第七次作业

赖显松 2021214726

1 书面作业题目

P222 T1

解: Gram-Schmide 方法:

$\varphi_0(x)=1$

$\varphi_1(x) = x^1 - \sum_{j=0}^{0} \frac{(x^1,\varphi_j)}{(\varphi_j,\varphi_j)}\varphi_j(x) = x - \frac{(x,\varphi_0)}{(\varphi_0,\varphi_0)}\varphi_0(x)$

$\varphi_2(x) = x^2 - \sum_{j=0}^{1} \frac{(x^2,\varphi_j)}{(\varphi_j,\varphi_j)}\varphi_j(x) = x^2 - \left(\frac{(x^2,\varphi_0)}{(\varphi_0,\varphi_0)}\varphi_0(x) + \frac{(x^2,\varphi_1)}{(\varphi_1,\varphi_1)}\varphi_1(x)\right)$

其中: $(x,\varphi_0) = \int_a^b \rho(x)\,x\,\varphi_0(x)\,dx = \int_{-1}^{1} x\,dx = \frac{x^2}{2}\Big|_{-1}^{1} = 0$

$\varphi_1(x) = x$

$(x^2,\varphi_0) = \frac{x^3}{3}\Big|_{-1}^{1} = \frac{2}{3}$

$(\varphi_0,\varphi_0) = \int_a^b \rho(x)[\varphi_0(x)]^2dx = \int_{-1}^{1}dx = 2$

$(x^2,\varphi_1) = \frac{x^4}{4}\Big|_{-1}^{1} = 0$

$(\varphi_1,\varphi_1) = \int_a^b \rho(x)[\varphi_1(x)]^2dx = \int_{-1}^{1}x^2dx = \frac{2}{3}$

$\varphi_2(x) = x^2 - \left(\frac{\frac{2}{3}}{2}\times 1 + 0\right) = x^2 - \frac{1}{3}$

∴构造的正交多项式序列为:

$\{\varphi_0(x), \varphi_1(x), \varphi_2(x), \cdots\} = \{1, x, x^2-\frac{1}{3}, \cdots\}$

$P_0(x)=1 \quad P_1(x)=x \quad P_2(x) = x^2-\frac{1}{3}$

P223 T10

解: 对 $y=ce^{ax}$ 的两边同时取对数:

$\ln y = ax + \ln c$

新的表格:

x_i	0	1	2	3	4
$\ln y_i$	0.41	0.92	1.25	1.61	2.01

线性拟合,选择 $\varphi_0(x)=1 \quad \varphi_1(x)=x \qquad \rho(x)\equiv 1$

$$(\varphi_0,\varphi_0)=\sum_{i=0}^{4}1=5$$

$$(\varphi_0,\varphi_1)=\sum_{i=0}^{5}x=10=(\varphi_1,\varphi_0)$$

$$(\varphi_1,\varphi_1)=\sum_{i=0}^{5}x^2=1+4+9+16=30$$

$$(\varphi_0,f)=\sum_{i=0}^{5}f(x_i)=6.2$$

$$(\varphi_1,f)=\sum_{i=0}^{5}x_i f(x_i)=16.29$$

∴ 法方程：$\begin{bmatrix}(\varphi_0,\varphi_0) & (\varphi_0,\varphi_1)\\(\varphi_1,\varphi_0) & (\varphi_1,\varphi_1)\end{bmatrix}\begin{bmatrix}a_0\\a_1\end{bmatrix}=\begin{bmatrix}(f,\varphi_0)\\(f,\varphi_1)\end{bmatrix}$

可化为：

$$\begin{bmatrix}5 & 10\\10 & 30\end{bmatrix}\begin{bmatrix}a_0\\a_1\end{bmatrix}=\begin{bmatrix}6.2\\16.29\end{bmatrix}$$

$$\begin{bmatrix}5 & 10\\0 & 10\end{bmatrix}\begin{bmatrix}a_0\\a_1\end{bmatrix}=\begin{bmatrix}6.2\\3.89\end{bmatrix}$$

得：$\begin{cases}a_1=0.389=a\\a_0=0.462=\ln c\end{cases}$ 得：$\begin{cases}a=0.389\\c=e^{0.462}=1.59\end{cases}$

$y=1.59e^{0.389x}$ 为拟合函数.

