

# 计算方法实验（实验 1）

2022 年 4 月 22 日

数据显示结果已保留 4 位小数。

## 1 实验题目 1：拉格朗日（Lagrange）插值

### 1.1 问题分析

准确描述并总结出实验题目（摘要），并准确分析原题的目的和意义。

#### 1.1.1 方法概要

给定平面上  $n+1$  个不同的数据点  $(x_k, f(x_k)), k=0, 1, \dots, n, x_i \neq x_j, i \neq j$  则满足条件

$$P_n(x_k) = f(x_k), k=0, 1, \dots, n$$

的  $n$  次拉格朗日插值多项式

$$P_n(x) = \sum_{k=0}^n f(x_k) l_k(x)$$

是存在唯一的。若  $x_k \in [a, b], k=0, 1, \dots, n$ ，且函数  $f(x)$  充分光滑，则当  $x \in [a, b]$  时，有误差估计式

$$f(x) - P_n(x) = \frac{f^{n+1}(\xi)}{(n+1)!} (x-x_0)(x-x_1) \cdots (x-x_n), \xi \in [a, b]$$

#### 1.1.2 实验目的

利用拉格朗日插值多项式  $P_n(x)$  求  $f(x)$  的近似值。

输入： $n+1$  个数据点  $x_k, f(x_k), k=0, 1, \dots, n$ 、插值点  $x$

输出:  $f(x)$  在插值点  $x$  的近似值  $P_n(x)$

## 1.2 数学原理

数学原理表达清晰且书写准确。

### 1.2.1 证明 $P_n(x)$ 存在且唯一

证明: 使用归纳法证明。

当  $n = 0$ , 一定存在  $P_0(x) = C = f_0(x)$  满足要求。

假设当  $n = k - 1$  时, 存在满足要求的  $P_{k-1}(x)$ , 则当  $n = k$ , 有

$$P_k(x) = P_{k-1}(x) + c(x - x_0)(x - x_1) \cdots (x - x_{k-1})(x - x_k), \text{ } c \text{ 为系数}$$

则  $\because P_n(x_n) = f(x_n), \therefore$  参数  $c$  是可求的, 故  $P_n(x)$  是存在的。

由多项式基本定理,  $\because P_n(x)$  的次数  $\leq n, \therefore P_n(x)$  是唯一存在的。

### 1.2.2 计算方法

对平面上  $n + 1$  个点  $(x_k, f(x_k)), k = 0, 1, \cdots, n, x_i \neq x_j, i \neq j$  定义  $n$  次多项式:

$$L_k(x) = \frac{(x - x_0)(x - x_1) \cdots (x - x_{k-1})(x - x_{k+1}) \cdots (x - x_n)}{(x_k - x_0)(x_k - x_1) \cdots (x_k - x_{k-1})(x_k - x_{k+1}) \cdots (x_k - x_n)}$$

则  $L_k(x_k) = 1, L_k(x_m) = 0, m \neq k$ 。

定义:

$$P_n(x) = f(x_0)L_0(x) + f(x_1)L_1(x) + \cdots + f(x_n)L_n(x) = \sum_{k=0}^n f(x_k)L_k(x)$$

为  $f(x)$  的  $n$  次拉格朗日插值多项式。

## 1.3 程序设计流程

编译通过, 根据输入能得到正确输出。

```
[11]: # 引入需要的包
import numpy as np
```

```
from pandas import DataFrame
from matplotlib import pyplot as plt
```

```
[12]: #  $L_k(x)$ 
def L(k, x_list: np.ndarray):
    def l(x: float):
        x_k = x_list[k]
        slice = np.array([*x_list[:k], *x_list[k+1:]])
        repeat = np.array([x, ] * (len(x_list) - 1))
        repeat_k = np.array([x_k, ] * (len(x_list) - 1))
        return np.prod(repeat - slice) / np.prod(repeat_k - slice)
    return l

# 拉格朗日多项式  $P_n(x)$ 
def P(f, x_list: np.ndarray, x: float):
    return np.sum([f(x_list[k]) * L(k, x_list)(x) for k in range(len(x_list))])

# 获得  $x\_list$ 
def linspace(r: int, n: int):
    return [-r + k * (2 * r / n) for k in range(0, n + 1)]
```

