《计算方法》上机作业报告

学生信息：

|  |  |  |
| --- | --- | --- |
| 姓名：何宇昊 | 学号：3190102182 | 授课教师：郑太英 |

## 上机作业内容：

**最小二乘法多项式拟合**

## 算法目的和意义

**多项式插值存在缺陷：**

由实验提供的数据通常带有测试误差。如果要求近似曲线y=p(x)严格地通过所给的每个数据点（xi,yi），就会使曲线保留着原有的测试误差。当个别数据的误差较大时，插值效果显然是不理想的。

**最小二乘的目的：**

在一般情况下，不能要求近似曲线*y*=*p*(*x*)严格地通过所有数据点（*xi*,*yi*）,亦即不能要求拟合函数在*xi*处的残差（亦称偏差）都严格地等于零。但是，为了使近似曲线能尽量反映所给数据点的变化趋势，要求|都较小还是需要的。为了便于计算、分析与应用，较多地根据“**使残差平方和最小**”的原则（称为最小二乘原则）来选取拟合曲线y=p（x），使用这个原则选择拟合曲线的方法就称作最小二乘法。

## 计算公式

由于本次实验中不考虑具体的实际原因，单纯使用多项式进行拟合，因此无需考虑其他的线性无关的函数族。





其中定义为内积，解出上面这个法方程组就可以得到最小二乘多项式拟合的系数。

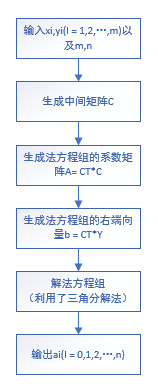
## 程序设计

本次程序设计中有如下这些需要注意的：

1. 考虑到用多项式进行拟合，在构建中间矩阵的时候就可以较为方便地使用整行整列进行赋值。
2. matlab集成好了对于矩阵的相关运算，在生成法方程组的时候可以巧妙设计矩阵的乘法进行计算
3. （本次代码中没有实现），考虑到线性无关的函数族其实可以手动输入而不是代码内实现，可以通过输入函数转化为函数句柄形成函数句柄的数列，将其作为一个算子矩阵，从而实现自定义函数族解最小二乘法。