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：

```
def myLeastSq(x, fx, order):
    A = np.empty((order+1, order+1))
    for i in range(order+1):
        for j in range(order+1):
            A[i, j] = sum(x**(i+j))

    b = np.empty((order+1))
    for i in range(order+1):
        b[i] = sum((x**i) * fx)

    a = np.linalg.solve(A, b)

    # 计算拟合值
    sx = x.copy()
    for i in range(len(sx)):
        sx[i] = fitFunction(x[i],a)

    # 计算误差
    err = 0
    for i in range(len(fx)):
        err = (sum((fx-sx)**2))**(0.5)

    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：

```
def fitFunction(x, a):
    s = 0
    for i in range(len(a)):
        s += (x**i)*a[i]
    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：

```
for i in range(1, 9):
    order = i
    a, err = fit.myLeastSq(x, fx, order)

    xx = np.arange(0, 1, 0.01)
    yy = np.arange(0, 1, 0.01)
    for i in range(len(xx)):
        yy[i] = fit.fitFunction(xx[i],a)
    plt.plot(xx, yy)
    plt.scatter(x,fx)
    plt.title('%d 阶线性无关多项式拟合' % (order))
    plt.show()

    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。

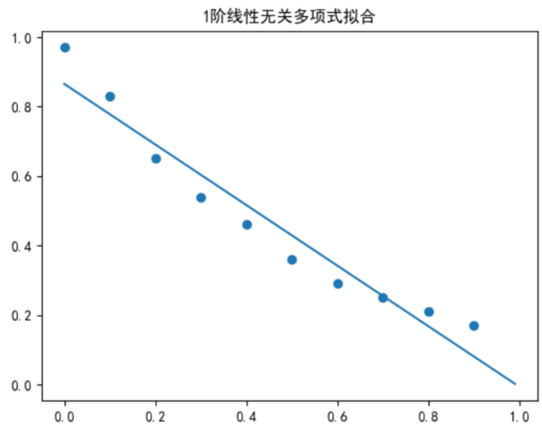
从拟合的结果以及误差可发现，不同阶次的多项式对数据都进行了逼近，曲线的变化趋势与数据的大致变化趋势也是相符合的。一阶多项式直接通过线性方程进行拟合，拟合的效果不够好，考虑到原数据可能符合指数函数的变化规律，在实际用函数进行逼近时可以考虑先将数据线性化，再用线性函数进行逼近。从课堂上的讲解也可以看出，数据确实是符合指数函数变化的，指数函数线性化再用线性函数进行拟合是最优的。

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。

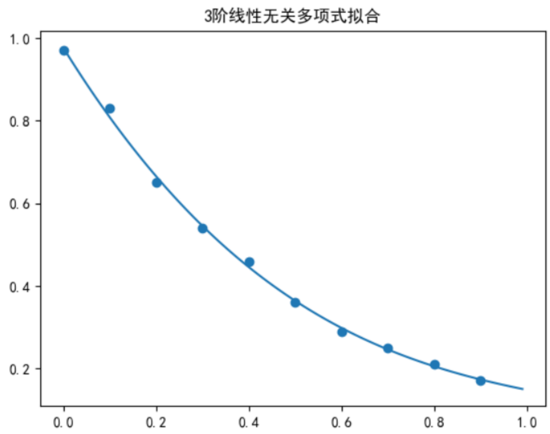
因此，进行函数逼近时，并不是多项式的阶数越高越好，不是误差越小越好，具体还需要根据观测数据的可能模型进行分析，以免出现过拟合的情况。

扰动法方程：

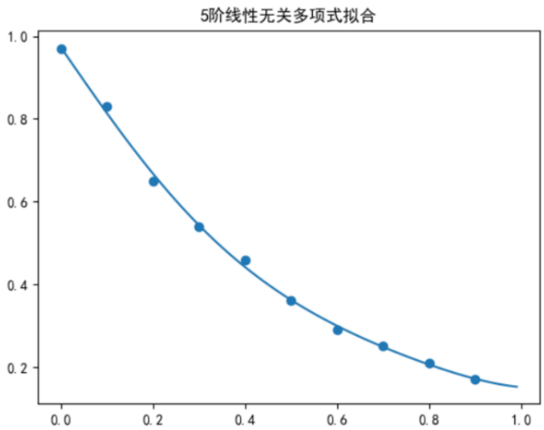
对法方程施加微小扰动 10^{-5}，以 6 阶多项式为例，分析扰动前后的拟合效果。



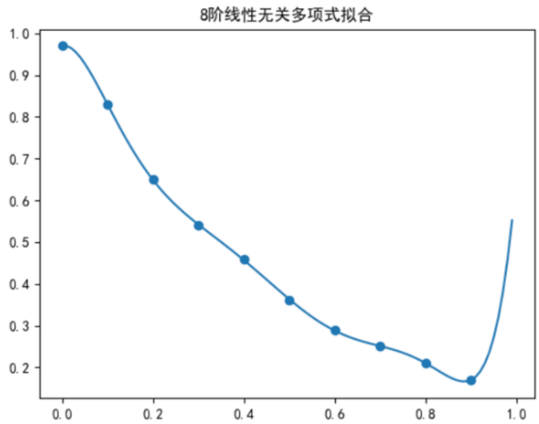
(a)



(b)



(c)



(d)

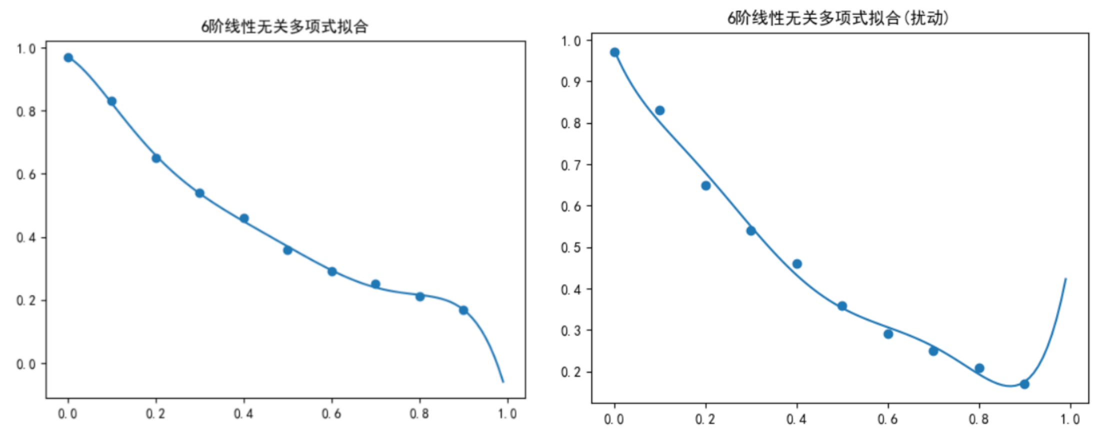
图 2. 1 不同阶次的线性无关多项式拟合结果

