实验报告

姓名	袁永润
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 10 日

实验报告

一、实验名称

1.最小二乘法拟合
2.圆周率的计算

二、 实验目的

1.理解最小二乘法的原理，并会用最小二乘法解决问题；
2.实现几种利用分析方法、概率方法求解圆周率的算法，并分析比较各种算法的特点；
3.培养编程与上机调试能力；
4.熟悉 Python 软件环境。

三、 最小二乘法计算

3.1 问题的背景

关于交通事故的调查。一辆汽车在拐弯时急刹车，结果冲到路边的沟里（见图 3.1）。交警立即赶到事故现场，而司机申辩说，当他进入弯道时刹车已失灵，他还一口咬定，进入弯道时其车速为 40 英里/小时，即该车在这类公路上的速度上限，相当于 17.9 米/秒，交警验车时证实该车的制动器在事故发生时的确失灵，然而司机所说的车速是否真实，我们可以利用最小二乘法进行计算验证，建立的模型也可推广到生活中，有助于交警合理判断交通违章行为。

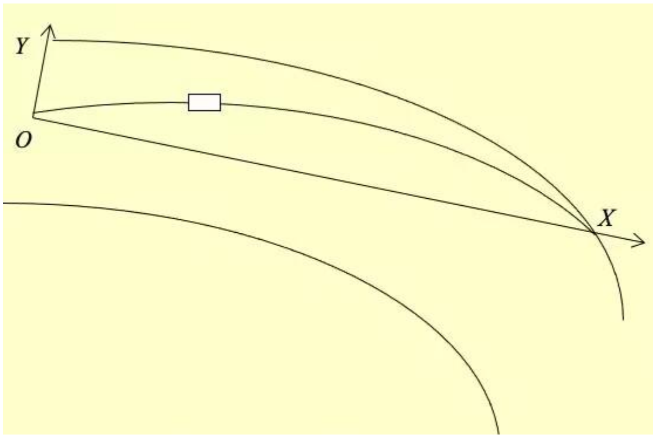


图 3.1 汽车轨迹图

3.2 问题重述

由交警获得的数据（见表 3.2）进行计算，通过 Python 绘制图像可得到一个类似于刹车痕迹的弧线，并计算司机此时的真实车速，由此计算来验证司机所说的车速 17.9 米/秒是

否属实。并且假定该车并没有偏离它的行驶转弯方向，车头一直指向转弯曲线的切线方向。

x	0	3	6	9	12	15	16.64	18	21	24	27	30	33.27
y	0	1.19	2.15	2.82	3.28	3.53	3.55	3.54	3.31	2.89	2.22	1.29	0

