数值实验报告

实验名称	高斯消去法与三角分解法的实现				实验时间	2025 年 4 月 26 日	
姓名	秦浩政 郭凯平 刘桂凡 刘佳鑫	班级	数据科学 2301	学号	2306030214 2306020510 2309050116 2309050117	成绩	

一、实验目的，内容

实验 4.3 多项式插值的 Runge 现象与分段低次插值

目的：

1. 观察 Runge 现象：通过高次多项式插值在等距节点下的表现，验证 Runge 现象的存在及特征。
2. 比较不同节点分布：分析等距节点与切比雪夫节点对插值误差的影响，理解节点选择的重要性。
3. 掌握分段低次插值：实现分段线性插值方法，对比其与全局多项式插值在稳定性和精度上的差异。
4. 评估模型适用性：通过不同函数的插值实验，总结不同插值方法的适用范围。

内容：

1. 求出 f1(x) = 1/(1+25x^2)在区间[-1, 1]上的以 xi = -1+2i/n, i = 0, 1, 2, …, n 为插值节点的插值多项式 p(x)，依次取 n = 2, 3, 4,…, 绘制两个函数的图像，比较分析实验结果。
2. 选择其他函数 f2(x) = x/(1+x^4)，x 区间为[-5, 5]，f3(x) = sin(Πx)，x 的区间为[-1, 1]。
3. 以 x1, x2, …, xn+1 为插值节点构造上述各函数的切比雪夫插值多项式，并绘制图形观察插值效果。
4. 分别绘制分段线性插值多项式 p(x)的图像，观察插值效果。

实验 5.2 最小二乘拟合问题

目的：

1. 掌握最小二乘法的基本原理：通过实际数据拟合问题，理解最小二乘法的数学原理及其在回归分析中的应用。
2. 探索不同模型的拟合效果：分别使用三次多项式和指数模型对数据进行拟合，比较不同数学模型的适用性。

内容：

1. 绘制 y 关于 x 的图像。
2. 用三次多项式拟合数据。
3. 用 y = ae^(bx)拟合数据。
4. 比较两种模型的拟合效果。

二、算法描述

1.拉格朗日插值算法

给定 n+1 个节点(xi, yi)，构造 n 次多项式 li(x) = [(x-x0)(x-x1)…(x-x(i-1))(x-x(i+1))…(x-xn)][(xi-x0)(xi-x1)…(xi-x(i-1))(xi-x(i+1))…(xi-xn)]，再令 f(xi)分别与 li 相乘累计求和得 n 次插值多项式 p(x)。

2.切比雪夫节点生成

利用区间[a, b]上切比雪夫点的定义 xk = (b+a)/2+[(b-a)/2]cos[(2k-1)Π/(2n+2)], k = 1, 2, …, n+1，构造切比雪夫插值节点。

3.分段线性插值算法

4. 最小二乘法拟合 n 次多项式系数

（1）构建正规方程组的系数矩阵：初始化一个 (n+1)×(n+1)(*n*+1)×(*n*+1) 的零矩阵 A，对矩阵的每个元素 A[i][j]，计算所有样本点的 x(i+j) 次方和。（2）构建正规方程组的右侧向量：初始化一个长度为 n+1 的零向量 b，对向量的每个元素 b[i]*b*[*i*]，计算所有样本点的 xi*y 的累加和.(3) 求解线性方程组：解方程 Aa=b，其中 a=[a0,a1,…,an]T 为多项式系数向量, 使用 **np.linalg.solve** 直接求解，得到最小二乘意义下的最优系数。