```
1 阶error:  0.19979686653896223
2 阶error:  0.03957922626220922
3 阶error:  0.03337957165986114
4 阶error:  0.032487221802111305
5 阶error:  0.03215202814369434
6 阶error:  0.022097474251562547
7 阶error:  0.005840327044936369
8 阶error:  0.005532894043441647
```

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）。

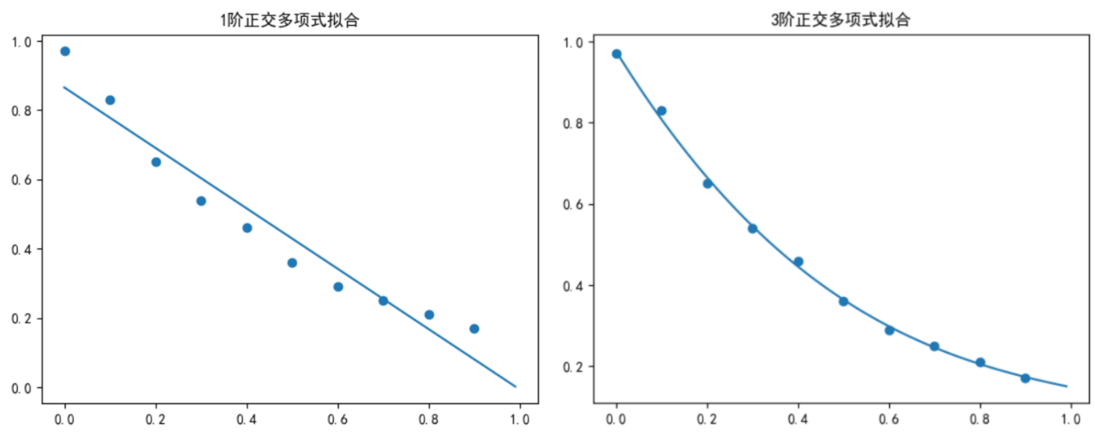
求解思路为正交多项式的迭代性质，先计算出 0 阶的多项式（1），根据公式求出 α 和 β，再通过迭代公式得到各个阶次的多项式。不展开的情况下，在求解的时候同时检验待拟合的函数值。之后计算出正交多项式的拟合系数，最后构造多项式并通过系数及多项式加权求和，求出正交多项式的拟合函数值。代码如下：

```python
def orthogonalPolyFit(x, fx, x_c, order):
    nk = x.size
    nc = x_c.size
    phi = np.empty((order+1, nk))
    phi_f = np.empty((order+1, nc))
    # 初始化展开系数矩阵
    para_phi = np.zeros((order+1, order+1))
    # 0 阶设置
    para_phi[0, 0] = 1 # 0 阶系数
    phi[0, :] = np.ones((1, nk))
    phi_f[0, :] = np.ones((1, nc))
    a = np.empty((order+1,))
    a[0] = np.sum(fx * phi[0, :]) / np.sum(phi[0, :]**2)
    # 开始迭代
```

```python
    for i in range(1, order+1):
        tmp = phi[i-1, :]**2
        alpha = np.sum(x * tmp)  / np.sum(tmp)
        phi[i, :] = (x-alpha) * phi[i-1, :]
        phi_f[i, :] = (x_c-alpha) * phi_f[i-1, :]
        # 系数展开相关
        pad_para_phi = np.pad(para_phi[i-1, :], (1, 0))
        kernal = np.array([1, -alpha]) # 定义卷积核
        for j in range(i+1):
            para_phi[i, j] = np.sum(kernal * pad_para_phi[j: j+2])
        # 阶次大于 1 时要考虑 beta 项
        if i > 1:
            beta = np.sum(tmp) / np.sum(phi[i-2, :]**2)
            phi[i, :] -= beta * phi[i-2, :]
            phi_f[i, :] -= beta * phi_f[i-2, :]
            para_phi[i, :] -= beta * para_phi[i-2, :]
        # 计算正交系数序列
        a[i] = np.sum(fx * phi[i, :]) / np.sum(phi[i, :]**2)
        # 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
        para = a @ para_phi
    # 计算拟合函数值
    sx = np.zeros((nc,))
    for i in range(nc):
        for j in range(order+1):
            sx[i] += a[j] * phi_f[j, i]
    # 返回
    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）。

```
> a: array([ 0.97737063, -1.81557887,  1.36888112, -0.38267288])
```

```
> para: array([ 0.97737063, -1.81557887,  1.36888112, -0.38267288])
```

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。