表 3.2 刹车痕迹测量值（单位：米）

x：刹车痕迹方向

y：垂直于 x 轴的方向

3.3 数学模型一

1.模型假设：

（1）车的重心沿一个半径为 r 的圆做圆周运动，现有公路弯道按圆弧段设计，需要检验。

（2）汽车速度 v 为常数。

（3）假设摩擦力 f 作用在汽车速度的法线上。

（4）摩擦系数为 k，汽车质量为 m。

（5）该问题摩擦系数取 kg=8.175m/s。

2.模型建立：

（1）根据牛顿运动学定律：

$$\mathrm{f} = \mathrm{kmg} = \frac{mv^2}{r}\#(1)$$

（2）圆半径的估计：假设已知圆的弦长为 c，弓形高度为 h，由勾股定理：

$$\left(\frac{c}{2}\right)^2 + (r-h)^2 = r^2\#(2)$$

（3）由①式得：

$$\mathrm{v} = \sqrt{\mathrm{kgr}}\#(3)$$

（4）通过 Python，用最小二乘法来拟合圆函数，从而可以求出半径，代入到③式中即可求得汽车的真实速度。

3.最小二乘圆拟合原理

最小二乘法是一种数学优化方法，通过最小化误差的平方和找到一组数据拟合为最佳的函数，最小二乘法通常用于曲线拟合和拟合圆曲线的公式推导过程。

最小二乘法拟合圆曲线：

$$R^2 = (x-A)^2 + (y-B)^2\#(4)$$

$$R^2 = x^2 - 2Ax + A^2 + y^2 - 2By + B^2\#(5)$$

令：

$$a =- 2A\#(6)$$

$$b =- 2B\#(7)$$

$$c = A^2 + B^2 - R^2\#(8)$$

可以得到圆曲线方程的另一个形式：

$$x^2+y^2+ax+by+c=0\#(9)$$

求解出参数a,b,c即可求得圆的相关参数：

$$A=\frac{a}{-2}\#(10)$$

$$B=\frac{b}{-2}\#(11)$$

$$R=\frac{1}{2}\sqrt{a^2+b^2-4c}\#(12)$$

中心到圆心的距离为d_i：

$$d_i^2=(X_i-A)^2+(Y_i-B)^2\#(13)$$

点（X_i，Y_i）到圆边缘的距离的平方与和半径平方的差为：

$$\delta_i=d_i^2-R^2=(X_i-A)^2+(Y_i-B)^2-R^2={X_i}^2+{Y_i}^2+aX_i+bY_i+c\#(14)$$

令$Q(a,b,c)$为δ_i的平方和：

$$Q(a,b,c)=\sum\delta_i^2=\sum\left[\left(X_i^2+Y_i^2+aX_i+bY_i+c\right)\right]^2\#(15)$$

求参数a,b,c使$Q(a,b,c)$的值为最小值。

关于$F(a,b,c)$对a,b,c求偏导，令偏导等于 0，得到极值点，比较所有极值点的函数值即可得到最小值：

$$\frac{\partial Q(a,b,c)}{\partial a}=\sum 2\left(X_i^2+Y_i^2+aX_i+bY_i+c\right)X_i=0\#(16)$$

$$\frac{\partial Q(a,b,c)}{\partial b}=\sum 2\left(X_i^2+Y_i^2+aX_i+bY_i+c\right)Y_i=0\#(17)$$

$$\frac{\partial Q(a,b,c)}{\partial c}=\sum 2\left(X_i^2+Y_i^2+aX_i+bY_i+c\right)=0\#(18)$$

解这个方程组解得：

$$Ca+Db+E=0\#(19)$$

$$Da+Gb+H=0\#(20)$$

$$a=\frac{HD-EG}{CG-D^2}\#(21)$$

$$b=\frac{HC-ED}{D^2-GC}\#(22)$$

$$c=-\frac{\sum\left(X_i^2+Y_i^2\right)+a\sum X_i+b\sum Y_i}{N}\#(23)$$

即可得到式（10）、（11）、（12）的值。

3.模型求解：

根据最小二乘拟合圆拟合得到汽车轨迹的半径为 r=40.67788553908053， v=18.235726316272224，即计算得到的汽车的真实速度约为 18.236m/s，与司机所说的速度 17.9m/s 相差不大，故可以认为司机所说的车速属实。

四、 圆周率的求解

4.1 问题的背景

关于π的计算。圆周率是人类获得的最古老告的数学概念之一，早在大约 3700 年前(即公元前 1700 年左右）的古埃及人就已经在用$\frac{81}{256}$（约 3.1605)作为几的近似值了。几千年来，人们一直没有停止过求几的努力。圆周率是人类获得的最古老告的数学概念之一，早在大约 3700 年前(即公元前 1700 年左右）的古埃及人就已经在用 256/81(约 3.1605)作为几的近似值了。几千年来,人们一直没有停止过求几的努力。古典方法采用的是用圆内接正多边形和圆外切正多边形来逼近π的值，阿基米德曾用圆内接 96 边形和圆外切 96 边形夹逼的方法证明了$\frac{223}{71} < \pi < \frac{22}{7}$，公元 5 世纪，祖冲之指出 3.1415926< π <3.1415927，其得到同样的结果比西方早了 1000 年。从十七世纪中叶起，人们开始使用更先进的分析方法来求π的近似值，其中应用的主要工具是收敛的无穷乘积和无穷级数。

4.2 问题重述

（1）沃里斯（Wallis）证明方法求解π近似值；
（2）利用泰勒级数求解π近似值；
（3）利用等式

$$\frac{\pi}{4} = \arctan 1 = \arctan\frac{1}{2} + \arctan\frac{1}{3} \#(1)$$

求解π近似值；
（4）利用麦琴（Machin）公式

$$\frac{\pi}{4} = \arctan 1 = 4\arctan\frac{1}{5} - \arctan\frac{1}{239} \#(2)$$

求解π近似值；
（5）利用概率方法求解π近似值；
（6）利用两种数值积分数值积分求解π近似值。

4.3 数学模型二

1.模型假设与建立：

（1）沃里斯（Wallis）证明方法求解π近似值：

沃里斯的方法证明了：

$$\pi = 2 \cdot \left(\frac{2}{1}\cdot\frac{2}{3}\right)\cdot\left(\frac{4}{3}\cdot\frac{4}{5}\right)\cdot\left(\frac{6}{5}\cdot\frac{6}{7}\right)\cdots = 2\cdot\prod_{k=1}^{\infty}\left(\frac{2k}{2k-1}\cdot\frac{2k}{2k+1}\right)\#(3)$$

