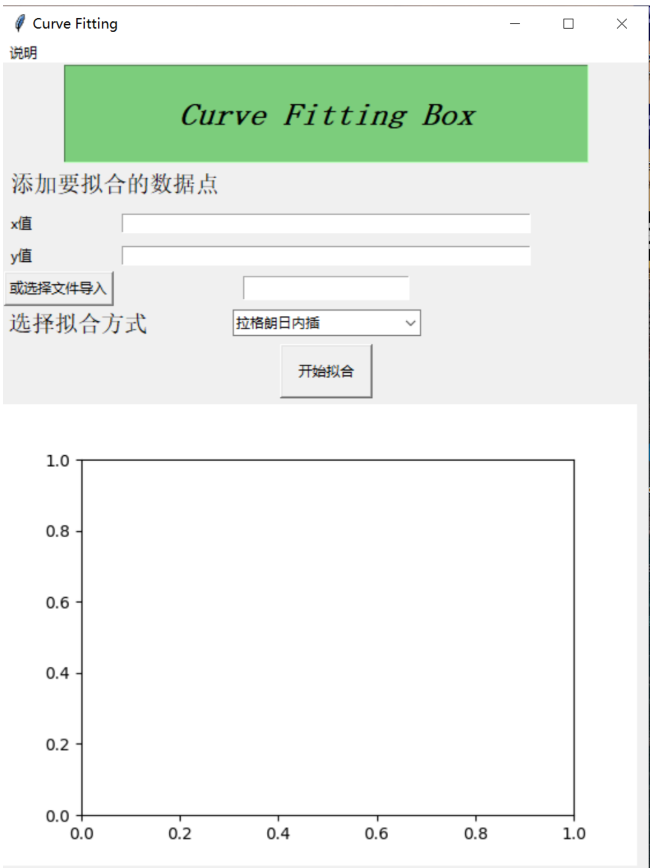
项目报告

设计思路

整个图形界面是用 `tkinder` 编写的，设计了两个文本框用来读取用户输入的数据，同时也设计了从文件导入的按钮，去根据用户输入的路径地址去读对应的文件，在将对应数据传入到 `x_data` ， `y_data` 里。

因为要提供不同的拟合方式，设计了复选框来读取用户选择的拟合方式，因为有些拟合方式需要额外数据（如三次样条压缩法要提供左右端点导数，所以设计了弹窗让用户进一步输入）。

根据用户输入的数据和选择的拟合方式，用对应的算法操作数据，这里在GUI中内嵌了`matplotlib`的画布，来可视化地展现拟合结果。



算法描述

拉格朗日插值

对 $x_1, x_2, \ldots, x_n$ ，先通过下面计算每个$\frac{f(x_k)}{(x_k-x_1)(x_k-x_2)\ldots(x_k-x_n)}$

```
while i < size:
        j = 0;temp = 1
        while j < size:
            if(i != j):
                temp*=data_x[i]-data_x[j]
            j+=1
        parameters.append(data_y[i]/temp)
        i += 1
    return parameters
```

然后带入每个x值，将上面的值与$(x-x_1)(x-x_2)\ldots(x-x_n)$ 相乘，计算得到在x点的函数值。

```
while i < len(parameters):
        temp = 1
        j = 0
        while j< len(parameters):
            if(i!=j):
                temp *=x-data_x[j]
            j+=1
        returnValue += temp * parameters[i]
        i += 1
    return returnValue
```

三次样条插值

假定有n+1个数据节点

(x_0, y_0) ， (x_1, y_1) ， (x_2, y_2) ，$\ldots$，(x_n, y_n)

1，计算步长 $h_i = x_{i+1} - x_i$

2，将数据节点和指定的首位端点条件带入矩阵方程

3，解矩阵方程，求得二次微分值 m_i 。该矩阵为三对角矩阵，常见解法为高斯消元法，可以对系数矩阵进行LU分解，分解为单位下三角矩阵和上三角矩阵。即
$B = Ax = (LU)x = L(Ux) = Ly$

对于三对角方程，我发现有种叫追赶法的方法可以简化计算：

由于是三对角方程，它的LU分解可以直接写出来，在带回求，所以只要一层for循环。

4，计算样条曲线的系数：

$a_i = y_i$

$b_i = \frac{y_{i+1}-y_i}{h_i} - \frac{h_i}{2}m_i - \frac{h_i}{6}(m_{i+1} - m_i)$

$c_i = \frac{m_i}{2}$

$d_i = \frac{m_{i+1}-m_i}{6h_i}$

5，在每个子区间 $x_i \leq x \leq x_{i+1}$ 中，创建方程

$g_i(x) = a_i + b_i\,(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$

线性拟合

有两种方法，最小二乘法or梯度下降法，这里我采用最小二乘法直接用公式求解，因为当数据小时候最下二乘法稳定性更好一点。

$$
\begin{gathered}
w = \frac{\sum x_i^2 \sum y_i - \sum x_i y_i \sum x_i}{m(\sum x_i^2) - (\sum x_i)^2} \\
b = \frac{m \sum x_i y_i - \sum x_i \sum y_i}{m(\sum x_i^2) - (\sum x_i)^2} \\
\Rightarrow y = wx + b
\end{gathered}
$$

多项式拟合

$$
\text{定义} S = \sum_{i=1}^{m} [f(x_i) - y_i]^2
$$

$$
\frac{\partial S}{\partial \theta_j} = \sum_{i=1}^{m} \left[2\left(\theta_0 + \theta_1 x_i + \theta_2 x_i^2 + \cdots + \theta_n x_i^n - y_i\right) x_i^j\right] = 0
$$

$$
\begin{cases}
m\theta_0 + (\sum_{i=1}^m x_i)\theta_1 + (\sum_{i=1}^m x_i^2)\theta_2 + \cdots + (\sum_{i=1}^m x_i^n)\theta_n = \sum_{i=1}^m y_i \\
(\sum_{i=1}^m x_i)\theta_0 + (\sum_{i=1}^m x_i^2)\theta_1 + (\sum_{i=1}^m x_i^3)\theta_2 + \cdots + (\sum_{i=1}^m x_i^{n+1})\theta_n = \sum_{i=1}^m (x_i y_i) \\
(\sum_{i=1}^m x_i^2)\theta_0 + (\sum_{i=1}^m x_i^3)\theta_1 + (\sum_{i=1}^m x_i^4)\theta_2 + \cdots + (\sum_{i=1}^m x_i^{n+2})\theta_n = \sum_{i=1}^m (x_i^2 y_i) \\
\cdots\cdots \\
(\sum_{i=1}^m x_i^n)\theta_0 + (\sum_{i=1}^m x_i^{n+1})\theta_1 + (\sum_{i=1}^m x_i^{n+2})\theta_2 + \cdots + (\sum_{i=1}^m x_i^{2n})\theta_n = \sum_{i=1}^m (x_i^n y_i)
\end{cases}
$$