函数运行过程如下程序框图所示，解法方程组调用了之前的三角分解法，因此有一部分添加了路径。

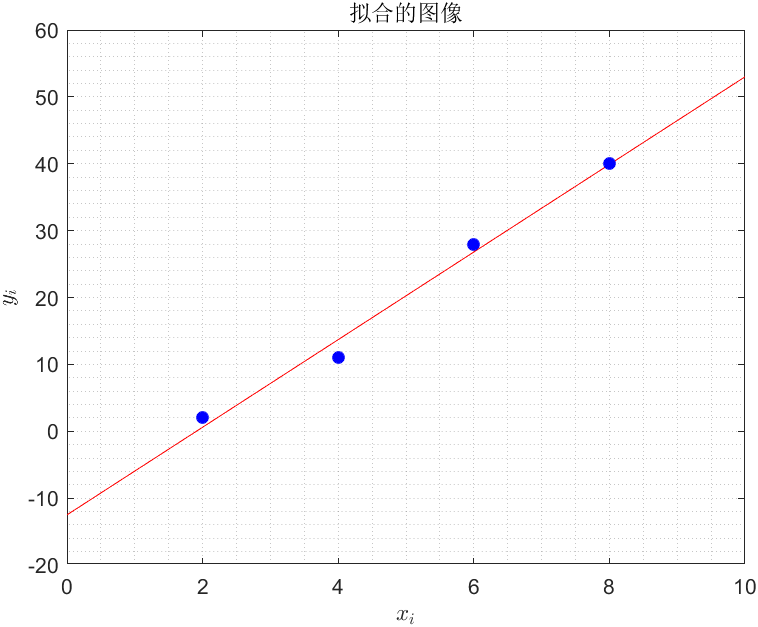
程序框图：



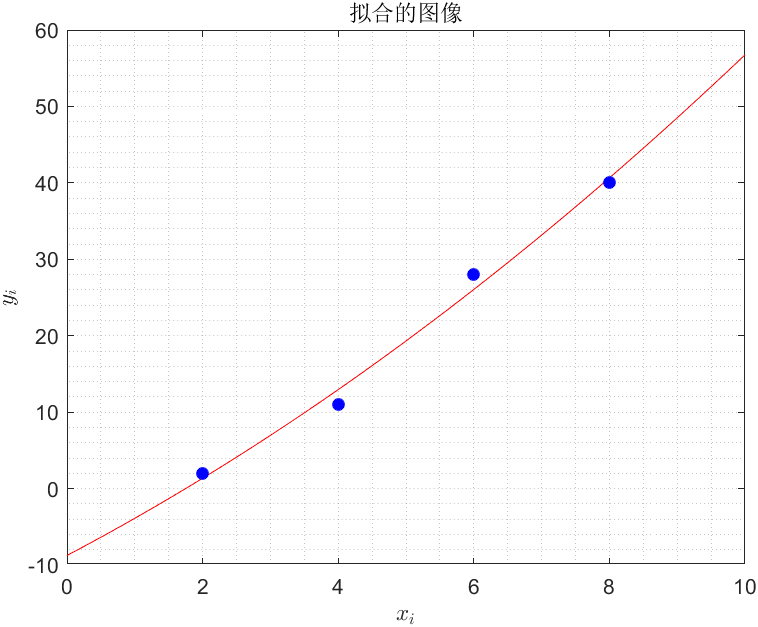
## 结果讨论与分析

对课本P89 第1题进行计算求解可以得到：

进行一阶拟合的最小二乘系数分别为-12.5，6.55，对应的函数为



进行二阶拟合的最小二乘系数分别为-8.75，4.675，0.1875对应的函数为



**程序评价：**

经过验证，算法结果无误。对程序的评价而言，比较好的完成了任务。

亮点：

1. 提出了一种利用自定义函数族的方法进行最小二乘的思路，实现起来非常简单。
2. 画出了图像，可以更好地看出拟合结果。
3. 利用了之前的成果，代码整体性更强
4. 巧妙地利用matlab对于函数运算的方便之处，更多设计了矩阵的乘法进行运算

## Appendix (coding in matlab)

【主函数Lab\_2.m】

clc;

clear;

%由于程序中用到了之前写的三角分解法，因此需要添加路径，前提是解压的时候是一起的，相对路径没有改变

addpath('../simplification\_of\_Matrix/');

level = input("请输入你想要拟合的多项式阶数\n");

TestSet = input("请根据\n[ x\_0, y\_0;\n  x\_1, y\_1;\n  ..., ...;\n  x\_n, y\_n ]\n的格式输入用于拟合的点集\n");

sizeofSet = size(TestSet);

if sizeofSet(1)<=level

%   判断如果输入的点集数量小于等于需要拟合的阶数，就无法通过最小二乘实现这点

    fprintf(' \n>> WARNING! 提供的点不足以拟合该阶曲线 <<\n\n');

    return;

end

% disp(TestSet);

A = LS\_Method(level,TestSet);

disp(A);

%%

% 往下是生成函数结果图像反馈的代码，

Min = min(TestSet(:,1)) - 2;

Max = max(TestSet(:,1)) + 2;

px = Min:0.1:Max;

py = zeros(size(px));

for i = level+1:-1:1

    py = py.\*px + A(i);

end

plot(px,py,'-r');

hold on;

plot(TestSet(:,1),TestSet(:,2),'\*b','linewidth',2);

hold off;

grid minor;

xlabel('$x\_i$','interpreter','latex');

ylabel('$y\_i$','interpreter','latex');

title("拟合的图像");

【最小二乘法函数】

function A = LS\_Method(level,testset)

    sizeofSet = size(testset);

%     提取出测试点集的y

    y = testset(:,2);

%     初始化中间矩阵c

    c = ones(sizeofSet(1),level+1);

%     利用测试点集给中间矩阵赋值，由于是多项式拟合所以可以这么暴力，如果是自定义函数集合，可以利用传入的函数句柄矩阵和点集的x进行计算赋值

    for cols = 2:level+1

        c(:,cols) = testset(:,1).^(cols-1);

    end

    C = c'\*c;

%     disp(C);

    Y = c'\*y;

%     disp(Y);

% 将处理好的C和Y矩阵合在一起形成可以传给自己写的三角分解法的参数形式

    all = [C,Y];

%     disp(all);

% 返回值即为所求

    [~,~,A] = Tri\_decomposition(all,level+1);

end