**综合实验题3\_实验报告**

贾树傲 2023210131号 2023211105班

**问题简述**

某地区作物生长所需的营养素主要是氮(N),磷(P),钾(K)。某作物研究所在该地区对土豆与生菜作了一定数量的实验，实验数据如下列表格所示，其中 ha 表示公顷，t 表示吨，kg 表示公斤。当一个营养素的施肥量变化时，总将另二个营养素的施肥量保持在第七个水平上。

通常情形，农作物的种植，氮、磷、钾 3 种肥料缺一不可，而每一种肥料元素的施用也不是多多益善，单一肥料施用量与农作物的产量曲线通常是“上单峰”式样，特别当释放量趋于无穷时，农作物产量会趋于“0”——这样也意味着，选择多项式函数拟合农作物产量与施肥组合的关联是不适宜的。

**实验目的与内容**

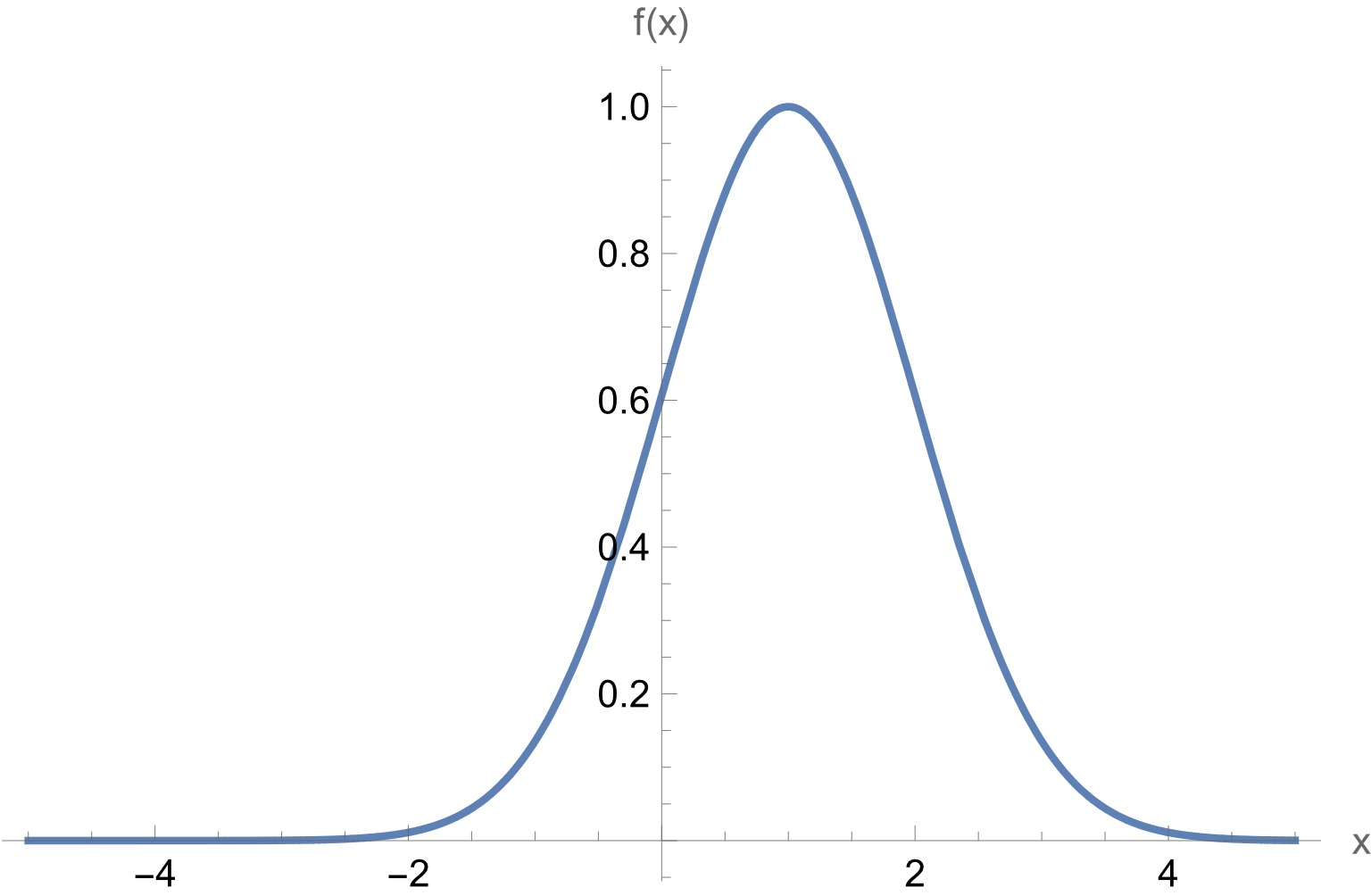
（1）试分析施肥量与生产量之间关系，并队对得结果从应用价值与如何改进等方面做出估价。

（2）请在“土豆”、“生菜”二者当中任选其一，基于赛题提供的数据，选择适 宜的拟合函数，对赛题给出尽可能完善的解答。

**实验思路**

由于实验目的是拟合施肥量与生产量的函数关系，并求出使得生产量最大时的施肥量和最大生产量。若分别对三种肥料采用多项式拟合，在进行“求出使得生产量最大时的施肥量和最大生产量”这一步骤时，调用Mathematica内置的FindMax[]函数的结果是：氮、磷、钾的施肥量都趋向于无穷大，生产量也趋向于无穷大，显然，这与——当释放量趋于无穷时，农作物产量会趋于“0”——这一事实是不符合的。我于是采用“上单峰”式样的函数去进行拟合，但这样拟合的结果也不是完美的，实验表明：采用“上单峰”式样的函数去进行拟合的结果还需要加上一个偏置作为补偿。此外，我还尝试利用神经网络去拟合施肥量与生产量的函数关系。