$$
X = \begin{bmatrix}
m & \sum_{i=1}^m x_i & \sum_{i=1}^m x_i^2 & \cdots & \sum_{i=1}^m x_i^n \\
\sum_{i=1}^m x_i & \sum_{i=1}^m x_i^2 & \sum_{i=1}^m x_i^3 & \cdots & \sum_{i=1}^m x_i^{n+1} \\
\sum_{i=1}^m x_i^2 & \sum_{i=1}^m x_i^3 & \sum_{i=1}^m x_i^4 & \cdots & \sum_{i=1}^m x_i^{n+2} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\sum_{i=1}^m x_i^n & \sum_{i=1}^m x_i^{n+1} & \sum_{i=1}^m x_i^{n+2} & \cdots & \sum_{i=1}^m x_i^{2n}
\end{bmatrix}, \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}, Y = \begin{bmatrix} \sum_{i=1}^m y_i \\ \sum_{i=1}^m (x_i y_i) \\ \sum_{i=1}^m (x_i^2 y_i) \\ \vdots \\ \sum_{i=1}^m (x_i^n y_i) \end{bmatrix}
$$

$$
\begin{gathered}
X\theta = Y \\
\Downarrow \\
\theta = X^{-1} Y
\end{gathered}
$$

即可得到系数向量θ .

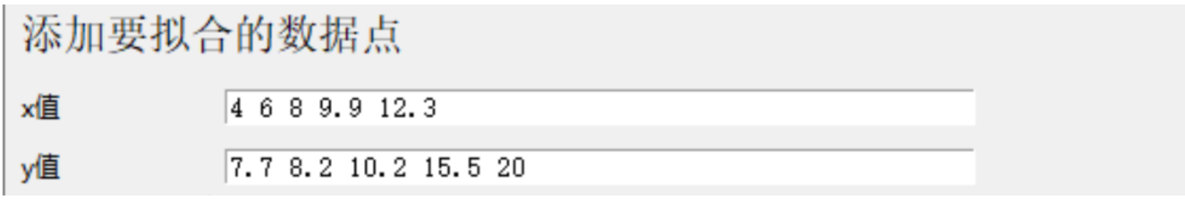
软件用法

手动输入数据:

假设有一些点 $(x_1,y_1),(x_2,y_2),...(x_n,y_n)$ 需要拟合

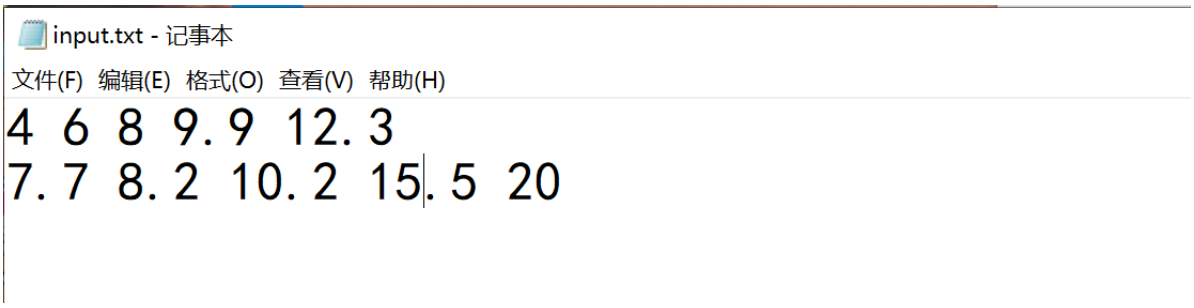
在界面的输入框分别输入 $x_1\quad x_2\quad ...\quad x_n$,$y_1\quad y_2\quad ...\quad y_n$

注意：中间用空格隔开



通过文件添加数据:

用txt方式，第一行放x数据，第二行放y数据，用空格隔开



然后选择拟合方式（有些拟合方式可能会弹出消息框输入额外参数）

然后点击 "**开始拟合**" , 即可在下面的框中生成图像。

性能分析

运行时间&准确性

在数据量小的时候运行良好，准确性高。若给的数据量太大，受限于Python语言本身的性能可能较慢，但总体可以接受。（5000个点用多项式拟合1s内出结果）但由于画布限制，可能一下显示这么多点显示效果会拥挤。

有时候算法的点给的很近（比如俩个x值分别是0.000001和0.000002，在计算机有限精度下，用某个数字（比如1000去除他们的差，结果就会很大，超出Python变量的保存上届，最后不能正确运行）

时间复杂度

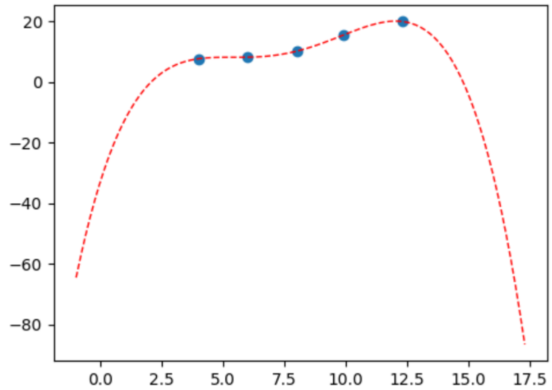
算法	时间复杂度	备注
拉格朗日插值	$O(n^2)$	两层循环，一层求各项之差，一层累加，时间复杂度即得
多项式拟合线性拟合	$O(n^3)$	因为采用高斯消元法解方程，其复杂度$O(n^3)$，因此最小二乘法的复杂度大体可表示为：$O(n^3)$
三次样条插值	$O(n)$	同样是解了矩阵方程，用了追赶法简化计算。

