一、初始化（种群数量、变异率、随机产生初代种群、最大迭代次数）

二、**while**（当前迭代次数<最大次数）

当代成本计算

1、计算当前种群中每个个体的成本

2、挑选出其中成本最小的个体，并记录（为绘制出迭代次数-成本的图）

当代适应度计算

1、适应度函数选为1/成本^2

下一代的产生

1、精英保留，当代适应度最高的直接克隆

2、父母选择，利用轮盘赌法选择；单点交叉互换，点位随机选择

3、基因变异，随机挑选点位进行变异。最终产生下一代

**end**

三、迭代完成后，从最后一代的种群里选择成本最低的个体，即为最终结果

clc;

clear;

tic;

Population = 100; %种群初始规模

Pvariation = 0.001; %基因变异概率

Iteration = 200; %最大迭代次数

currentIteration = 1; %当前迭代次数

currentPopulation = randi([0,1], Population, 90); %种群情况初始化

costRecord = zeros(Iteration, 1); %记录每次迭代的成本

while(currentIteration <= Iteration) %当迭代次数小于最大迭代次数

Adaptation = zeros(Population, 1); %适应度函数

for i = 1:Population

Adaptation(i) = costofallsample(currentPopulation(i,:)); %当前迭代次数下种群内每个个体的成本

end

costRecord(currentIteration) = min(Adaptation); %记录当前迭代次数下种群内最小个体的成本

Adaptation = 1./ Adaptation.^2; %适应度=成本平方的倒数

elite = find(Adaptation == max(Adaptation)); %精英保留

nextPopulation = zeros(Population, 90); %nextPopulation为下一代的种群情况

nextPopulation(1, :) = currentPopulation(elite(1), :); %前两行因为精英保留

nextPopulation(2, :) = currentPopulation(elite(1), :);

for i = 2:Population

Adaptation(i) = Adaptation(i) + Adaptation(i - 1); %轮盘赌法（i）：确定每个个体被选中的对应区间，即累加适应度

end

select = zeros(Population, 1); %select矩阵表示每个个体被选中的次数

for i = 1:Population - 2

tmp = find((Adaptation / Adaptation(Population))>=rand()); %轮盘赌法（ii):tmp记录所有适应度大于随机值的个体

select(tmp(1)) = select(tmp(1)) + 1; %轮盘赌法（iii):个体每被选中一次，其select值加一。除了两个精英，都参与此轮

end

%交叉

pin = 3; %新种群中前两个个体均为上一代中适应度最高的，所以新子代从第三行开始。

while (sum(select) ~= 0)

a = find(select); %找出所有被选中的个体，记为a矩阵

b = randi(length(a)); %b为a中任选一个个体的下标

father = a(b); %father为被选中个体中任选的那个

b = randi(length(a));

mother = a(b);

select(father) = select(father) - 1; %为了保证所有被选中个体都参与交叉，为了公平，参与一次，其select次数减去一

select(mother) = select(mother) - 1;

location = randi(89); %开始交叉，产生交换点

nextPopulation(pin, 1:location) = currentPopulation(father, 1:location); %通过单点交叉，每次产生两个新子代

nextPopulation(pin, (location + 1):90) = currentPopulation(mother, (location + 1):90); %前半不变，后半交叉

nextPopulation(pin + 1, 1:location) = currentPopulation(mother, 1:location);

nextPopulation(pin + 1, (location + 1):90) = currentPopulation(father, (location + 1):90);

pin = pin + 2; %通过单点交叉，每次产生两个新子代

end

mutation = rand(Population, 90) < Pvariation; %开始变异，当随机值小于阈值，开始变异，mutation矩阵的对应数值为1

