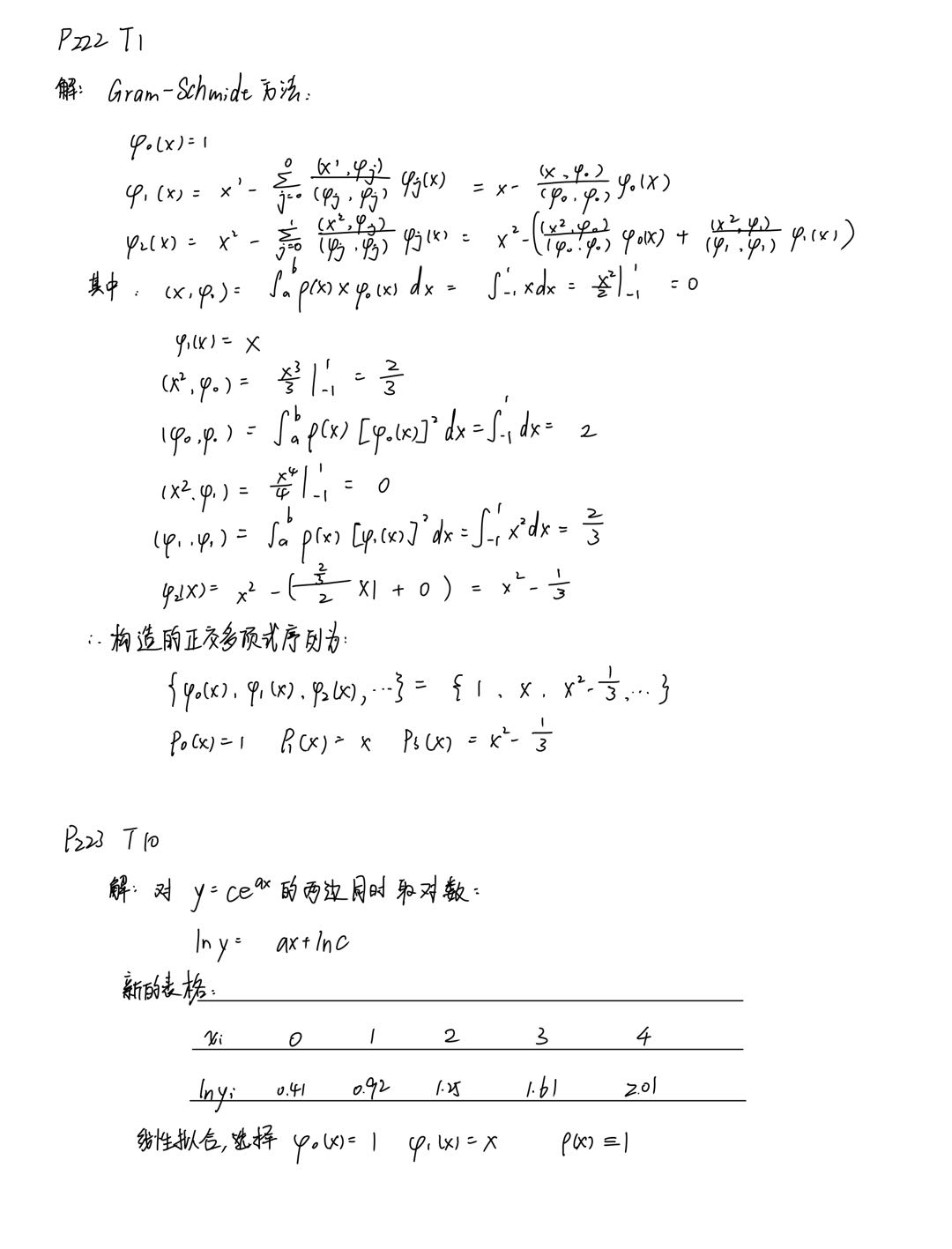
