

Problem 1

(1)

线性最小二乘法模型如下：

$$a_1 = \frac{m \sum x_i y_i - \sum x_i \sum y_i}{m \sum x_i^2 - (\sum x_i)^2}$$
$$a_0 = \frac{\sum x_i^2 \sum y_i - \sum x_i \sum y_i}{m \sum x_i^2 - (\sum x_i)^2}$$
$$y = a_1 x + a_0$$

线性模型代码如下：

```
#include<stdio.h>
//使用线性最小二乘拟合以下数据
//y: 1 43 75 155 196 219 271 318 351 385 425 466 503 537 575 599 626
//x: 1761 1767 1771 1772 1774 1775 1777 1779 1780 1782 1783 1785 1786 1788 1789
1791 1791
int main(){
    int x[17] =
{1761,1767,1771,1772,1774,1775,1777,1779,1780,1782,1783,1785,1786,1788,1789,1791,1791};
    int y[17] =
{1,43,75,155,196,219,271,318,351,385,425,466,503,537,575,599,626};
    double a0,a1;
    double m = 17;
    double sumXi = 0,sumYi=0,sumXi2=0,sumYi2=0,sumXiYi=0;
    for(int i=0;i<17;i++){
        sumXi += x[i];
        sumYi += y[i];
        sumXi2 += x[i]*x[i];
        sumYi2 += y[i]*y[i];
        sumXiYi += x[i]*y[i];
    }
    a1 = (m*sumXiYi-sumXi*sumYi)/(m*sumXi2-sumXi*sumXi);
    a0 = (sumXi2*sumYi-sumXiYi*sumXi)/(m*sumXi2-sumXi*sumXi);
    printf("a0 = %f\n a1 = %f\n",a0,a1);
    //y=173时预测x
    printf("y=173时预测x = %f\n", (173-a0)/a1);
    //y=726时预测x
    printf("y=726时预测x = %f\n", (726-a0)/a1);
}
```

运行结果截图如下：

```
[Running] cd "f:\桌面\一些文件\主修课程\大
a0 = -40344.267003
a1 = 22.861973
y=173时预测x = 1772.255936
y=726时预测x = 1796.444575
```

二次最小二乘代码如下:

```
import numpy as np
from numpy.linalg import *

import io
import sys
sys.stdout = io.TextIOWrapper(sys.stdout.buffer,encoding='utf-8')

#使用二次最小二乘拟合以下数据
#y: 1 43 75 155 196 219 271 318 351 385 425 466 503 537 575 599 626
#x: 1761 1767 1771 1772 1774 1775 1777 1779 1780 1782 1783 1785 1786 1788 1789
1791 1791

y = np.array([1, 43, 75, 155, 196, 219, 271, 318, 351, 385, 425, 466, 503, 537,
575, 599, 626],dtype='int64')
x =
np.array([1761,1767,1771,1772,1774,1775,1777,1779,1780,1783,1785,1786,1788,1789,
1791,1791],dtype='int64')
m = 17

sumXi=sumYi=0          #x, y的和
sumXi2=sumYi2=0        #x^2,y^2的和
sumXiYi=0              #x*y的和
sumXi3=0               #x^3的和
sumXi4=0               #x^4的和
sumXi2Yi=0             #x^2*y的和

for i in range(0,16):
    sumXi += x[i]
    sumYi += y[i]
    sumXi2 += x[i]*x[i]
    sumYi2 += y[i]*y[i]
    sumXiYi += x[i]*y[i]
    sumXi3 += x[i]*x[i]*x[i]
    sumXi4 += x[i]*x[i]*x[i]*x[i]
    sumXi2Yi += x[i]*x[i]*y[i]

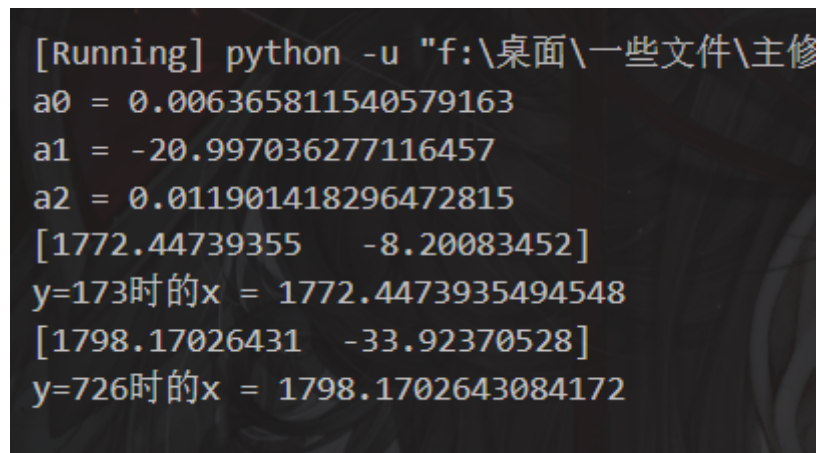
A = np.array([
    [m,sumXi,sumXi2],
    [sumXi,sumXi2,sumXi3],
    [sumXi2,sumXi3,sumXi4]
])
B = np.array([
    [sumYi],
    [sumXiYi],
    [sumXi2Yi]
])
```

```

a0=float(solve(A,B)[0])
a1=float(solve(A,B)[1])
a2=float(solve(A,B)[2])
print("a0 =",a0)
print("a1 =",a1)
print("a2 =",a2)
#预测y=173时的x
p=np.array([a2,a1,a0-173])
print(np.roots(p))
print("y=173时的x =",np.roots(p)[0])
#预测y=726时的x
p=np.array([a2,a1,a0-726])
print(np.roots(p))
print("y=726时的x =",np.roots(p)[0])