三．程序代码

1.多项式插值的 Runge 现象

```python
import numpy as np
import matplotlib.pyplot as plt
# 设置中文字体
plt.rcParams['font.sans-serif'] = ['SimHei']  # 黑体
# 解决负号显示异常
plt.rcParams['axes.unicode_minus'] = False
# 目标函数
def f1(x):
    return 1 / (1 + 25 * x ** 2)
1 个用法
def f2(x):
    return x / (1 + x**4)
1 个用法
def f3(x):
    return np.sin(np.pi * x)
# 生成等距节点
1 个用法
def generate(a, b, n):
    return np.linspace(a, b, n + 1)
# 拉格朗日插值实现
1 个用法
def lagrange_interp(x, x_list, y_list):
    n = len(x_list)
    result = np.zeros_like(x)
    for i in range(n):
        z = np.ones_like(x)
        for j in range(n):
            if i != j:
                z *= (x - x_list[j]) / (x_list[i] - x_list[j])
        result += y_list[i] * z
    return result
3 个用法
```

```python
def run(a, b, func, n_list, label):
    x = np.linspace(a, b,  num: 1000)
    for n in n_list:
        # 生成节点和函数值
        x_list = generate(a, b, n)
        y_list = func(x_list)

        # 计算插值多项式
        p = lagrange_interp(x, x_list, y_list)

        # 绘制图像
        plt.figure(figsize=(8, 5))
        plt.plot( *args: x, func(x), label=label, linewidth=2)
        plt.plot( *args: x, p, '--', label=f'p(x) n={n}')
        plt.scatter(x_list, y_list, color='red', zorder=5)
        plt.title(f'Runge现象(n={n})')
        plt.xlabel('x'), plt.ylabel('y')
        plt.legend()
        plt.grid(True)
        plt.ylim( *args: -0.5, 1.5)
        plt.show()

if __name__ == "__main__":
    # 实验参数设置
    n_list = [2, 3, 4, 5, 10, 15]
    run(-1,  b: 1, f1, n_list,  label: "f1(x)")
    run(-5,  b: 5, f2, n_list,  label: "f2(x)")
    run(-1,  b: 1, f3, n_list,  label: "f3(x)")
```

2.切比雪夫插值多项式

```python
def generate(a, b, n, type):
    if type == 'chebyshev':
        k = np.arange(1, n + 2)
        return (a + b) / 2 + (b - a) / 2 * np.cos((2 * k - 1) * np.pi / (2 * (n + 1)))
    else:
        return np.linspace(a, b, n + 1)
# 拉格朗日插值实现
2 个用法
def lagrange_interp(x, x_list, y_list):
    n = len(x_list)
    result = np.zeros_like(x)
    for i in range(n):
        z = np.ones_like(x)
        for j in range(n):
            if i != j:
                z *= (x - x_list[j]) / (x_list[i] - x_list[j])
        result += y_list[i] * z
    return result
3 个用法
def run(a, b, func, n_list, label):
    x = np.linspace(a, b, num: 1000)
    for n in n_list:
        # 生成节点和函数值
        x1_list = generate(a, b, n, type: 'on')
        x2_list = generate(a, b, n, type: 'chebyshev')
        y1_list = func(x1_list)
        y2_list = func(x2_list)
        # 计算插值多项式
        p1 = lagrange_interp(x, x1_list, y1_list)
        p2 = lagrange_interp(x, x2_list, y2_list)
```

```python
        # 绘制对比图
        plt.figure(figsize=(8, 5))
        # 原函数
        plt.plot( *args: x, func(x), 'k-', lw=2, label=label)
        # 等距插值
        plt.plot( *args: x, p1, 'r--', lw=1.5, label='等距')
        plt.scatter(x1_list, y1_list, c='red', s=40, zorder=5, marker='o')
        # 切比雪夫插值
        plt.plot( *args: x, p2, 'b-.', lw=1.5, label='切比雪夫')
        plt.scatter(x2_list, y2_list, c='blue', s=40, zorder=5, marker='s')
        # 图表装饰
        plt.title(f'节点对比 (n={n})')
        plt.xlabel('x'), plt.ylabel('y')
        plt.legend()
        plt.grid(True)
        plt.xlim( *args: a - 0.1 * (b - a), b + 0.1 * (b - a))
        plt.ylim( *args: min(func(x)) - 0.5, max(func(x)) + 0.5)
        plt.tight_layout()
        plt.show()
if __name__ == "__main__":
    # 实验参数设置
    n_list = [2, 3, 4, 5, 10, 15]
    run(-1, b: 1, f1, n_list, label: "f1(x)")
    run(-5, b: 5, f2, n_list, label: "f2(x)")
    run(-1, b: 1, f3, n_list, label: "f3(x)")
```