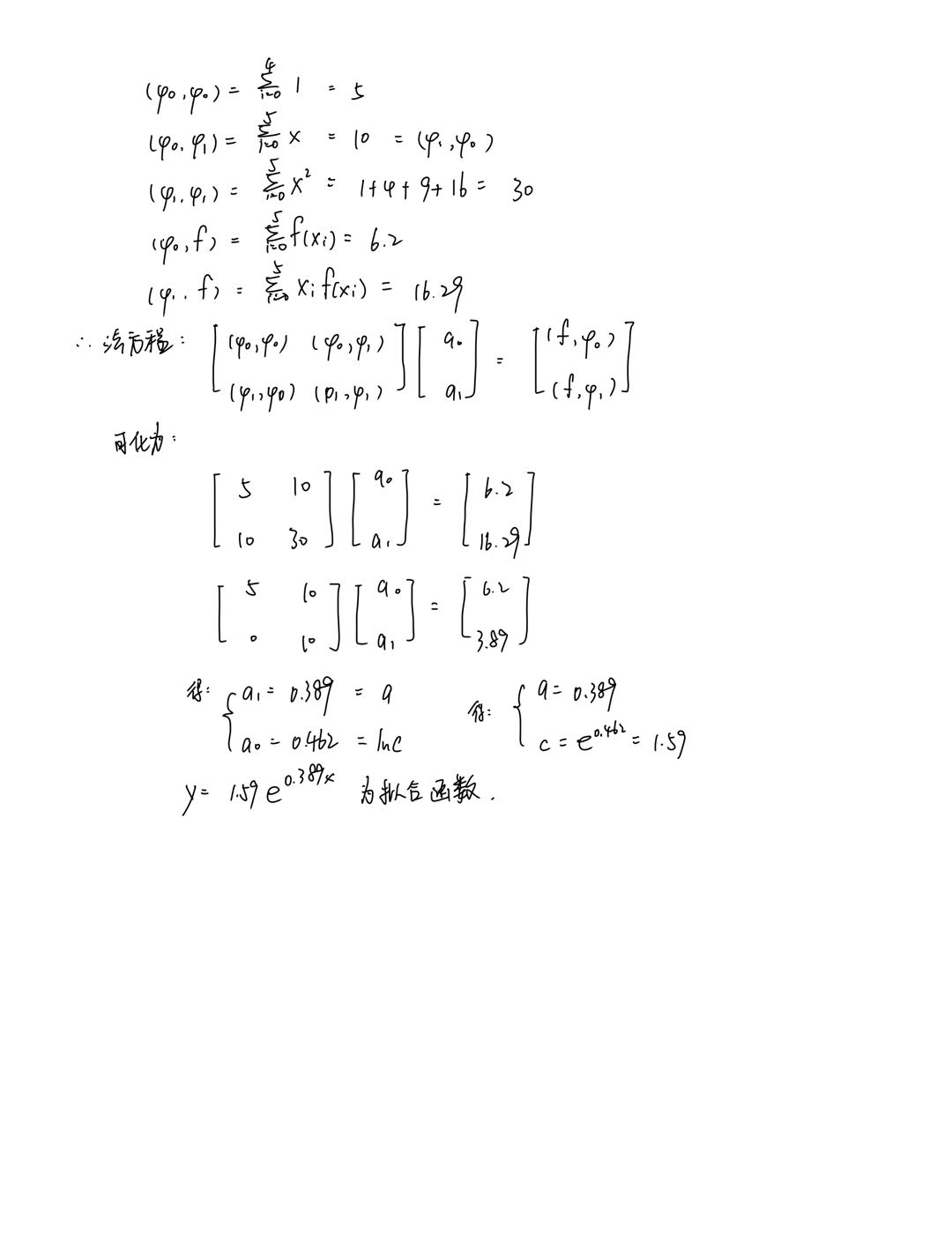
第七次作业