### 1.3.1 问题一

```
[13]: def problem1():
    print("问题 1: 拉格朗日插值多项式的次数 n 越大越好吗? ")

    def sub(targets_x, targets_n, solve):
        return DataFrame({n: {x: solve(n, x) for x in targets_x} for n in
        ↪ targets_n}).round(4)

    print("(1) 考虑  $f(x) = 1 / (1 + x^2)$  in  $[-5, 5]$ ")
    targets_x_1 = [0.75, 1.75, 2.75, 3.75, 4.75]
    targets_n = [5, 10, 20]
    print(sub(
        targets_x=targets_x_1,
        targets_n=targets_n,
```

```

        solve=lambda n_i, x: P(
            lambda x_i: 1 / (1 + x_i ** 2), linspace(5, n_i), x
        )
    ))
print("差值: ")
print(sub(
    targets_x=targets_x_1,
    targets_n=targets_n,
    solve=lambda n_i, x: ((1 / (1 + x ** 2)) - P(
        lambda x_i: 1 / (1 + x_i ** 2), linspace(5, n_i), x
    ))
))

print("(2) 考虑  $f(x) = e^x$  in  $[-1, 1]$ ")
targets_x_2 = [-0.95, -0.05, 0.05, 0.95]
print(sub(
    targets_x=targets_x_2,
    targets_n=targets_n,
    solve=lambda n_i, x: P(lambda x_i: np.e ** x_i,
        linspace(5, n_i), x)
))
print("差值: ")
print(sub(
    targets_x=targets_x_2,
    targets_n=targets_n,
    solve=lambda n_i, x: (np.e ** x - P(lambda x_i: np.e ** x_i,
        linspace(5, n_i), x))
))
print("画出两个函数以及其拉格朗日多项式的图像: ")
slice_fluent_size = 1000
x_linespace_2 = np.linspace(-1, 1, slice_fluent_size)
x_linespace_10 = np.linspace(-5, 5, slice_fluent_size)
y1 = 1 / (1 + x_linespace_10**2)
Ls1 = {n_i: np.array([P(lambda x_i: np.e ** x_i, linspace(5, n_i), x)
    for x in x_linespace_10]) for n_i in targets_n}
plt.figure(dpi=150)

```

```

plt.title("(1) Consider $f(x) = 1 / (1 + x^2)$")
plt.legend(handles=plt.plot(x_linespace_10, y1, label="$f(x) = 1 / (1 + \r\n\r\nx^2)$"), loc='best')
plt.figure(dpi=150)
plt.title("(1) Consider $f(x) = 1 / (1 + x^2)$, select n")
plt.legend(handles=[*plt.plot(x_linespace_10, Ls1[n_i], \r\n\r\nlabel=f"$P_n(x), n={n_i}$") [0] for n_i in Ls1],
            plt.plot(x_linespace_10, y1, label="$f(x) = 1 / (1 + \r\n\r\nx^2)$") [0]], loc='best')
y2 = np.e**x_linespace_2
Ls2 = {n_i: np.array([P(lambda x_i: np.e ** x_i, linspace(5, n_i), x)
                    for x in x_linespace_2]) for n_i in targets_n}
plt.figure(dpi=150)
plt.title("(2) Consider $f(x) = e^x$")
plt.legend(handles=plt.plot(x_linespace_2, y2, label="$f(x) = e^x$"), \r\n\r\nloc='best')
plt.figure(dpi=150)
plt.title("(2) Consider $f(x) = e^x$, select n")
plt.legend(handles=[*plt.plot(x_linespace_2, Ls2[n_i], \r\n\r\nlabel=f"$P_n(x), n={n_i}$") [0] for n_i in Ls2],
            plt.plot(x_linespace_2, y2, label="$f(x) = e^x$") [0]], \r\n\r\nloc='best')

problem1()

```

问题 1: 拉格朗日插值多项式的次数  $n$  越大越好吗?

(1) 考虑  $f(x) = 1 / (1 + x^2)$  in  $[-5, 5]$

	5	10	20
0.75	0.5290	0.6790	0.6368
1.75	0.3733	0.1906	0.2384
2.75	0.1537	0.2156	0.0807
3.75	-0.0260	-0.2315	-0.4471
4.75	-0.0157	1.9236	-39.9524

差值:

	5	10	20
0.75	0.1110	-0.0390	0.0032
1.75	-0.1272	0.0556	0.0077
2.75	-0.0369	-0.0988	0.0361
3.75	0.0923	0.2979	0.5134
4.75	0.0582	-1.8812	39.9949

