|  |  |
| --- | --- |
| 姓名 |  |
| 评分 |  |

《数学建模》实验报告

课题名称： 数学建模

专 业： 信息与计算科学

姓 名： 袁永润

班 级： 123212

完成日期： 2024 年 3 月 16 日

实验报告

## 一、实验名称

1.多项式拟合

## 二、 实验目的

1.理解多项式拟合的原理，学会使用曲线拟合的最小二乘法；

2.学会使用python内置函数来拟合多项式，并分析比较不同次数的多项式对某一函数的拟合效果，体会欠拟合、过拟合现象；

3.培养编程与上机调试能力；

4.熟悉Python软件环境。

## 三、 多项式拟合

3.1 问题的背景

给定一个多项式，对其进行多项式拟合，对所给数据添加干扰后再次进行拟合与原系数比较，比较结果，体会欠拟合、过拟合现象。

3.2 问题重述

假定多项式为，产生一组数据,再在上添加随机干扰，此时使用rand产生均匀分布随机数，然后用和添加了随机干扰的分别作1、2、3、4、6次多项式拟合，比较分析其结果，并将所作的3次多项式拟合与原系数进行比较。

3.3 模型假设

（1）设定数据为0到10之间等距离的点，间隔为0.5，为的计算值；

（2）设定干扰产生的随机数范围利用python中的rand函数。

3.4 模型的建立

将原始散点数据用多项式函数进行拟合，再通过最小二乘法求解多项式拟合函数的系数。

假设拟合函数为，为原函数，设误差平方和为：

使误差平方和最小的拟合称为曲线拟合的最小二乘法。

多项式拟合的原理为：

对于给定的一组数据拟合其一个次数不超过次的多项式

其拟合函数与原函数的误差平方和为最小：

即计算：

根据如下关于的线性方程组计算系数：

通过矩阵的运算计算出拟合多项式的系数:

分别计算矩阵：

和矩阵：

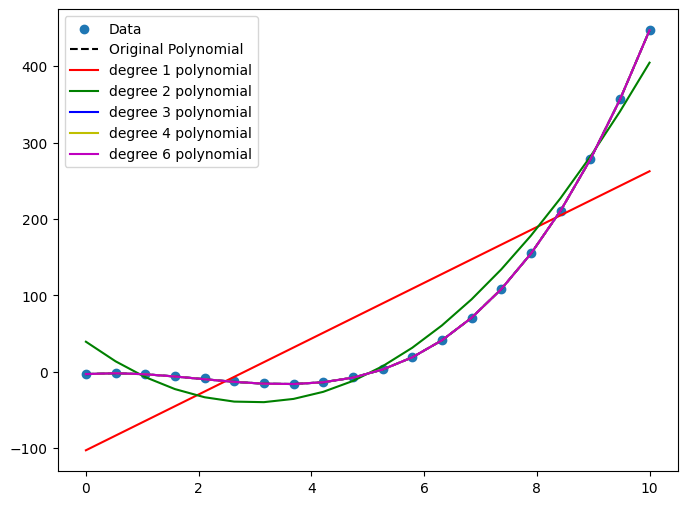
即可求得线性方程组的系数矩阵。

此处使用python中的polyfit函数来计算拟合多项式的系数，polyval计算拟合多项式的值。首先将拟合出的1、2、3、4、6次多项式拟合结果进行分析对比，再将干扰后的数值拟合多项式与原拟合的多项式进行对比。

