系统优化期末大作业

陈政培 17363011

第一题——main.m

题目分析

本题在群里也有多位同学询问是否是题目要求出现问题，根据群里同学普遍提出的解决方案，本题我才用了复合型法来求解这个问题。

唯一的参数自定义就是设置**复合型顶点的数目为N+1**

matlab程序

```
clc
clear

% 优化计算的参数设定
N=2;%自变量的个数
Numg=4; %约束的数目
KD=N+1;%复合形顶点数目（N+1~2N之间）
x0=[-0.9;-0.5];%初始点

x1(1)=-0.9;
x2(1)=-0.5;

AB=[-1 1;
    -1 1]; %自变量的估计区间。这是在找到符合要求的初始点x0后，自己定义的一个自变量的估计区间

ebsn1=1e-6;%%精度，是自定义的

% 开始计算

   gx=yueshu_g(x0);
   if(max(gx)>0);   %如果不满足约束条件
       'error,初始点x0为外点！！';
       return;
   end;
% 生成KD个顶点的复合形
 XFU=x0;

 for j=2:KD
     wai=1.0;
     while(wai>0)
       for i=1:N
         XFU(i,j)=AB(i,1)+rand(1,1)*(AB(i,2)-AB(i,1));
       end
       xj=XFU(:,j);
       gx=yueshu_g(xj);
       if(max(gx)<=0);
           wai=-1;
       end;
     end
 end

'ok 初始复合形正常';

%迭代开始
```

```
error1=100;

k=1;
 while(error1>ebsn1)
    k=k+1;
  % 比较各个顶点的函数值，找出最小的和最大的
  for j=1:KD
      xj=XFU(:,j);
      FX(j)=fun2(xj);
  end
     [fLx,n_xL]=min(FX);
     [fHx,n_xH]=max(FX);%找出最小、大函数值对应的函数值 与号码
     XL=XFU(:,n_xL);  %复合形的最优点
     x_opt=XFU(:,n_xL);
     fx_opt=FX(n_xL);

     if (mod(k,20)==0)%每迭代20次，输出一次结果
         k
     '当前最优解：'
     x_opt=XFU(:,n_xL)
     fx_opt=FX(n_xL)
     end

     x1(k)=x_opt(1);
     x2(k)=x_opt(2);

     ee=0;
     for j=1:KD
       ee=ee+(FX(j)-fLx)^2;
     end
     error1=sqrt(1.0/(KD-1)*ee);
     if error1<ebsn1;
         break;
     end;% 收敛判断

     %% 求去除 最坏点 后的 形心，
     Xc=x0;
     Xc(:)=0;

  for j=1:KD
     if (j<n_xH || j>n_xH);
         Xc=Xc+XFU(:,j);
     end; %求形心Xc
  end
     Xc=Xc/(KD-1);
     gx_XC=yueshu_g(Xc);

  if(max(gx_XC)>0)%如果除去最坏点 的形心 在可行域外面，则重新形成复合形，否则进行变形计算
      AB(:,1)=XL;
      AB(:,2)=Xc;  %替换设计变量上下界
     for j=1:KD
       wai=1.0;
       while(wai>0)
         for i=1:N
           XFU(i,j)=AB(i,1)+rand(1,1)*(AB(i,2)-AB(i,1));
         end
         xj=XFU(:,j);
         gx=yueshu_g(xj);
```

```
          if(max(gx)<=0);wai=-1;end;
        end
      end
  else
     %'求反射点'
     alpha=1.3;
     while (alpha>1.0e-8)
       alpha;
       XR=Xc+alpha*(Xc-XFU(:,n_xH));
       gx_XR=yueshu_g(XR);
       f_XR=fun2(XR);
       if(max(gx_XR)>0 || f_XR>fHx); %如果反射点不可行 或者 反射点函数值比原最坏点函数值
还要大，则减小反射因子
           alpha=0.5*alpha;%缩小反射因子
       else
           break;%
       end;
     end

      %% 如果 反射方向上找不到 函数更小的点，则更改反射方向
     if (alpha<1.0e-8)
         FXP=FX;FXP(n_xH)=fLx; %定义新的函数数组，将原数组FX中的最大值去掉，以便寻找 第二
大的值
         [fGx,n_xG]=max(FXP);%找出次坏点
         Xc(:)=0; %% 求形心，去除 最坏点
         for j=1:KD
             if (j<n_xG || j>n_xG);
                 Xc=Xc+XFU(:,j);
             end; %求去掉次坏点之后的形心XC
         end
         Xc=Xc/(KD-1);

         gx_XC=yueshu_g(Xc);
        if(max(gx_XC)>0);
          AB(:,1)=XL;
          AB(:,2)=Xc;
          for j=1:KD
             wai=1.0;
           while(wai>0)
             for i=1:N
               XFU(i,j)=AB(i,1)+rand(1,1)*(AB(i,2)-AB(i,1));
             end
             xj=XFU(:,j);
             gx=yueshu_g(xj);
             if(max(gx)<=0);wai=-1;end;
           end
          end
        else  %'求次坏点的反射点'
           alpha=1.3;
           while (alpha>1.0e-8)
             XR=Xc+alpha*(Xc-XFU(:,n_xH));
             gx_XR=yueshu_g(XR);
             f_XR=fun2(XR);
             if(max(gx_XR)>0 || f_XR>fHx); %如果反射点不可行或者 反射点函数值比 原最坏
点函数值还要大，则减小反射因子
                 alpha=0.5*alpha;%缩小反射因子
             else
               break;%
```

```
                end;
            end
        end
    end
  end

   XFU(:,n_xH)=XR;
       if k>10000;
           '强行终止';
           break;
       end;
 end

  '最优解为：'
 x_opt=XL
  '最优值为：'
 fx_opt=fun2(XL)
    figure;
    plot(x1,x2); %绘制迭代点

function gx=yueshu_g(X)
% 不等式约束函数，就是帖子里说的gi(x1,x2,...,xn)，这是根据您的实际情况需要修改的。
gx=[ -1-X(1);
    -1-X(2);
    X(1)-1;
    X(2)-1;];
end

function F=fun2(xk)
% 目标函数，就是帖子里说的f(x1,x2,...,xn)，这是根据您的实际情况需要修改的。
F=(xk(2)-xk(1))^4+12*xk(1)*xk(2)-xk(1)+xk(2)-3;
end
```