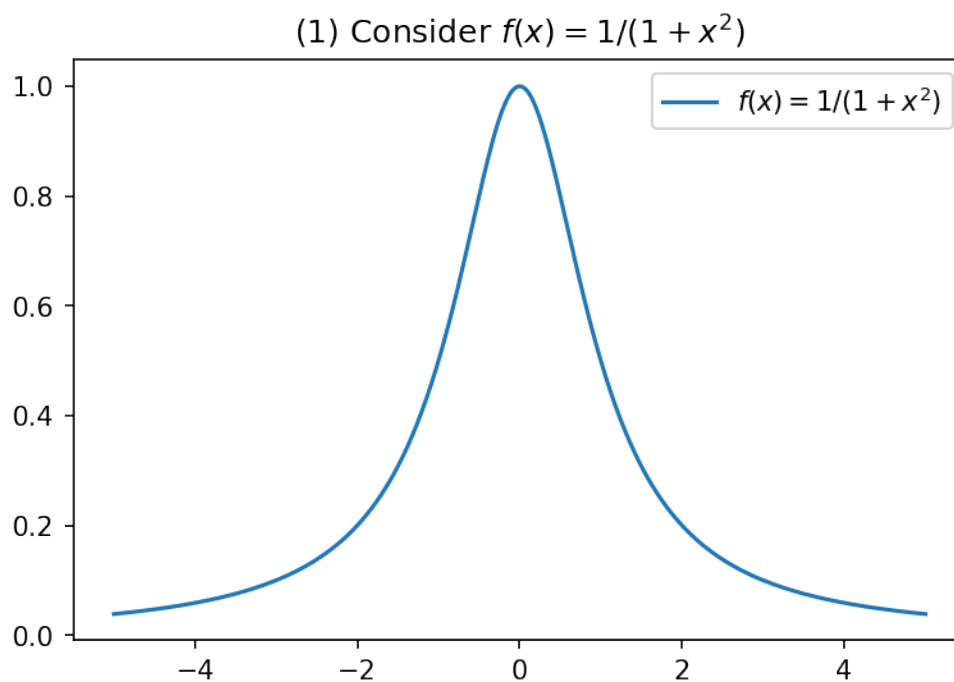
(2) 考虑  $f(x) = e^x$  in  $[-1, 1]$

	5	10	20
-0.95	0.4306	0.3867	0.3867
-0.05	1.5272	0.9512	0.9512
0.05	1.6345	1.0513	1.0513
0.95	2.6414	2.5857	2.5857

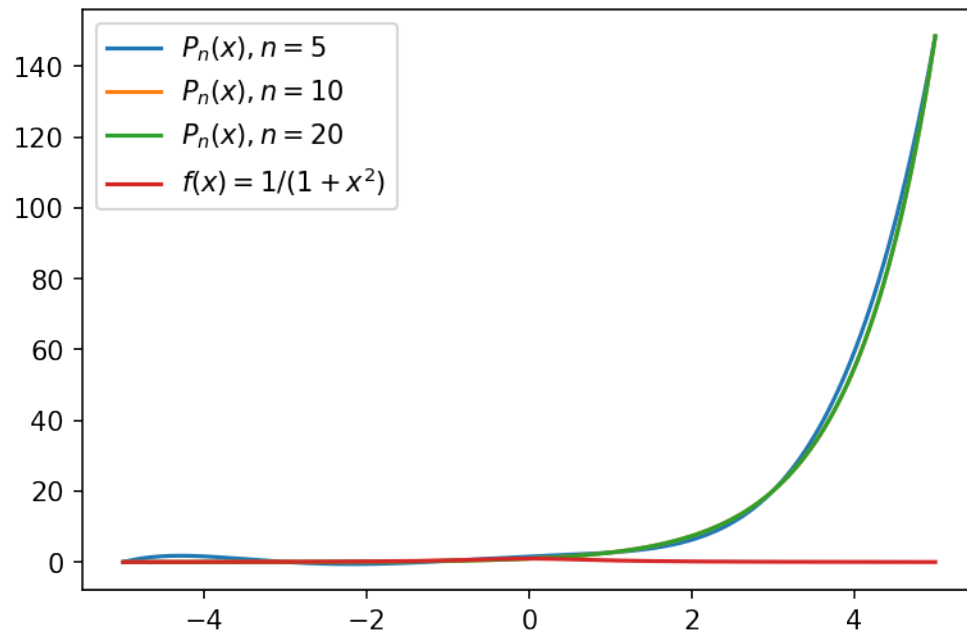
差值:

	5	10	20
-0.95	-0.0439	0.0	0.0
-0.05	-0.5760	0.0	-0.0
0.05	-0.5832	-0.0	0.0
0.95	-0.0557	-0.0	-0.0