3.5 结果与分析

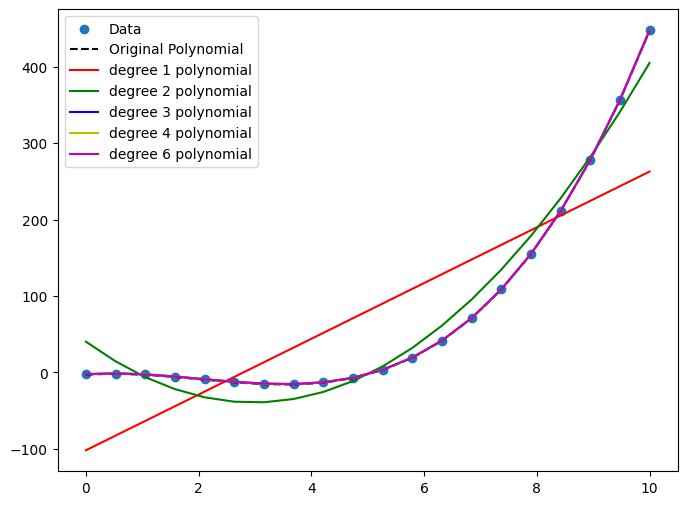
3.5.1 结果一：

绘制出未添加随机干扰的拟合多项式图像以及设定的数据点如下图1：



**图1 不添加随机干扰拟合多项式曲线与原数据点对比图**

绘制出添加随机干扰的拟合多项式图像以及设定的数据点如下图2：



**图2 添加随机干扰拟合多项式曲线与原数据点对比图**

3.5.2 结论一：

根据图像，拟合得到的一次和二次多项式数值与原数据点差距较大，拟合效果较差，拟合得到的三次、四次和六次多项式曲线对原数据点的逼近程度较好，拟合效果优于一次和二次的。原始数据添加干扰前后的拟合曲线变化不大，且在拟合的多项式为三次之后的拟合效果相差不大，都有较好的拟合效果。原函数为三次多项式，可以观察到一次和二次多项式为欠拟合的情况，而四次和六次多项式为过拟合的情况。

3.5.3 结果二：

在使用原数据点情况下，拟合1、2、3、4、6次多项式的系数与原系数对比如下表1，系数对应项的次数从低到高。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 原函数 | 一次多项式 | 二次多项式 | 三次多项式 | 四次多项式 | 六次多项式 |
|  | -3 | -102.7229 | 39.3822 | -3.0000 | -3.0000 | -3.0000 |
|  | 5 | 36.52 | -53.4764 | 5.0000 | 5.0000 | 5.0000 |
|  | -6 |  | 9.0000 | -6.0000 | -6.0000 | -6.0000 |
|  | 1 |  |  | 1.0000 | 1.0000 | 1.0000 |
|  |  |  |  |  | -0.0000 | -0.0000 |
|  |  |  |  |  |  | 0.0000 |
|  |  |  |  |  |  | -0.0000 |

**表1 拟合多项式系数与原系数对比**

根据计算结果，在对原数据添加随机干扰情况下，拟合1、2、3、4、6次多项式的系数与原系数对比如下表2，系数对应项的次数从低到高。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 原函数 | 一次多项式 | 二次多项式 | 三次多项式 | 四次多项式 | 六次多项式 |
|  | -3 | -102.2806 | 40.2194 | -2.5306 | -2.5306 | -2.5306 |
|  | 5 | 36.4500 | -53.5500 | 5.0000 | 5.0000 | 5.0000 |
|  | -6 |  | 9.0000 | -6.0000 | -6.0000 | -6.0000 |
|  | 1 |  |  | 1.0000 | 1.0000 | 1.0000 |
|  |  |  |  |  | -0.0000 | -0.0000 |
|  |  |  |  |  |  | 0.0000 |
|  |  |  |  |  |  | -0.0000 |

**表2 添加随机干扰拟合多项式系数与原系数对比**

3.5.4 结果二：

根据表格，无干扰情况下拟合得到的一次和二次多项式系数与原函数系数差距较大，拟合效果较差，拟合得到的三次、四次和六次多项式系数与原函数系数一致，拟合效果较好。

在对数据进行随机干扰的情况下，拟合得到的一次和二次多项式系数与原函数系数差距较大，三次、四次和六次多项式系数与原函数系数较为接近，且从三次及之后的多项式系数都相等，可以认为三次、四次、六次多项式拟合的效果较好。原函数为三次多项式，再结合图像显示的结果，可以得到一次和二次多项式为欠拟合的情况，而四次和六次多项式为过拟合的情况。

## 四、 总结

（1）对于离散的数据点，可以采用以上多项式模型进行拟合，通过最小二乘法的原理，可以找到最优的多项式次数，拟合次数不同的多项式其拟合效果也不同；

（2）找到最优多项式次数，在此之前次数的拟合多项式可能会出现欠拟合现象，之后的可能会出现过拟合现象，但并不是拟合的多项式次数越高其拟合效果越好，还可能出现龙格现象，在本例中在拟合20次多项式之后就出现了拟合效果变差的情况，如图3。



