\*h)

return sum

# 复化Simpson求积公式

def ComplexSimpson(f,a,b,n):

h=(b-a)/n

sum =0

for k in range(n):

sum += Simpson(f,a+k\*h,a+(k+1)\*h)

return sum

def CompexGuassLegendreI(f,a,b,n):

h = (b-a)/n

sum = 0

for k in range(n):

sum += GuassLegendre(f,a+k\*h,a+(k+1)\*h)

return sum

def f1(x):

return 1/(x\*x-1)

def f2(x):

return 1/(x\*x+1)

def f3(x):

return math.pow(3,x)

def f4(x):

return x\*math.exp(x)

def test():

print("(1):")

CompTrap = -2\*ComplexTrapzoid(f1,2,3,1792)

CompSimp = -2\*ComplexSimpson(f1,2,3,21)

CompGuaLeg = -2\*CompexGuassLegendreI(f1,2,3,19)

Exact = math.log(2,math.exp(1))-math.log(3,math.exp(1))

print("复化梯形求积计算值",CompTrap,"误差",Exact-CompTrap)

print("复化Simpson求积计算值",CompSimp,"误差",Exact-CompSimp)

print("复化Guass-Legendre求积计算值",CompGuaLeg,"误差",Exact-CompGuaLeg)

print("准确值",Exact)

print("(2):")

CompTrap = 4\*ComplexTrapzoid(f2,0,1,3652)

CompSimp = 4\*ComplexSimpson(f2,0,1,29)

CompGuaLeg = 4\*CompexGuassLegendreI(f2,0,1,26)

Exact = math.pi

print("复化梯形求积计算值",CompTrap,"误差",Exact-CompTrap)

print("复化Simpson求积计算值",CompSimp,"误差",Exact-CompSimp)

print("复化Guass-Legendre求积计算值",CompGuaLeg,"误差",Exact-CompGuaLeg)

print("准确值",Exact)

print("(3):")

CompTrap = ComplexTrapzoid(f3,0,1,2457)

CompSimp = ComplexSimpson(f3,0,1,14)

CompGuaLeg = CompexGuassLegendreI(f3,0,1,12)

Exact = 2/math.log(3,math.exp(1))

print("复化梯形求积计算值",CompTrap,"误差",Exact-CompTrap)

print("复化Simpson求积计算值",CompSimp,"误差",Exact-CompSimp)

print("复化Guass-Legendre求积计算值",CompGuaLeg,"误差",Exact-CompGuaLeg)

print("准确值",Exact)

print("(4):")

CompTrap = ComplexTrapzoid(f4,1,2,7019)

CompSimp = ComplexSimpson(f4,1,2,24)

CompGuaLeg = CompexGuassLegendreI(f4,1,2,22)

Exact = math.exp(1)\*math.exp(1)

print("复化梯形求积计算值",CompTrap,"误差",Exact-CompTrap)

print("复化Simpson求积计算值",CompSimp,"误差",Exact-CompSimp)

print("复化Guass-Legendre求积计算值",CompGuaLeg,"误差",Exact-CompGuaLeg)

print("准确值",Exact)

return

if \_\_name\_\_ == '\_\_main\_\_':

test();

代码的运行结果如下图所示：



### 第三问

比较算法的计算量：

由于计算机的乘除运算的时间远大于加减运算的时间，所以在分析运算量的时候，选择忽略加减运算的时间，只考虑乘除运算的时间，并且假设求函数在某点对应的函数值与乘除运算所用时间相同。

假设在复化求积时积分区间被等分为n等分。

对于复化梯形求积公式，每个区间上需要进行2次乘法运算，，总共n个区间，需要进行2n次乘除法，所有区间上总共需要计算 n+1次函数值，所以总共需要3n+1次运算量。

对于复化Simpson求积公式，总共需要计算2n+1个点的函数值，求f((a+b)/2)需要一次除法，n和区间总共n次，每个区间上需要进行3次乘除法，总共n个区间，所以乘除法运算量为3n。总共需要6n+1的运算量。

对于复化Guass-Legendre I型求积公式，总共需要计算2n个点的函数值。计算每个区间上的求积节点需要2次乘除法，总共需要2n次乘除法。每个区间上需要进行2次乘除法，总共n个区间，需要2n次。总共需要6n次计算量。

各个算法的计算值与精确值的误差也在上图之中给出：

我们可以发现，在刚好满足绝对误差限的条件下，复化Simpson和复化Guass-Legendre公式所需要选取的求积节点比复化梯形公式所需要的求积节点少得多，上述4题中这两者的求积节点个数都相差几十倍。复化Simpson公式和复化Guass-Legendre公式所需要选取的求积节点个数差别不大。

从误差来看，在刚好满足绝对误差限的条件下，即使是用到几十倍的求积节点，复化梯形求积公式求出来的积分值与精确值的误差也比复化Simpson公式和复化Guass-Legendre公式大，在（1）（2）题的情况中，两者的误差甚至达到了10倍甚至更大的情况。而复化Simpson公式和复化Guass-Legendre公式两者的误差比较接近，这跟公式本身的精度有着密切的关系。

# 实验4.2比较复化Simpson和变步长Simpson

### 实验目的:

比较复化Simpson方法和变步长(区间逐次分半求积法)Simpson方法

### 实验题目：

计算下列定积分

(1)

(2)

(3)

### 实验要求

1. 分别用复化Simpson公式和变步长Simpson公式计算，要求绝对误差限为，输出每种方法所需的节点数和积分近似值。
2. 分析比较计算结果

### 第一问

解答：复化Simpson公式的余项为 其中

变步长Simpson公式的误差满足关系：

1. 积分区间 [0,2]

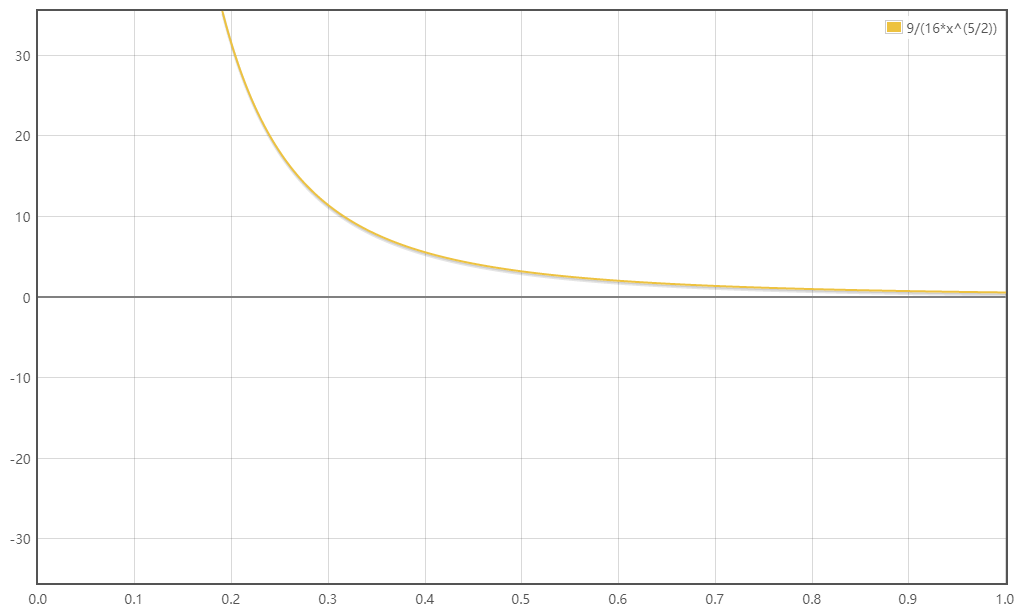
在[0,2]上单调递增且为正。所以在[0,2]上的绝对值的最大值为

带入复化Simpson余项公式得到:

解得n并取整得到：

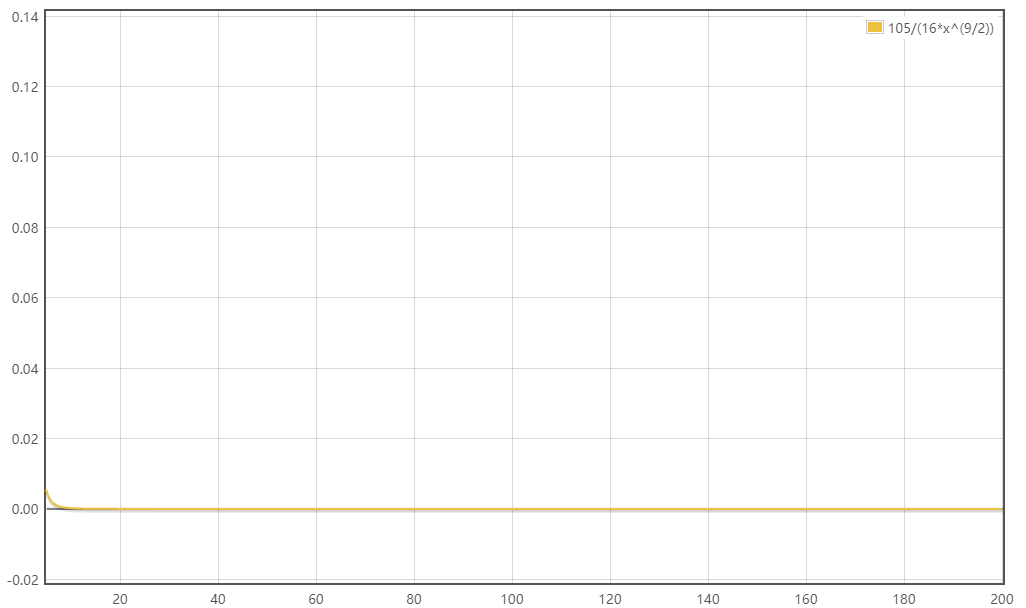
1. 积分区间[0,1]

在[0,1]上单调递减且为正，所以在[0,1]上的最大值为+∞，所以无法直接通过事前误差估计的方法确定步长，只能通过事后误差估计的方法得到一个合适的步长。



1. 积分区间[5,200]

在[5,200]上单调递减且为正，所以在[5,200]上绝对值的最大值为



代入复化Simpson公式的余项得到：

解得n并取整得到：

计算以上积分值的python代码如下所示：

import math

from sympy import \*

def Simpson(f, a, b):

return (b - a) \* (f(a) + 4 \* f((a + b) / 2) + f(b)) / 6

# 复化Simpson求积公式 n表示分段数 步长1/n

def ComplexSimpson(f,a,b,n):

h=(b-a)/n

sum =0

for k in range(n):

sum += Simpson(f,a+k\*h,a+(k+1)\*h)

return sum

# 变步长Simpson求积公式

def VariableStepSimpson(f,a,b,e):

i = 1

Sn = ComplexSimpson(f,a,b,i)# Sn初始化为S1

S2n = ComplexSimpson(f,a,b,2\*i)# S2n初始化为S2

e\_S2n = (1/15)\*abs(S2n - Sn)

while(e\_S2n > e):

i = i\*2

Sn = S2n

S2n = ComplexSimpson(f,a,b,2\*i)

e\_S2n = (1 / 15) \* abs(S2n - Sn)

return 2\*i,S2n

def f1(x):

return math.pow(x,6)/10-x\*x+x

def f2(x):

return x\*math.sqrt(x)

def f3(x):

return 1/(math.sqrt(x))

def main():

print("(1):")

VariStep = VariableStepSimpson(f1,0,2,0.5\*math.pow(10,-7))

print("复化Simpson:",ComplexSimpson(f1,0,2,76))

print("变步长Simpson:","2n:",VariStep[0],"积分值:",VariStep[1])

print("(2):")

VariStep = VariableStepSimpson(f2,0,1,0.5\*math.pow(10,-7))

print("复化Simpson:",ComplexSimpson(f2,0,1,100))

print("变步长Simpson:","2n:",VariStep[0],"积分值:",VariStep[1])

print("(3):")

VariStep = VariableStepSimpson(f3,5,200,0.5\*math.pow(10,-7))

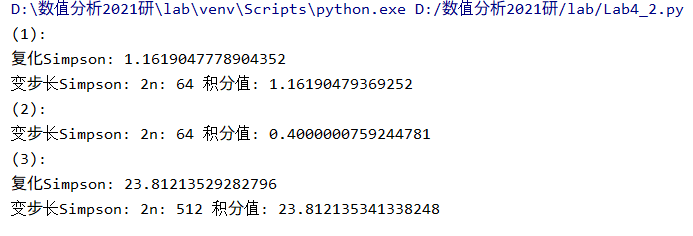
print("复化Simpson:",ComplexSimpson(f3,5,200,1742))

print("变步长Simpson:","2n:",VariStep[0],"积分值:",VariStep[1])

if \_\_name\_\_ == '\_\_main\_\_':

main();

运行程序的输出结果如下图所示：



综上所述：

题目(1)：利用复化Simpson公式需要的节点数为 2\*76+1 =153。变步长Simpson需要节点数为 2\*64+1 = 129

题目(2):无法利用复化Simpson公式用事前估计的方法求出n，但是可以利用事后误差估计的方法求出积分的近似值，变步长Simpson方法求出来需要的节点数为2\*64+1 =129

题目(3):利用复化Simpson公式需要的节点数为2\*1742+1=3485，利用变步长Simpson需要节点数为2\*512+1 = 1025

各题的积分近似值如上图所示

### 第二问

通过第一问的计算过程和结果，我们可以发现。

1. 有些时候用事前误差估计无法确定n值，但是用事后误差估计可以求出来。
2. 用事前误差估计，因为余项表达式中的取的是积分区间上的最大值，所以会导致求出来的n往往会更大，n更大就会需要取更多的求积节点以及更多的计算量。

# 实验2.1 多项式插值的震荡现象

### 问题提出

考虑在一个固定的区间上用插值逼近一个函数，显然Lagrange插值中使用的节点越多，插值多项式的次数就越高，我们自然关心插值多项式的次数增加时，是否也更加靠近被逼近的函数，Runge给出的一个例子是极著名并富有启发性的，设区间[-1,1]上函数：

### 实验内容

考虑在区间[-1,1]的一个等距划分，分点为

则Lagrange插值多项式为

其中是n次的Lagrange插值基函数。

### 实验要求

1. 选择不断增大的分点数目画出原函数f(x)以及插值多项式函数在[-1,1]上的图像，比较并分析实验结果。
2. 选择其他的函数，例如定义在区间[-5,5]上的函数

重复上述的实验看结果如何。

### 第一问

求解并画出和图像的python代码如下所示，n的取值为2到15：

import math

import numpy as np

import matplotlib.pyplot as plt

from pylab import mpl

def f1(x):

return 1/(1+25\*x\*x)

def f2(x):

return x/(1+math.pow(x,4))

def f3(x):

return np.arctan(x)