**实验过程（算法与程序）**



我采用的用于拟合的“上单峰”式样的函数**（高斯函数）**如上图所示，

### 除这个函数外，我还对P磷肥尝试了利用洛伦兹函数去做拟合，效果好于高斯函数。

**（1）关于N做拟合**

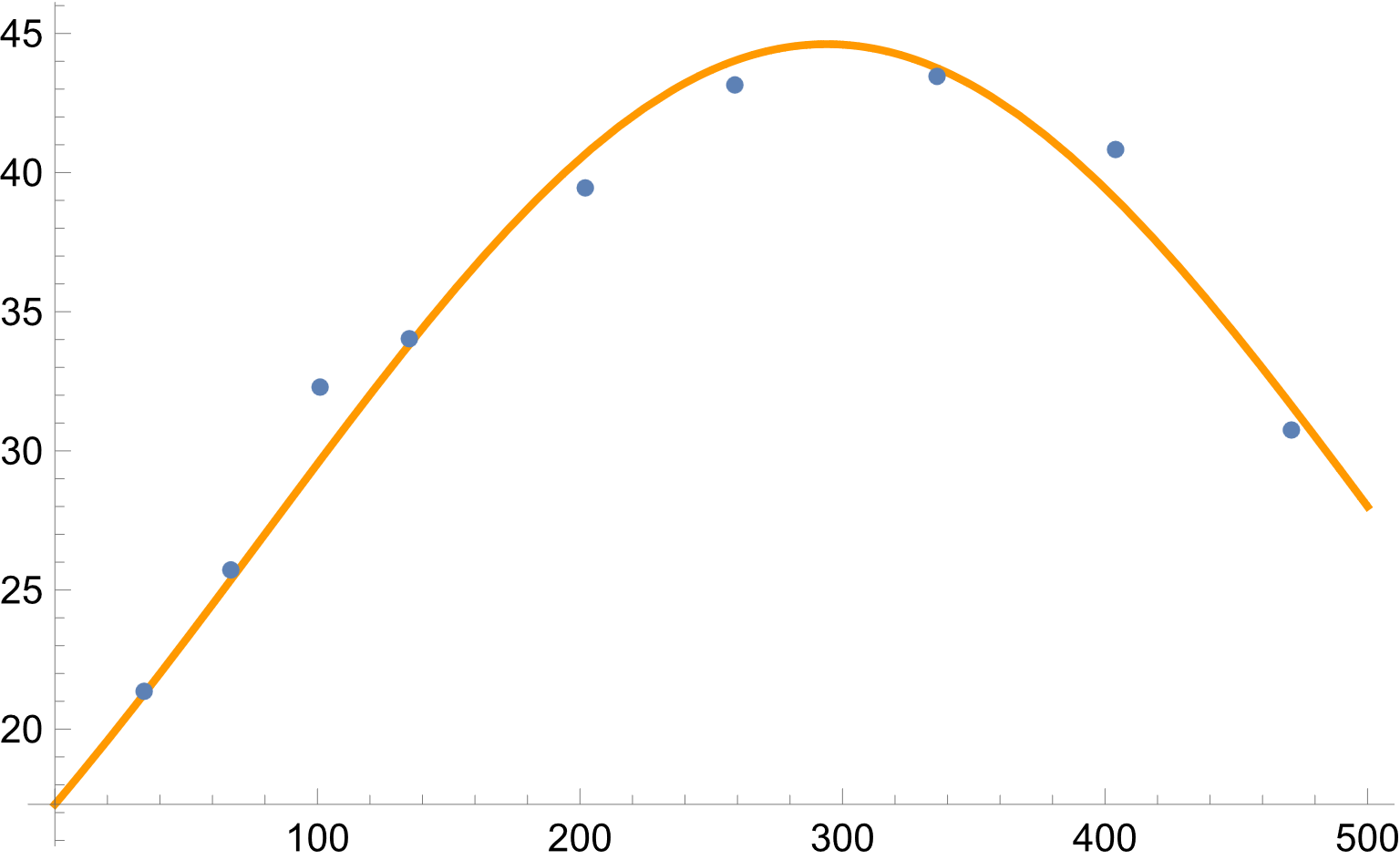
slu=FindFit[dataN,aa\*Exp[-(t-uu)^2/(2\*cc^2)],{aa,uu,cc},{t},PrecisionGoal->20,WorkingPrecision->4];

(\*在FindFit函数中添加PrecisionGoal和WorkingPrecision选项，以使用更高的精度进行拟合计算。这可以减少机器精度溢出的可能性。\*)

exp03=(aa\*Exp[-(x-uu)^2/(2\*cc^2)])/.slu

G03=Plot[exp03,{x,0,500},PlotStyle->Hue[0.1]];Show[{G03,GN}]

拟合出的函数：



——上图是拟合出的函数图像

**（二）关于P做拟合（洛伦兹函数）**

GP=ListPlot[dataP];

slu=FindFit[dataP,aa/(1+((x-bb)/dd)^2),{aa,bb,dd},{x},PrecisionGoal->20,WorkingPrecision->4]