运行结果

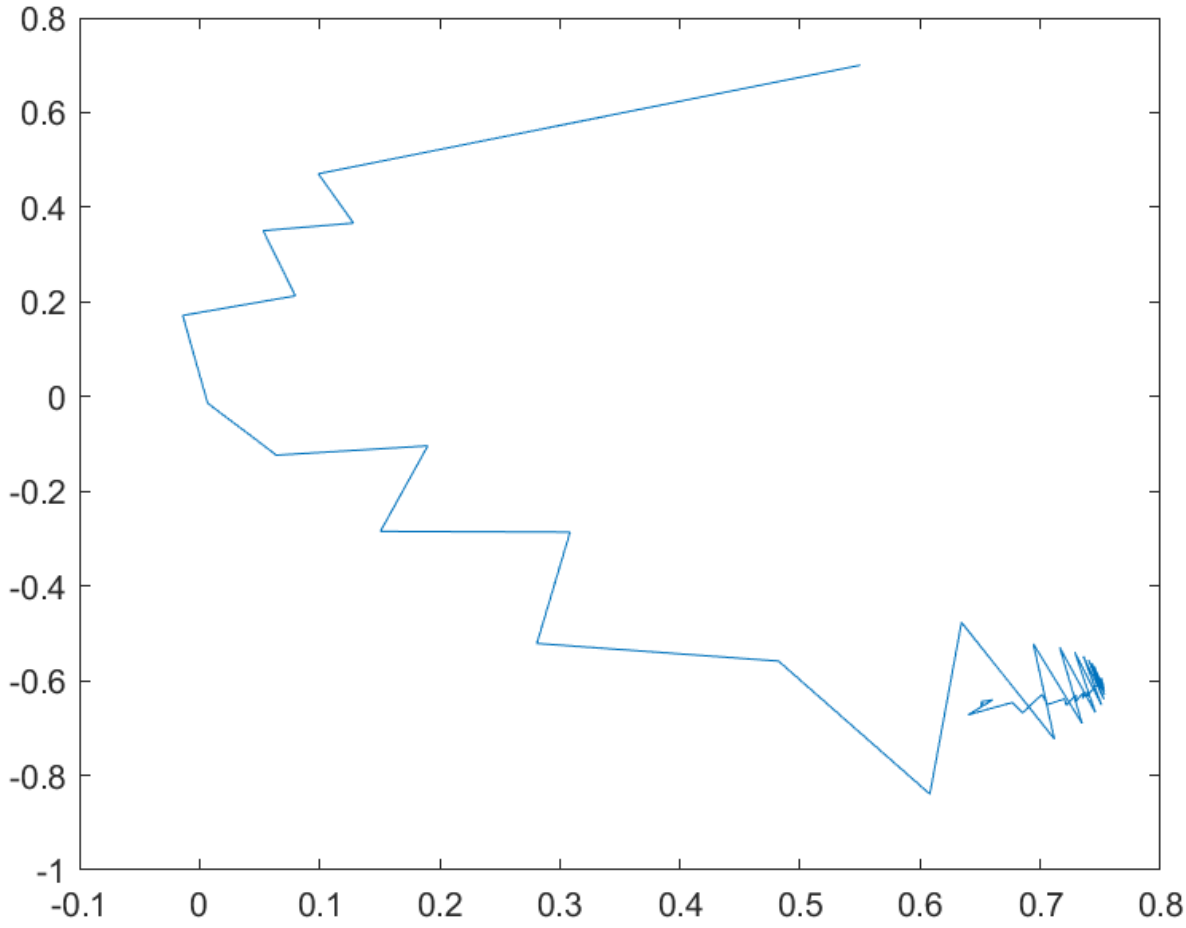
不同初始点迭代路径不同，但是得到的迭代结果极小点结果一致

初始点为 [0.55,0.7] 时，运算得到的极小点是 [0.6503611,-0.6503637]，极小值为 -6.51390494

命令行窗口

```
    '最优解为: x*'


x_opt =

   0.650361106024492
  -0.650363694692447


ans =

    '最优值为:  f(x*)'


fx_opt =

  -6.513904944708647
```