# xn yn 插值节点 x是需要求值的点

# xn 是一个数组，包含 x0 x1 .. xn

def Lanrange(xn,yn,x):

n = len(xn)-1

lagrange = 0

for i in range(n+1):

li =1

for j in range(n+1):

if(j!=i):

li\*=(x-xn[j])/(xn[i]-xn[j])

lagrange += yn[i]\*li

return lagrange

# xn yn 用来画原函数的点

# LagXn LagYn 用来画Lagrange插值函数的点

# DiscreteXn DiscreteYn 插值节点。

def Draw(xn, yn, LagXn, LagYn,DiscreteXn,DiscreteYn,info):

plt.plot(LagXn, LagYn, label="拟合曲线Ln(x)", color="black")

plt.plot(xn,yn,label="实际曲线f(x)",color="green")

plt.scatter(DiscreteXn, DiscreteYn, label="插值节点", color="red")

#使用中文字符集

mpl.rcParams['font.sans-serif'] = ['SimHei']

mpl.rcParams['axes.unicode\_minus'] = False

plt.title("拉格朗日插值拟合数据 n="+str(len(DiscreteXn)-1))

plt.legend(loc="upper left")

plt.savefig("./"+info+".png")

plt.show()

def main():

for n in range(2,16):

xn=[]

yn=[]

DiscreteXn = []

DiscreteYn = []

LagXn=[]

LagYn=[]

for i in range(n+1):

xi = -1+2\*i/n

DiscreteXn.append(xi)

DiscreteYn.append(f1(xi))

for i in range(100\*n+1):

xi = -1+2\*i/(100\*n)

LagXn.append(xi)

LagYn.append(Lanrange(DiscreteXn,DiscreteYn,xi))

for i in range(100\*n):

xi = -1+2\*i/(100\*n)

xn.append(xi)

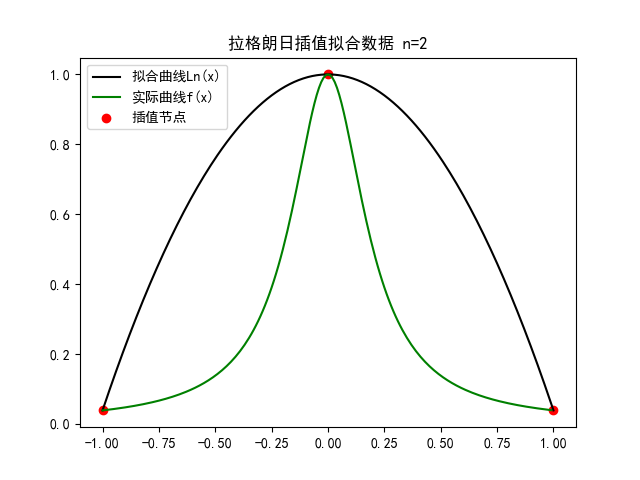
yn.append(f1(xi))

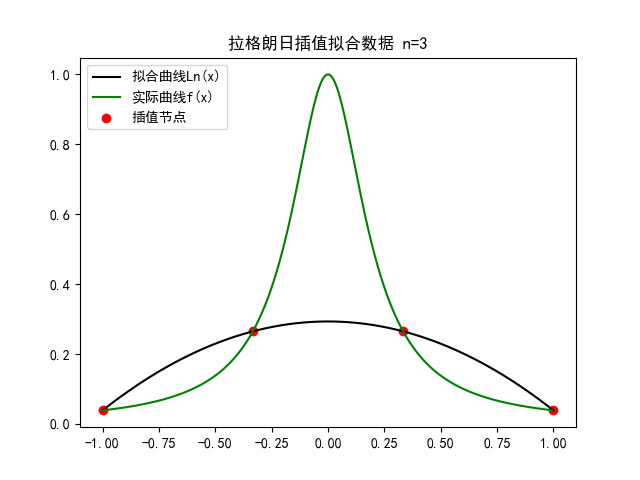
Draw(xn, yn, LagXn, LagYn,DiscreteXn,DiscreteYn,"f1\_n="+str(n))

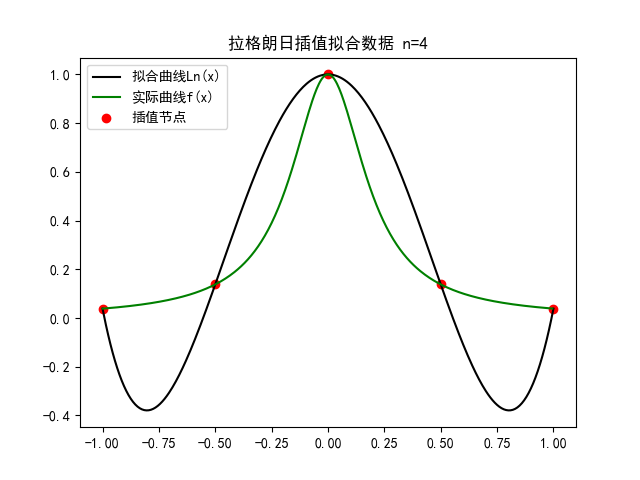
if \_\_name\_\_ == "\_\_main\_\_":

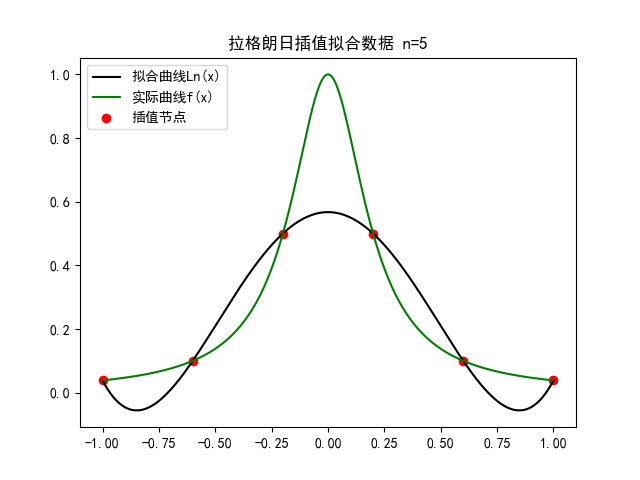
main()