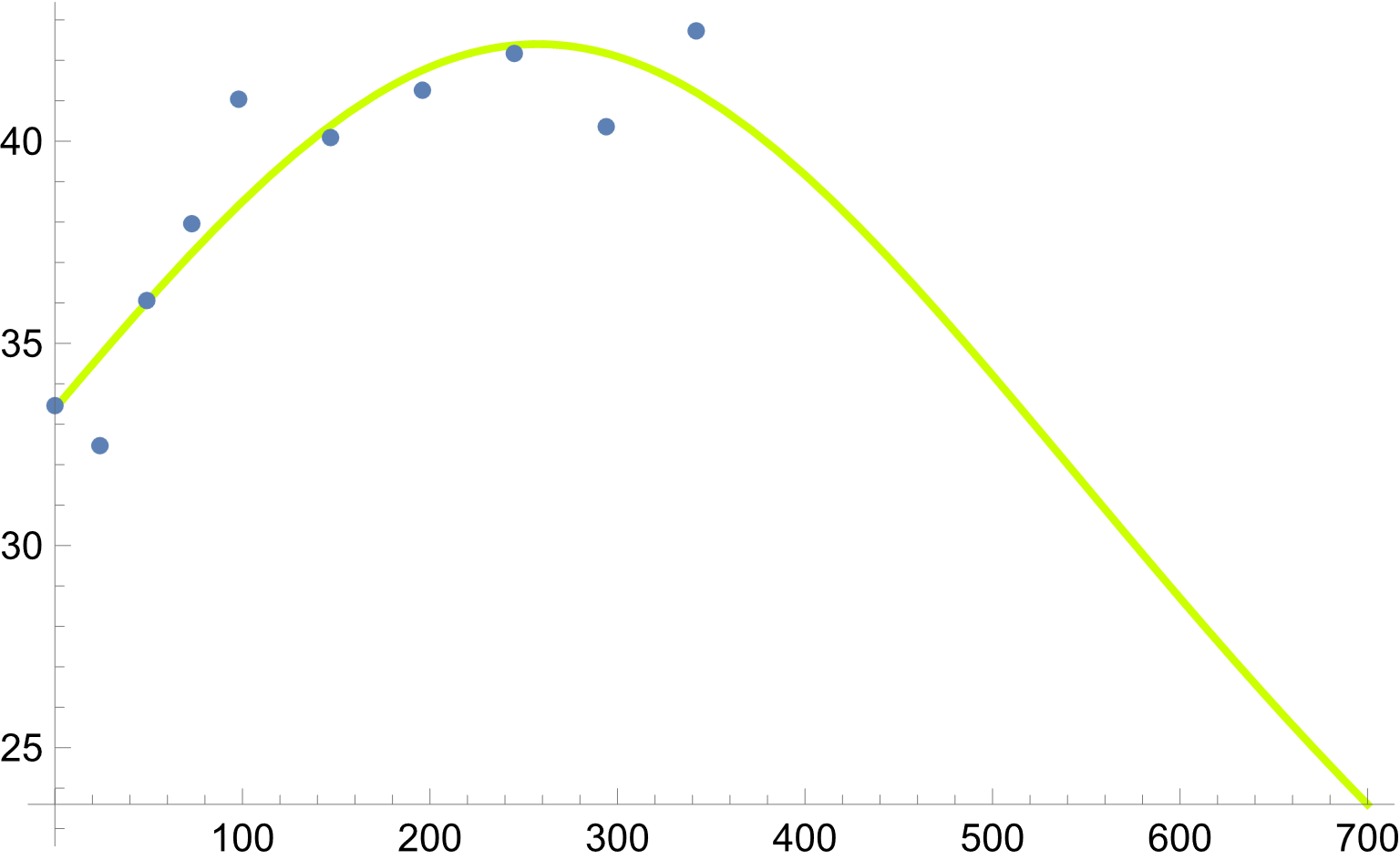
(\*在FindFit函数中添加PrecisionGoal和WorkingPrecision选项，以使用更高的精度进行拟合计算。这可以减少机器精度溢出的可能性。\*)

exp04=(aa/(1+((x-bb)/dd)^2))/.slu

G04=Plot[exp04,{x,0,700},PlotStyle->Hue[0.2]];

Show[{G04,GP}]

函数表示式：



——这是函数图像

**（三）关于K做拟**

GK=ListPlot[dataK];

slu=FindFit[dataK,aa\*Exp[-(t-uu)^2/(2\*cc^2)],{aa,uu,cc},{t},PrecisionGoal->20,WorkingPrecision->4];