迭代点变化如图

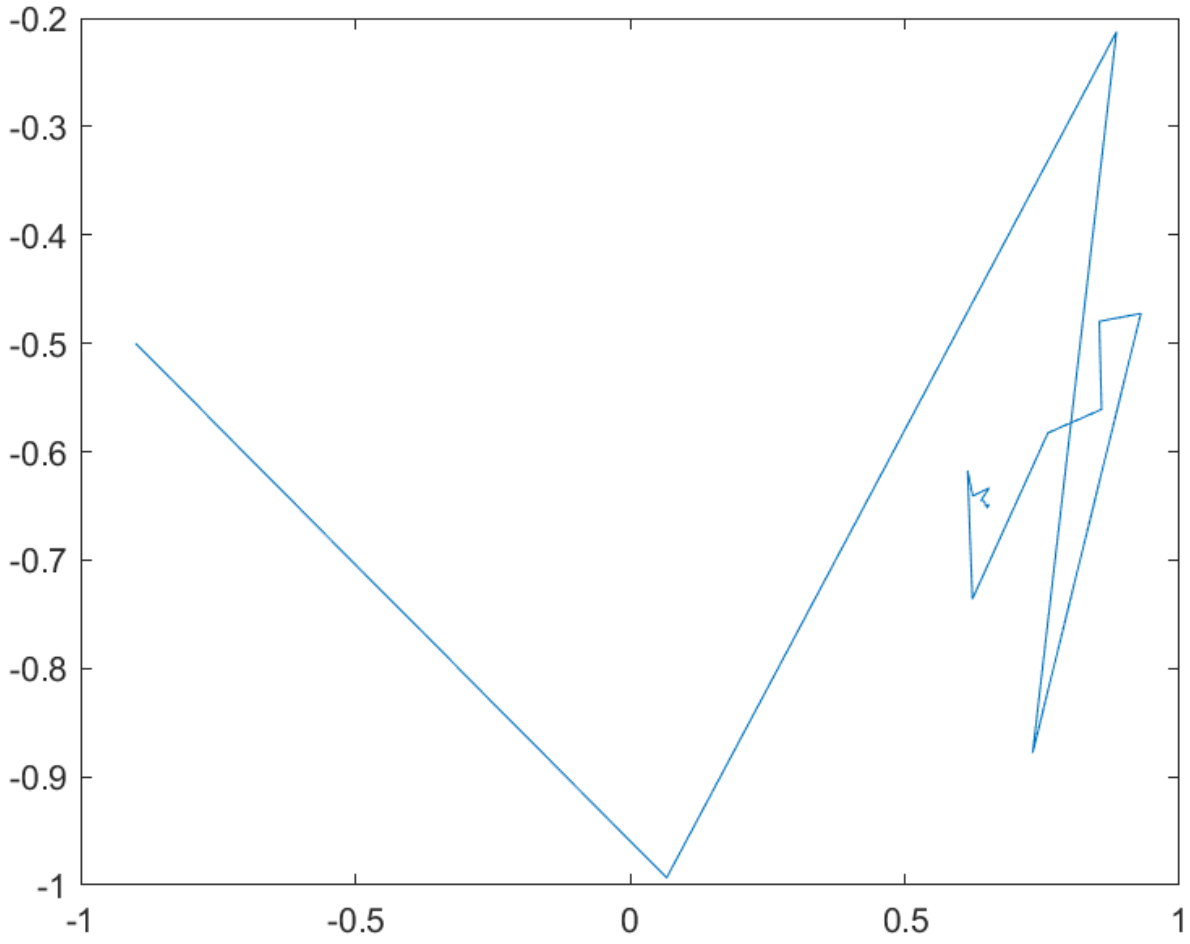


初始点为 [-0.9,-0.5] 时，运算得到的极小点也是是 [0.6503611,-0.6503637]，极小值为 -6.51390494

命令行窗口

```
    '最优解为: x*'


x_opt =

   0.650611348936971
  -0.649682410105981


ans =

    '最优值为:  f(x*)'


fx_opt =

  -6.513900320052795
```

迭代点变化如图

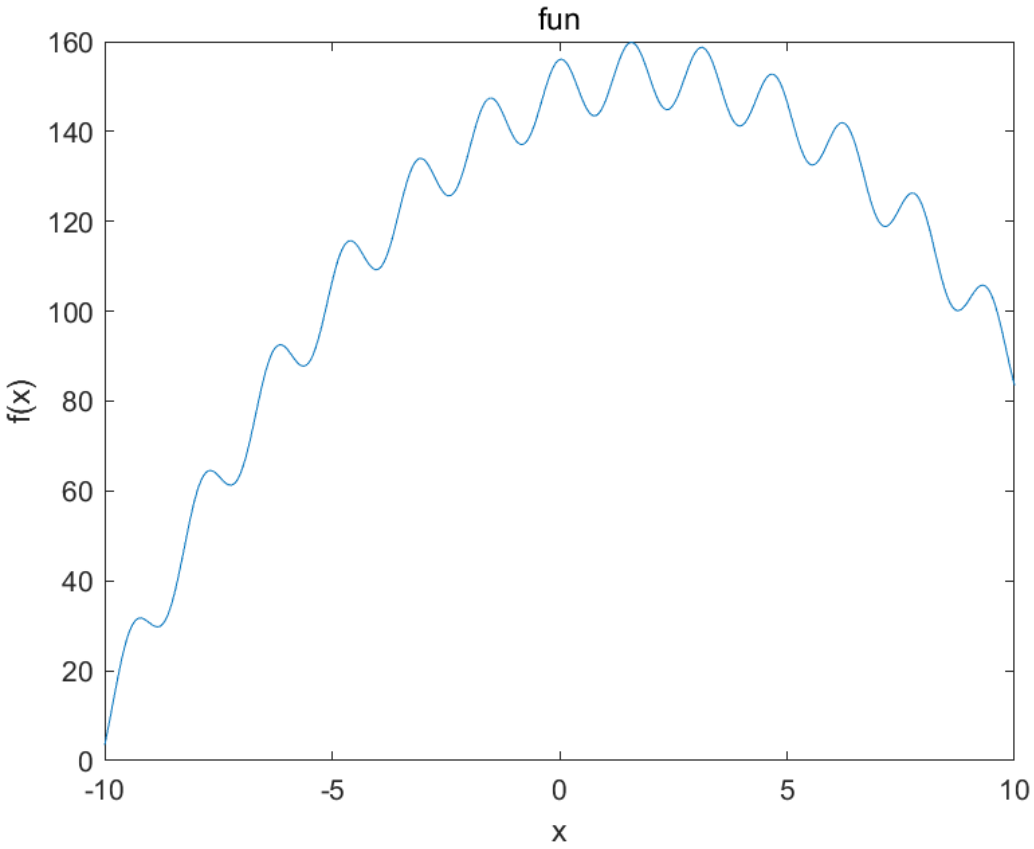


第二题——gm.m

题目分析

先绘制出函数的图像

```
clear;clc;close all;
x=-10:0.01:10;
y=-15*sin(2*x).*sin(2*x)-(x-2).*(x-2)+160;
plot(x,y)
xlabel('x')
ylabel('f(x)')
title('fun')
```



可以看到最优值区间大概在【-1，2】，为了减少运算时间和运算难度我们将求极值的区间缩小，加快运算性能

matlab程序

程序中自定义**初始区间为【-1，2】，初始种群数量100，迭代次数10000，杂交率0.7，选择率0.8，变异率0.001**

```
clear;clc;close all;
%%遗传参数设置
NUMPOP=100;%初始种群大小
irange_l=-1; %问题解区间
irange_r=2;
LENGTH=22; %二进制编码长度
ITERATION = 10000;%迭代次数
CROSSOVERRATE = 0.7;%杂交率
SELECTRATE = 0.8;%选择率
VARIATIONRATE = 0.001;%变异率
```

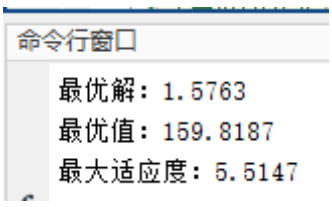
```
%初始化种群
pop=m_InitPop(NUMPOP,irange_l,irange_r);
pop_save=pop;
%绘制初始种群分布
x=linspace(-1,2,1000);
y=m_Fx(x);
plot(x,y);
hold on
for i=1:size(pop,2)
    plot(pop(i),m_Fx(pop(i)),'ro');
end
hold off
title('初始种群');

%开始迭代
for time=1:ITERATION
    %计算初始种群的适应度
    fitness=m_Fitness(pop);
    %选择
    pop=m_Select(fitness,pop,SELECTRATE);
    %编码
    binpop=m_Coding(pop,LENGTH,irange_l);
    %交叉
    kidsPop = crossover(binpop,NUMPOP,CROSSOVERRATE);
    %变异
    kidsPop = Variation(kidsPop,VARIATIONRATE);
    %解码
    kidsPop=m_Incoding(kidsPop,irange_l);
    %更新种群
    pop=[pop kidsPop];
end
figure
x=linspace(-1,2,1000);
y=m_Fx(x);
plot(x,y);
hold on
for i=1:size(pop,2)
    plot(pop(i),m_Fx(pop(i)),'ro');
end
hold off
title('终止种群');

disp(['最优解：' num2str(max(pop))]);
disp(['最优解：' num2str(max(m_Fx(pop)))]);
disp(['最大适应度：' num2str(max(m_Fitness(pop)))]);

%% 要求解的函数
function y=m_Fx(x)
    y=-15*sin(2*x).^2-(x-2).^2+160;
end

%适应度函数为1/距离
function fitness=m_Fitness(pop)
%y=-15*sin(2*x).^2-(x-2).^2+160在[-1,2]上，最大值也不会超过160
%所以计算函数值到2的距离，距离最小时，即为最优解
for n=1:size(pop,2)
    fitness(n)=1/(160-m_Fx(pop(:,n)));
```