再通过分别取十二组 k 值进行计算。

（2）泰勒级数求解π近似值：

其中公式为：

$$\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5} - \cdots = \sum_{k=0}^{\infty}(-1)^k\cdot\frac{x^{2k+1}}{2k+1}\#(4)$$

当 x 取 1 时有：

$$\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\cdots=\sum_{k=0}^{\infty}(-1)^k\cdot\frac{1}{2k+1}\#(5)$$

再通过分别取十二组 k 值进行计算。

（3）利用等式：

$$\frac{\pi}{4}=\arctan 1=\arctan\frac{1}{2}+\arctan\frac{1}{3}\#(6)$$

方法同（1）、（2）相同，编写 Python 程序直接求解π近似值。

（4）利用麦琴（Machin）公式求解π近似值

其中公式为：

$$\frac{\pi}{4}=\arctan 1=4\arctan\frac{1}{5}-\arctan\frac{1}{239}\#(7)$$

方法同上述步骤，直接求解π近似值。

（5）概率方法求解π近似值：

取一个二维数组（x，y），取一个充分大的正整数 n，重复 n 次，每次独立地从（0，1）中随机地抽取一对数 x 和 y，分别检验$x^2+y^2\leq 1$ 是否成立。设 n 次实验中等式成立的共有 m 次，令$\pi\approx\frac{4m}{n}$，编写 Python 程序进行求解。

（6）两种数值积分求解π近似值：

$$\pi=4\int_0^1\sqrt{1-x^2}dx\#(8)$$

$$\pi=4\int_0^1\frac{1}{1+x^2}dx\#(9)$$

以上两种方法均采用 Romberg 算法求其积分近似解π。

2.模型求解：

（1）沃里斯（Wallis）证明方法求解π近似值：

k	10	20	50	100
π	3.0677038066434976	3.1035169615392304	3.1260789002154086	3.133787490628159
k	300	500	700	1000
π	3.138980103882125	3.1400238186005947	3.1404716572102904	3.140807746030398
k	3000	5000	7000	10000
π	3.141330908733516	3.1414355935898683	3.14148046386916	3.141514118681864

表 4. 1 沃里斯方法求π近似值

由表 4.1 可以发现沃里斯方法得到的π的近似值，在 k=500 时的值才到 3.14，在 k=10000 时达到 3.1415，但精确度还不够，收敛速度较慢。

（2）泰勒级数法求解π近似值：

k	10	20	50	100
π	3.232315809405594	3.189184782277596	3.1611986129870506	3.1514934010709914
k	300	500	700	1000
π	3.1449149035588526	3.143588659585789	3.1430191863875865	3.1425916543395442
k	3000	5000	7000	10000
π	3.1419258758397897	3.141792613595791	3.141735490326666	3.1416926435905346

表 4.2 泰勒级数方法求π近似值

（3）利用等式法求解π近似值：

k	2	4	6	8
π	3.1455761316872426	3.1417411974336886	3.141599340966198	3.1415929813345667
k	10	12	14	16
π	3.1415926704506854	3.141592654485348	3.141592653638457	3.1415926535924825
k	18	20	22	24
π	3.141592653589943	3.141592653589801	3.141592653589793	3.141592653589792

表 4.3 等式方法求π近似值

（4）利用麦琴公式求解π近似值：

k	2	4	6	8
π	3.1416210293250346	3.1415926824043994	3.141592653623555	3.141592653589836
k	10	12	14	16
π	3.141592653589794	3.141592653589794	3.141592653589794	3.141592653589794
k	18	20	22	24
π	3.141592653589794	3.141592653589794	3.141592653589794	3.141592653589794

表 4.4 麦琴公式法求π近似值

将（1）、（2）、（3）、（4）进行比较分析，沃里斯计算方法与泰勒级数法得到的π的近似值的收敛速度相近，都较慢，在 k=10000 时在数值 3.141 附近，但从结果上来看，泰勒级数法的收敛速度要比沃里斯计算方法慢。等式法与麦琴公式收敛速度均比前面两种计算方法要快，等式法在 k 值较小的情况下能够收敛到 3.141，麦琴公式在 k=10 时收敛到了更为精确的值 3.14159，由此可知，计算π近似值时麦琴公式法收敛速度更快，且更为精确。

（4）利用概率方法求解π近似值：

k	20000	40000	60000	80000
π	3.156	3.1476	3.1281333333333334	3.1455
k	100000	120000	140000	160000
π	3.13192	3.146733333333333	3.147885714285714	3.140225
k	180000	200000	220000	240000
π	3.146377777777778	3.13942	3.1394727272727274	3.1418833333333334