nextPopulation = abs(mutation - nextPopulation); %对应基因的基因型翻转，即0、1互换

currentPopulation = nextPopulation; %下一代递进为当代

currentIteration

currentIteration = currentIteration + 1; %开始下一次迭代工作

end

lastCost = zeros(Population, 1); %记录最终种群中每个个体的成本

for i = 1:Population

lastCost(i) = costofallsample(currentPopulation(i, :)); %计算最终种群每个个体的成本

end

finalCost = min(lastCost) %找出最终迭代后最小成本的个体，输出最小成本

chosenIndividual = currentPopulation(lastCost == finalCost,:); %找出最终迭代后最小成本的个体，找出其基因型

chosenPoint = find(chosenIndividual(1, :)) - 20 %找出最终迭代后最小成本的个体，输出基因为1的编号，考虑到表格里的是从-20开始的，故减去20

toc;

figure; %绘制出“迭代次数-当此最小成本”的函数图像

plot(1:currentIteration-1, costRecord);

title('Relationship between Times And Cost');

xlabel('Times of Iteration');

ylabel('Cost');

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

function c1 = costofallsample(Matrix)

table = dlmread('dataform\_train.csv');

Content= table(1,:);

Number = table(2:2:1000,:);

equal1 = find(Matrix);

Numberchosen = Number(:, equal1);

Contentchosen = Content(:, equal1);

residual = zeros(500,90);

costsingle = zeros(500,90);

n = 1;

for j = 1:90

if ( Matrix(1,j) == 1)

n = n + 1;

end

end

for i = 1:500

residual(i,:) = abs(Content-interp1(Numberchosen(i,:),Contentchosen,Number(i,:),'pchip'));

end

for i = 1:500

for j = 1:90

if (residual(i,j) <= 0.5)

costsingle(i,j) = 0;

end

if (residual(i,j) > 0.5 && residual(i,j) <= 1.0)

costsingle(i,j) = 1;

end

if (residual(i,j) >1.0 && residual(i,j) <= 1.5)

costsingle(i,j) = 6;

end

if (residual(i,j) >1.5 && residual(i,j) <=2.0)

costsingle(i,j) = 20;

end

if (residual(i,j) >2.0)

costsingle(i,j) = 10000;

end

end

end

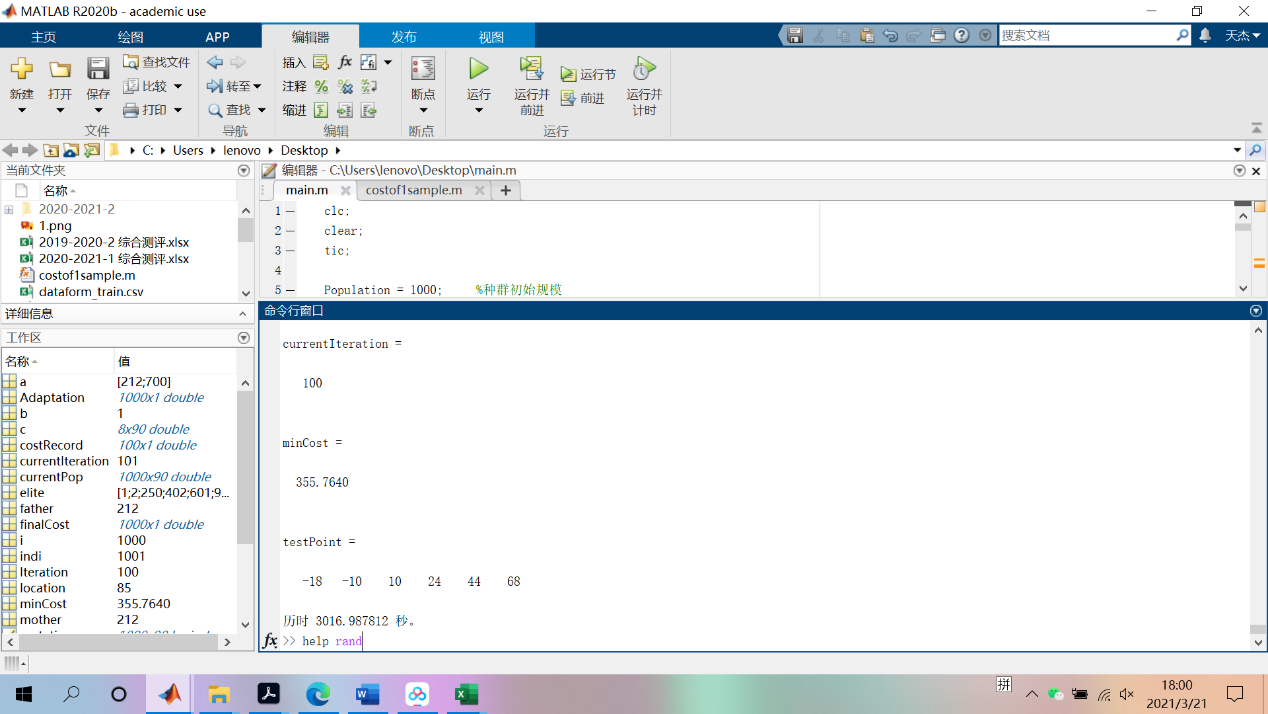
c1 = sum(costsingle(:))/500 + 50 \* n;