3.分段线性插值多项式

```python
# 分段线性插值实现
1 个用法
def fenduan_interp(x, x_list, y_list):
    # 确保节点排序
    idx = np.argsort(x_list)
    x_1 = x_list[idx]
    y_1 = y_list[idx]
    # 找到每个x_eval对应的区间索引
    i = np.searchsorted(x_1, x) - 1
    # 处理边界情况
    i = np.clip(i, a_min: 0, len(x_1) - 2)
    # 获取相邻节点
    x0 = x_1[i]
    x1 = x_1[i + 1]
    y0 = y_1[i]
    y1 = y_1[i + 1]
    # 计算线性插值
    s = (y1 - y0) / (x1 - x0)
    return y0 + s * (x - x0)
3 个用法
def run(a, b, func, n_list, label):
    x = np.linspace(a, b, num: 1000)
    for n in n_list:
        # 生成节点和函数值
        x_list = generate(a, b, n)
        y_list = func(x_list)
        # 计算分段线性插值多项式
        p = fenduan_interp(x, x_list, y_list)
```

```python
        # 绘制图像
        plt.figure(figsize=(8, 5))
        plt.plot( *args: x, func(x), label=label, linewidth=2)
        plt.plot( *args: x, p, '--', label=f'p(x) n={n}')
        plt.scatter(x_list, y_list, color='red', zorder=5)
        plt.title(f'分段插值多项式(n={n})')
        plt.xlabel('x'), plt.ylabel('y')
        plt.legend()
        plt.grid(True)
        plt.ylim( *args: -0.5, 1.5)
        plt.show()

if __name__ == "__main__":
    # 实验参数设置
    n_list = [2, 3, 4, 5, 10, 15]
    run(-1, b: 1, f1, n_list, label: "f1(x)")
    run(-5, b: 5, f2, n_list, label: "f2(x)")
    run(-1, b: 1, f3, n_list, label: "f3(x)")
```

3. 最小二乘拟合

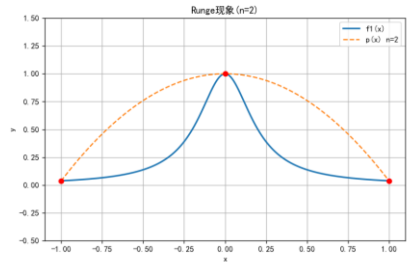
```
import numpy as np
from matplotlib import pyplot as plt
# 设置字体支持中文
plt.rcParams['font.sans-serif'] = ['SimHei']  # 或使用其他支持中文的字体
plt.rcParams['axes.unicode_minus'] = False  # 解决负号显示问题
2 个用法
def minsquare(x, y, n):
    # 构建法方程组的系数矩阵A
    A = np.zeros((n+ 1, n + 1))
    for i in range(n + 1):
        for j in range(n + 1):
            A[i, j] = np.sum(x ** (i + j))
    # 构建法方程组的右侧向量b
    b = np.zeros(n + 1)
    for i in range(n + 1):
        b[i] = np.sum(x ** i * y)

    # 求解法方程组得到多项式的系数
    a = np.linalg.solve(A, b)
    return a
1 个用法
def nihe(a,x):
    i=0
    s=0
    for j in a:
        s+=j*(x**i)
        i=i+1
    return s
1 个用法
def zhishu(a,x):
    return a[0]*np.exp(a[1]*x)
```