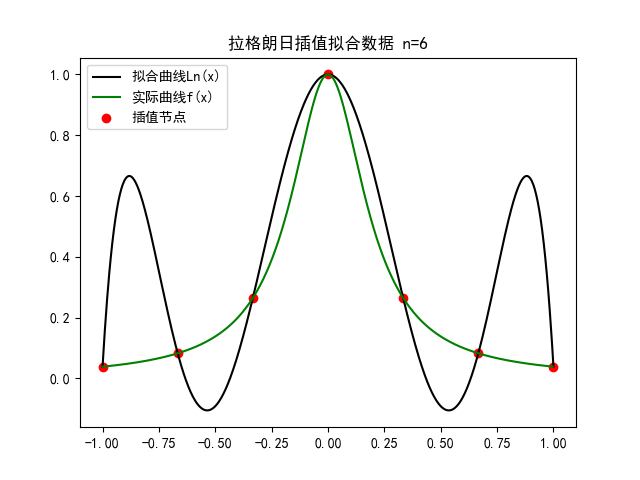
输出的图像如下：

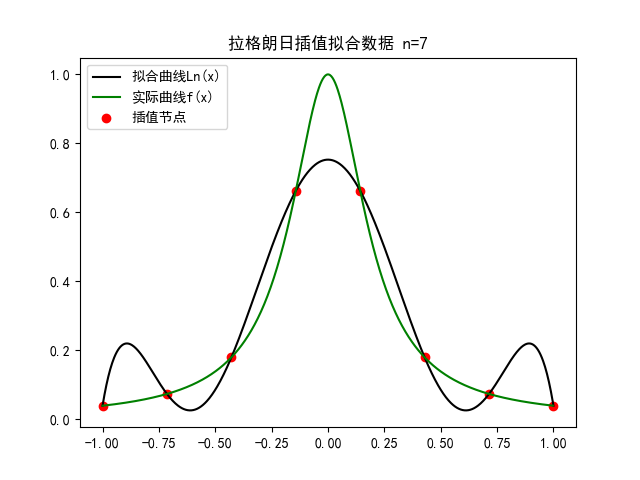


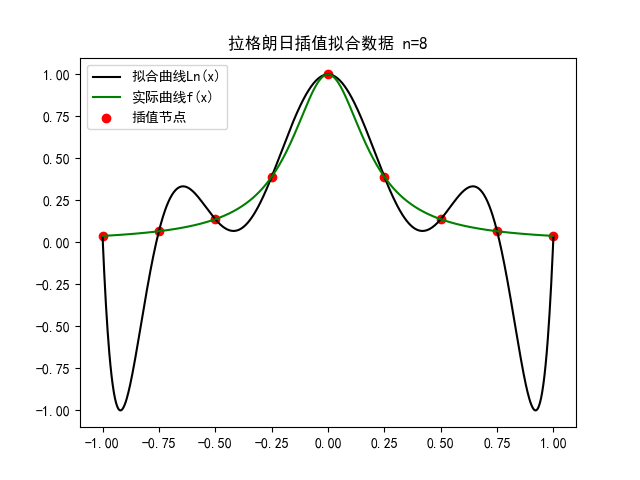


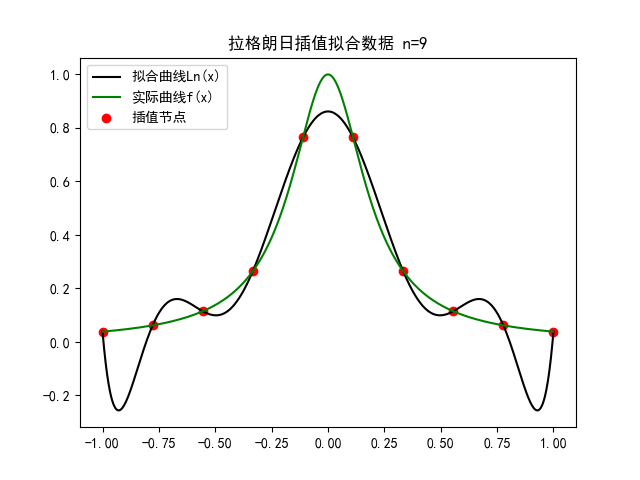


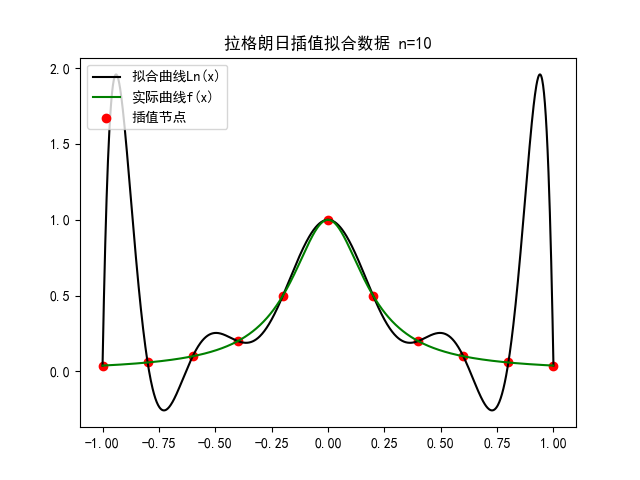


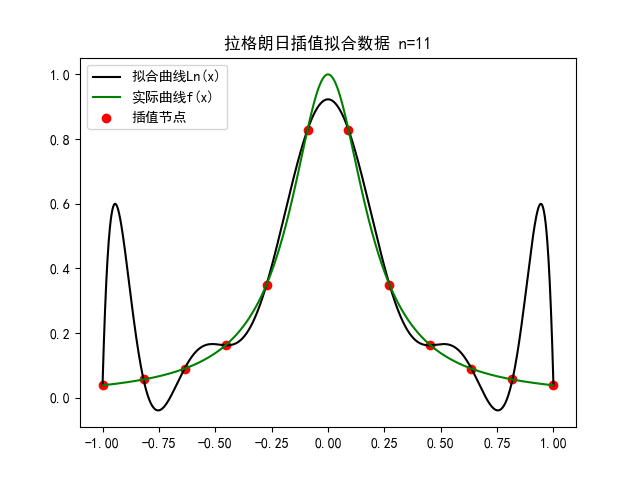


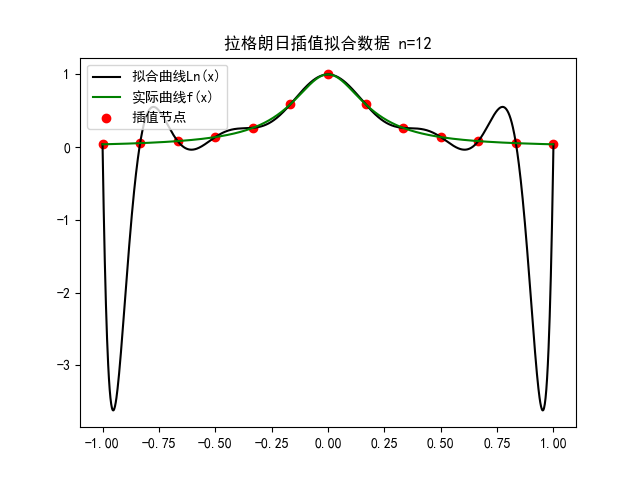


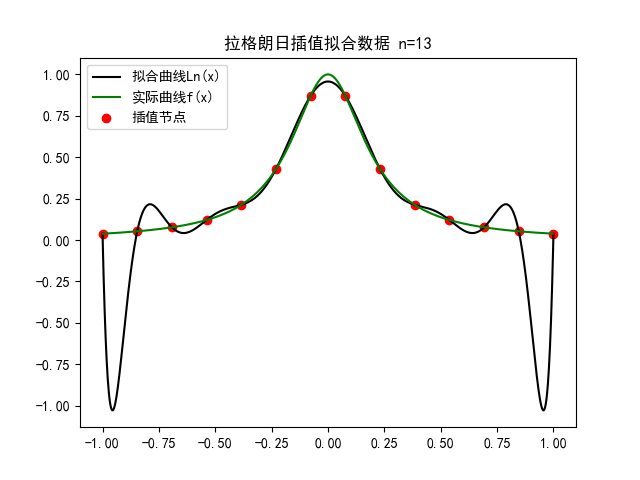


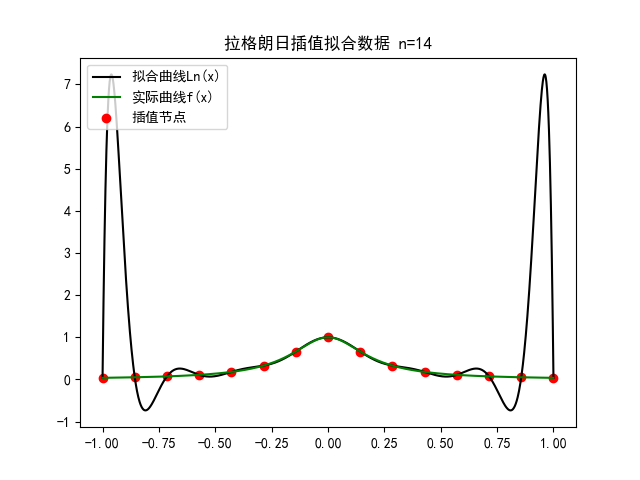


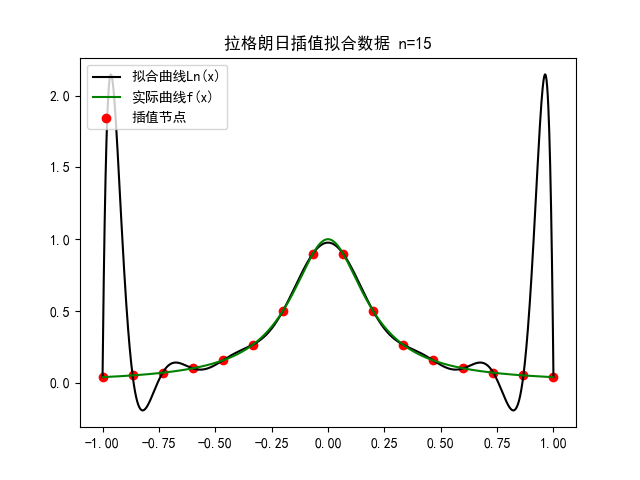












分析:

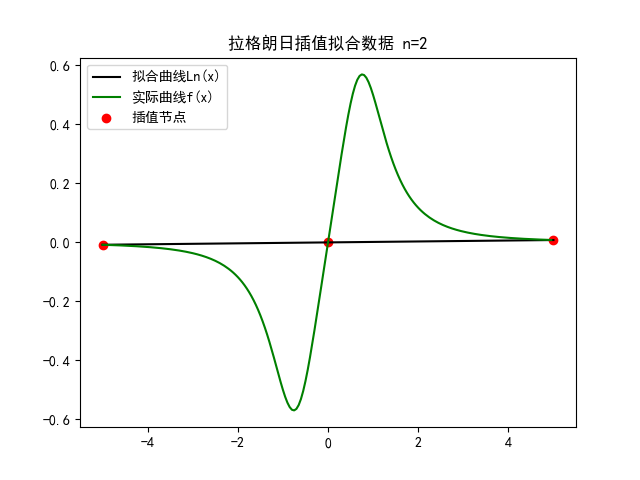
根据输出的图像，我们可以发现，当采用等间距的插值接点时，随着分点数n的不断增大，Lagrange插值函数在插值区间的中间部分和原函数的接近程度越来越好。但是在靠近插值区间的两端附近，随着n的增大，会出现明显的抖动，也即Runge现象。