**图3 高次多项式拟合情况**

## 五、 附录

|  |
| --- |
| **1.拟合多项式** |
| （1）源代码 |
| import numpy as np  import matplotlib.pyplot as plt  # 生成数据  n = 20  np.random.seed()  x = np.linspace(0, 10, n)  y\_original = original\_polynomial(x)  y = original\_polynomial(x) + np.random.rand(n)  # 定义原始多项式  def original\_polynomial(x):  return x\*\*3 - 6\*x\*\*2 + 5\*x - 3  # 拟合多项式的系数  def return\_xishu(x,y,n):  return np.polyfit(x, y, n)  # 拟合多项式函数  def return\_coefficient(x,y,n):  coefficients = np.polyfit(x, y, n)  return np.poly1d(coefficients)  print("未加噪点的拟合系数:")  for i in [1,2,3,4,6]:  print(f'The coefficients of degree {i} polynomial are:', return\_xishu(x, y\_original, i))  # 绘图  c1 = return\_coefficient(x, y\_original, 1)  c2 = return\_coefficient(x, y\_original, 2)  c3 = return\_coefficient(x, y\_original, 3)  c4 = return\_coefficient(x, y\_original, 4)  c6 = return\_coefficient(x, y\_original, 6)  #未添加噪点的图像  plt.figure(figsize=(8, 6))  ax = plt.gca()  plt.scatter(x, y, label='Data')  plt.plot(x, original\_polynomial(x), label='Original Polynomial', color='black', linestyle='--')  plt.plot(x, c1(x), 'r', label='degree 1 polynomial')  plt.plot(x, c2(x), 'g', label='degree 2 polynomial')  plt.plot(x, c3(x), 'b', label='degree 3 polynomial')  plt.plot(x, c4(x), 'y', label='degree 4 polynomial')  plt.plot(x, c6(x), 'm', label='degree 6 polynomial')  plt.legend() |
| （2）结果 |
| 未加噪点的拟合系数:  The coefficients of degree 1 polynomial are: [ 36.52354571 -102.72299169]  The coefficients of degree 2 polynomial are: [ 9. -53.47645429 39.38227147]  The coefficients of degree 3 polynomial are: [ 1. -6. 5. -3.]  The coefficients of degree 4 polynomial are: [-3.43025193e-16 1.00000000e+00 -6.00000000e+00 5.00000000e+00  -3.00000000e+00]  The coefficients of degree 6 polynomial are: [ 4.62384683e-17 -1.46618040e-15 1.68028964e-14 1.00000000e+00  -6.00000000e+00 5.00000000e+00 -3.00000000e+00] |

|  |
| --- |
| **2.添加干扰后的多项式拟合** |
| （1）源代码 |
| c1 = return\_coefficient(x, y, 1)  c2 = return\_coefficient(x, y, 2)  c3 = return\_coefficient(x, y, 3)  c4 = return\_coefficient(x, y, 4)  c6 = return\_coefficient(x, y, 6)  #未添加噪点的图像  plt.figure(figsize=(8, 6))  ax = plt.gca()  plt.scatter(x, y, label='Data')  plt.plot(x, original\_polynomial(x), label='Original Polynomial', color='black', linestyle='--')  plt.plot(x, c1(x), 'r', label='degree 1 polynomial')  plt.plot(x, c2(x), 'g', label='degree 2 polynomial')  plt.plot(x, c3(x), 'b', label='degree 3 polynomial')  plt.plot(x, c4(x), 'y', label='degree 4 polynomial')  plt.plot(x, c6(x), 'm', label='degree 6 polynomial')  plt.legend()  print("加噪点的拟合系数:")  for i in [1,2,3,4,6]:  print(f'The coefficients of degree {i} polynomial are:', return\_xishu(x, y, i)) |
| （2）结果 |
| 加噪点的拟合系数:  The coefficients of degree 1 polynomial are: [ 36.50245579 -102.15205061]  The coefficients of degree 2 polynomial are: [ 9.00597667 -53.55731093 40.04758104]  The coefficients of degree 3 polynomial are: [ 1.00524427 -6.07268733 5.22580951 -2.55695438]  The coefficients of degree 4 polynomial are: [ 9.12811563e-04 9.86988036e-01 -5.95720421e+00 4.98378983e+00  -2.46387281e+00]  The coefficients of degree 6 polynomial are: [ 2.88115207e-04 -8.83454244e-03 1.03131096e-01 4.36748400e-01  -4.59924861e+00 3.73527035e+00 -2.29166612e+00] |