赖显松 2021214726

1 书面作业题目





2 编程题目

2.1 课堂任务1

**线性无关多项式拟合函数：**

用最基本的多项式序列构造拟合函数（输入：*x*序列，对应的函数值序列fx，求解所致最高阶次order；输出：多项式系数序列a，误差err）。

求解步骤为：根据离散逼近的公式构造法方程，解出法方程，得到系数序列，同时根据拟合结果与采样点的结果计算出拟合误差。

代码如下：

1. **def** myLeastSq(x, fx, order):
2. A = np.empty((order+1, order+1))
3. **for** i **in** range(order+1):
4. **for** j **in** range(order+1):
5. A[i, j] = sum(x\*\*(i+j))
7. b = np.empty((order+1))
8. **for** i **in** range(order+1):
9. b[i] = sum((x\*\*i) \* fx)
11. a = np.linalg.solve(A, b)
13. # 计算拟合值
14. sx = x.copy()
15. **for** i **in** range(len(sx)):
16. sx[i] = fitFunction(x[i],a)
18. # 计算误差
19. err = 0
20. **for** i **in** range(len(fx)):
21. err = (sum((fx-sx)\*\*2))\*\*(0.5)
23. **return** a, err

其中，根据多项式系数序列得到对应的拟合值的函数代码如下：

1. **def** fitFunction(x, a):
2. s = 0
3. **for** i **in** range(len(a)):
4. s += (x\*\*i)\*a[i]
5. **return** s

**不同阶次拟合并分析误差：**

通过循环采用不同阶次（1至8）的线性无关多项式进行拟合，画出拟合图线并得到拟合误差。

代码如下：

1. **for** i **in** range(1, 9):
2. order = i
3. a, err = fit.myLeastSq(x, fx, order)
5. xx = np.arange(0, 1, 0.01)
6. yy = np.arange(0, 1, 0.01)
7. **for** i **in** range(len(xx)):
8. yy[i] = fit.fitFunction(xx[i],a)
9. plt.plot(xx, yy)
10. plt.scatter(x,fx)
11. plt.title('%d阶线性无关多项式拟合' % (order))
12. plt.show()
14. **print**(order, "阶error：", err)

选取其中某些阶次的拟合效果（图2. 1）：

不同阶次拟合函数的误差如图2. 2所示。

从拟合的结果以及误差可发现，不同阶次的多项式对数据都进行了逼近，曲线的变化趋势与数据的大致变化趋势也是相符合的。一阶多项式直接通过线性方程进行拟合，拟合的效果不够好，考虑到原数据可能符合指数函数的变化规律，在实际用函数进行逼近时可以考虑先将数据线性化，再用线性函数进行逼近。从课堂上的讲解也可以看出，数据确实是符合指数函数变化的，指数函数线性化再用线性函数进行拟合是最优的。

从拟合函数计算得到的误差可以发现，多项式的阶次越高，随着阶次的增加，拟合的误差也在不断下降。然而，在拟合函数阶次达到6次时就已经出现了过拟合的情况，具体表现为：函数逼近的误差确实在减小，但是曲线过于追求在每个点处的观测值逼近，导致曲线整体较奇怪，并且在离观测值较远的地方变化趋势不太符合观测数据的变化情况。