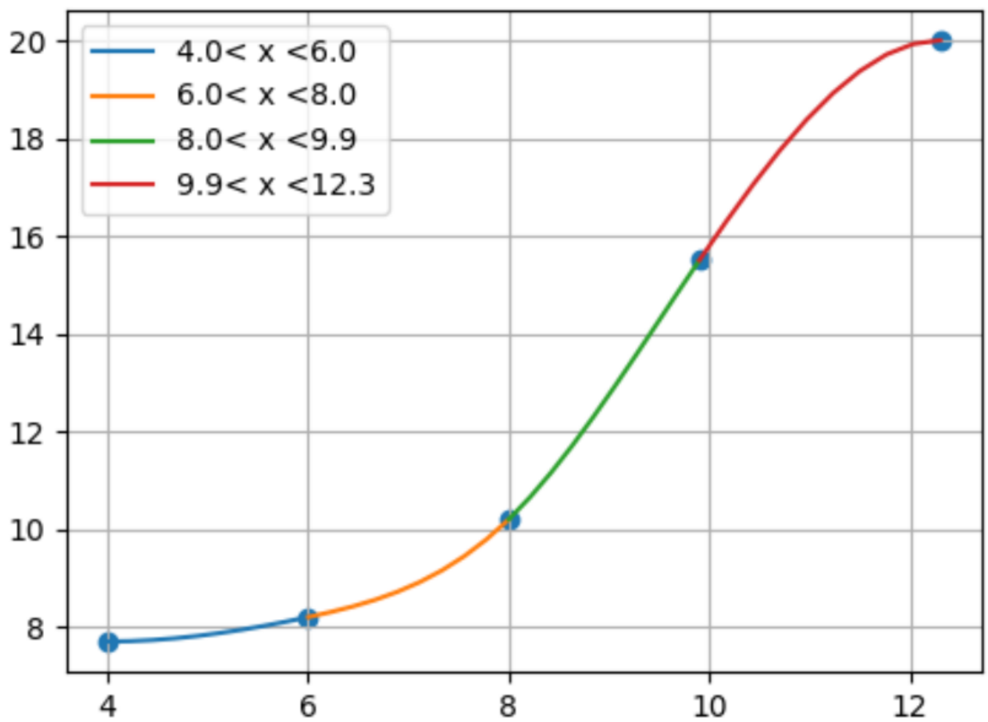
应用示例

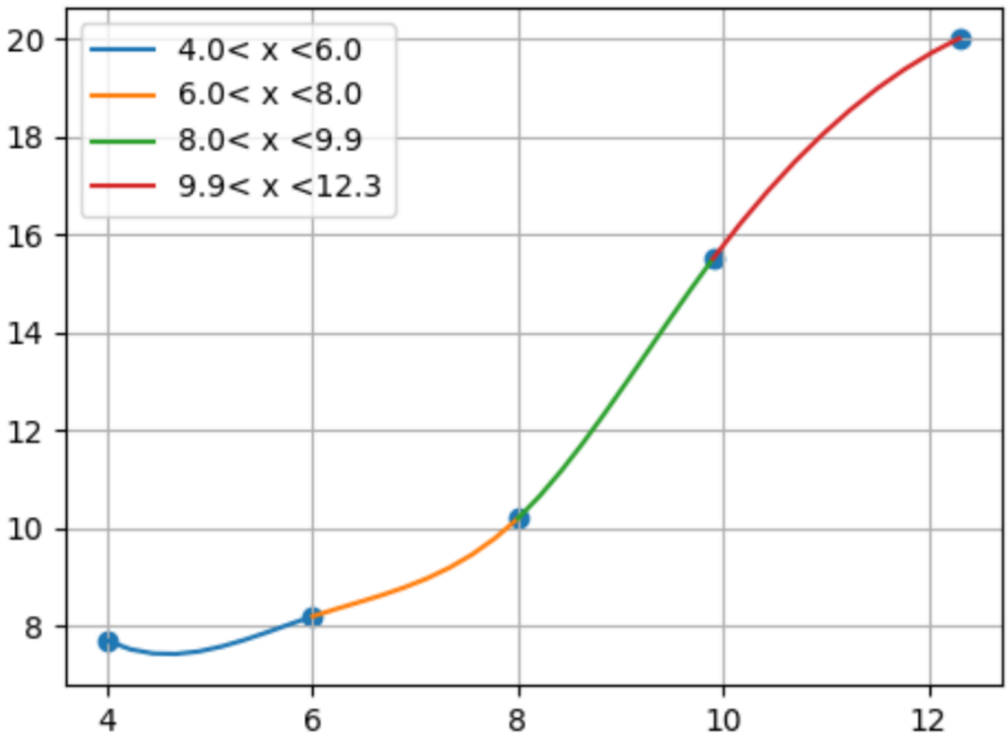
```
数据1
4 6 8 9.9 12.3
7.7 8.2 10.2 15.5 20
```

拟合图像按的顺序给出

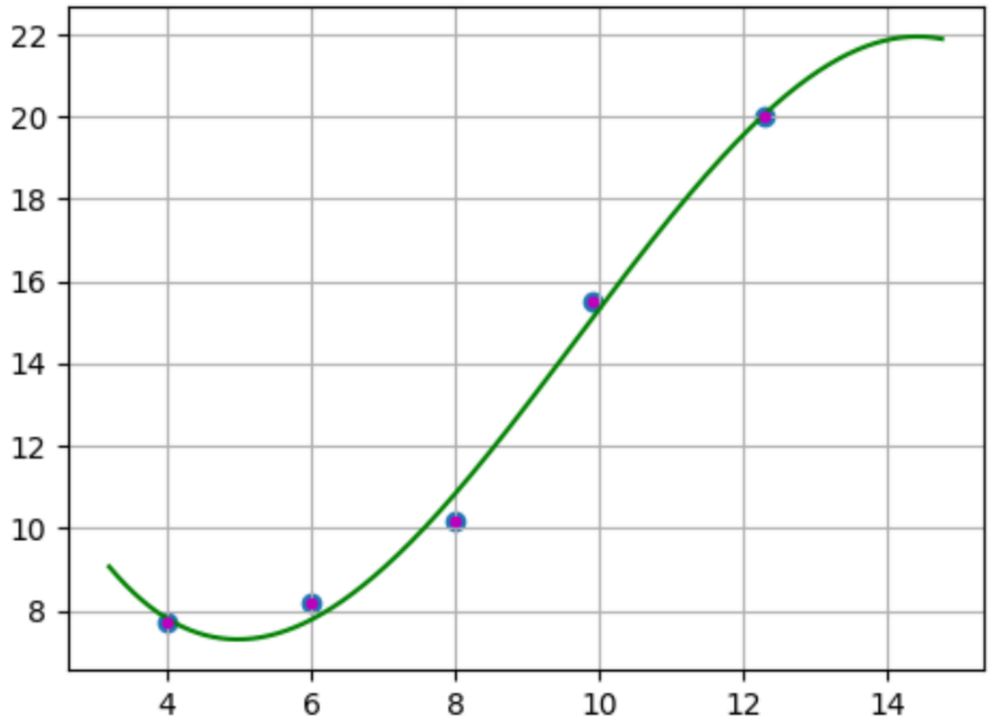
1. 拉格朗日拟合
2. 三次样条插值（自然条件）
3. 三次样条插值（压缩条件）（左右端点导数-1,1）
4. 多项式拟合
5. 线性拟合