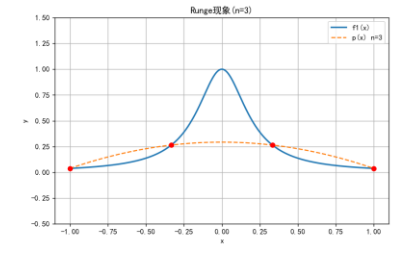
```
def plot_xy(x, y,x2,y2,x3,y3):
    # 创建图形
    plt.figure(figsize=(8, 6))
    # 绘制函数关系图
    plt.plot( *args: x, y, marker='.',linestyle='-', color='b',label='原关系')
    # 绘制多项式拟合函数关系图
    plt.plot( *args: x2, y2, linestyle='-', color='r', label='多项式拟合关系')
    #绘制指数拟合函数关系图
    plt.plot( *args: x3, y3, linestyle='-', color='k', label='指数拟合关系')
    # 添加标题和标签
    plt.title( label: 'x和y的关系图', fontsize=14)
    plt.xlabel( xlabel: 'x', fontsize=12)
    plt.ylabel( ylabel: 'y', fontsize=12)
    # 添加网格和图例
    plt.grid( visible: True, linestyle='--', alpha=0.6)
    plt.legend()
    # 显示图形
    plt.show()
x=[1,2,3,4,5,6,7,8,9,10]
y=[2615,1943,1494,1087,765,538,484,290,226,204]
#求解多项式拟合关系系数
n1=3
a1 = minsquare(np.array(x), np.array(y), n1)
x2 = np.linspace(min(x), max(x),  num: 1000)
y2 = np.array([nihe(a1, xi) for xi in x2])
#求解指数拟合关系系数
```

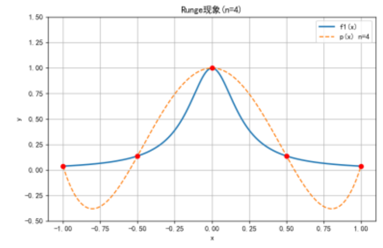
```
#求解指数拟合关系系数
# 计算自然对数
log_y = np.log(np.array(y))
n2=1
a2 = minsquare(np.array(x), np.array(log_y), n2)
a3 =[np.exp(a2[0]),a2[1]]
x3 =x2
y3 = np.array([zhishu(a3, xi) for xi in x3])
#绘制图像
plot_xy(x,y,x2,y2,x3,y3)
print("三次拟合的各项系数[a0 a1 a2 a3]:",list(a1))
print("指数拟合的各项系数: ",list(a3))
```

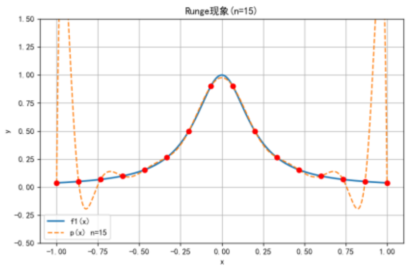
四、数值结果

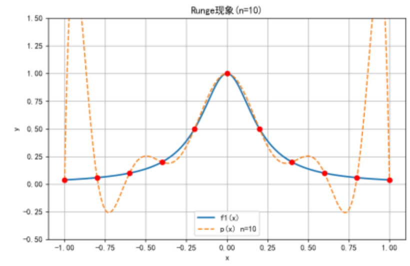
1. f1(x),f2(x),f3(x)以 xi = -1+2i/n 为插值节点构造的拉格朗日插值多项式