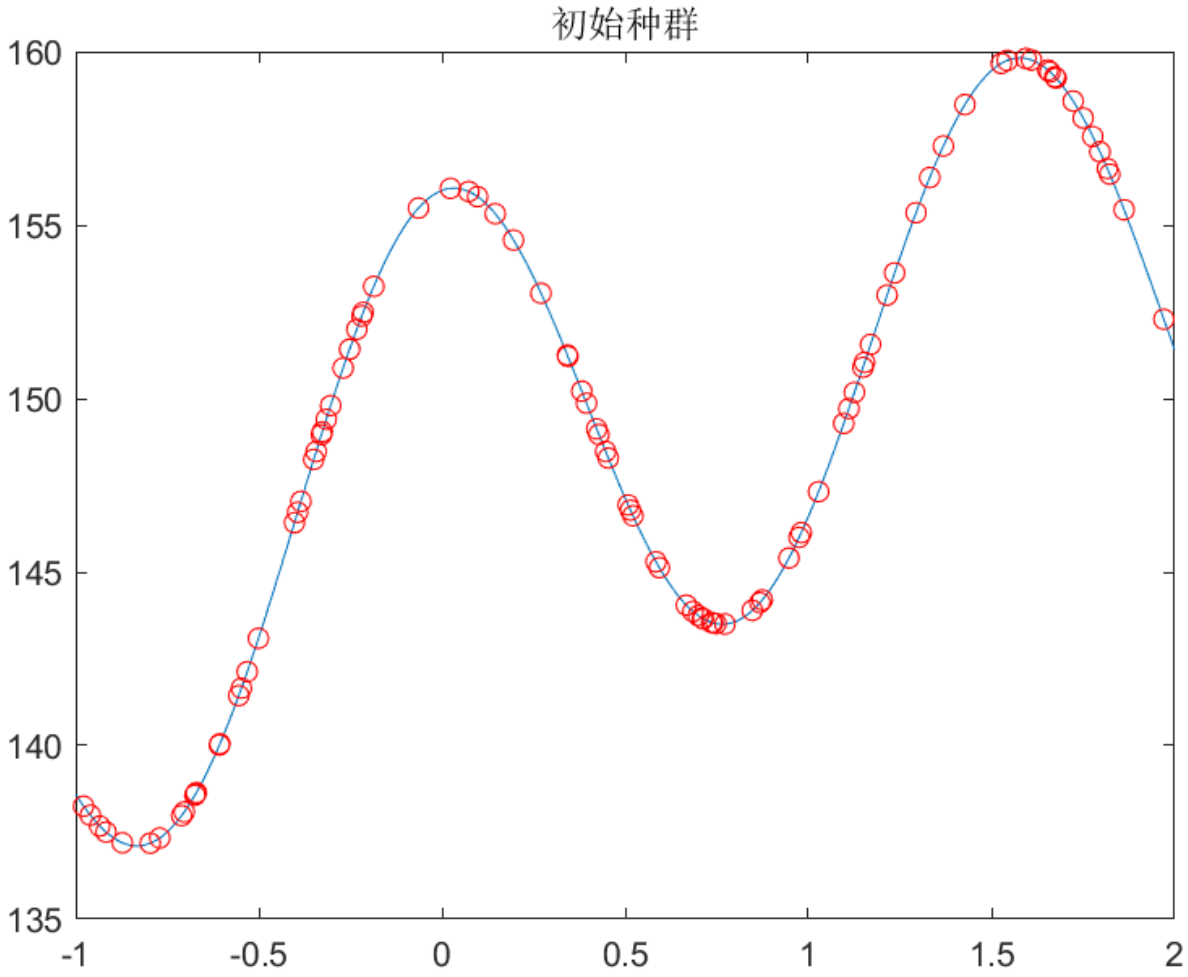
```
end
end

function pop=m_InitPop(numpop,irange_l,irange_r)
%% 初始化种群
%  输入：numpop--种群大小；
%        [irange_l,irange_r]--初始种群所在的区间
pop=[];
for i=1:numpop
    pop(:,i)=irange_l+(irange_r-irange_l)*rand;
end
end

%Crossover函数
%输    入：    parentsPop        上一代种群
%             NUMPOP            种群大小
%             CROSSOVERRATE     交叉率
%输    出：    kidsPop           下一代种群
function kidsPop = crossover(parentsPop,NUMPOP,CROSSOVERRATE)
kidsPop = {[]};n = 1;
while size(kidsPop,2)<NUMPOP-size(parentsPop,2)
    %选择出交叉的父代和母代
    father = parentsPop{1,ceil((size(parentsPop,2)-1)*rand)+1};
    mother = parentsPop{1,ceil((size(parentsPop,2)-1)*rand)+1};
    %随机产生交叉位置
    crossLocation = ceil((length(father)-1)*rand)+1;
    %如果随即数比交叉率低，就杂交
    if rand<CROSSOVERRATE
        father(1,crossLocation:end) = mother(1,crossLocation:end);
        kidsPop{n} = father;
        n = n+1;
    end
end
end

%% 二进制编码（生成染色体）
function binPop=m_Coding(pop,pop_length,irange_l)
% 输入：pop--种群
%       pop_length--编码长度
pop=round((pop-irange_l)*10^6);
for n=1:size(pop,2) %列循环
    for k=1:size(pop,1) %行循环
        dec2binpop{k,n}=dec2bin(pop(k,n));%dec2bin的输出为字符向量；
                                          %dec2binpop是cell数组
        lengthpop=length(dec2binpop{k,n});
        for s=1:pop_length-lengthpop %补零
            dec2binpop{k,n}=['0' dec2binpop{k,n}];
        end
    end
    binPop{n}=dec2binpop{k,n};   %取dec2binpop的第k行
end
end

%% 解码
function pop=m_Incoding(binPop,irange_l)
popNum=1;
popNum = 1;%染色体包含的参数数量
for n=1:size(binPop,2)
```