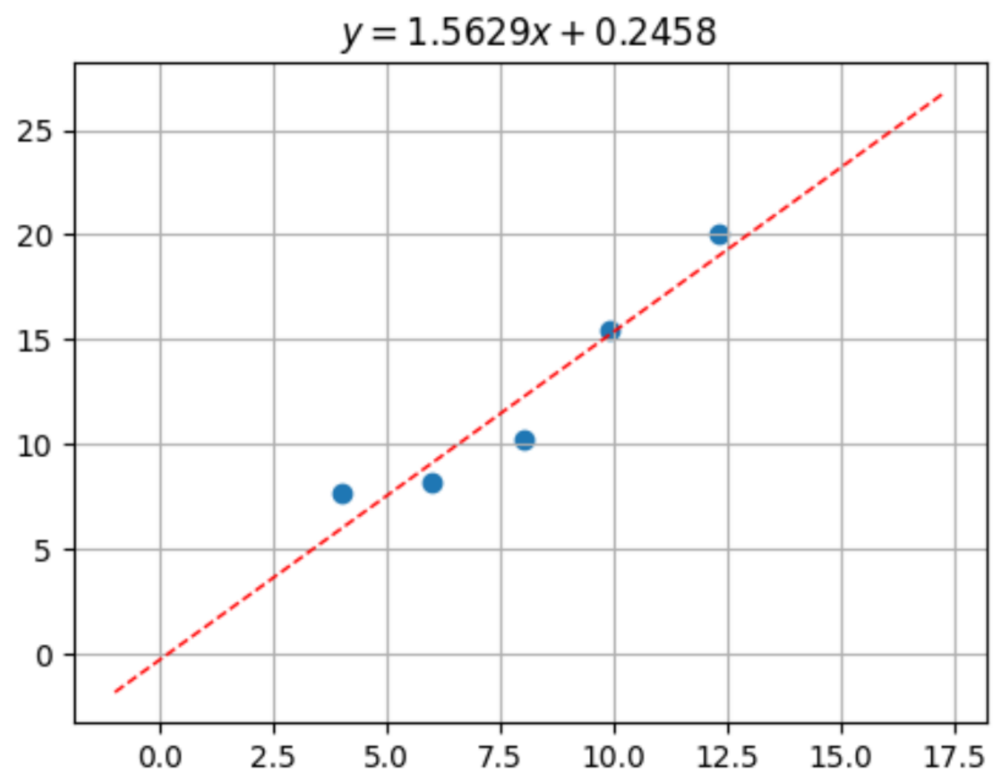




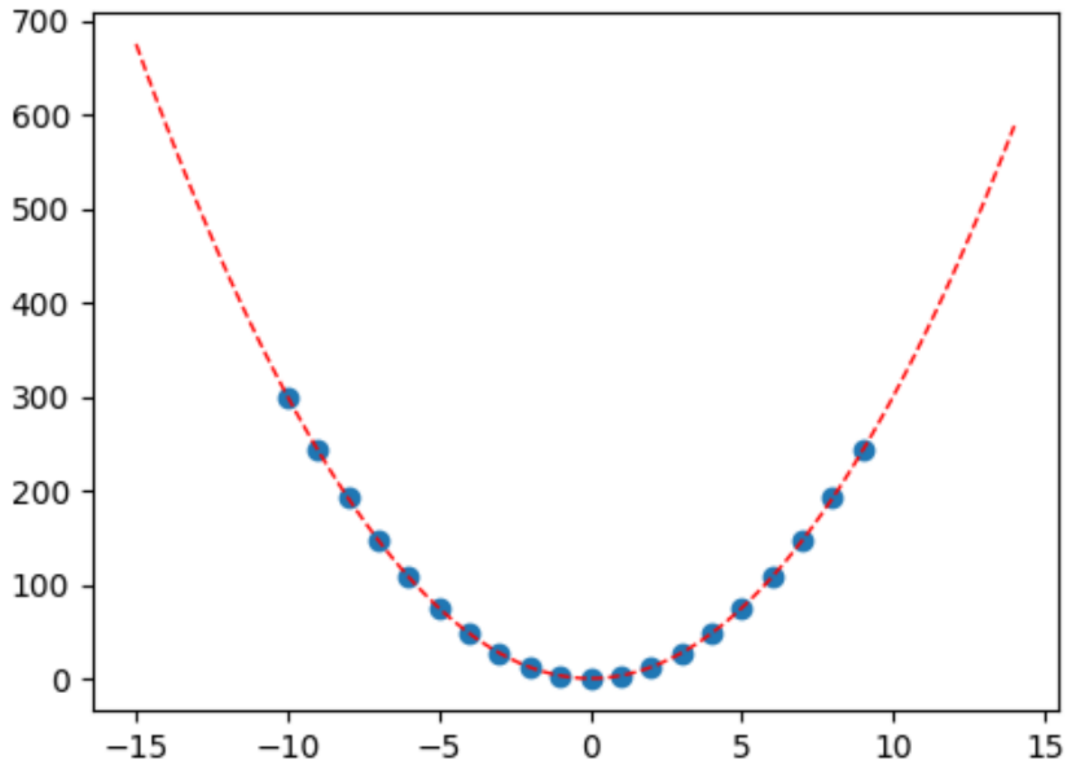


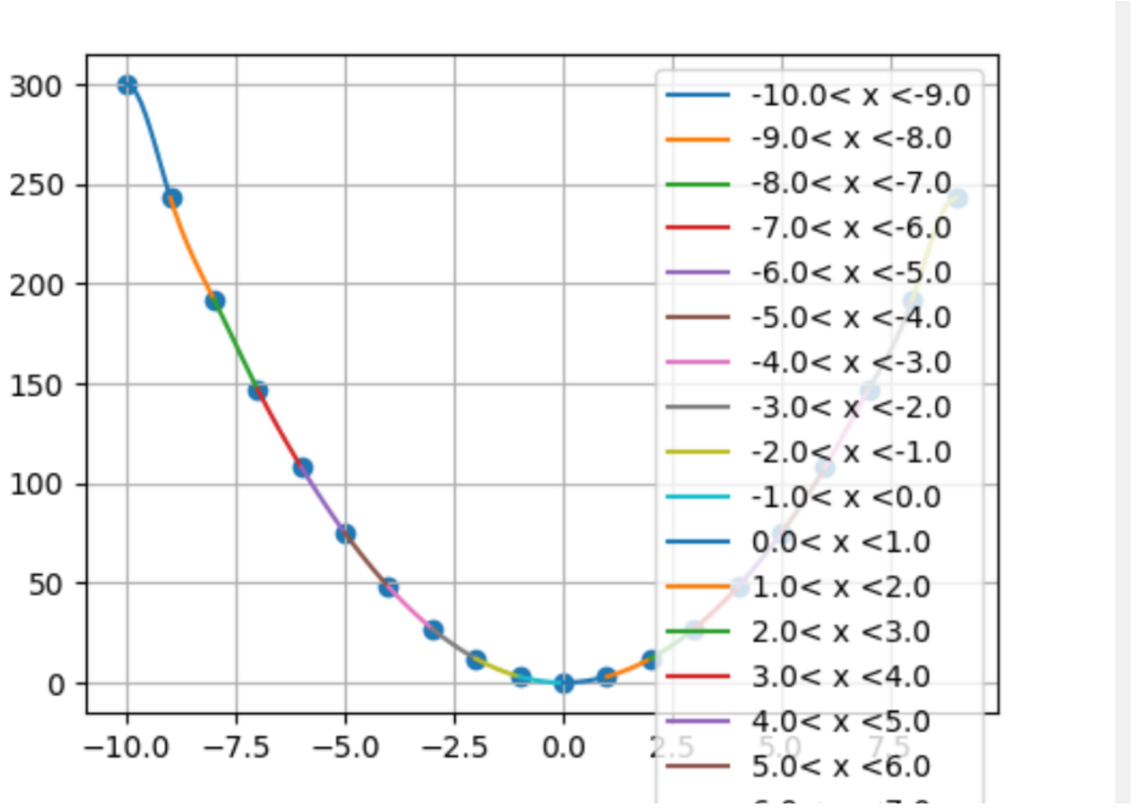
f(x)=(23.90)x^0+(-7.53)x^1+(1.02)x^2+(-0.03)x^3