```

运行结果截图如下：



```

[Running] python -u "f:\桌面\一些文件\主修
a0 = 0.006365811540579163
a1 = -20.997036277116457
a2 = 0.011901418296472815
[1772.44739355 -8.20083452]
y=173时的x = 1772.4473935494548
[1798.17026431 -33.92370528]
y=726时的x = 1798.1702643084172

```

三次最小二乘代码如下：

```

import numpy as np
from numpy.linalg import *

import io
import sys
sys.stdout = io.TextIOWrapper(sys.stdout.buffer,encoding='utf-8')

#使用三次最小二乘拟合以下数据
#y: 1 43 75 155 196 219 271 318 351 385 425 466 503 537 575 599 626
#x: 1761 1767 1771 1772 1774 1775 1777 1779 1780 1782 1783 1785 1786 1788 1789
1791 1791

y = np.array([1, 43, 75, 155, 196, 219, 271, 318, 351, 385, 425, 466, 503, 537,
575, 599, 626],dtype='int64')
x =
np.array([1761,1767,1771,1772,1774,1775,1777,1779,1780,1783,1785,1786,1788,1789,
1791,1791],dtype='int64')
m = 17

sumXi=sumYi=0          #x, y的和
sumXi2=sumYi2=0        #x^2,y^2的和
sumXiYi=0              #x*y的和
sumXi3=0               #x^3的和
sumXi4=0               #x^4的和
sumXi5=0               #x^5的和

```

```

sumXi6=0          #x^6的和
sumXi2Yi=0        #x^2*y的和
sumXi3Yi=0        #x^3*y的和

for i in range(0,16):
    sumXi += x[i]
    sumYi += y[i]
    sumXi2 += x[i]*x[i]
    sumYi2 += y[i]*y[i]
    sumXiYi += x[i]*y[i]
    sumXi3 += x[i]*x[i]*x[i]
    sumXi4 += x[i]*x[i]*x[i]*x[i]
    sumXi5 += x[i]*x[i]*x[i]*x[i]*x[i]
    sumXi6 += pow(float(x[i]),6)
    sumXi2Yi += x[i]*x[i]*y[i]
    sumXi3Yi += x[i]*x[i]*x[i]*y[i]

A = np.array([
    [m,sumXi,sumXi2,sumXi3],
    [sumXi,sumXi2,sumXi3,sumXi4],
    [sumXi2,sumXi3,sumXi4,sumXi5],
    [sumXi3,sumXi4,sumXi5,sumXi6]
])
B = np.array([
    [sumYi],
    [sumXiYi],
    [sumXi2Yi],
    [sumXi3Yi]
])
res = solve(A,B)
a0 = float(res[0])
a1 = float(res[1])
a2 = float(res[2])
a3 = float(res[3])
print("a0 = ",a0)
print("a1 = ",a1)
print("a2 = ",a2)
print("a3 = ",a3)
#预测y=173时的x
p=np.array([a3,a2,a1,a0-173])
print(np.roots(p))
print("y=173时的x =", np.roots(p)[0])
#预测y=726时的x
p=np.array([a3,a2,a1,a0-726])
print(np.roots(p))
print("y=726时的x =", np.roots(p)[0])