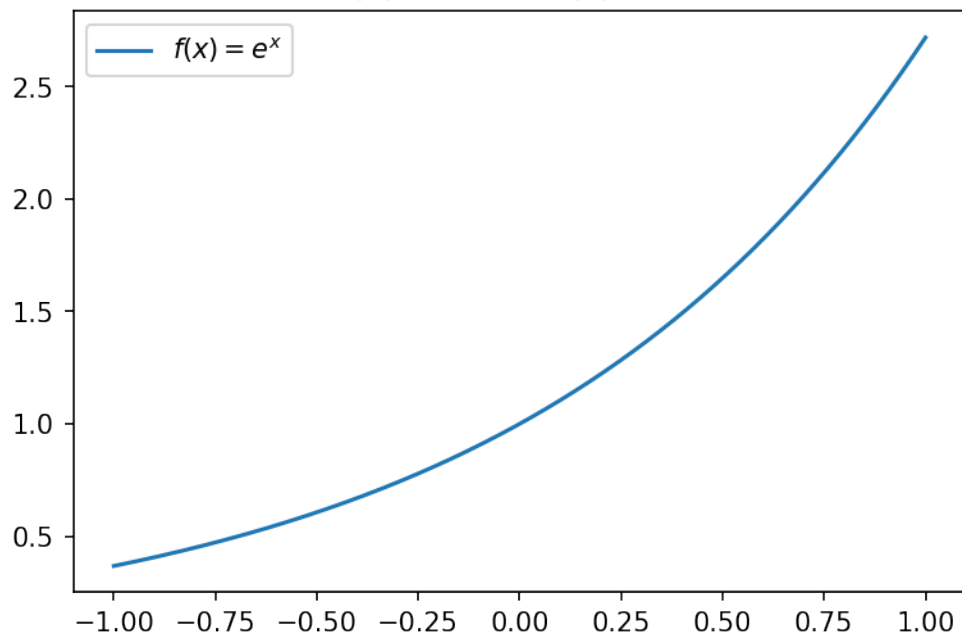
画出两个函数以及其拉格朗日多项式的图像:



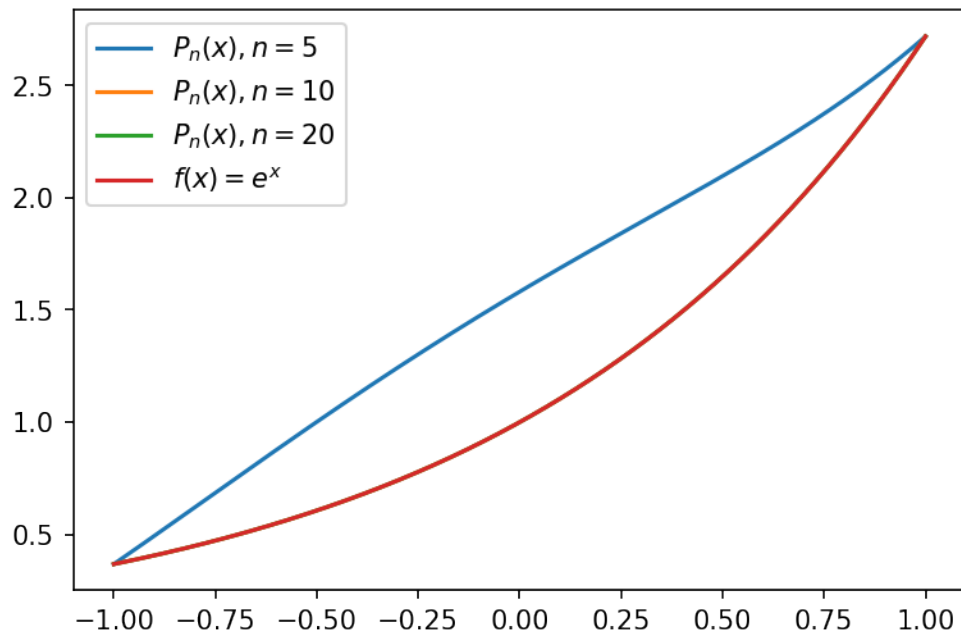
(1) Consider  $f(x) = 1/(1 + x^2)$ , select  $n$