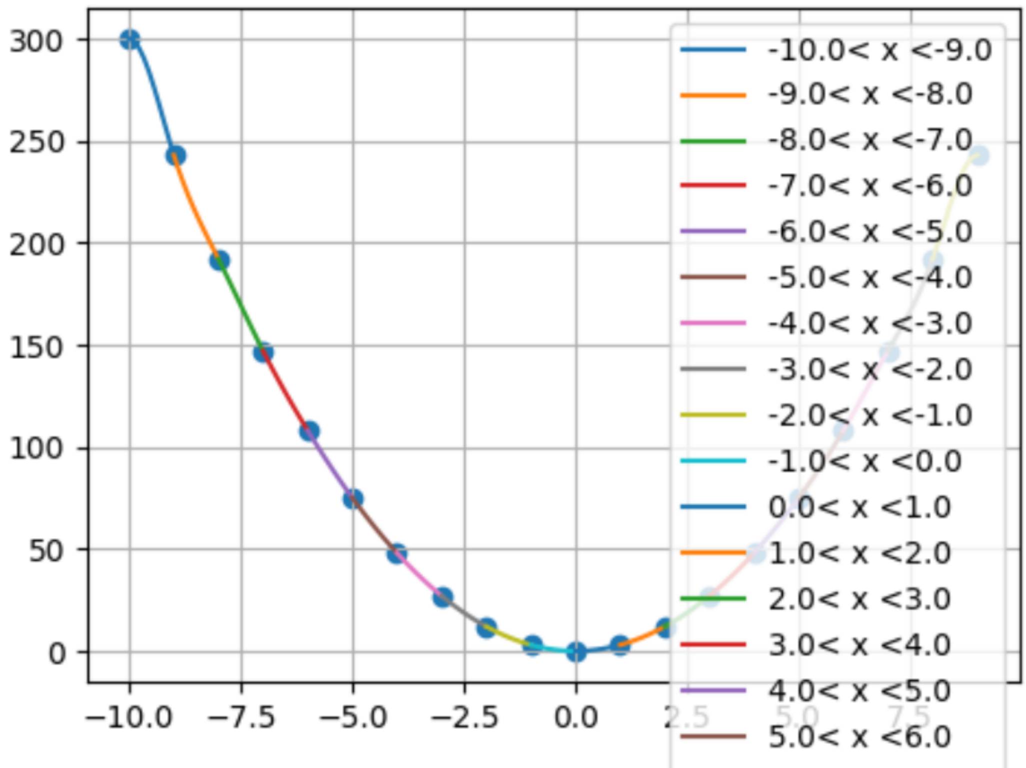


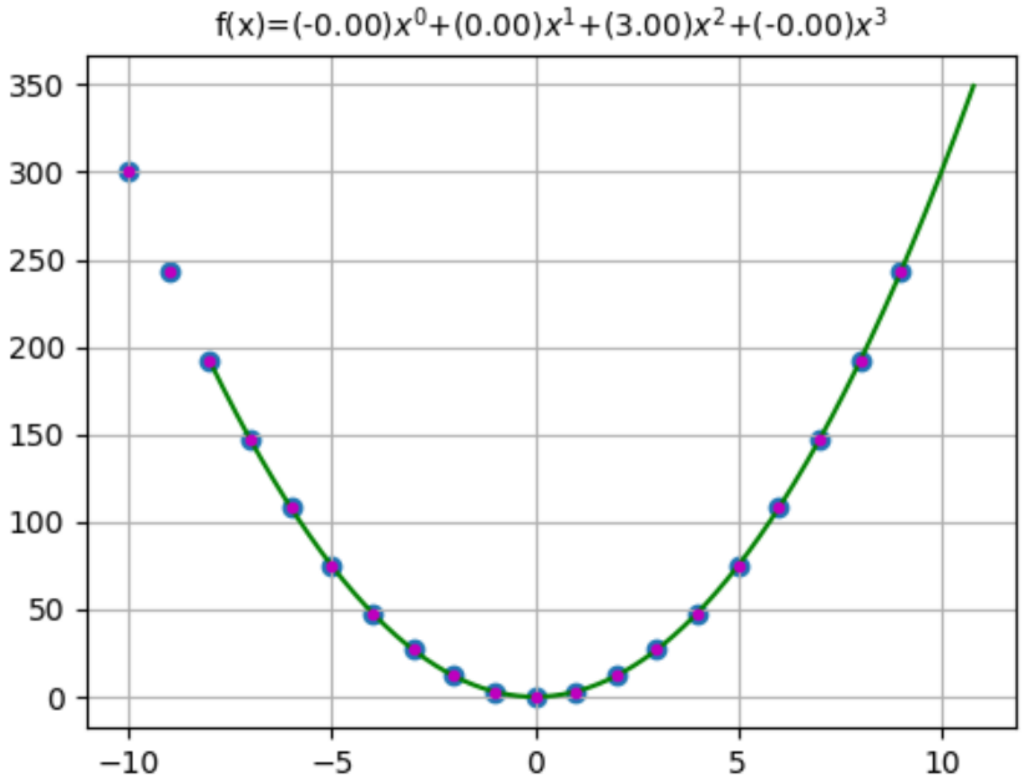


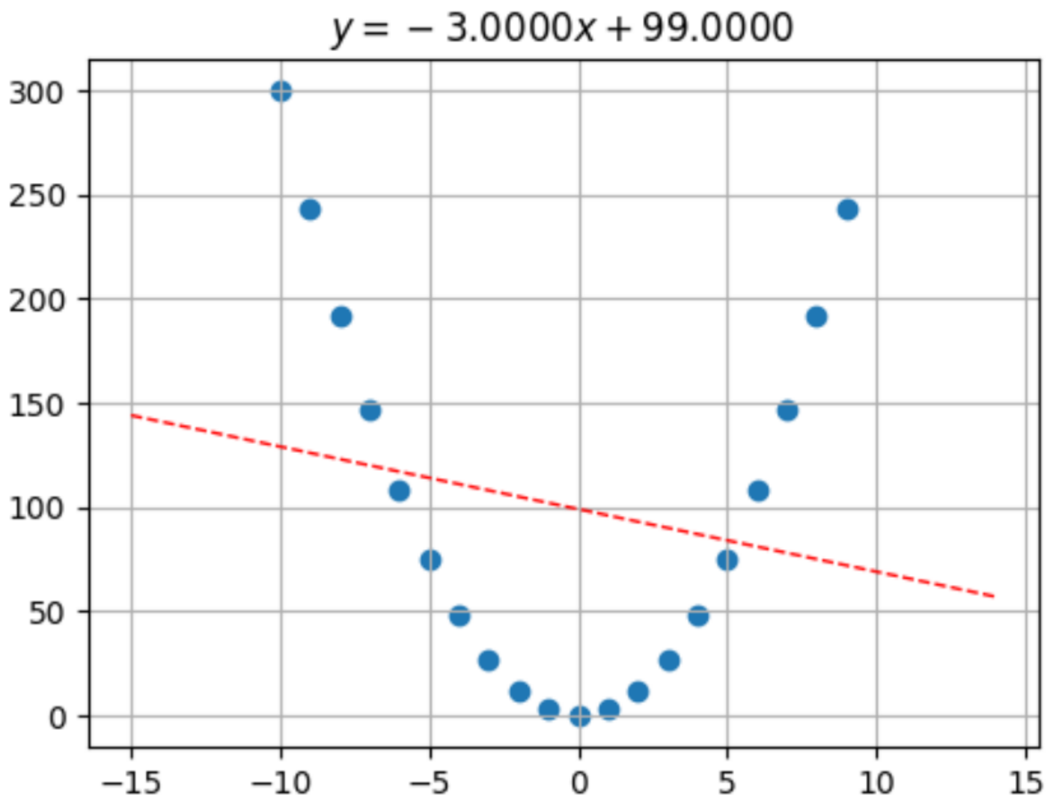

```
数据2
-10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9
300 243 192 147 108 75 48 27 12 3 0 3 12 27 48 75 108 147 192 243
```