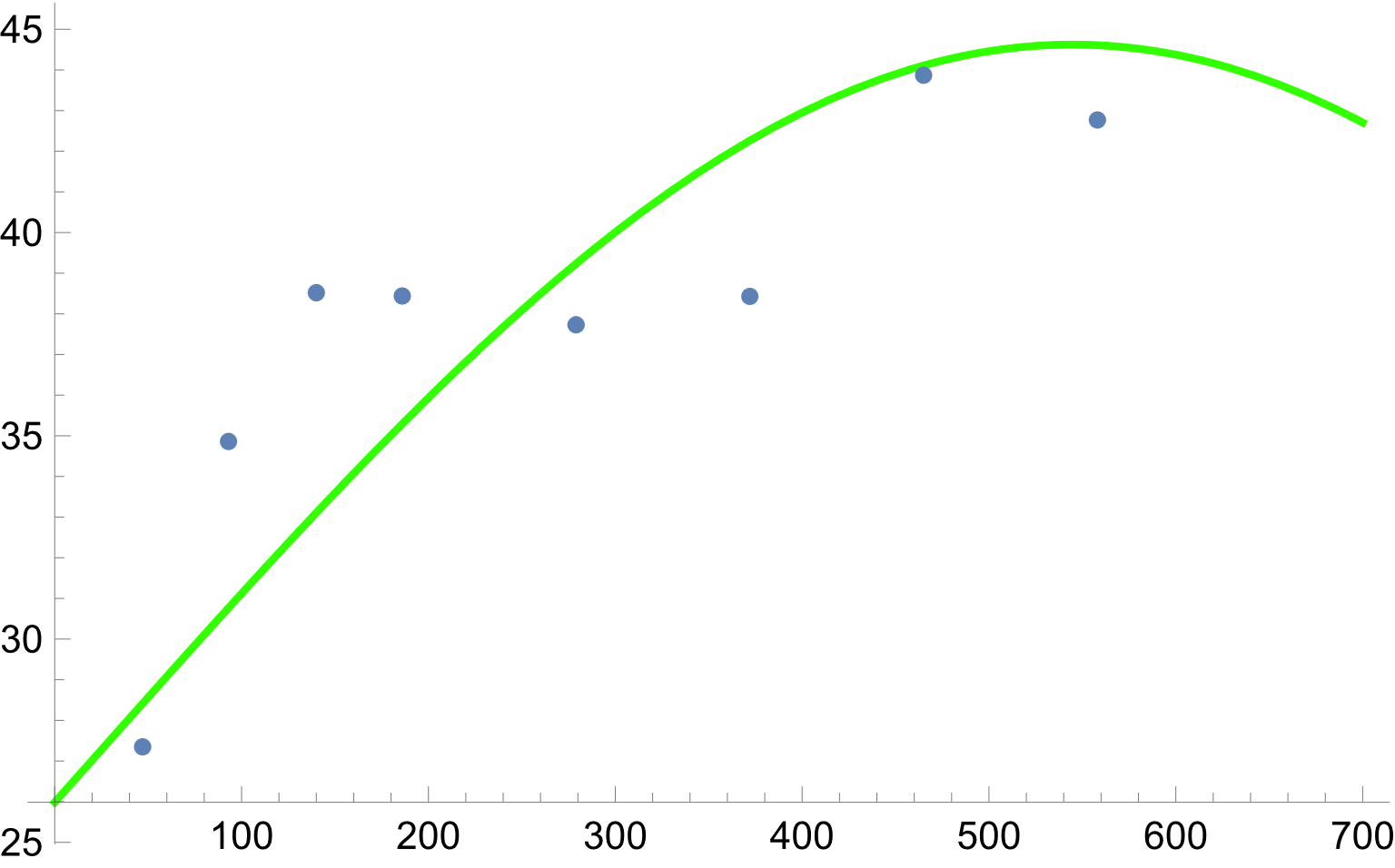
(\*在FindFit函数中添加PrecisionGoal和WorkingPrecision选项，以使用更高的精度进行拟合计算。这可以减少机器精度溢出的可能性。\*)

exp05=(aa\*Exp[-(x-uu)^2/(2\*cc^2)])/.slu

G05=Plot[exp05,{x,0,700},PlotStyle->Hue[0.3]];

Show[{G05,GK}]

函数表示式：



——这是函数图像

（4）定义三元目标函数

ObjectiveFunction[xN\_,xP\_,xK\_]:=Module[{},((exp03/.x->xN)+(exp04/.x->xP)+(exp05/.x->xK))-2/3(dataN[[7,2]]+dataP[[7,2]]+6+dataK[[7,2]])];

GGN1=Plot[ObjectiveFunction[u,dataP[[7,1]],dataK[[7,1]]],{u,0,500}];

Show[{GGN1,GN}]

GGN2=Plot[ObjectiveFunction[dataN[[7,1]],u,dataK[[7,1]]],{u,0,500},PlotStyle->Hue[0.8],PlotRange->All,PlotPoints->100,MaxRecursion->5];