(2) Consider  $f(x) = e^x$



(2) Consider  $f(x) = e^x$ , select n



### 1.3.2 问题二

```
[14]: def problem2():
    print("问题 2: 插值区间越小越好吗? ")

    def sub(targets_x, targets_n, solve):
        return DataFrame({n: {x: solve(n, x) for x in targets_x} for n in
        ↪ targets_n}).round(4)

    print("(1) 考虑  $f(x) = 1 / (1 + x^2)$  in  $[-1, 1]$ ")
    targets_x_1 = [-0.95, -0.05, 0.05, 0.95]
    targets_n = [5, 10, 20]
    print(sub(
        targets_x=targets_x_1,
        targets_n=targets_n,
        solve=lambda n_i, x: P(
            lambda x_i: 1 / (1 + x_i ** 2), linspace(1, n_i), x
```



```

    )
))
print("差值: ")
print(sub(
    targets_x=targets_x_1,
    targets_n=targets_n,
    solve=lambda n_i, x: ((1 / (1 + x ** 2)) - P(
        lambda x_i: 1 / (1 + x_i ** 2), linspace(1, n_i), x
    ))
))

print("(2) 考虑  $f(x) = e^x$  in  $[-5, 5]$ ")
targets_x_2 = [0.75, 1.75, 2.75, 3.75, 4.75]
print(sub(
    targets_x=targets_x_2,
    targets_n=targets_n,
    solve=lambda n_i, x: P(lambda x_i: np.e ** x_i,
        linspace(5, n_i), x)
))
print("差值: ")
print(sub(
    targets_x=targets_x_2,
    targets_n=targets_n,
    solve=lambda n_i, x: (np.e ** x - P(lambda x_i: np.e ** x_i,
        linspace(5, n_i), x))
))
print("画出两个函数以及其拉格朗日多项式的图像: ")
slice_fluent_size = 1000
x_linespace_2 = np.linspace(-1, 1, slice_fluent_size)
x_linespace_10 = np.linspace(-5, 5, slice_fluent_size)
y1 = 1 / (1 + x_linespace_2**2)
Ls1 = {n_i: np.array([P(lambda x_i: np.e ** x_i, linspace(5, n_i), x)
    for x in x_linespace_2]) for n_i in targets_n}
plt.figure(dpi=150)
plt.title("(1) Consider  $f(x) = 1 / (1 + x^2)$ ")

```

```

plt.legend(handles=plt.plot(x_linespace_2, y1, label="$f(x) = 1 / (1 + \u221ax^2)$"), loc='best')

plt.figure(dpi=150)
plt.title("(1) Consider $f(x) = 1 / (1 + x^2)$, select n")
plt.legend(handles=[*plt.plot(x_linespace_2, Ls1[n_i], \u221a
\u2192label=f"$P_n(x), n={n_i}$") [0] for n_i in Ls1],
            plt.plot(x_linespace_2, y1, label="$f(x) = 1 / (1 + \u221ax^2)$") [0]], loc='best')

y2 = np.e**x_linespace_10
Ls2 = {n_i: np.array([P(lambda x_i: np.e ** x_i, linspace(5, n_i), x)
                      for x in x_linespace_10]) for n_i in targets_n}

plt.figure(dpi=150)
plt.title("(2) Consider $f(x) = e^x$")
plt.legend(handles=plt.plot(x_linespace_10, y2, label="$f(x) = e^x$"), \u221a
\u2192loc='best')

plt.figure(dpi=150)
plt.title("(2) Consider $f(x) = e^x$, select n")
plt.legend(handles=[*plt.plot(x_linespace_10, Ls2[n_i], \u221a
\u2192label=f"$P_n(x), n={n_i}$") [0] for n_i in Ls2],
            plt.plot(x_linespace_10, y2, label="$f(x) = e^x$") [0]], \u221a
\u2192loc='best')

problem2()

```

问题 2: 插值区间越小越好吗?

(1) 考虑  $f(x) = 1 / (1 + x^2)$  in  $[-1, 1]$

	5	10	20
-0.95	0.5171	0.5264	0.5256
-0.05	0.9928	0.9975	0.9975
0.05	0.9928	0.9975	0.9975
0.95	0.5171	0.5264	0.5256

差值:

	5	10	20
-0.95	0.0085	-0.0008	0.0

-0.05	0.0047	-0.0000	-0.0
0.05	0.0047	-0.0000	-0.0
0.95	0.0085	-0.0008	0.0

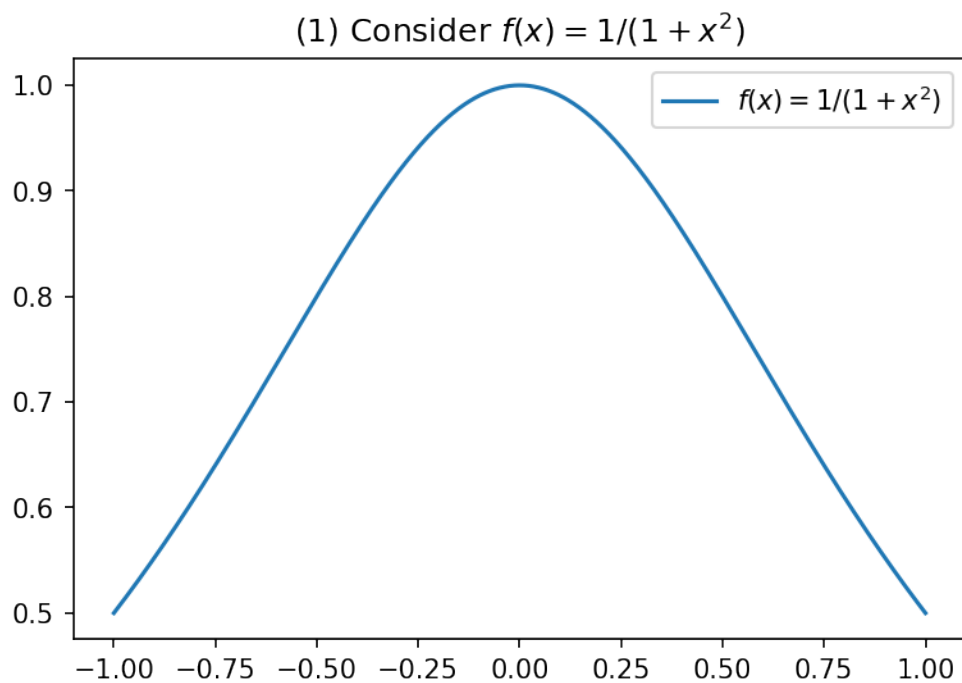
(2) 考虑  $f(x) = e^x$  in  $[-5, 5]$

	5	10	20
0.75	2.3740	2.1171	2.1170
1.75	4.8716	5.7544	5.7546
2.75	15.0081	15.6432	15.6426
3.75	45.8623	42.5184	42.5211
4.75	119.6210	115.6074	115.5843

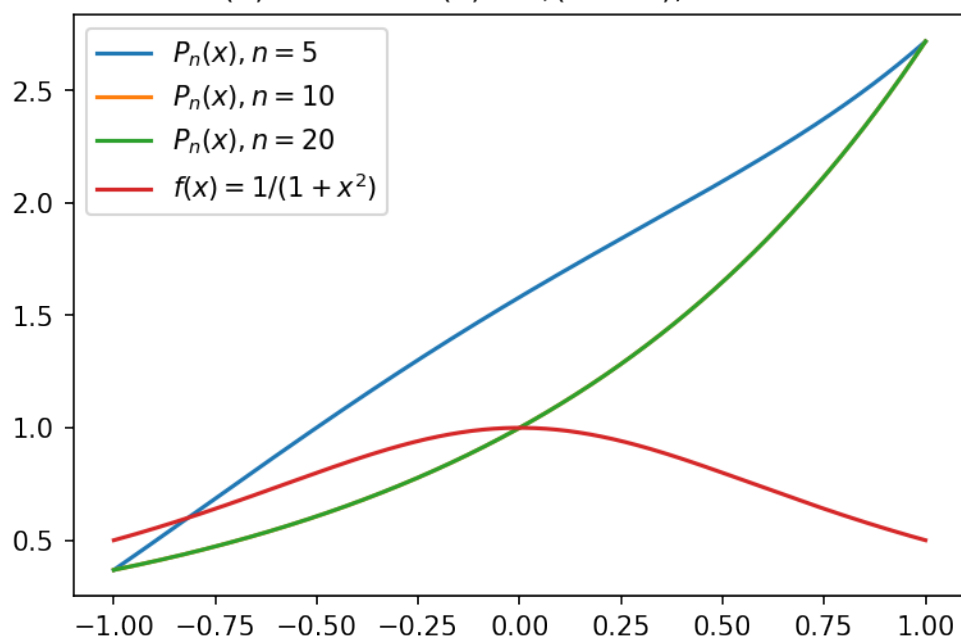
差值:

	5	10	20
0.75	-0.2570	-0.0001	-0.0
1.75	0.8830	0.0002	-0.0
2.75	0.6346	-0.0006	-0.0
3.75	-3.3412	0.0027	-0.0
4.75	-4.0367	-0.0231	-0.0