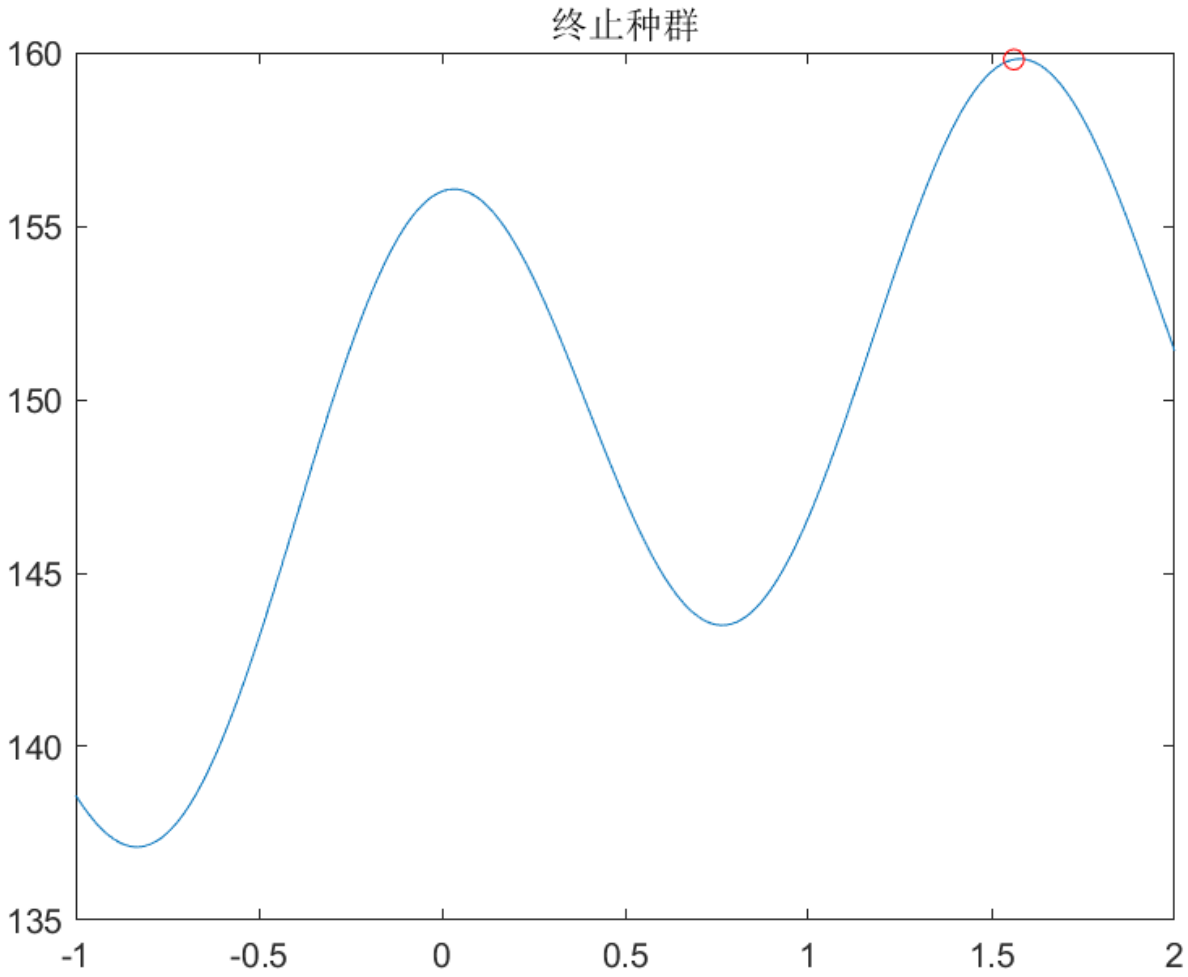
```
    Matrix = binPop{1,n};
    for num=1:popNum
        pop(num,n) = bin2dec(Matrix);
    end
end
pop = pop./10^6+irange_l;
end

%% 选择
function parentPop=m_Select(matrixFitness,pop,SELECTRATE)
% 输入：matrixFitness--适应度矩阵
%       pop--初始种群
%       SELECTRATE--选择率

sumFitness=sum(matrixFitness(:));%计算所有种群的适应度

accP=cumsum(matrixFitness/sumFitness);%累积概率
%轮盘赌选择算法
for n=1:round(SELECTRATE*size(pop,2))
    matrix=find(accP>rand); %找到比随机数大的累积概率
    if isempty(matrix)
        continue
    end
    parentPop(:,n)=pop(:,matrix(1));%将首个比随机数大的累积概率的位置的个体遗传下去
end
end

%% Variation函数
%输    入：    pop                种群
%              VARIATIONRATE      变异率
%输    出：    pop                变异后的种群
%%
function kidsPop = Variation(kidsPop,VARIATIONRATE)
for n=1:size(kidsPop,2)
    if rand<VARIATIONRATE
        temp = kidsPop{n};
        %找到变异位置
        location = ceil(length(temp)*rand);
        temp = [temp(1:location-1) num2str(~temp(location))...
            temp(location+1:end)];
       kidsPop{n} = temp;
    end
end
end
```