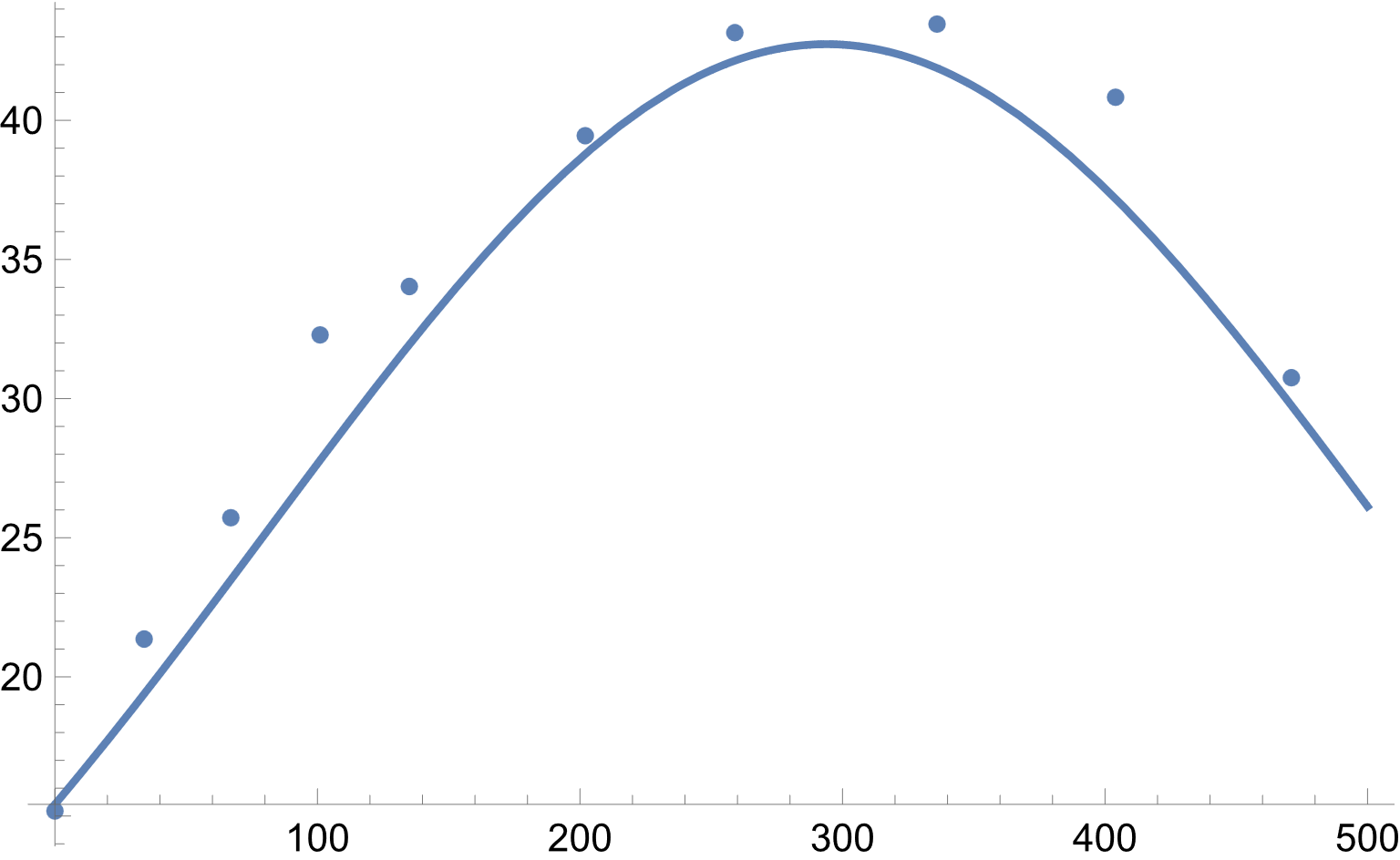
Show[{GGN2,GP}]

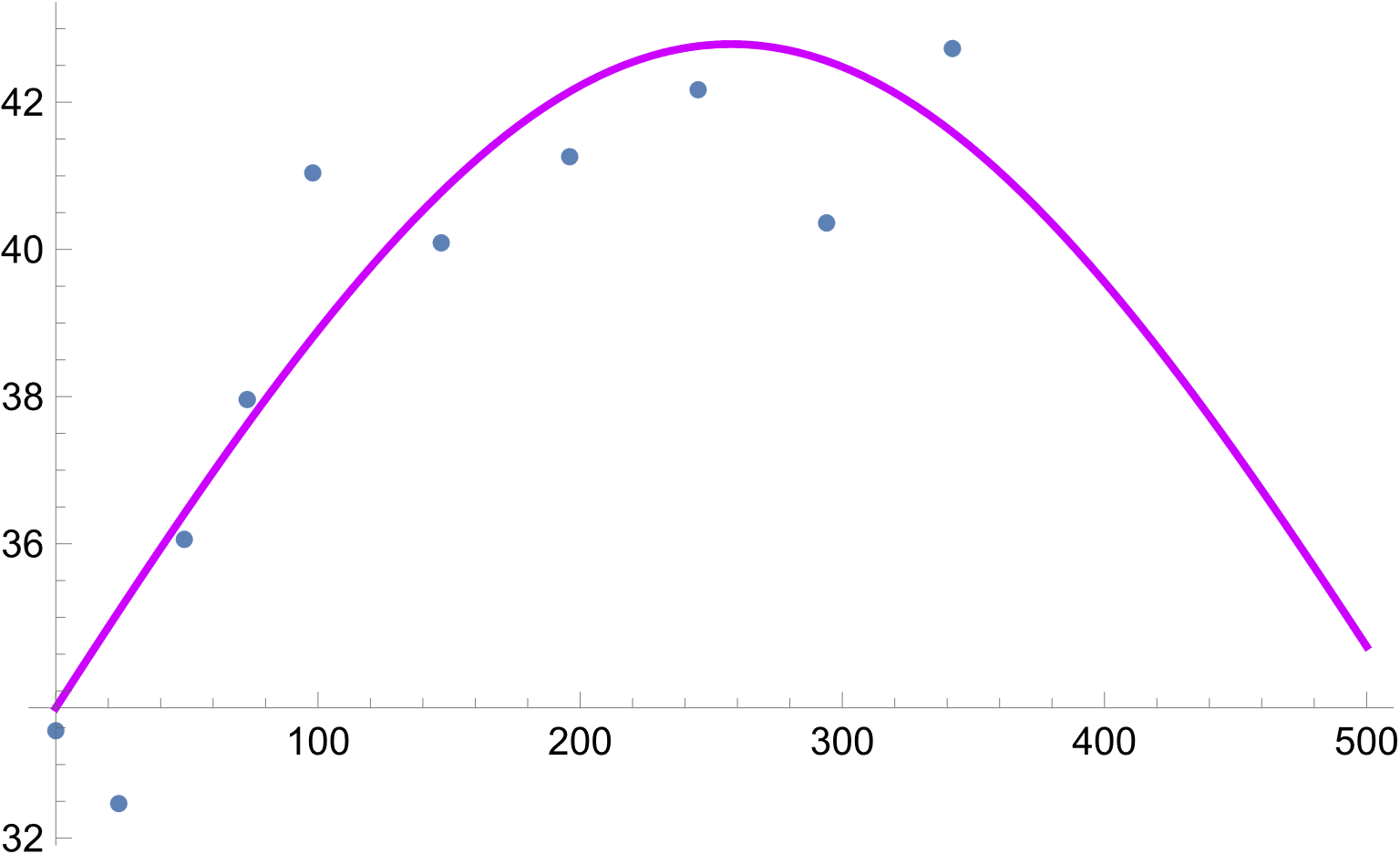
GGN3=Plot[ObjectiveFunction[dataN[[7,1]],dataP[[7,1]],u],{u,0,700},PlotStyle->Hue[0.8],PlotPoints->100,MaxRecursion->5];