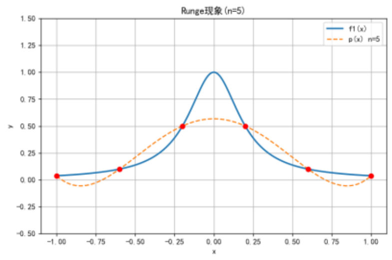


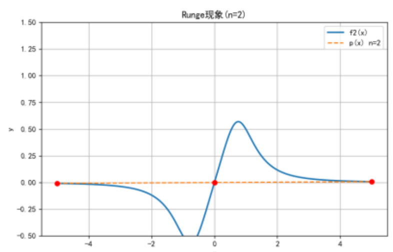


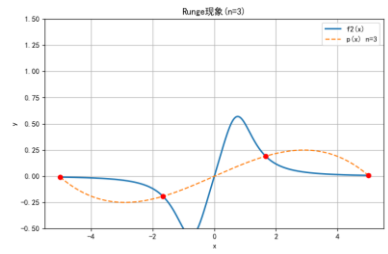


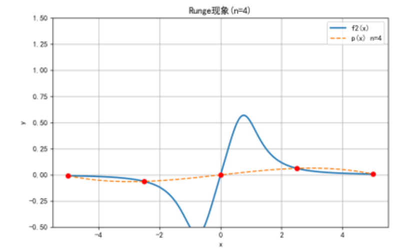


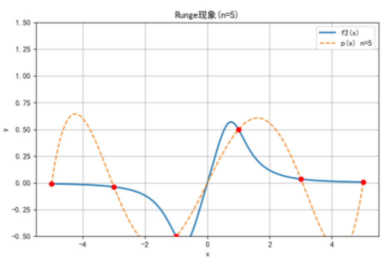


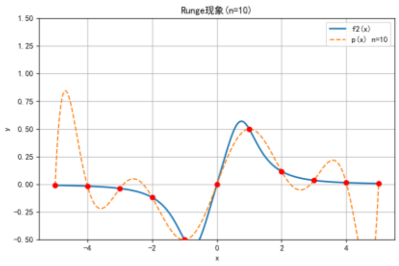


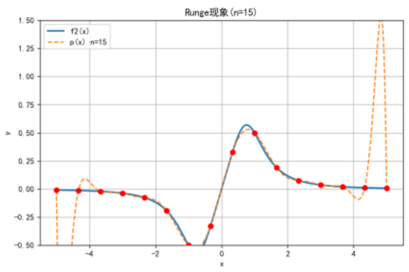


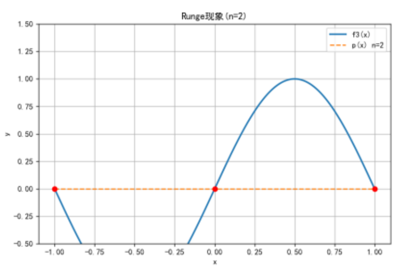


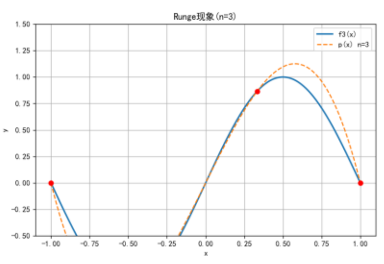