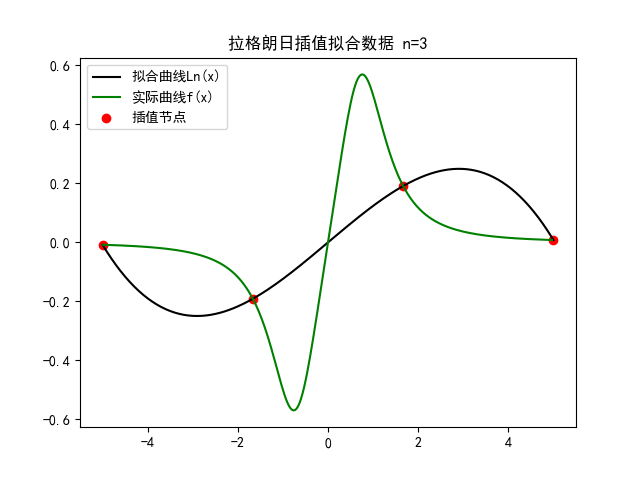
### 第二问

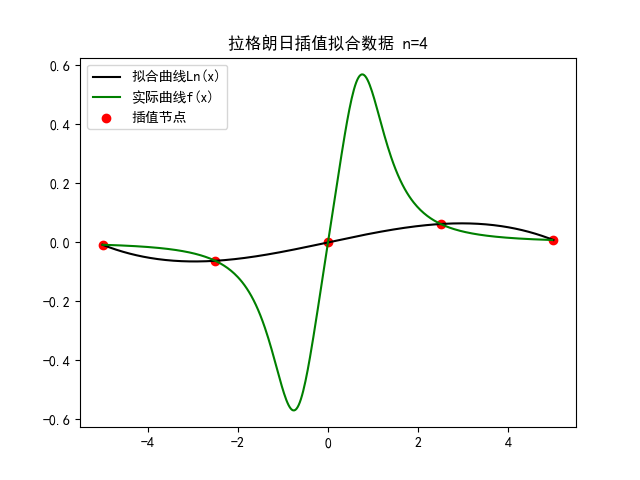
只用对上述第一问的main函数进行稍微的修改，即可用于计算

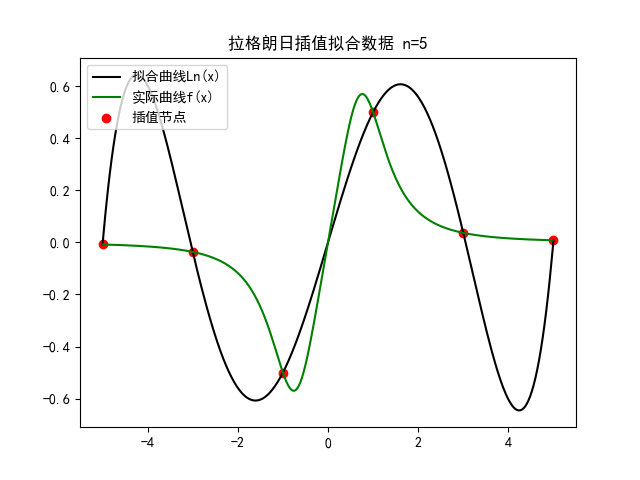
的Lagrange插值

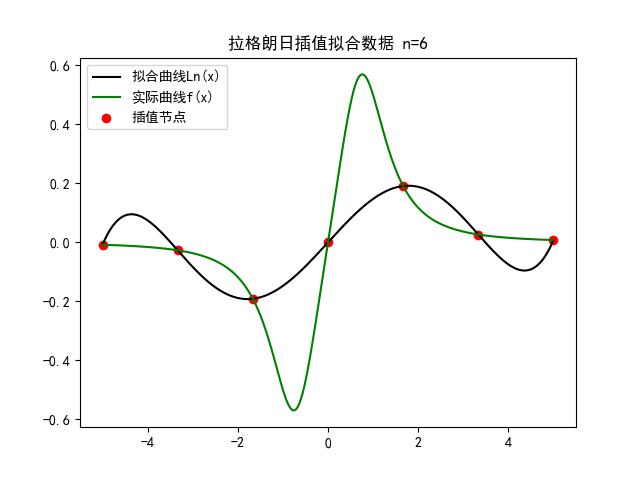
画出的和对应的Lagrange函数的图像如下所示：

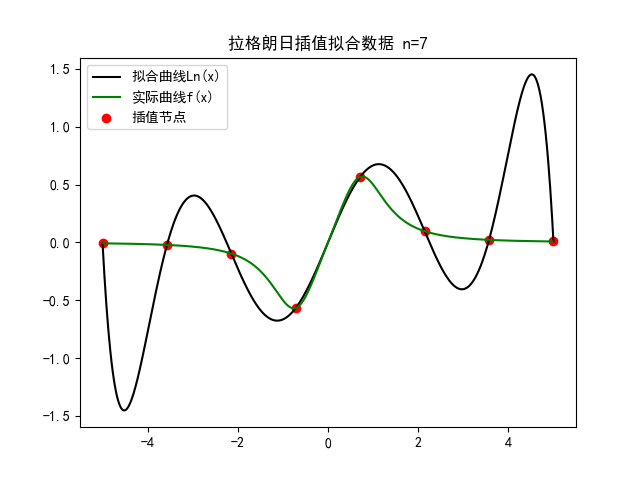


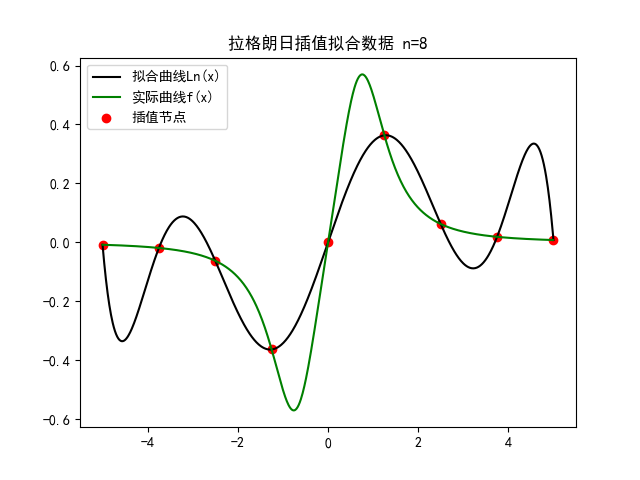


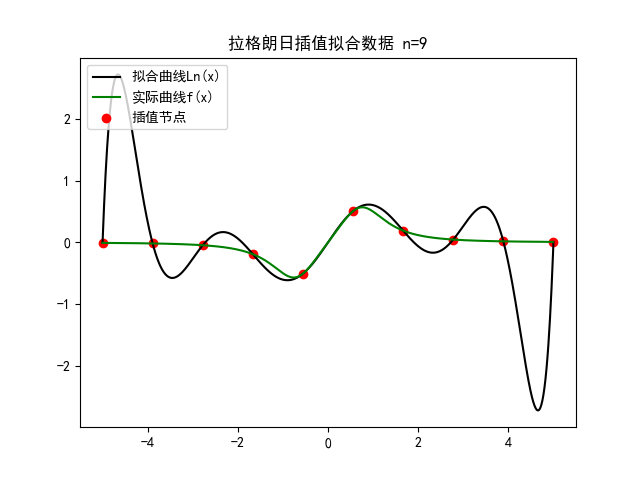


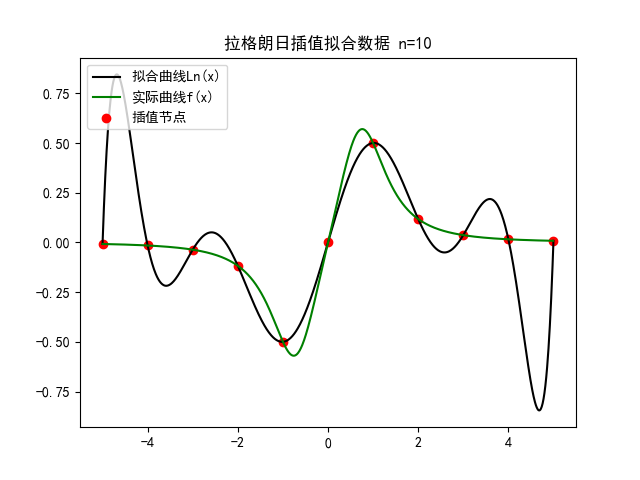


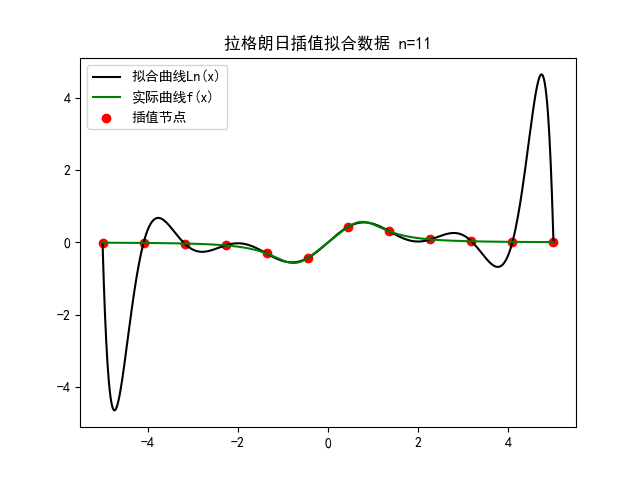


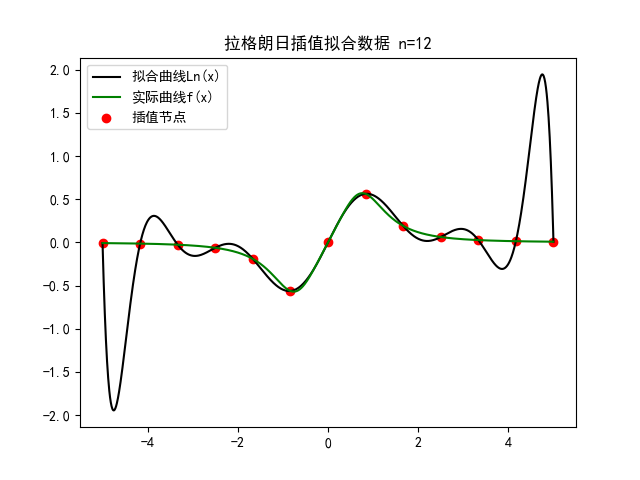


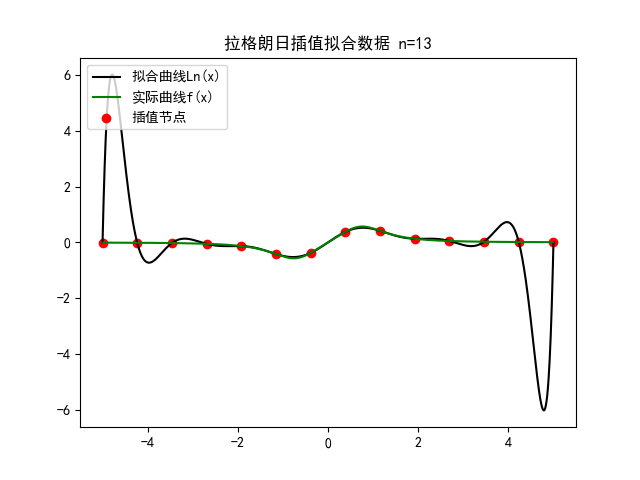


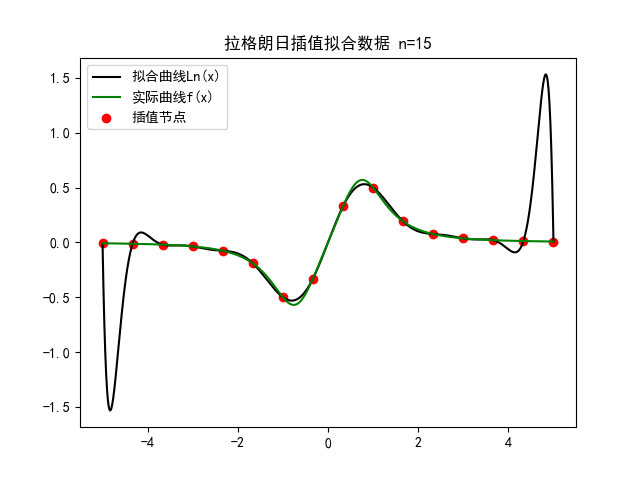
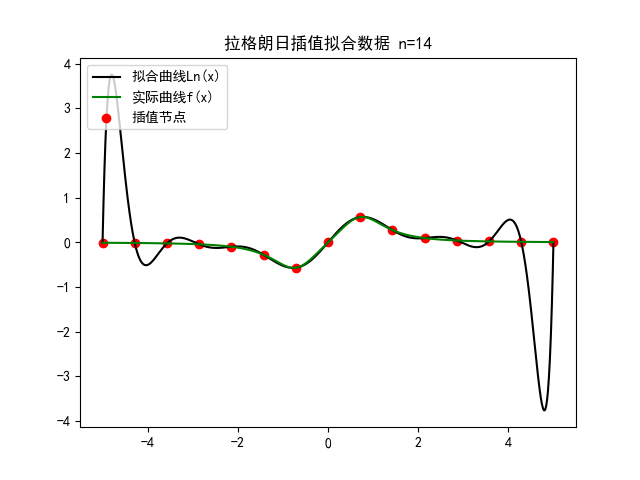