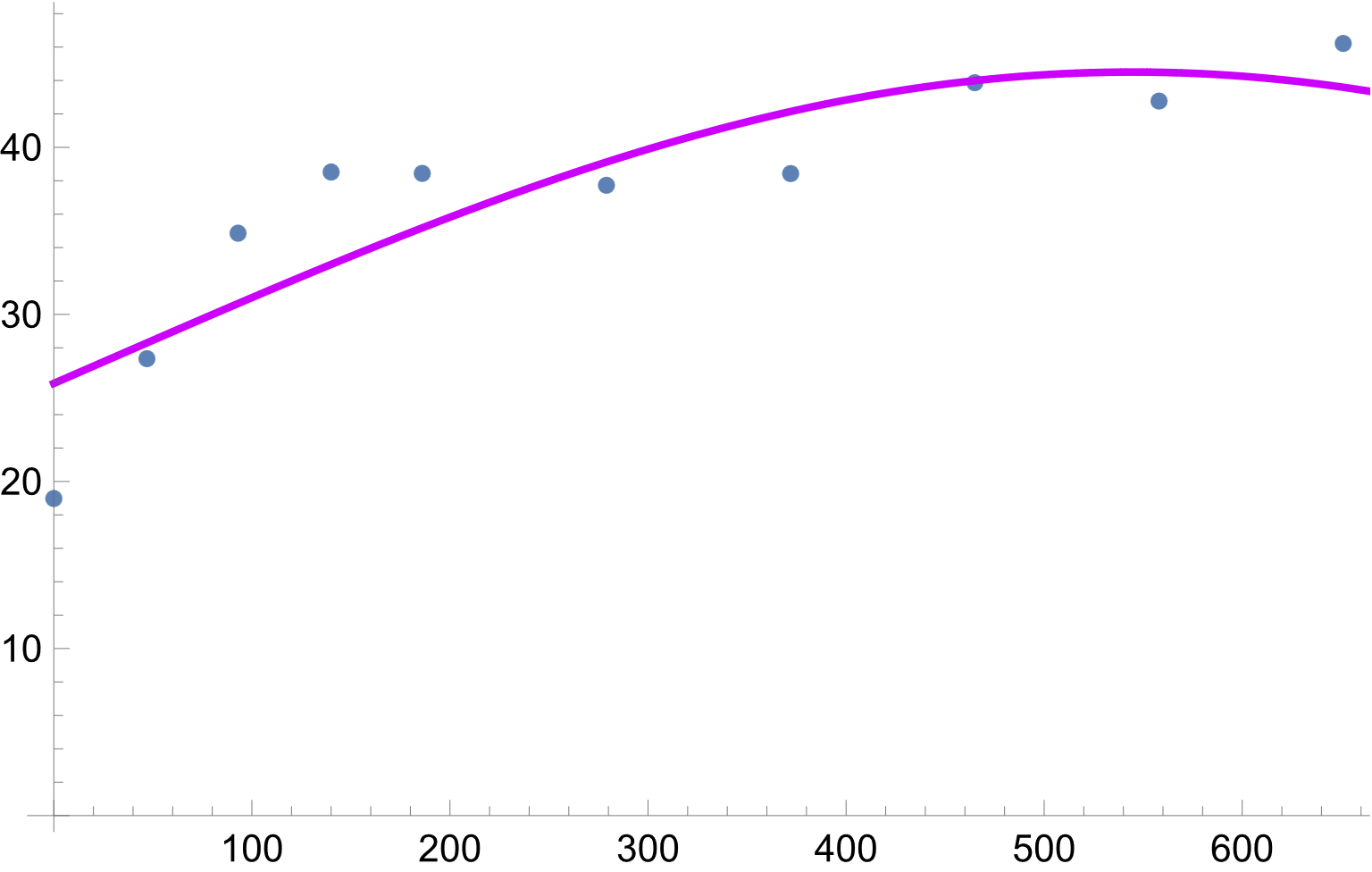
Show[{GK,GGN3}]

**三元函数定义的算法：**((exp03/.x->xN)+(exp04/.x->xP)+(exp05/.x->xK))-2/3(dataN[[7,2]]+dataP[[7,2]]+6+dataK[[7,2]])这样的算法效果好于((exp03/.x->xN)+(exp04/.x->xP)+(exp05/.x->xK))/3