运行结果

最终运算结果，极大解是1.5763，极大值是159.8187




初始种群
160
155
150
145
140
135
-1
-0.5
0
0.5
1
1.5
2


终止种群
160
155
150
145
140
135
-1
-0.5
0
0.5
1
1.5
2

第三题——fa.m

题目分析

此题为等式约束的罚函数法，直接进行编程即可

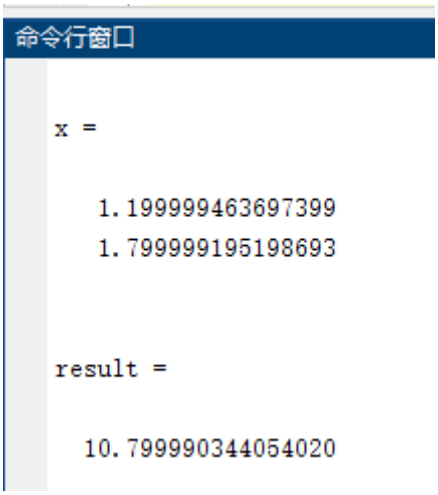
matlab程序

我们设定**初始值为【3，0】，优化精度0.000001，惩罚因子0.01，放大系数2**

```
clear;clc;close all;
%% 外点惩罚函数法-等式约束
syms x1 x2
f=3*x1.^2+2*x2.^2;
hx=[x1+x2-3]; % hx 约束函数

x0=[3;0]; % x0 初始值
M=0.01; % M 初始罚因子
C=2; % C 罚因子放大系数
eps=1e-6; % eps 容差
[x,result]=waidian_EQ(f,x0,hx,M,C,eps)

% f 目标函数
function [x,result]=waidian_EQ(f,x0,hx,M,C,eps)
%计算惩罚项
CF=sum(hx.^2);  %惩罚项P(x)

while 1
    F=matlabFunction(f+M*CF); %目标函数，使用之前的牛顿法，需要转换成句柄
    x1=Min_Newton(F,x0,eps,100);
    if norm(x1-x0)<eps
        x=x1;
        result=double(subs(f,symvar(f),x'));
        break;
    else
        M=M*C;
        x0=x1;
    end
end
end
%牛顿法
function [X,result]=Min_Newton(f,x0,eps,n)
%f为目标函数 x0为初始点 eps为迭代精度 n为迭代次数
TiDu=gradient(sym(f),symvar(sym(f)));% 计算出梯度表达式
Haisai=jacobian(TiDu,symvar(sym(f)));
Var_Tidu=symvar(TiDu);
Var_Haisai=symvar(Haisai);
Var_Num_Tidu=length(Var_Tidu);
Var_Num_Haisai=length(Var_Haisai);
TiDu=matlabFunction(TiDu);
flag = 0;
if Var_Num_Haisai == 0  %也就是说海塞矩阵是常数
    Haisai=double((Haisai));
    flag=1;
end
%求当前点梯度与海赛矩阵的逆
```

```
f_cal='f(';
TiDu_cal='TiDu(';
Haisai_cal='Haisai(';
for k=1:length(x0)
    f_cal=[f_cal,'x0(',num2str(k),'),'];

    for j=1: Var_Num_Tidu
        if char(Var_Tidu(j)) == ['x',num2str(k)]
            TiDu_cal=[TiDu_cal,'x0(',num2str(k),'),'];
        end
    end

    for j=1:Var_Num_Haisai
        if char(Var_Haisai(j)) == ['x',num2str(k)]
            Haisai_cal=[Haisai_cal,'x0(',num2str(k),'),'];
        end
    end
end
Haisai_cal(end)=')';
TiDu_cal(end)=')';
f_cal(end)=')';
switch flag
    case 0
        Haisai=matlabFunction(Haisai);
        dk='-eval(Haisai_cal)^(-1)*eval(TiDu_cal)';
    case 1
        dk='-Haisai^(-1)*eval(TiDu_cal)';
        Haisai_cal='Haisai';
end
i=1;
while i < n
    if abs(det(eval(Haisai_cal))) < 1e-6
        disp('逆矩阵不存在! ');
        break;
    end
    x0=x0(:)+eval(dk);
    if norm(eval(TiDu_cal)) < eps
        X=x0;
        result=eval(f_cal);
        return;
    end
    i=i+1;
end
disp('无法收敛! ');
X=[];
result=[];
end
```

运行结果



运行得到精确解，x*=[1.2，1.8]，最小值为10.8

第四题——Simulated_Annealing.m

题目分析

按照题目要求编写代码即可

matlab程序

```
%结束条件为根据上一个最优解与最新的一个最优解的之差小于某个容差。
%使用METROPOLIS接受准则进行模拟

function [BestX,BestY]=SimulateAnnealing1
clear;
clc;
%// 要求最优值的目标函数,搜索的最大区间即 a
 a = 3
  XMAX = a;
  YMAX = a;
%冷却表参数
 MarkovLength = 10000; %// 马可夫链长度
 DecayScale = 0.95; %// 衰减参数
 StepFactor = 0.02; %// 步长因子
 Temperature=30; %// 初始温度
 Tolerance = 1e-8; %// 容差
AcceptPoints = 0.0; %// Metropolis过程中总接受点
rnd =rand;

% 随机选点 初值设定
PreX = -XMAX * rand ;
PreY = -YMAX * rand;
PreBestX  = PreX;
PreBestY = PreY;

PreX = -XMAX * rand ;
PreY = -YMAX * rand;
BestX = PreX;
```

```
 BestY = PreY;
% 每迭代一次退火一次(降温)，直到满足迭代条件为止
mm=abs( ObjectFunction( BestX,BestY)-ObjectFunction (PreBestX, PreBestY));
while mm > Tolerance

Temperature=DecayScale*Temperature;
AcceptPoints = 0.0;
% 在当前温度T下迭代loop(即MARKOV链长度)次
for i=0:MarkovLength:1
% 1) 在此点附近随机选下一点
p=0;
while p==0
    NextX = PreX + StepFactor*XMAX*(rand-0.5);
    NextY = PreY + StepFactor*YMAX*(rand-0.5);
    if p== (~(NextX >= -XMAX && NextX <= XMAX && NextY >= -YMAX && NextY <= YMAX
&& NextX >= -3 && NextY <= 3))
       p=1;
    end
end

% 2) 是否全局最优解
if (ObjectFunction(BestX,BestY) > ObjectFunction(NextX,NextY))
% 保留上一个最优解
PreBestX =BestX;
PreBestY = BestY;
% 此为新的最优解
BestX=NextX;
BestY=NextY;
end
% 3) Metropolis过程

if( ObjectFunction(PreX,PreY) - ObjectFunction(NextX,NextY) > 0 )
  %// 接受，此处lastPoint即下一个迭代的点以新接受的点开始
PreX=NextX;
PreY=NextY;
AcceptPoints=AcceptPoints+1;
else
 changer = -1 * ( ObjectFunction(NextX,NextY) - ObjectFunction(PreX,PreY) ) /
Temperature ;
rnd=rand;
p1=exp(changer);
double (p1);

if p1 > rand             %// 不接受，保存原解
   PreX=NextX;
   PreY=NextY;
   AcceptPoints=AcceptPoints+1;
end
end
end
mm=abs( ObjectFunction( BestX,BestY)-ObjectFunction (PreBestX, PreBestY));
end

disp('最小值在点:');
BestX
BestY
disp( '最小值为:{0}');
ObjectFunction(BestX, BestY)
```

```
end



%子函数，目标函数值计算
function  value=ObjectFunction(x,y)
         value=-3*(1-x)^2*exp(-x^2-(y+1)^2)+10*(x/5-x^3-y^5)*exp(-x^2-y^2)+exp(-
(x+1)^2-y^2)/3;
end
```