```

运行结果截图如下：

```
[Running] python -u "f:\桌面\一些文件\主修课程\大
a0 = 2.001041428024293e-05
a1 = 346.99455387430015
a2 = -0.402113418244975
a3 = 0.00011644566135166216
[1.77298139e+03 1.67974775e+03 4.98855190e-01]
y=173时的x = 1772.9813934539657
[1795.77923319 1655.35142039 2.09734631]
y=726时的x = 1795.7792331874111
```

取三次最小二乘逼近模型为最优逼近方案

(2)

三种模型的运行结果截图如下，从上到下依次是线性模型、二次最小二乘模型、三次最小二乘模型

```
[Running] cd "f:\桌面\一些文件\主修课程\大二上\数值分析方法\Project\" && gcc 线性最小二乘.c -o
a0 = -40344.267003
a1 = 22.861973
y=173时预测x = 1772.255936
y=726时预测x = 1796.444575

[Done] exited with code=0 in 0.941 seconds

[Running] python -u "f:\桌面\一些文件\主修课程\大二上\数值分析方法\Project\二次最小二乘.py"
a0 = 0.006365811540579163
a1 = -20.997036277116457
a2 = 0.011901418296472815
[1772.44739355 -8.20083452]
y=173时的x = 1772.4473935494548
[1798.17026431 -33.92370528]
y=726时的x = 1798.1702643084172

[Done] exited with code=0 in 0.687 seconds

[Running] python -u "f:\桌面\一些文件\主修课程\大二上\数值分析方法\Project\三次最小二乘.py"
a0 = 2.001041428024293e-05
a1 = 346.99455387430015
a2 = -0.402113418244975
a3 = 0.00011644566135166216
[1.77298139e+03 1.67974775e+03 4.98855190e-01]
y=173时的x = 1772.9813934539657
[1795.77923319 1655.35142039 2.09734631]
y=726时的x = 1795.7792331874111

[Done] exited with code=0 in 0.524 seconds
```

三次最小二乘模型预测到y=173时，x=1772.9，预测了正确的日期

(3)

如第二题的截图所示，线性模型预测值为1796.4，二次最小二乘模型预测值为1798.2，三次最小二乘模型预测为1795.8，猜测三次最小二乘模型更有效，即莫扎特将会在1796年完成K.726，因为y=173时，三次最小二乘模型的预测值最准确

Problem 2

(1)

logistics模型的微分方程如下：

$$y'(t) = ky(t)\left(1 - \frac{y(t)}{L}\right)$$
$$k = 0.8 \quad L = 10$$

使用如下代码进行迭代：

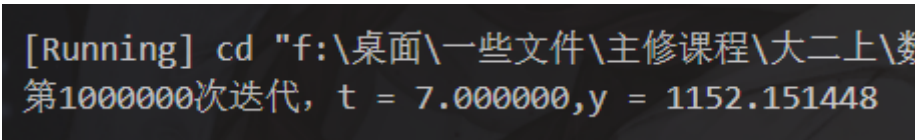
```
#include<stdio.h>

//使用logistic模型进行预测
//使用欧拉方法近似解logistic模型
//y(0) = 10
//求t=7时的y(t)
//y'(t) = k*y(t)*(1-y(t)/L)
//k = 0.8

double fn(double t,double y){
    double res = 0;
    res = 0.8*y*(1-y/2000);
    return res;
}

int main(){
    int N = 1000000;
    double h = (7.0-0)/N;
    double t = 0;
    double y = 10;
    int iterateTime = 0;
    //printf("第%d次迭代, t = %lf,y = %lf\n",iterateTime,t,y);
    for(int i=0;i<N;i++){
        t += h;
        double tmp = y;
        y = tmp + h*fn(t,tmp);
        iterateTime++;
        //printf("第%d次迭代, t = %lf,y = %lf\n",iterateTime,t,y);
    }
    printf("第%d次迭代, t = %lf,y = %lf\n",iterateTime,t,y);
}
```

迭代1000000次以后得到的结果如下：



```
[Running] cd "f:\桌面\一些文件\主修课程\大二上\数
第1000000次迭代, t = 7.000000,y = 1152.151448
```

除了倡议小组以外的，大约还有1142个同学知道关于音乐会的消息

(2)