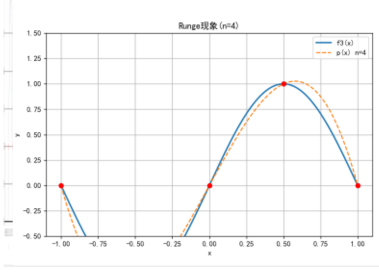


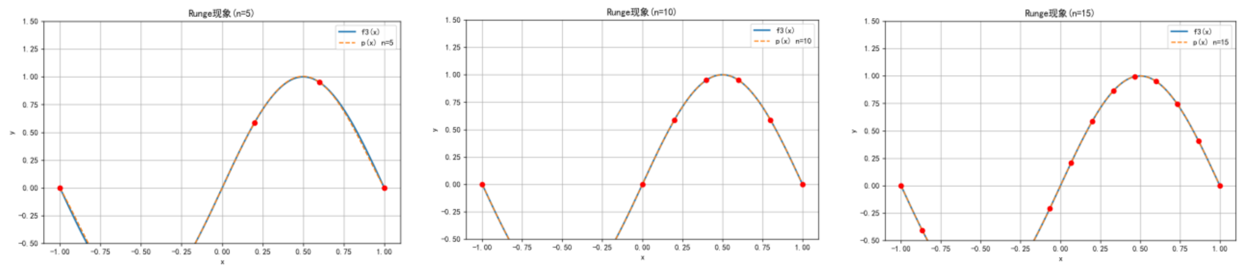




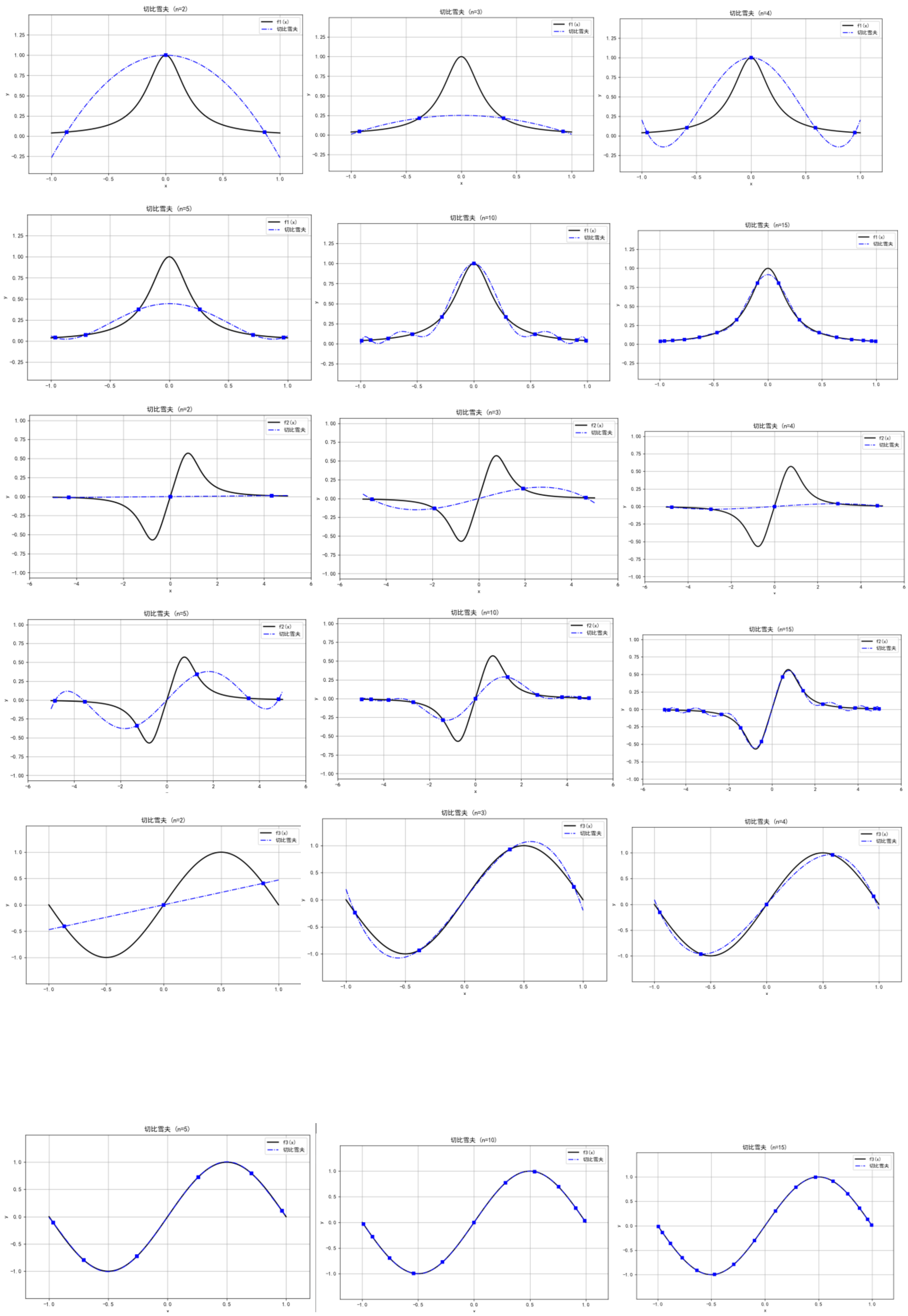




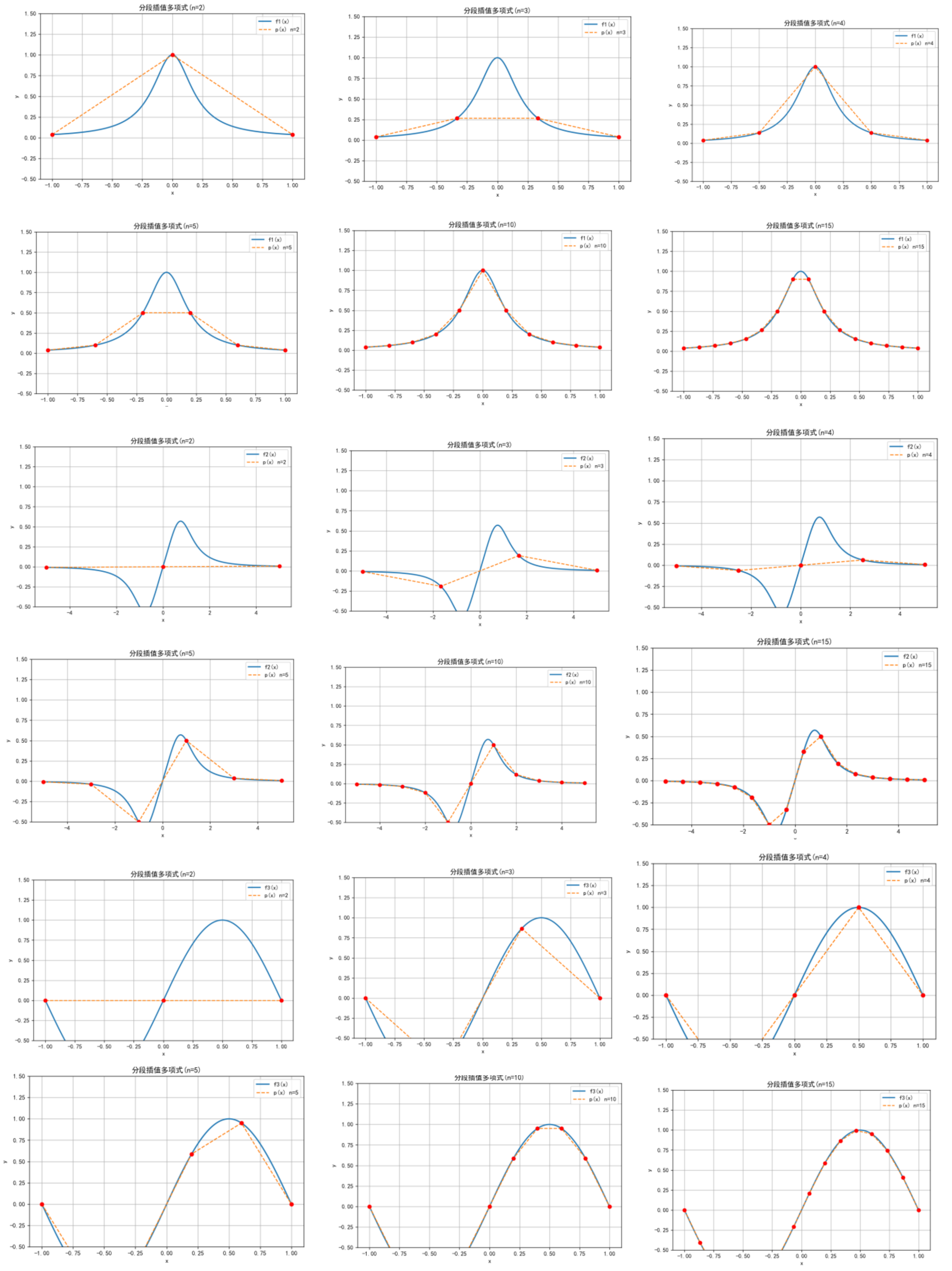




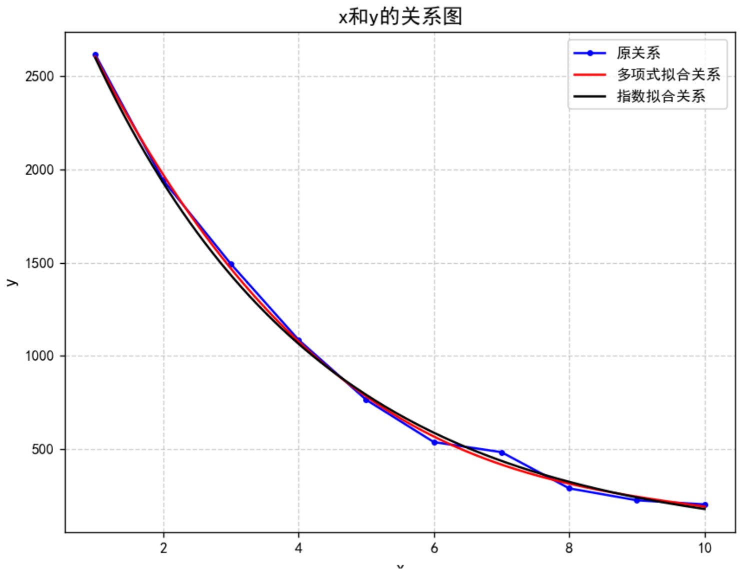
2.切比雪夫插值多项式



3.分段线性插值多项式


分段插值多项式(n=2)
分段插值多项式(n=3)
分段插值多项式(n=4)
分段插值多项式(n=5)
分段插值多项式(n=10)
分段插值多项式(n=15)
分段插值多项式(n=2)
分段插值多项式(n=3)