表 4.5 概率方法求π近似值

由上表可知用概率方法计算出的π近似值收敛情况不容易确定，且数值之间差距有的较大，得到的近似值也不够精确。

（5）两种数值积分求解π近似值分别为：3.141592653638244 和 3.141580817524002，分析对比发现第二种数值积分计算出来的π近似值要更为精确。

五、 总结

1.当需要判断汽车转弯时是否有超速行为，可根据汽车轨迹数据，利用最小二乘拟合圆来进行判定，对生活中相关的模型具有较好的借鉴作用。

2.估计π近似值时，不同算法的效率不同。为计算快捷，可以选择收敛速度较快的等式法以

及麦琴公式进行计算，其中麦琴公式的效果更好，计算结果更精确也更快捷。
3.编写更为高效的程序能提高计算速率，也能更好地分析相关的建模问题。

六、 附录

1.最小二乘法

(1)源代码

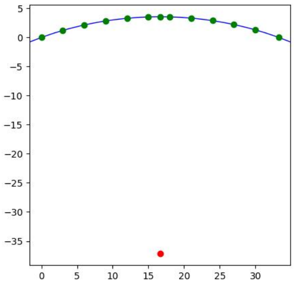
```
import numpy as np
import math
import matplotlib.pyplot as plt
from scipy.optimize import least_squares

# 定义拟合函数及其残差函数
def circle_residuals(params,points):
    x0,y0,r = params
    residuals = []
    for point in points:
        residual = (point[0]-x0)**2+(point[1]-y0)**2-r**2
        residuals.append(residual)
    return residuals

def fit_circle(points):
    x_data = [point[0] for point in points ]
    y_data = [point[1] for point in points]
    # 初始估计值
    x0 = np.mean(x_data)
    y0 = np.mean(y_data)
    r = np.mean([np.sqrt((x-x0)**2+(y-y0)**2) for x,y in zip(x_data,y_data)])
    # 最小二乘法拟合
    params0 = [x0,y0,r]
    result = least_squares(circle_residuals,params0,args=(points,))
    x0_fit,y0_fit,r_fit = result.x
    # 计算弓形高度
    arc_height = abs(r_fit)-abs(y0_fit)
    # 计算弦长
    x_intersect_length = 2*math.sqrt(abs(r_fit)**2-(abs(r_fit)-abs(arc_height))**2)
    return x0_fit,y0_fit,abs(r_fit),arc_height,x_intersect_length

    points = 
[(0,0),(3,1.19),(6,2.15),(9,2.82),(12,3.28),(15,3.53),(16.64,3.55),(18,3.54),(21,3.31),(24,2.89),(27,
2.22),(30,1.29),(33.27,0)]
    center_1,center_2,radius,arc_height,xianchang= fit_circle(points)
    kg = 8.175
    v = math.sqrt(kg*radius)
    print(f"拟合圆的半径为:{radius}\n 弓形高度为{arc_height}\n 弦长为{xianchang}\n 拟合圆
的中心为{center_1,center_2}\n 速度为{v}")

#绘制拟合图像
plt.figure(figsize=(5,5))
ax = plt.gca()
ax.plot([point[0] for point in points],[point[1] for point in points],'go',label="Original Points")
ax.plot(center_1,center_2, 'ro',label="Fitted Circle")
circle = plt.Circle((center_1,center_2), radius, color='b', fill=False)
```

```
ax.add_artist(circle)
plt.show()
```

（2）结果

拟合圆的半径为:40.67788553908053
弓形高度为 3.554144967069668
弦长为 33.25767627917329
拟合圆的中心为(16.64079097607902, -37.12374057201086)
速度为 18.235726316272224

拟合图像(红点表示拟合圆的圆心):



2.圆周率的求解

（1）源代码

```
#沃里斯（Wallis）方法求解 Π
def Wallis_caculate_pi(n):
    k = 1
    multiple = 1
    while(k<=n):
        temp=2*k/(2*k-1)*2*k/(2*k+1)
        multiple = multiple*temp
        k+=1
    return multiple*2
print("沃利斯求解 π :")
for i in ([10,20,50,100,300,500,700,1000,3000,5000,7000,10000]):
    print(f"k={i}时近似值为:{Wallis_caculate_pi(i)}")
```