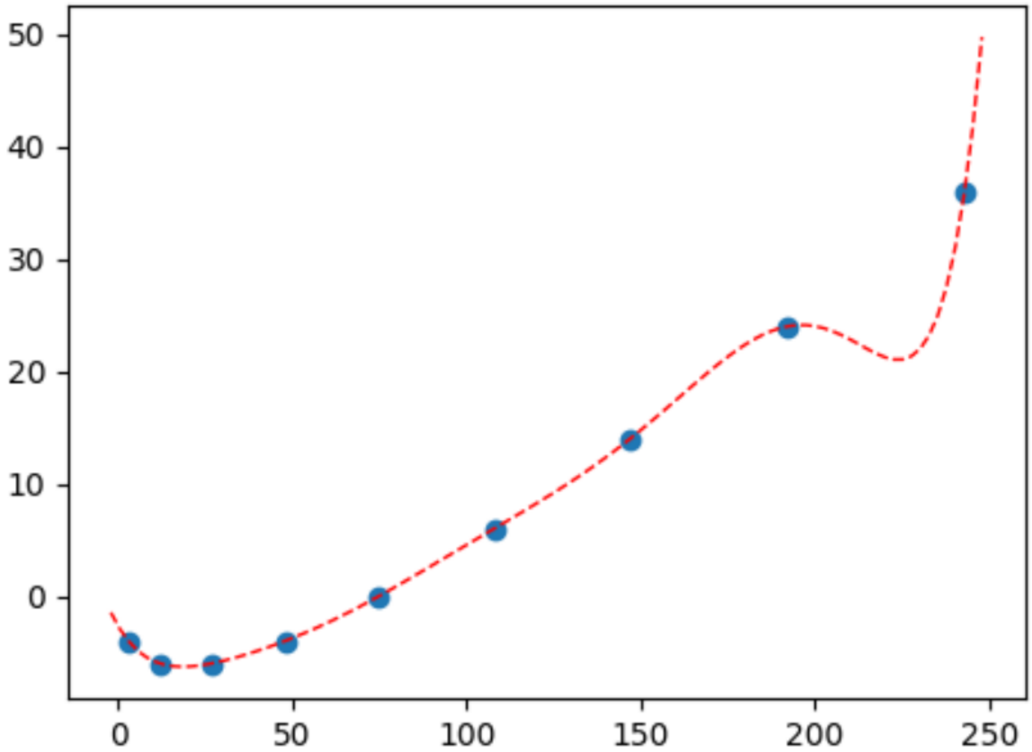

300
250
200
150
100
50
0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
-10.0< x <-9.0
-9.0< x <-8.0
-8.0< x <-7.0
-7.0< x <-6.0
-6.0< x <-5.0
-5.0< x <-4.0
-4.0< x <-3.0
-3.0< x <-2.0
-2.0< x <-1.0
-1.0< x <0.0
0.0< x <1.0
1.0< x <2.0
2.0< x <3.0
3.0< x <4.0
4.0< x <5.0
5.0< x <6.0


300
250
200
150
100
50
0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