分段插值多项式(n=4)
分段插值多项式(n=5)
分段插值多项式(n=10)
分段插值多项式(n=15)
分段插值多项式(n=2)
分段插值多项式(n=3)
分段插值多项式(n=4)
分段插值多项式(n=5)
分段插值多项式(n=10)
分段插值多项式(n=15)

4. 最小二乘拟合




```
三次拟合的各项系数[a0 a1 a2 a3]: [3380.100000000201, -852.8323620825203, 79.95221445224608, -2.654817404819217]
指数拟合的各项系数,[a b]: [3495.0950966915193, -0.2969346607458727]
```

五、计算结果分析

1.等距插值多项式：Runge 现象显著，高次多项式插值在等距节点下具有不稳定性。

2.切比雪夫插值多项式：震荡抑制效果显著，相同节点数下，切比雪夫节点插值的最大误差下降明显，误差分布更均匀。

3.分段插值多项式：无震荡但精度有限。

六、计算中出现的问题、解决方法及体会

1.数值不稳定问题

2.切比雪夫节点排序问题生成的切比雪夫节点按余弦值降序排列，导致插值时索引混乱。

解决方法：显式调用 **np.sort** 对节点排序，确保 **x** 递增。

体会：

1. 通过实验直观验证了 Runge 现象和切比雪夫节点的理论优势，加深了对插值余项公式的理解。

2. 高次插值虽理论精度高，但受限于数值稳定性；分段插值牺牲光滑性换取稳定性，适合不同场景。

3. 模型选择的重要性，指数模型因符合物理意义（价格持续衰减）表现更优，强调建模时需结合实际问题背景。

教 师 评 语	指导教师：　　　　　　　年　月　日