使用如下代码进行逼近：

```
#include<stdio.h>
```

```

//使用logistic模型进行预测
//使用欧拉方法近似解logistic模型
//y(0) = 10
//求t=7时的y(t)
//y'(t) = k*y(t)*(1-y(t)/L)
//k = 0.8

double fn(double t,double y,double k,double L){
    double res = 0;
    res = k*y*(1-y/L);
    return res;
}

int main(){
    int N = 1000000;
    double k = 0.8;
    int L = 2000;
    double h = (4.0-0)/N;
    double t = 0;
    double y = 10;
    int iterateTime = 0;
    //printf("第%d次迭代, t = %lf,y = %lf\n",iterateTime,t,y);
    for(double j = 0;j<0.8;j+=0.001){
        for(int i=0;i<N;i++){
            t += h;
            double tmp = y;
            y = tmp + h*fn(t,tmp,j,L);//欧拉方法迭代
            iterateTime++;
            //printf("第%d次迭代, t = %lf,y = %lf\n",iterateTime,t,y);
        }
        printf("k = %lf时迭代得到的结果, t = %lf,y = %lf\n",j,t,y);
        if(y>=40)//当知道的人数大于40时, 停止迭代
            break;
        t = 0;
        y = 10;
    }
}

```

运算结果截图如下：

```
k = 0.314000时迭代得到的结果, t = 4.000000,y = 34.678010
k = 0.315000时迭代得到的结果, t = 4.000000,y = 34.814580
k = 0.316000时迭代得到的结果, t = 4.000000,y = 34.951679
k = 0.317000时迭代得到的结果, t = 4.000000,y = 35.089308
k = 0.318000时迭代得到的结果, t = 4.000000,y = 35.227469
k = 0.319000时迭代得到的结果, t = 4.000000,y = 35.366164
k = 0.320000时迭代得到的结果, t = 4.000000,y = 35.505395
k = 0.321000时迭代得到的结果, t = 4.000000,y = 35.645165
k = 0.322000时迭代得到的结果, t = 4.000000,y = 35.785475
k = 0.323000时迭代得到的结果, t = 4.000000,y = 35.926327
k = 0.324000时迭代得到的结果, t = 4.000000,y = 36.067723
k = 0.325000时迭代得到的结果, t = 4.000000,y = 36.209665
k = 0.326000时迭代得到的结果, t = 4.000000,y = 36.352156
k = 0.327000时迭代得到的结果, t = 4.000000,y = 36.495197
k = 0.328000时迭代得到的结果, t = 4.000000,y = 36.638790
k = 0.329000时迭代得到的结果, t = 4.000000,y = 36.782938
k = 0.330000时迭代得到的结果, t = 4.000000,y = 36.927642
k = 0.331000时迭代得到的结果, t = 4.000000,y = 37.072905
k = 0.332000时迭代得到的结果, t = 4.000000,y = 37.218728
k = 0.333000时迭代得到的结果, t = 4.000000,y = 37.365114
k = 0.334000时迭代得到的结果, t = 4.000000,y = 37.512065
k = 0.335000时迭代得到的结果, t = 4.000000,y = 37.659582
k = 0.336000时迭代得到的结果, t = 4.000000,y = 37.807669
k = 0.337000时迭代得到的结果, t = 4.000000,y = 37.956326
k = 0.338000时迭代得到的结果, t = 4.000000,y = 38.105557
k = 0.339000时迭代得到的结果, t = 4.000000,y = 38.255363
k = 0.340000时迭代得到的结果, t = 4.000000,y = 38.405746
k = 0.341000时迭代得到的结果, t = 4.000000,y = 38.556709
k = 0.342000时迭代得到的结果, t = 4.000000,y = 38.708254
k = 0.343000时迭代得到的结果, t = 4.000000,y = 38.860382
k = 0.344000时迭代得到的结果, t = 4.000000,y = 39.013096
k = 0.345000时迭代得到的结果, t = 4.000000,y = 39.166399
k = 0.346000时迭代得到的结果, t = 4.000000,y = 39.320292
k = 0.347000时迭代得到的结果, t = 4.000000,y = 39.474777
k = 0.348000时迭代得到的结果, t = 4.000000,y = 39.629857
k = 0.349000时迭代得到的结果, t = 4.000000,y = 39.785534
k = 0.350000时迭代得到的结果, t = 4.000000,y = 39.941810
k = 0.351000时迭代得到的结果, t = 4.000000,y = 40.098688
```

当 $k=0.351$ 时，可以控制4天内知情人数为40