return;

end

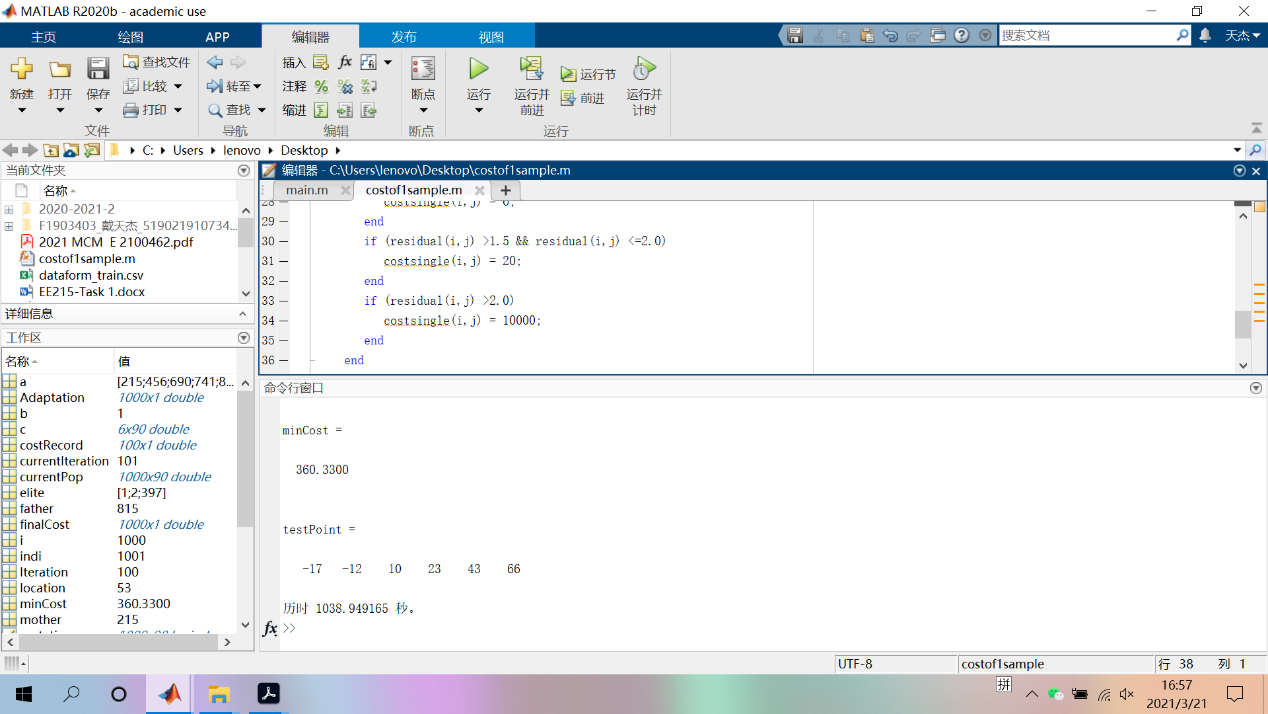
种群：1000

迭代：100