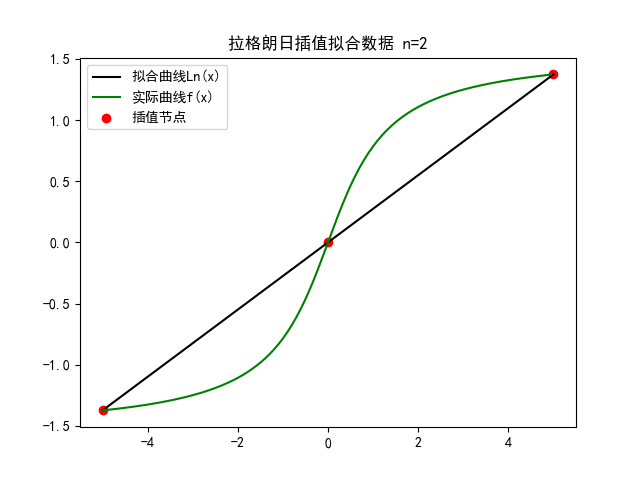


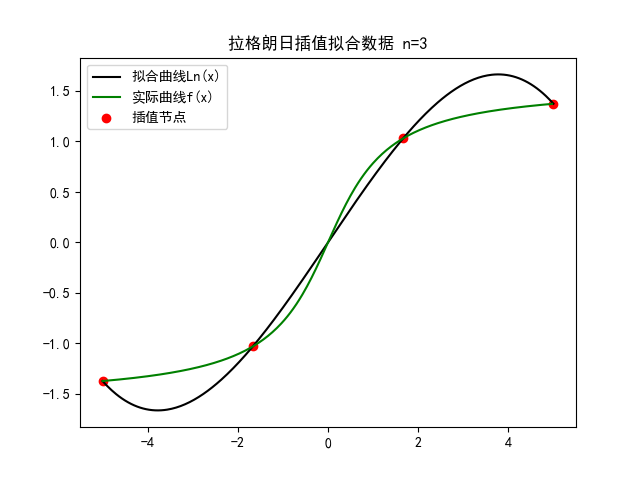


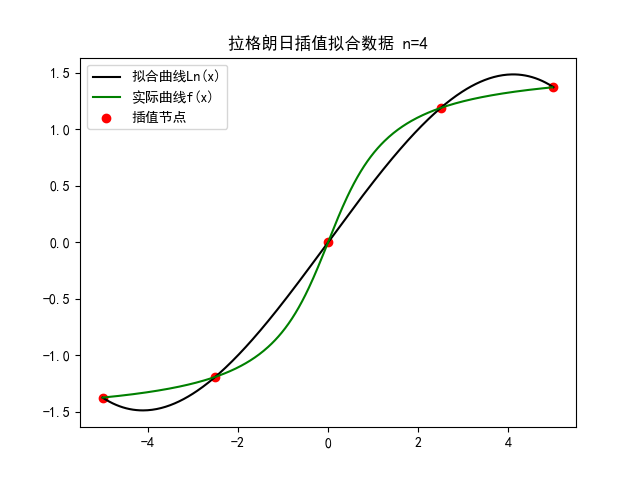


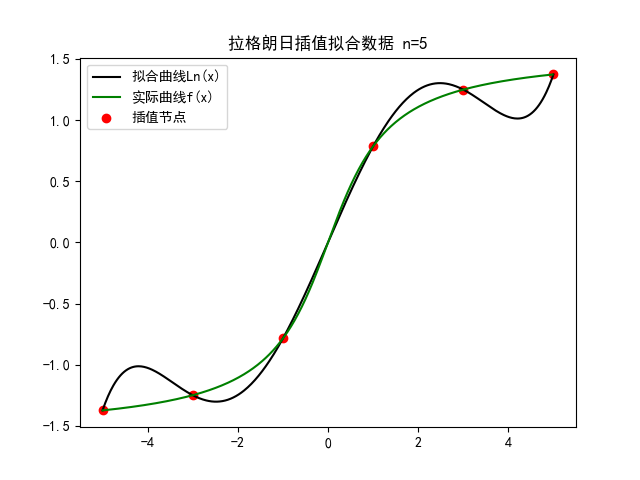


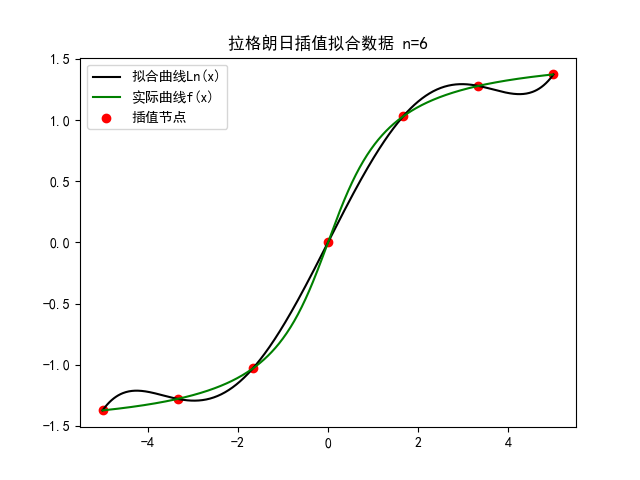
画出的及对应的Lagrange插值函数的图像如下所示：

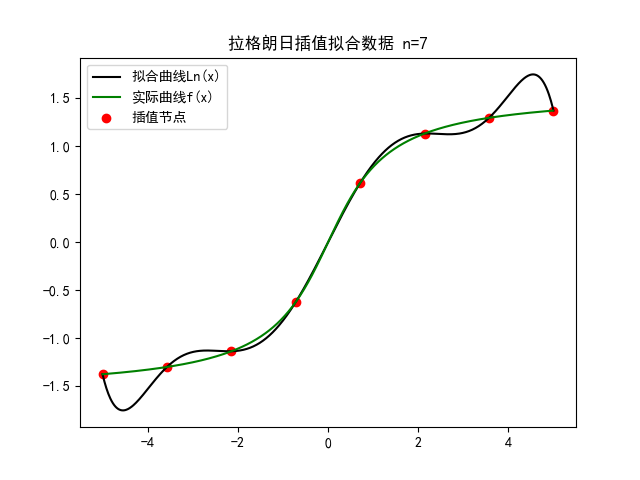


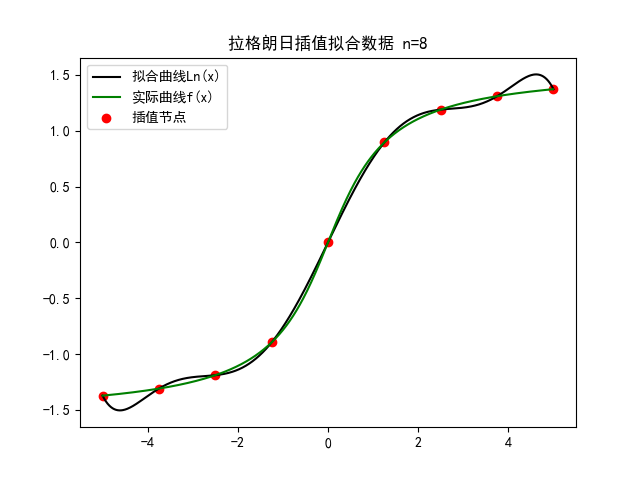


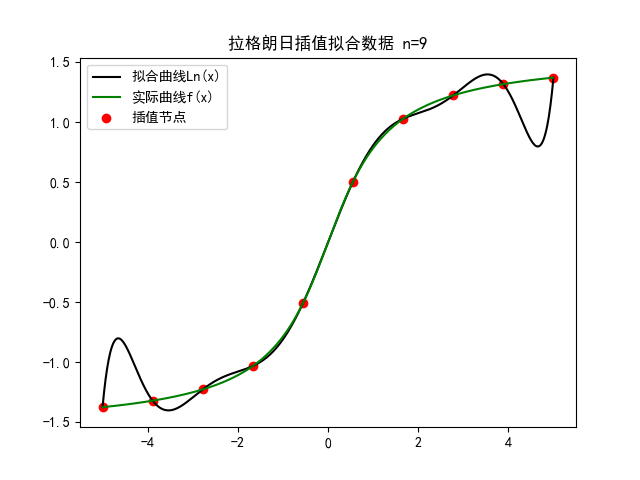