（2）结果

沃利斯求解 π :
k=10时近似值为:3.0677038066434976
k=20时近似值为:3.1035169615392304
k=50时近似值为:3.1260789002154086
k=100时近似值为:3.1337874906281593
k=300时近似值为:3.138980103882125
k=500时近似值为:3.1400238186005947
k=700时近似值为:3.1404716572102904
k=1000时近似值为:3.1408077460303976
k=3000时近似值为:3.141330908733516
k=5000时近似值为:3.1414355935898683
k=7000时近似值为:3.14148046386916
k=10000时近似值为:3.1415141186818643

（3）源代码

```
%泰勒级数法
def tayor_caculate(x,n):
    k = 0
    sum = 0
    while(k<=n):
        temp = (-1)**k*(x**(2*k+1)/(2*k+1))
        sum += temp
        k += 1
    return sum*4

print("Tayor 法求解:")
for i in ([10,20,50,100,300,500,700,1000,3000,5000,7000,10000]):
    print(f"k={i}时近似值为:{tayor_caculate(1,i)}")
```

（4）结果

```
Tayor法求解:
k=10时近似值为:3.232315809405594
k=20时近似值为:3.189184782277596
k=50时近似值为:3.1611986129870506
k=100时近似值为:3.1514934010709914
k=300时近似值为:3.1449149035588526
k=500时近似值为:3.143588659585789
k=700时近似值为:3.1430191863875865
k=1000时近似值为:3.1425916543395442
k=3000时近似值为:3.1419258758397897
k=5000时近似值为:3.1417926135957908
k=7000时近似值为:3.141735490326666
k=10000时近似值为:3.1416926435905346
```

（5）源代码

```
print("等式法直接求解:")
for i in range(2,26,2):
    print(f"k={i}时近似值为:{tayor_caculate(1/2,i)+tayor_caculate(1/3,i)}")
```

（6）结果

```
等式法直接求解:
k=2时近似值为:3.1455761316872426
k=4时近似值为:3.1417411974336886
k=6时近似值为:3.141599340966198
k=8时近似值为:3.1415929813345667
k=10时近似值为:3.1415926704506854
k=12时近似值为:3.141592654485348
k=14时近似值为:3.1415926536384573
k=16时近似值为:3.1415926535924825
k=18时近似值为:3.141592653589943
k=20时近似值为:3.141592653589801
k=22时近似值为:3.1415926535897927
k=24时近似值为:3.1415926535897922
```

（7）源代码

```
print("麦瑟法直接求解:")
for i in range(2,26,2):
    print(f"k={i}时近似值为:{4*tayor_caculate(1/5,i)-tayor_caculate(1/239,i)}")
```

（8）结果
麦瑟法直接求解: k=2时近似值为:3.1416210293250346 k=4时近似值为:3.1415926824043994 k=6时近似值为:3.141592653623555 k=8时近似值为:3.141592653589836 k=10时近似值为:3.141592653589794 k=12时近似值为:3.141592653589794 k=14时近似值为:3.141592653589794 k=16时近似值为:3.141592653589794 k=18时近似值为:3.141592653589794 k=20时近似值为:3.141592653589794 k=22时近似值为:3.141592653589794 k=24时近似值为:3.141592653589794

（9）源代码

```
# 利用概率方法求解 π
import random
def monte_carlo_pi(points):
    inside_circle = 0
    for _ in range(points):
        x, y = random.random(), random.random()
        if (x ** 2 + y ** 2) <= 1.0:
            inside_circle += 1
    return 4 * (inside_circle / points)

print("概率方法求解:")
for i in range(20000,260000,20000):
    print(f"k={i}时近似值为:{monte_carlo_pi(i)}")
```

（10）结果
概率方法求解: k=20000时近似值为:3.156 k=40000时近似值为:3.1476 k=60000时近似值为:3.1281333333333334 k=80000时近似值为:3.1455 k=100000时近似值为:3.13192 k=120000时近似值为:3.1467333333333333 k=140000时近似值为:3.147885714285714 k=160000时近似值为:3.140225 k=180000时近似值为:3.146377777777778 k=200000时近似值为:3.13942 k=220000时近似值为:3.1394727272727274 k=240000时近似值为:3.1418833333333334

（11）源代码

```
import math
from scipy.integrate import romberg
def pi_integrand1(x):
    return 1 / (1 + x ** 2)
def calculate_pi1(function):
    a, b = 0, 1  # 积分下限和上限
```

```
    return romberg(function, a, b) * 4
print(f"π的计算值为: {calculate_pi1(pi_integrand1)}")
```

（12）结果

π 的计算值为: 3.141592653638244

（13）源代码

```
%数值积分求解法二
def pi_integrand2(x):
    return math.sqrt(1 - x**2)
print(f"π的计算值为: {calculate_pi1(pi_integrand2)}")
```

（14）结果

π 的计算值为: 3.141580817524002