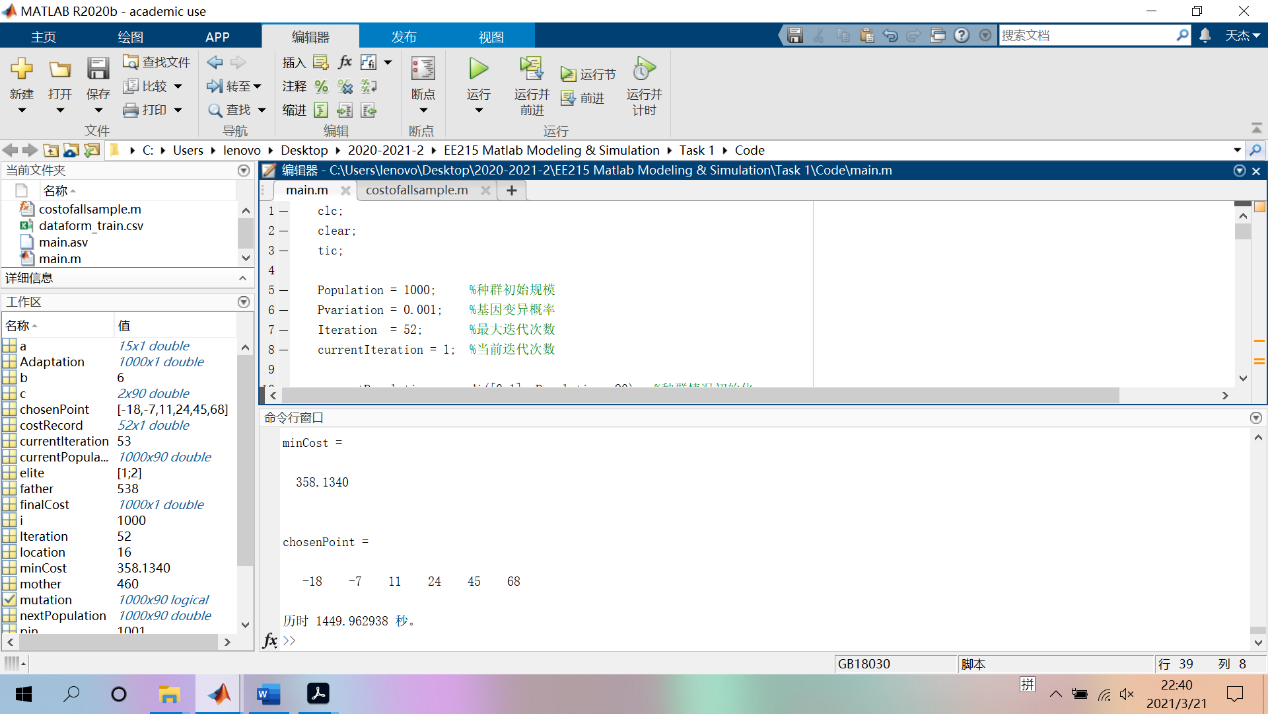
种群：1000

迭代：50



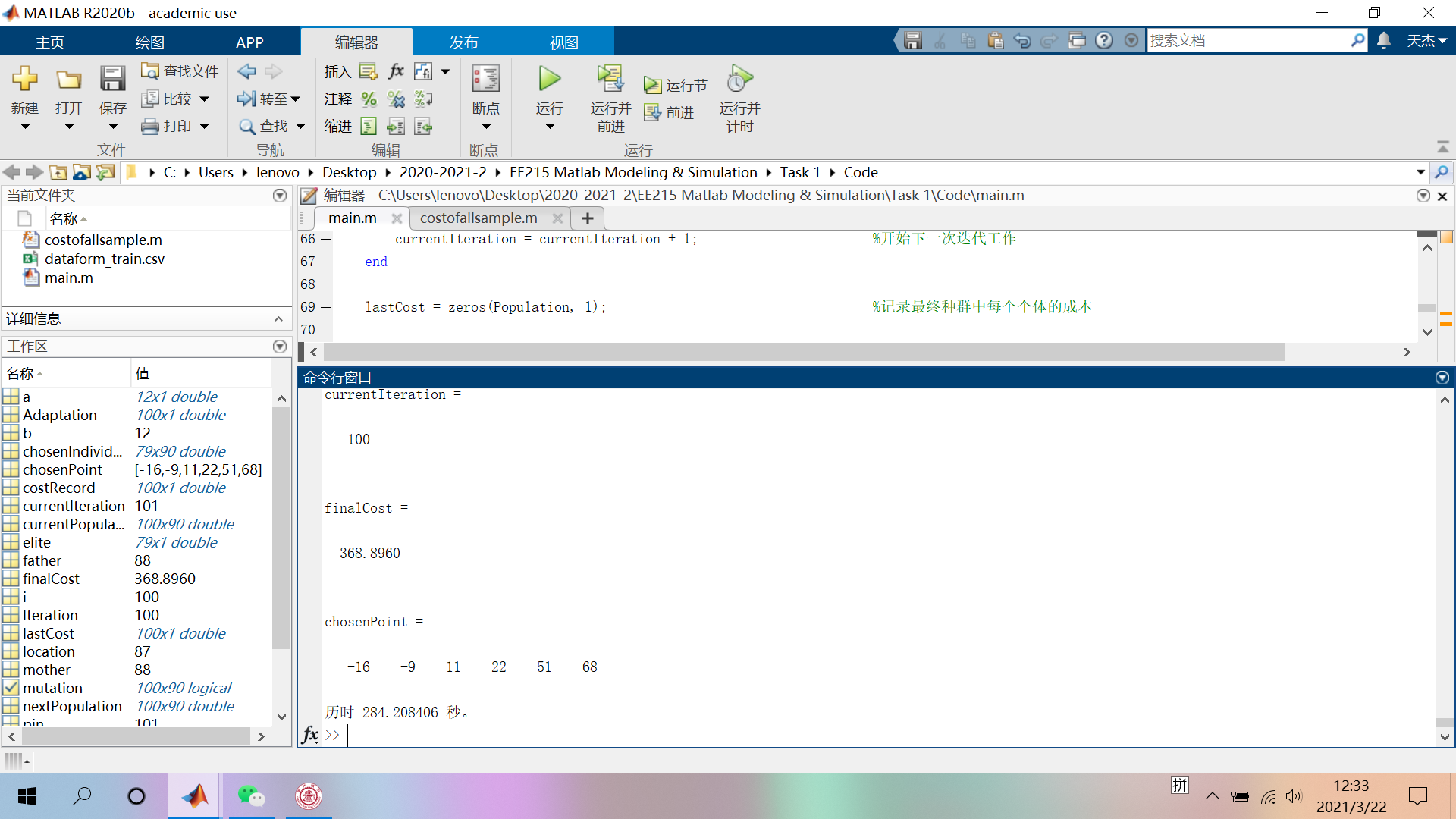