运行结果

分别用a=2，3，4，5进行测试，发现a越小运行得到的结果越精确，但最值上已经比较精准了

a=2

```
a =

     2

最小值在点:-0.46001
最小值在点:-0.62916
最小值为:-3.7766
```

a=3

```
a =

     3

最小值在点:-0.46001
最小值在点:-0.62918
最小值为:-3.7766
```

a=4

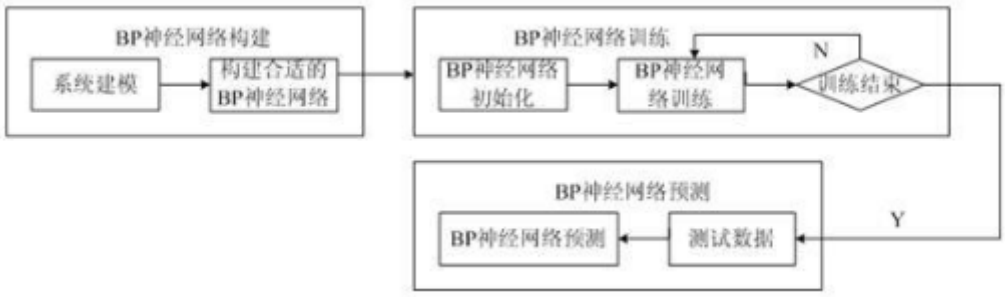
```
a =

     4

最小值在点:-0.46001
最小值在点:-0.62915
最小值为:-3.7766
```

a=5

```
a =

     5

最小值在点:-0.46003
最小值在点:-0.6292
最小值为:-3.7766
```

第五题——基于BP神经网络的线性函数拟合

给出一些点集，通过BP神经网路进行线性回归预测并对比BP拟合效果和最小二乘拟合效果



点集为网上找到的一批测试数据

```
t=[0 3.9 4.1 7.3 8.4 13.1 14.8 16.4 17.7 19 19.7 20.3 21.2 24.5 26.3 27.8 28.9
29 29.8 31.1 32.8 33.5 34.5 35.6 36.2 37.6 37.8 38.7 39.4 40.3 41 41.4 42.5 43.9
45 45.7 46.9 47.8 49 49.4 51.4 53 54 55.6 56.9 57.5 58.9 ];
R=[100.16 101.87 101.97 102.99 103.43 105.23 105.89 106.54 107.01 107.52 107.77
108.01 108.39 109.64 110.33 110.90 111.32 111.41 111.86 112.53 112.63 113.10
113.52 113.94 114.39 114.52 114.92 115.26 115.87 115.90 116.27 116.96 117.32
117.71 118.13 118.34 118.62 118.96 119.59 120.20 120.68 121.33 121.90 122.17
122.94 123.27 123.85];
```

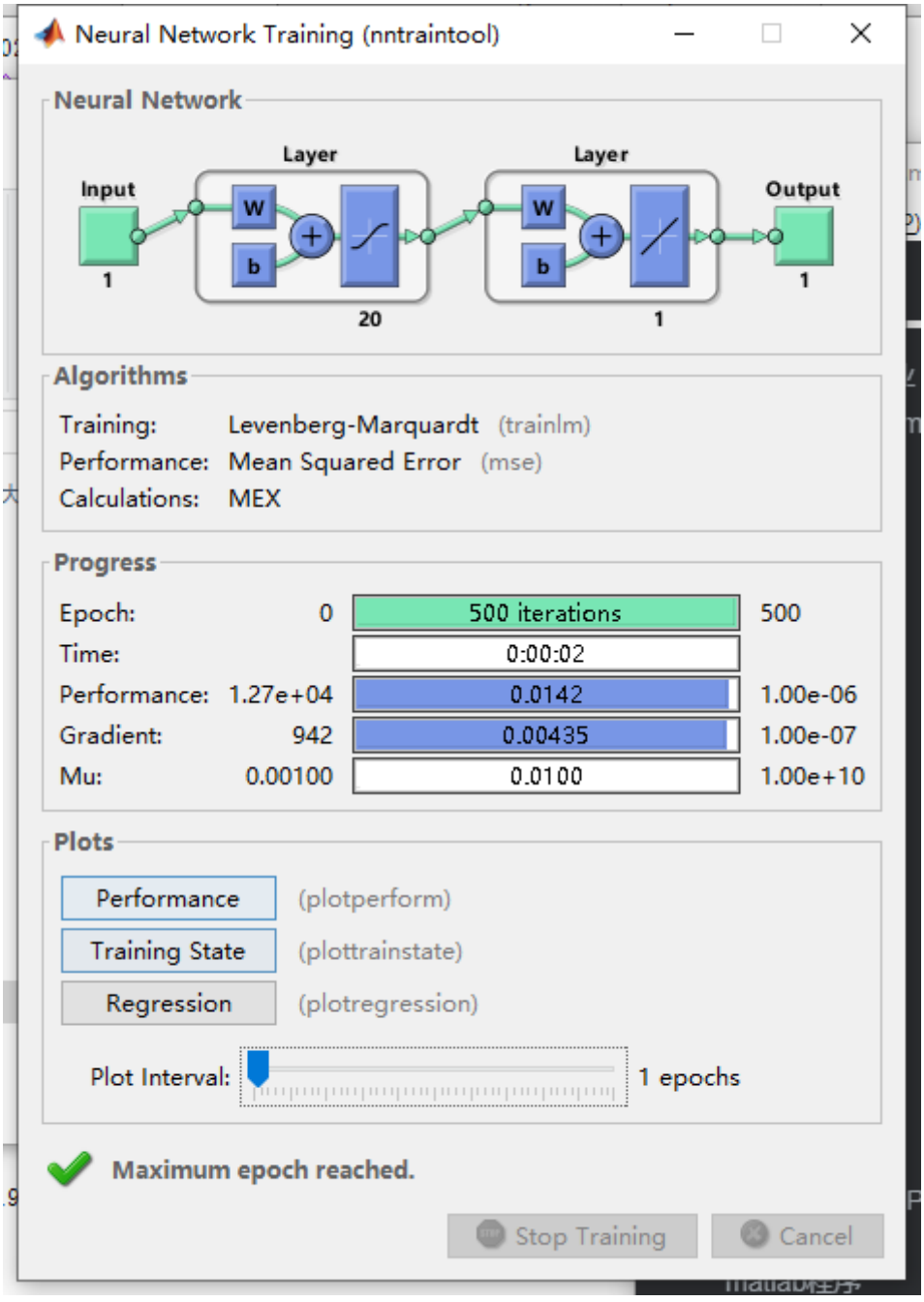
matlab程序

```
t=[0 3.9 4.1 7.3 8.4 13.1 14.8 16.4 17.7 19 19.7 20.3 21.2 24.5 26.3 27.8 28.9
29 29.8 31.1 32.8 33.5 34.5 35.6 36.2 37.6 37.8 38.7 39.4 40.3 41 41.4 42.5 43.9
45 45.7 46.9 47.8 49 49.4 51.4 53 54 55.6 56.9 57.5 58.9 ];
R=[100.16 101.87 101.97 102.99 103.43 105.23 105.89 106.54 107.01 107.52 107.77
108.01 108.39 109.64 110.33 110.90 111.32 111.41 111.86 112.53 112.63 113.10
113.52 113.94 114.39 114.52 114.92 115.26 115.87 115.90 116.27 116.96 117.32
117.71 118.13 118.34 118.62 118.96 119.59 120.20 120.68 121.33 121.90 122.17
122.94 123.27 123.85];

subplot(2,2,1);plot(t,R,'r*'); hold on;
% R=0.0002*t^2+0.3676*t+100.3780;
plot(t,0.0002*t.^2+0.3676*t+100.3780,':');    %绘制不含噪声的余弦曲线
legend('训练样本','正确的曲线')
title('样本数据');

p=polyfit(t,R,2)
y1=polyval(p,t);
subplot(223),plot(t,R,'r*',t,0.0002*t.^2+0.3676*t+100.3780,'b:',t,y1,'g');
legend('训练样本','正确的曲线','拟合曲线'),grid;
% xlabel(sprintf('多项式:y=%.2fx^2+%.2fx+%.2f',p(1),p(2),p(3)));
% pretty(poly2sym(p))
 xlabel(sprintf('多项式:%s',poly2str(p,'x')));
title('最小二乘法的多项式拟合');

net=newff(minmax(t),[20,1],{'tansig','purelin'}); %创建一个新的前向神经网络
net.trainFcn='trainlm'; %设置训练方法及参数
net.trainParam.epochs=500;
net.trainParam.goal=1e-6;
```