**偏置：**dataN[[7,2]]+dataP[[7,2]]+6+dataK[[7,2]]，这个数字6是通过实验得出的，不加偏置，下面第二幅图的误差会很大。







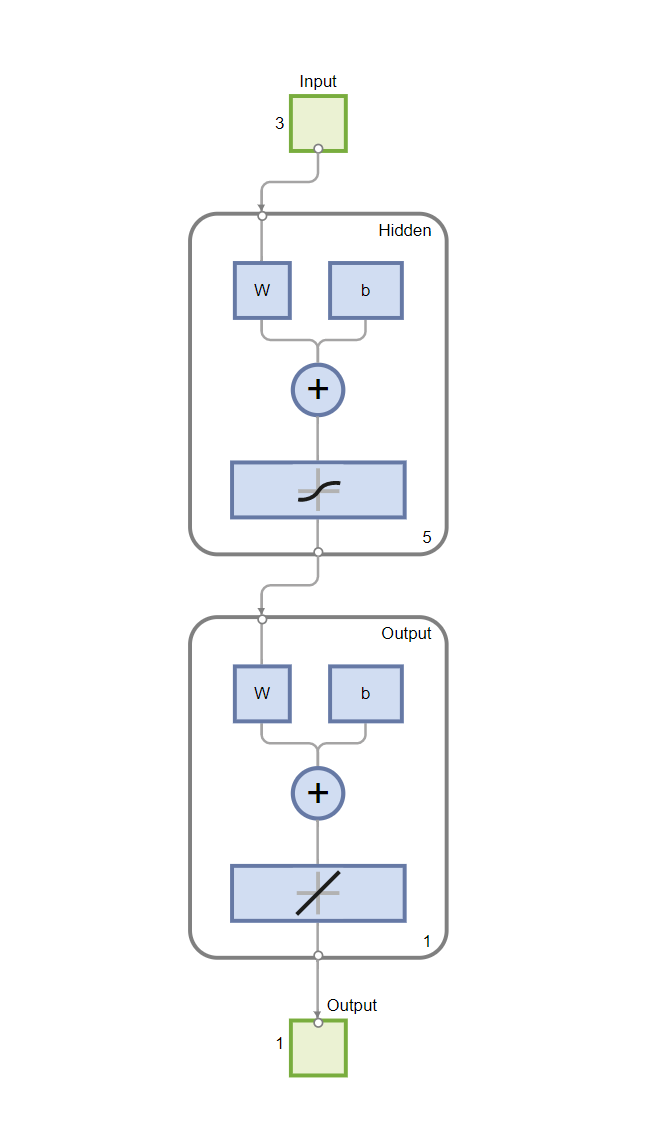
**实验结果**

调用FindMaximum函数求出使得生产量最大时的施肥量和最大生产量

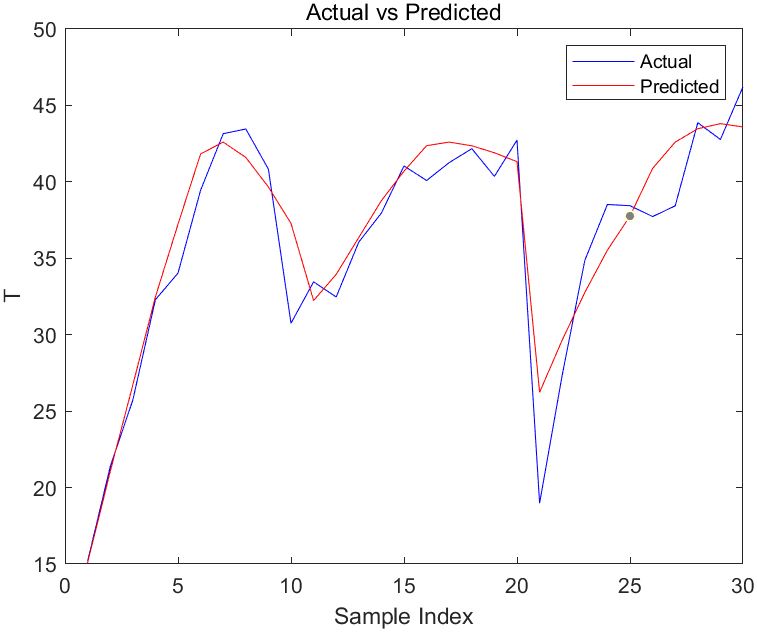
结果：

当氮肥量为293.92，磷肥量为257.6，钾肥量为544.73时，最大生产量为45.743

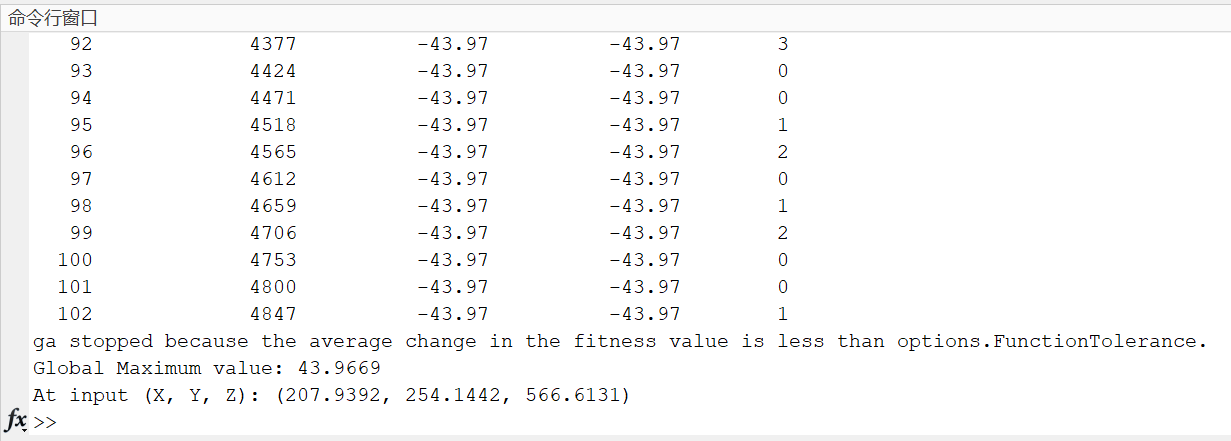
**尝试用神经网络拟合**



我选取的神经网络，只有一层HiddenLayer，这层HiddenLayer有五个神经元。通过实验发现，这样的神经网络架构的拟合结果是最好的，可能是因为数据量太少了，模型一旦复杂起来，就容易overfitting。