种群：1000

迭代：52



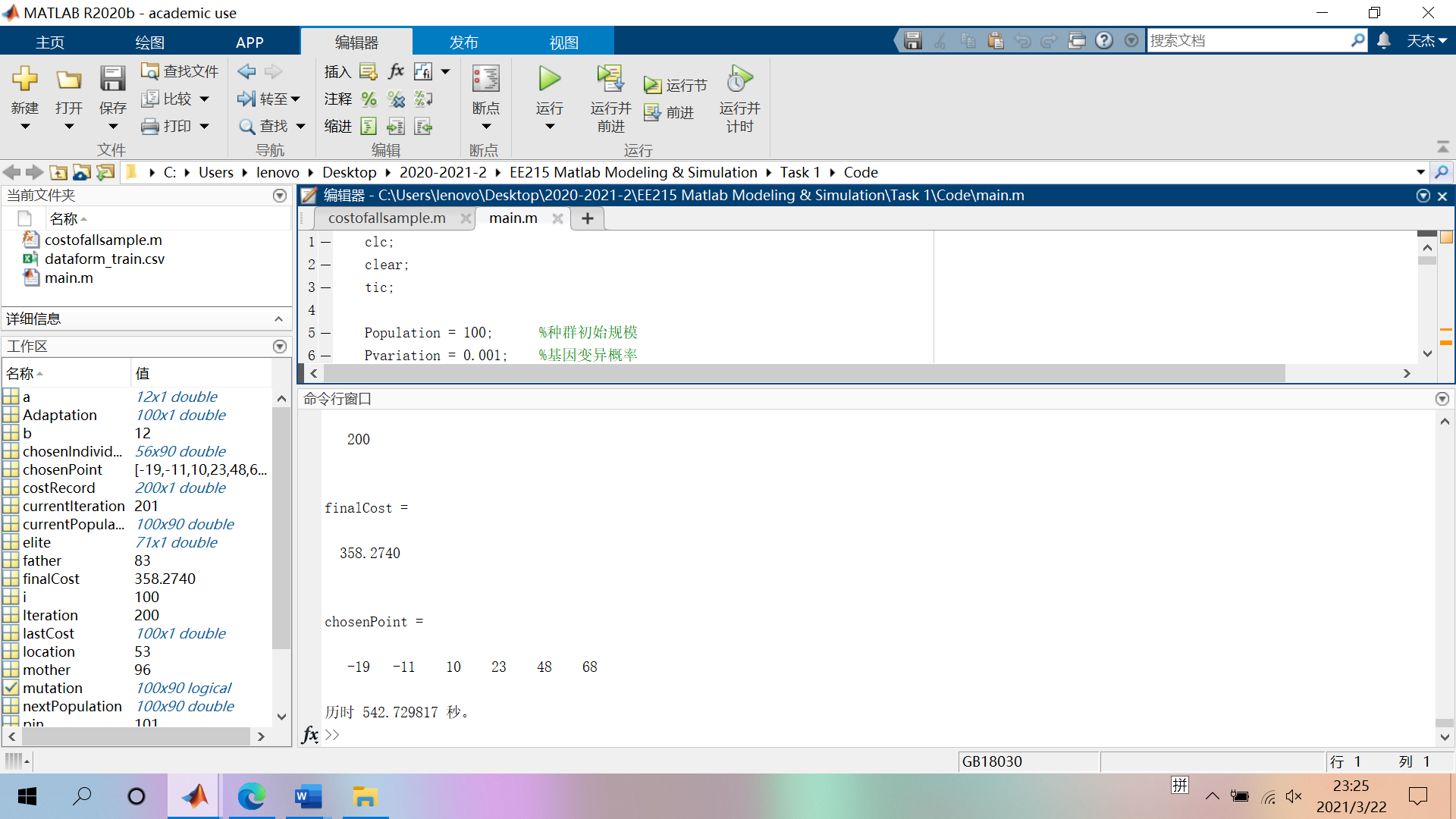
种群：100