因此，进行函数逼近时，并不是多项式的阶数越高越好，不是误差越小越好，具体还需要根据观测数据的可能模型进行分析，以免出现过拟合的情况。

**扰动法方程：**

对法方程施加微小扰动10-5，以6阶多项式为例，分析扰动前后的拟合效果。

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |
|  |  |
| (c) | (d) |

图2. 1 不同阶次的线性无关多项式拟合结果

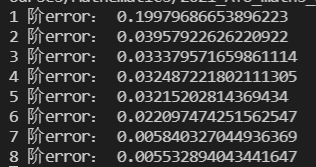


图2. 2 不同阶次的线性无关多项式拟合误差

扰动前后的6阶多项式拟合结果对比如图2. 3所示。

对比扰动前后的拟合效果，发现即使施加的是微小扰动，也会对拟合的结果产生较大的影响，特别是阶数高时，影响的效果更加显著。这与课堂上的内

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

图2. 3 6阶线性无关多项式扰动前后拟合效果

容相符，法方程是一个病态方程。

2.2 课堂任务2

**正交多项式作最小二乘拟合函数（包含选做\*卷积的思想进行系数展开）：**

用正交的多项式序列构造拟合函数（输入：*x*序列，对应的函数值序列fx，待拟合的数据点x\_c，求解所致最高阶次order；输出：拟合结果sx，对应的线性无关多项式的系数序列para）。

求解思路为正交多项式的迭代性质，先计算出0阶的多项式（1），根据公式求出α和β，再通过迭代公式得到各个阶次的多项式。不展开的情况下，在求解的时候同时检验待拟合的函数值。之后计算出正交多项式的拟合系数，最后构造多项式并通过系数及多项式加权求和，求出正交多项式的拟合函数值。代码如下：

1. **def** orthogonalPolyFit(x, fx, x\_c, order):
2. nk = x.size
3. nc = x\_c.size
4. phi = np.empty((order+1, nk))
5. phi\_f = np.empty((order+1, nc))
6. # 初始化展开系数矩阵
7. para\_phi = np.zeros((order+1, order+1))
8. # 0阶设置
9. para\_phi[0, 0] = 1 # 0阶系数
10. phi[0, :] = np.ones((1, nk))
11. phi\_f[0, :] = np.ones((1, nc))
12. a = np.empty((order+1,))
13. a[0] = np.sum(fx \* phi[0, :]) / np.sum(phi[0, :]\*\*2)
14. # 开始迭代
15. **for** i **in** range(1, order+1):
16. tmp = phi[i-1, :]\*\*2
17. alpha = np.sum(x \* tmp)  / np.sum(tmp)
18. phi[i, :] = (x-alpha) \* phi[i-1, :]
19. phi\_f[i, :] = (x\_c-alpha) \* phi\_f[i-1, :]
20. # 系数展开相关
21. pad\_para\_phi = np.pad(para\_phi[i-1, :], (1, 0))
22. kernal = np.array([1, -alpha]) # 定义卷积核
23. **for** j **in** range(i+1):
24. para\_phi[i, j] = np.sum(kernal \* pad\_para\_phi[j: j+2])
25. # 阶次大于1时要考虑beta项
26. **if** i > 1:
27. beta = np.sum(tmp) / np.sum(phi[i-2, :]\*\*2)
28. phi[i, :] -= beta \* phi[i-2, :]
29. phi\_f[i, :] -= beta \* phi\_f[i-2, :]
30. para\_phi[i, :] -= beta \* para\_phi[i-2, :]
31. # 计算正交系数序列
32. a[i] = np.sum(fx \* phi[i, :]) / np.sum(phi[i, :]\*\*2)
33. # 得到对应的线性无关多项式系数序列，用来验证和非线性相关的多项式系数是否一致
34. para = a @ para\_phi
35. # 计算拟合函数值
36. sx = np.zeros((nc,))
37. **for** i **in** range(nc):
38. **for** j **in** range(order+1):
39. sx[i] += a[j] \* phi\_f[j, i]
40. # 返回
41. **return** sx, para

**正交多项式拟合结果：**

同样的思路进行循环选择多项式拟合，拟合结果如图2. 4所示：

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

图2. 4 正交多项式拟合效果

拟合效果看起来与线性无关多项式的拟合结果基本一样，而实际上，通过对比同一阶次正交多项式求出的对应线性无关多项式阶次系数，可以发现两种方法得到的系数序列是相等的（图2. 5）。





图2. 5 正交多项式对应的线性关多项式系数对比

参考文献

1. 关治．数值方法[M]．北京：清华大学出版社，2006：66-92．

附录

见附件python源代码。