如上图所示，拟合的结果也还是蛮不错的。



通过神经网络拟合的结果：

**当氮肥量为207.94，磷肥量为254.14，钾肥量为566.61时，最大生产量约为44。**

matlab代码附在最后。

**实验结果分析：**

（1）通过神经网络拟合的结果：

**当氮肥量为207.94，磷肥量为254.14，钾肥量为566.61时，最大生产量约为44。**

（2）通过拟合函数得到的结果

**当氮肥量为293.92，磷肥量为257.6，钾肥量为544.73时，最大生产量为45.743**

很容易可以发现，两种拟合方式得出的结果是差不多的，通过两种不同的方法得出一个相同的答案，可以说明结果是比较准确的。

但在时间复杂度上，选取拟合函数拟合要远好于神经网络拟合。通过因为通过神经网络得出的拟合函数无法写出具体的表达式，所以求出它的最大值需要通过遗传算法遍历很大的范围。此外，由于数据量太小，一旦神经网络的复杂度上升，就容易出现过度拟合（overfitting）。在数据量小的情况下（本题），更适合通过人脑拟合，选出一个合适的拟合函数的方式。

在实际问题中 ，产量受作物种类 、植株密度 、施肥量 、气候条件等各种因素的作用。

1. 考虑植株密度： 在原有数据基础上 ， 加上一组植株密度变化数据 。
2. 土壤肥力影响： 在实际环境中 ， 每块地肥力不等 ，有高产 田与低产田之分 将土壤肥力也当作影响作物产量的一个变量 ， 同样可进行分析

附件：MatLab代码

% 输入数据

X = [0, 34, 67, 101, 135, 202, 259, 336, 404, 471, 259, 259, 259, 259, 259, 259, 259, 259, 259, 259, 259, 259, 259, 259, 259, 259, 259, 259, 259, 259];

Y = [196, 196, 196, 196, 196, 196, 196, 196, 196, 196, 0, 24, 49, 73, 98, 147, 196, 245, 294, 342, 196, 196, 196, 196, 196, 196, 196, 196, 196, 196];

Z = [372, 372, 372, 372, 372, 372, 372, 372, 372, 372, 372, 372, 372, 372, 372, 372, 372, 372, 372, 372, 0, 47, 93, 140, 186, 279, 372, 465, 558, 651];

T = [15.18, 21.36, 25.72, 32.29, 34.03, 39.45, 43.15, 43.46, 40.83, 30.75, 33.46, 32.47, 36.06, 37.96, 41.04, 40.09, 41.26, 42.17, 40.36, 42.73, 18.98, 27.35, 34.86, 38.52, 38.44, 37.73, 38.43, 43.87, 42.77, 46.22];

% 转换为适合网络的输入格式

input = [X; Y; Z];

output = T;

% 创建一个具有10个隐藏层神经元的前馈神经网络

hiddenLayerSize = 5;

net = fitnet(hiddenLayerSize);

net.performParam.regularization = 0.1;

% 分割数据为训练集、验证集和测试集

net.divideParam.trainRatio = 100/100;

net.divideParam.valRatio = 15/100;

net.divideParam.testRatio = 15/100;

% 训练网络

[net, tr] = train(net, input, output);

% 预测输出

output\_pred = net(input);

% 计算性能

performance = perform(net, output, output\_pred);

% 可视化结果

figure;

plot(output, 'b', 'DisplayName', 'Actual');

hold on;

plot(output\_pred, 'r', 'DisplayName', 'Predicted');

legend('show');

xlabel('Sample Index');

ylabel('T');

title('Actual vs Predicted');

% 计算均方误差

mse = mean((output - output\_pred).^2);

% 显示性能指标

disp(['Mean Squared Error: ', num2str(mse)]);

disp(['Performance: ', num2str(performance)]);

% 定义目标函数（用于优化）

objective\_function = @(xyz) -net(xyz'); % 由于遗传算法是求最小值，所以我们取负值

% 定义输入变量的上下界

lb = [0, 0, 0]; % 下界（可以根据实际情况调整）

ub = [471, 342, 651]; % 上界（可以根据实际情况调整）

% 使用遗传算法进行优化

options = optimoptions('ga', 'Display', 'iter', 'PopulationSize', 50);

[x\_opt, fval\_opt] = ga(objective\_function, 3, [], [], [], [], lb, ub, [], options);

% 最优解

max\_value = -fval\_opt; % 记得取负值还原

max\_X = x\_opt(1);

max\_Y = x\_opt(2);

max\_Z = x\_opt(3);

% 显示最大值及其对应的输入数据点

disp(['Global Maximum value: ', num2str(max\_value)]);

disp(['At input (X, Y, Z): (', num2str(max\_X), ', ', num2str(max\_Y), ', ', num2str(max\_Z), ')']);