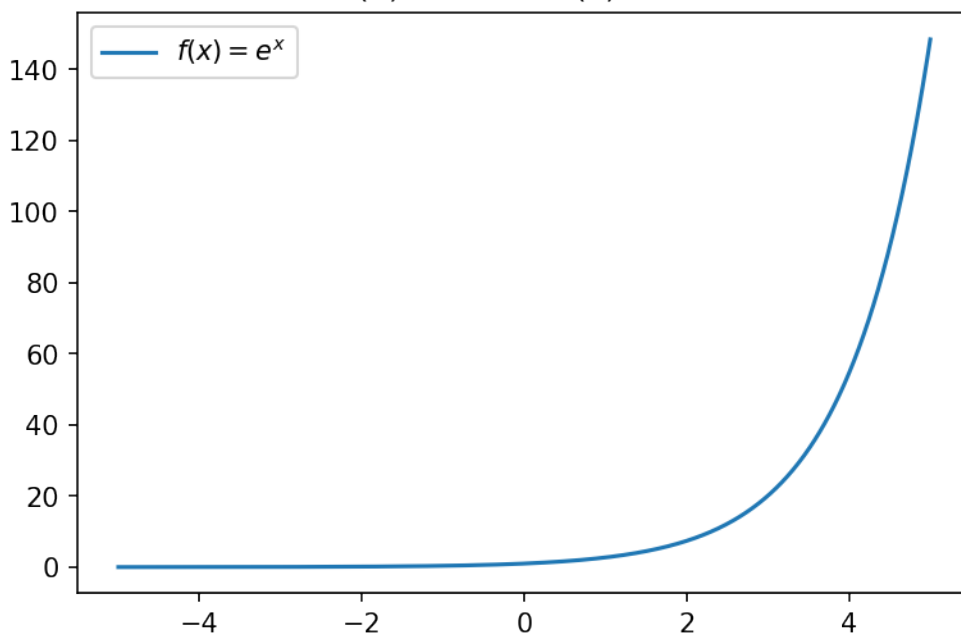
画出两个函数以及其拉格朗日多项式的图像:



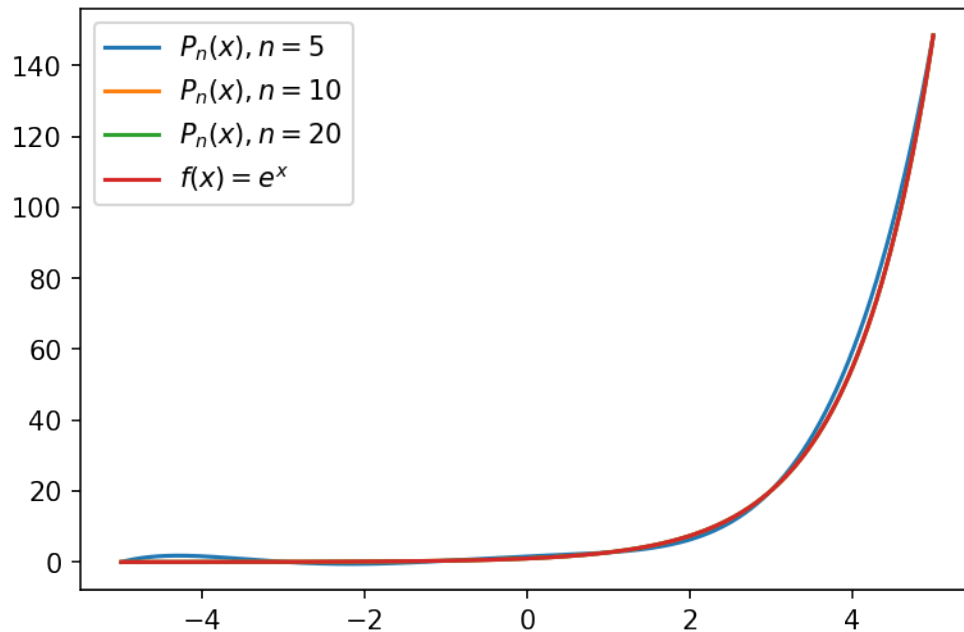
(1) Consider  $f(x) = 1/(1 + x^2)$ , select  $n$



(2) Consider  $f(x) = e^x$



(2) Consider  $f(x) = e^x$ , select  $n$



### 1.3.3 问题四

```
[15]: def problem4():
    print("问题 4: 考虑拉格朗日插值问题, 内插比外推更可靠吗? ")

    def f(x: float):
        return np.sqrt(x)

    target_x = [5, 50, 115, 185]

    print("考虑函数  $f(x) = \sqrt{x}$ ")

    def do_by_node(index: int, nodes: list):
        print(f"({index}) 考虑以", ", ".join(
            [f"x_{i} = {nodes[i]}" for i in range(3)]), "为节点的拉格朗日插值多项式  $P_2(x)$ ")
        print(DataFrame({
            "函数值": {x: P(f, nodes, x) for x in target_x},
```

```

        "f(x)": {x: f(x) for x in target_x},
        "差值": {x: f(x) - P(f, nodes, x) for x in target_x}
    }).round(4))
mean = np.mean([f(x) - P(f, nodes, x) for x in target_x])
print(f"平均差值: {mean:.4f}")
return mean

nodes_data = [
    [(i + 1) ** 2 for i in range(3)],
    [(i + 5) ** 2 for i in range(3)],
    [(i + 9) ** 2 for i in range(3)],
    [(i + 12) ** 2 for i in range(3)]
]
means = [do_by_node(i + 1, nodes=nodes_data[i])
          for i in range(len(nodes_data))]
return DataFrame({
    "x_0,x_1,x_2": [",".join([str(n) for n in nodes_data[i]]) for i in
↪range(len(nodes_data))],
    "平均差值": [means[i] for i in range(len(nodes_data))]
}).round(4)

problem4()

```

问题 4: 考虑拉格朗日插值问题, 内插比外推更可靠吗?