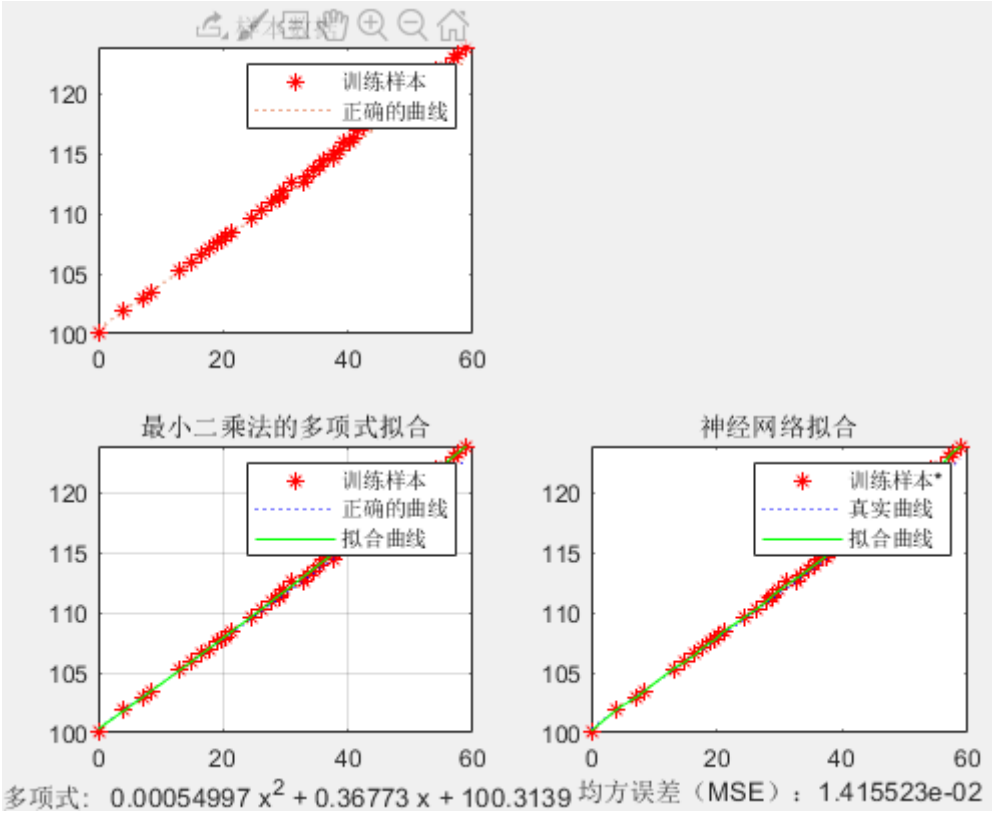
```
net=init(net);%初始化网络

[net,tr]=train(net,t,R); %调用相应算法训练BP网络

A=sim(net,t); %对BP网络进行仿真
E=R-A; %计算仿真误差
MSE=mse(E);%均方误差

subplot(224);plot(t,R,'*r',t,0.0002*t.^2+0.3676*t+100.3780,'b:',t,A,'g'); %绘制拟合结果曲线
legend('训练样本*','真实曲线','拟合曲线');
xlabel(sprintf('均方误差（MSE）：%e',MSE));
title('神经网络拟合');
```

运行结果

BP网络参数如下



训练结果对比



误差为1.415523e-02，可以看到拟合效果非常好

参考文献

1. Matlab论坛 https://www.ilovematlab.cn/thread-313764-1-1.html
2. strawberry-magic-pocket/Genetic-Algorithm https://github.com/strawberry-magic-pocket/Genetic-Algorithm
3. matlab实现基于遗传算法求解二元函数的最大值 https://blog.csdn.net/qjt19950610/article/details/88934020
4. MATLAB约束优化之惩罚函数法 https://blog.csdn.net/STM89C56/article/details/105745129
5. matlab 程序实现 模拟退火算法程序 函数求极值（引用后修改）http://blog.sina.com.cn/s/blog_6377a31001019foe.html
6. 基于Matlab的BP神经网络在非线性函数拟合中的应用 https://blog.csdn.net/LSGO_MYP/article/details/54425751