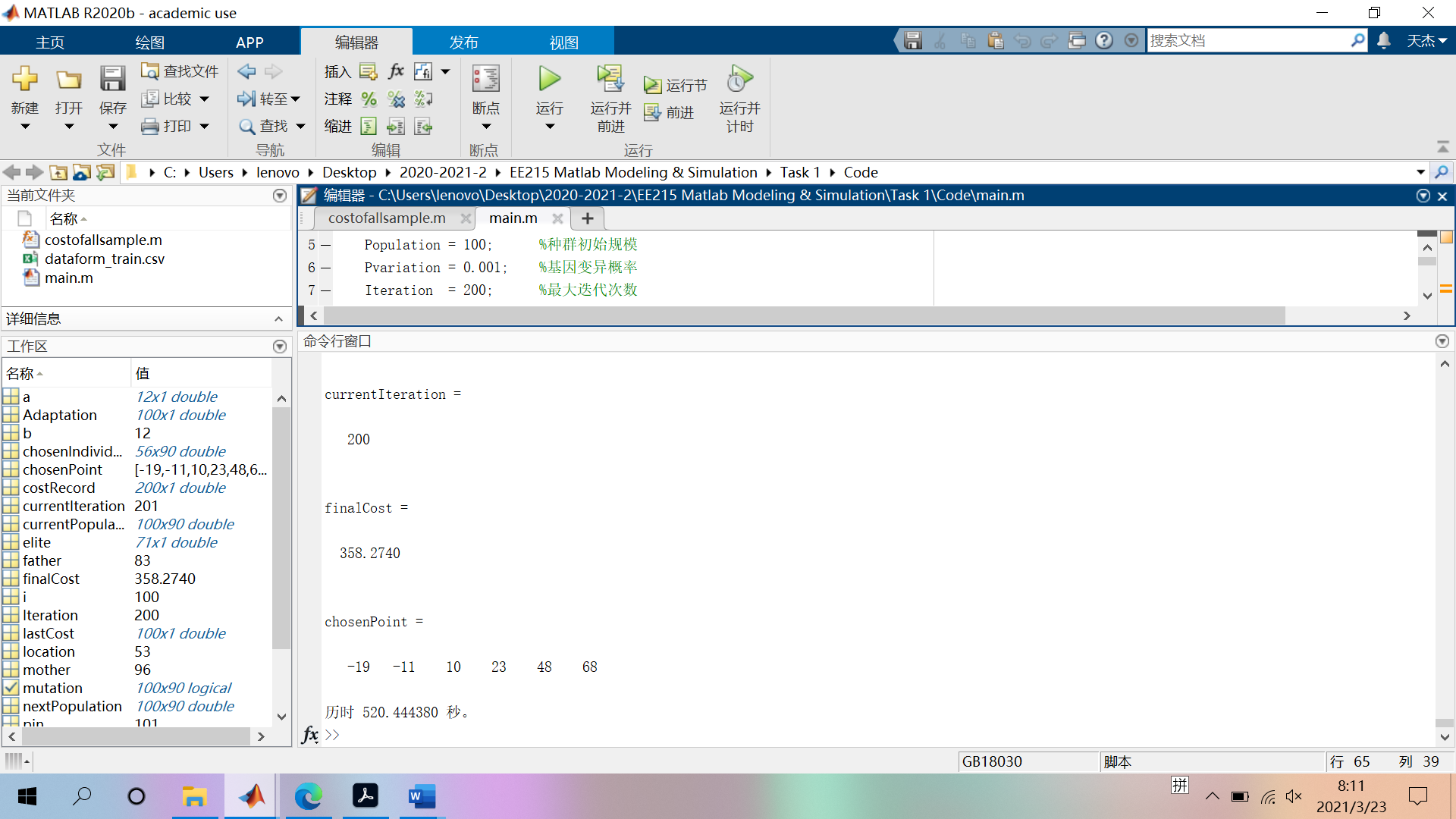
迭代：100



种群：100

迭代：200





1、码重；

2、参数；

3、拟合方法（pchip等）;

4、利用假设检验看看误差是不是呈现正态分布；

5、适应度函数的选择，是不是要根据迭代次数变化呢？

6、精英保留的数量？