考虑函数  $f(x) = \sqrt{x}$

(1) 考虑以  $x_0 = 1$ ,  $x_1 = 4$ ,  $x_2 = 9$  为节点的拉格朗日插值多项式  $P_2(x)$

	函数值	$f(x)$	差值
5	2.2667	2.2361	-0.0306
50	-20.2333	7.0711	27.3044
115	-171.9000	10.7238	182.6238
185	-492.7333	13.6015	506.3348

平均差值: 179.0581

(2) 考虑以  $x_0 = 25$ ,  $x_1 = 36$ ,  $x_2 = 49$  为节点的拉格朗日插值多项式  $P_2(x)$

	函数值	$f(x)$	差值
5	2.8205	2.2361	-0.5844

50	7.0688	7.0711	0.0023
115	9.0385	10.7238	1.6853
185	5.6527	13.6015	7.9488

平均差值: 2.2630

(3) 考虑以  $x_0 = 81$ ,  $x_1 = 100$ ,  $x_2 = 121$  为节点的拉格朗日插值多项式  $P_2(x)$

	函数值	$f(x)$	差值
5	4.0952	2.2361	-1.8592
50	7.1742	7.0711	-0.1031
115	10.7256	10.7238	-0.0018
185	13.3659	13.6015	0.2356

平均差值: -0.4321

(4) 考虑以  $x_0 = 144$ ,  $x_1 = 169$ ,  $x_2 = 196$  为节点的拉格朗日插值多项式  $P_2(x)$

	函数值	$f(x)$	差值
5	5.1411	2.2361	-2.9050
50	7.6026	7.0711	-0.5316
115	10.7508	10.7238	-0.0270
185	13.6026	13.6015	-0.0012

平均差值: -0.8662

[15]:

	$x_0, x_1, x_2$	平均差值
0	1, 4, 9	179.0581
1	25, 36, 49	2.2630
2	81, 100, 121	-0.4321
3	144, 169, 196	-0.8662

## 1.4 实验结果

准确规范地给出各个实验题目的结果, 并对相应的思考题给出正确合理的回答与说明。

由题目(1)代码、数据和图像可知:

1. 对  $f(x) = 1/(1+x^2)$  函数而言, 在  $[-5, 5]$  范围内, 并不是  $n$  越大越好,  $n$  越大反而误差增大。
2. 对  $f(x) = e^x$  函数而言, 在  $[-1, 1]$  范围内,  $n$  越大拟合效果越好。

所以不是  $n$  越大越好, 需要结合具体函数考虑。

由题目(2)代码、数据和图像, 并且结合题目(1)的数据可知:

1. 对  $f(x) = 1/(1+x^2)$  函数而言,  $[-1, 1]$  差值区间效果要比  $[-5, 5]$  好。
2. 对  $f(x) = e^x$  函数而言,  $[-5, 5]$  差值区间效果要比  $[-1, 1]$  好。

所以不是差值区间越小越好，需要结合具体函数考虑。

由题目（4）代码、数据和图像，对函数  $f(x) = \sqrt{x}$ ，内插确实比外推可靠。

### 思考题

对问题一存在的问题，应该如何解决？

问题一中， $f(x) = \frac{1}{1+x^2}$  在  $[-5, 5]$  的差值区间、 $n \in \{10, 20\}$  的情况下拟合效果并不好， $n = 10, n = 20$  的时候多项式在  $x$  较大的时候明显偏大。

由实验数据可知不应选择过大的插值多项式次数， $n$  应该  $< 10$ 。

对问题二中存在的问题的回答，试加以说明。

插值区间不是越小越好，如这两个函数： $f(x) = \frac{1}{1+x^2}$  和  $f(x) = e^x$ ，前者在  $[-1, 1]$  上插值效果较好而在  $[-5, 5]$  上效果不好；后者在  $[-1, 1]$  上效果不好而在  $[-5, 5]$  上效果较好。

如何理解插值问题中的内插和外推？

内插即只对已知数据集内部范围的点的插值运算，外推即对已知数据集外部范围的点进行插值运算。

内插运算比外推更可靠，偏差更小的原因是内插能够更加有效地利用已知数据集的限制条件，尽量利用已知的信息进行计算推测，故更加可靠。