-10.0< x <-9.0
-9.0< x <-8.0
-8.0< x <-7.0
-7.0< x <-6.0
-6.0< x <-5.0
-5.0< x <-4.0
-4.0< x <-3.0
-3.0< x <-2.0
-2.0< x <-1.0
-1.0< x <0.0
0.0< x <1.0
1.0< x <2.0
2.0< x <3.0
3.0< x <4.0
4.0< x <5.0
5.0< x <6.0

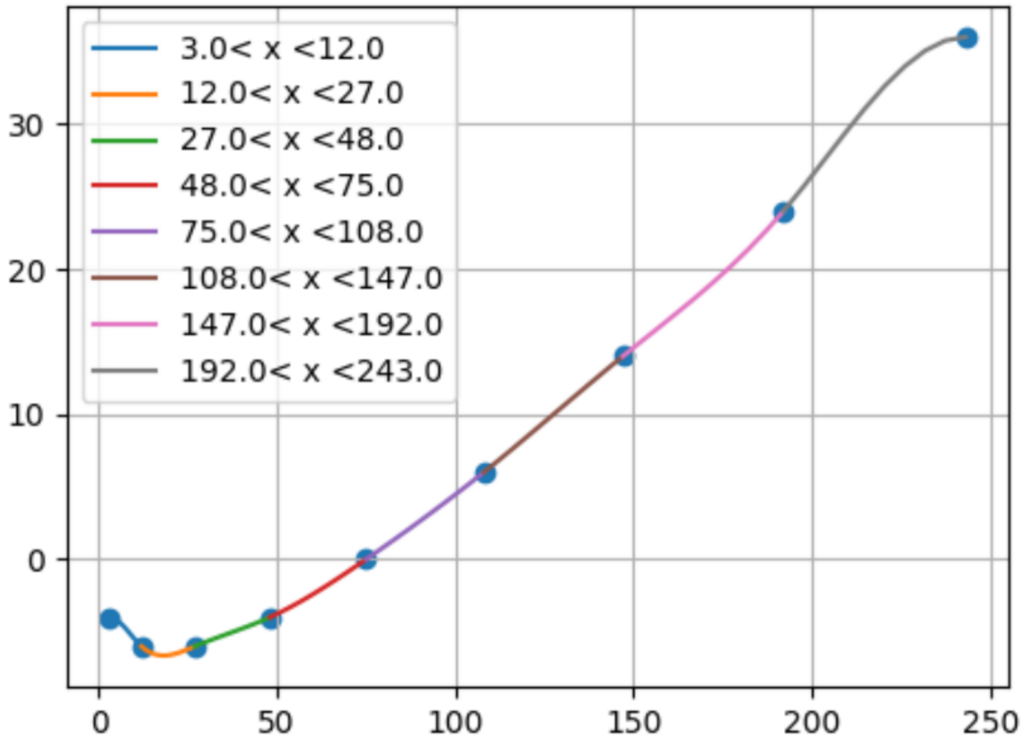




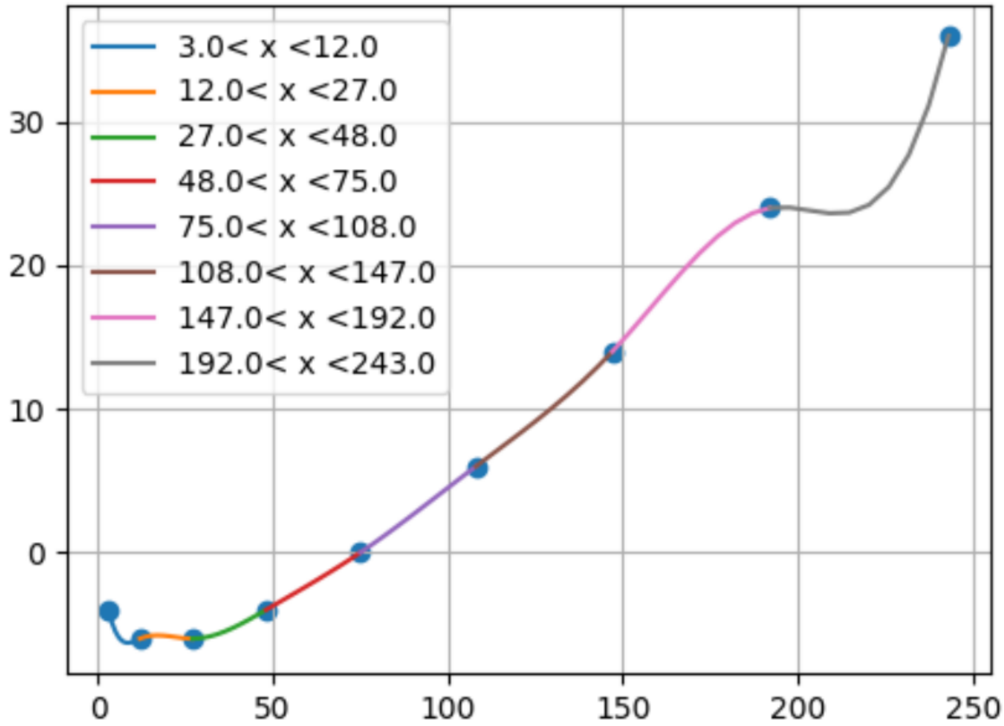

```
数据3
3 12 27 48 75 108 147 192 243
-4 -6 -6 -4 0 6 14 24 36
```