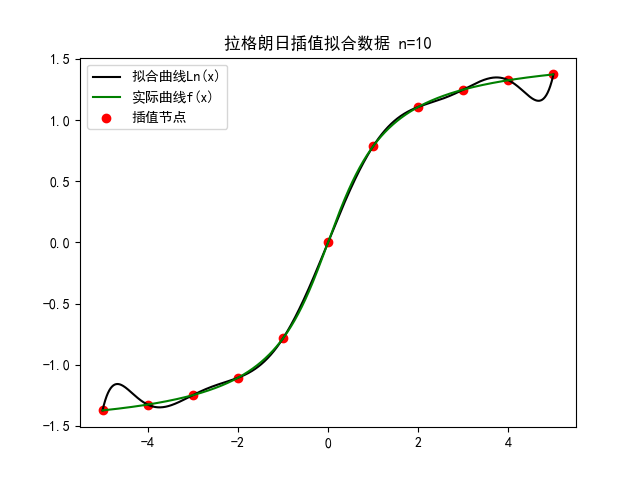


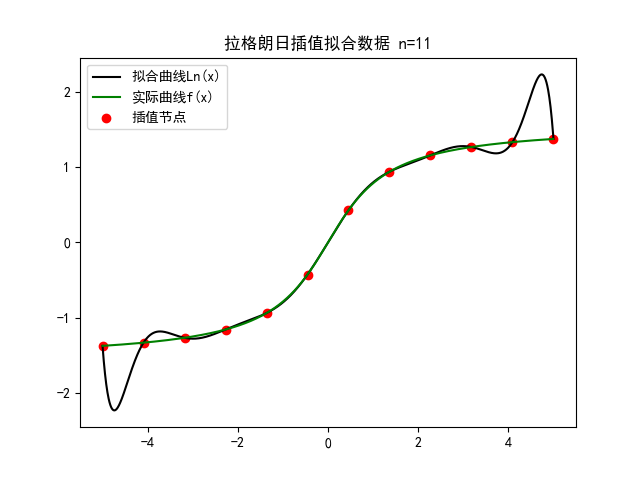


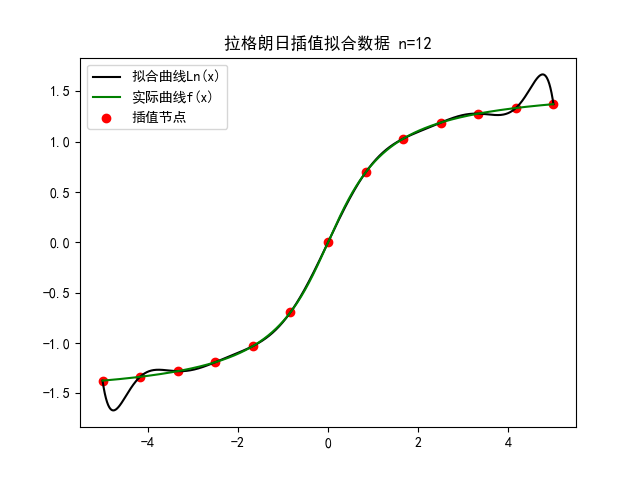


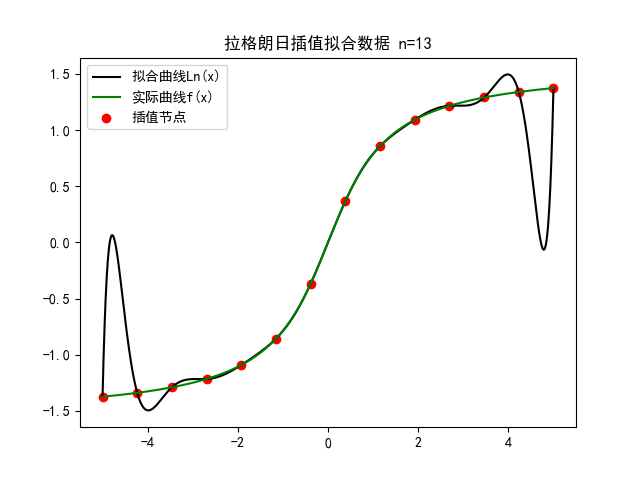


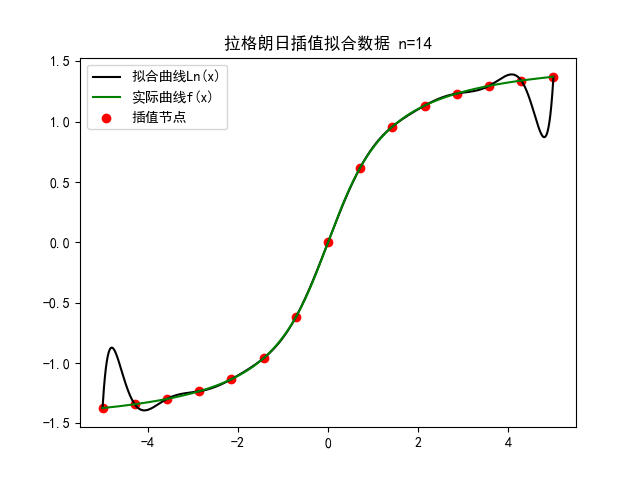


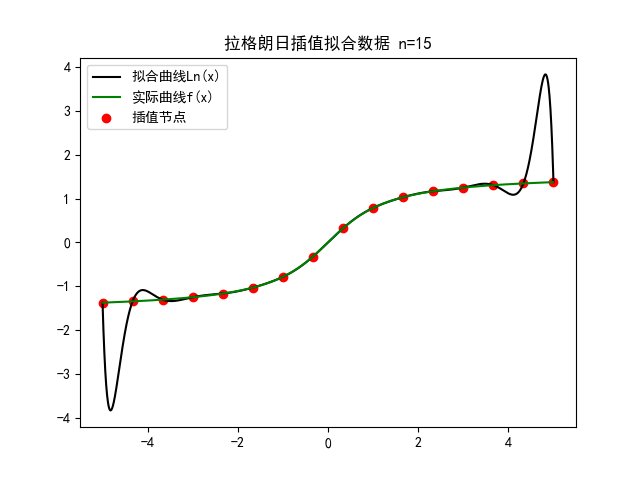












分析：

通过画出及对应的Lagrange函数的图像。我们仍然可以看到，当采用等间距节点进行插值时，随着分点数n的增大，也出现了和第一问一样的问题，插值区间中间部分随着n的增大越来越接近，但是在插值区间的两端，会随着n的增大出现震荡，也即Runge现象。