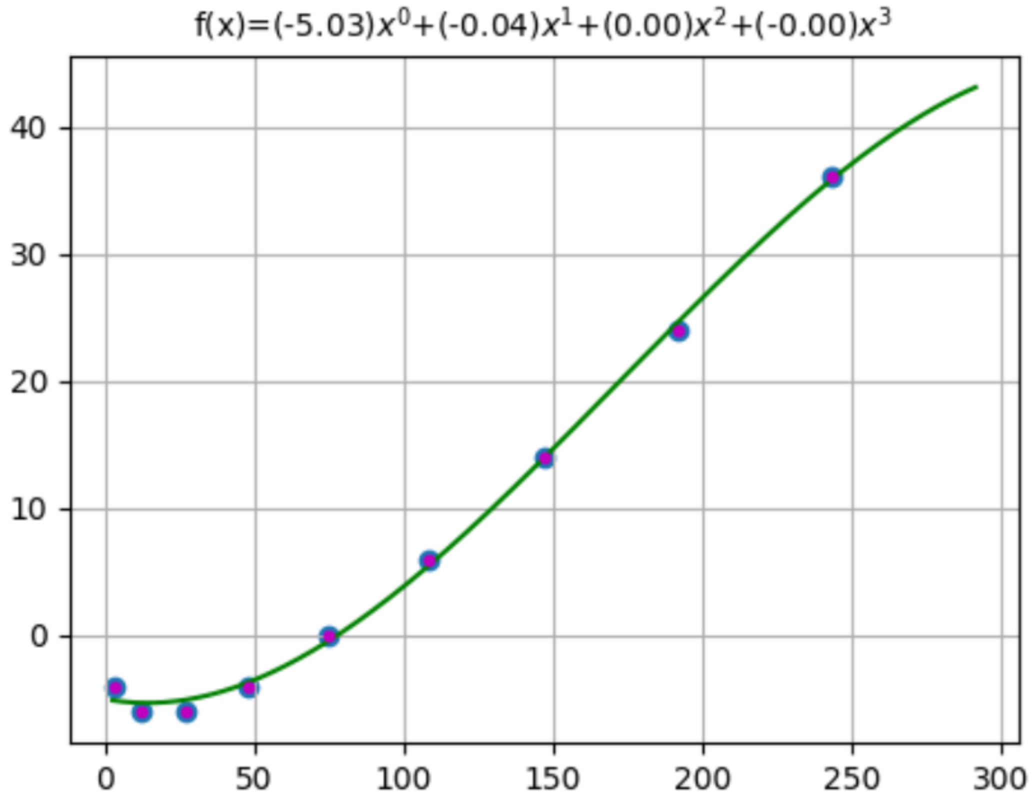


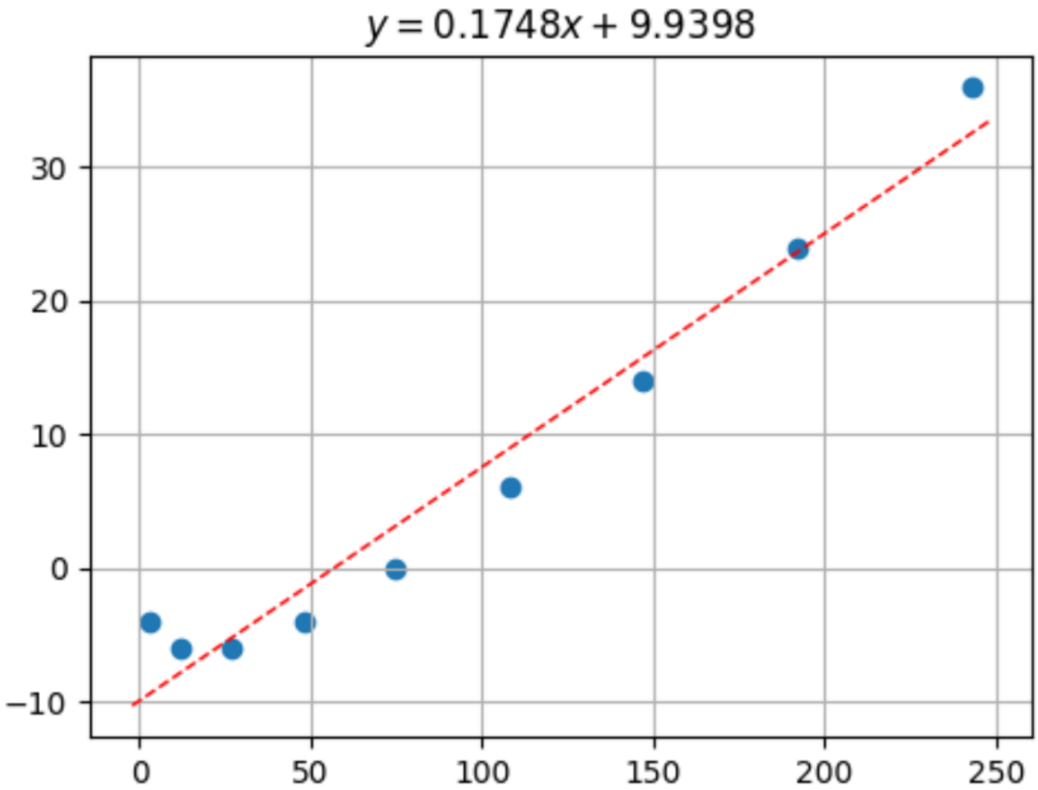
50
40
30
20
10
0
0
50
100
150
200
250



3.0< x <12.0
12.0< x <27.0
27.0< x <48.0
48.0< x <75.0
75.0< x <108.0
108.0< x <147.0
147.0< x <192.0
192.0< x <243.0
30
20
10
0
0
50
100
150
200
250


3.0< x <12.0
12.0< x <27.0
27.0< x <48.0
48.0< x <75.0
75.0< x <108.0
108.0< x <147.0
147.0< x <192.0
192.0< x <243.0


f(x)=(-5.03)x^0+(-0.04)x^1+(0.00)x^2+(-0.00)x^3



$y = 0.1748x + 9.9398$
30
20
10
0
−10
0
50
100
150
200
250