附 录

1　GM(1,1)灰色模拟源代码

```
% IIi 为预测点
% x 为预测序列（一行）
% XOX 为预测值（对有 IIi 而言）
% QOQ 为预测精度
% pp 为小误差概率
% C 为后验误差比
% pp>0.95  或 C<0.35  为 1 级(好)
% pp>0.80  或 C<0.50  为 2 级(合格)
% pp>0.70  或 C<0.65  为 3 级(勉强合格)
% pp<=0.70 或 C>=0.65 为 4 级(不合格)
%
% AUA=[a;u;ua;xua];
% XEG=[xg;eg];
% PPC=[pp;C];
function [XOX,QOQ,PPC,pddstr,XIi,AUA,XEG]=GM11(x,IIi);
%%%%%%%%%%%%%%%%%%%%%%

%例子
%x=[3.711 3.723 3.716 3.721 3.728]
%x=[2.67 3.13 3.25 3.36 3.56 3.72]
% x=[2.97 3.23 3.29 3.46 3.59 3.71]
%x=[43.45 47.05 52.75  57.14 62.64 68.52]
% x=[3.38 4.27 4.55 4.69 5.59]
% x=[4.24 4.33 5.20 6.42 7.32 8.53 8.82 10.72]
%
% x=[3.38 4.27 4.55 4.69 5.59]
% IIi=[  5 6 7 8 9 10 11]
% IIi=[]

%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                %%
%%          一、数据处理            %%
%%                                %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%1 建立 X1 生成数列
%format long
a12=size(x);
n=a12(2);
x1=ones(1,n);
```

```
x1(1)=x(1);
for i=2:n
    x1(i)=x1(i-1)+x(i);
end
x1;
%2 构造数据距阵 B 和数据向量 Yn:
B=ones(n-1,2);
for i=1:n-1
    B(i,1)=(-0.5)*(x1(i)+x1(i+1));
end
Yn=ones(n-1,1);
for i=2:n
    Yn(i-1,1)=x(i);
end
B;
Yn;
%3 计算 B'B,inv(B'B)和 B'Yn；得到参数 a 和 u
%BTB=B'*B;%
%BTb=inv(BTB);
%BTYn=B'*Yn;
%au=BTb*BTYn;
au=inv(B'*B)*B'*Yn;


a=au(1,1);
u=au(2,1);

% fprintf('参数 a：%f\n',a);
% fprintf('参数 u：%f\n',u);
ua=u/a;
xua=x(1)-ua;
%4 得出预测模型
%预测第 I 的预测值
%有问题

%   XI1=xua*exp((-a)*I)+ua;%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                %%
%%          二、模型检验          %%
%%                                %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%5:残差检验
%51 计算 X1
X1=ones(1,n);
for i=1:n
    X1(i)=xua*exp((-a)*(i-1))+ua;
end
X1;
%52 累减生成 X0 序列
X0=ones(1,n);
X0(1)=X1(1);
for i=2:n
    X0(i)=X1(i)-X1(i-1);
end
X0;
%53 计算绝对误差及相对误差序列
Dd=abs(X0-x);%计算绝对误差序列
DDd=(Dd./x).*100;%计算相对误差序列 x%
% fprintf('相对误差序列：');
% fprintf('%f%%  ',DDd);
% fprintf('\n');
%6 进行关联度检验
%61 计算序列绝对误差 Xx
Xx=ones(1,n);
for i=1:n
    Xx(i)=x(i)-X0(i);
end
% fprintf('序列绝对误差：');
% fprintf('%f ',Xx);
% fprintf('\n');

minXx=min(Dd);
% fprintf('最小差：');
% fprintf('%f ',minXx);
% fprintf('\n');

maxXx=max(Dd);
% fprintf('最大差：');
% fprintf('%f ',maxXx);
% fprintf('\n');
%62 计算关联系数 P=0.5 (则 r=0.6)
P=0.5;
nI=ones(1,n);
```

```
for i=1:n
    nI(i)=(minXx+P*maxXx)/(Dd(i)+P*maxXx);
end
%63 计算关联度,(P=0.5 时， 则 r=0.6)
r=(1/n)*sum(nI);
% fprintf('请查找 P=0.5 是的检验准则 r=0.6 是否大于%f\n',r);
%7 后验差检验
%71
xm=mean(x);
%72 求的均方差
s1=(sum((x-xm).^2)/(n-1))^(1/2);
%73 计算残差的均值
Ddm=mean(Dd);
%74 计算残差的均方差
s2=(sum((Dd-Ddm).^2)/(n-1))^(1/2);
%75 计算后验误差比 C:
C=s2/s1;
% fprintf('验误差比 C：%f\n',C);
%76 计算小误差概率
%%%%%%%%%%%%%%

pr=abs(Xx-mean(Xx))<0.6745*s1; %m 满足条件的样本
pa=size(find(pr==1));
ps=pa(1,2); %m 满足条件的样本个数
pb=size(pr);
pS=pb(1,2); %m 总样本个数

pp=ps/pS; %小误差概率
% fprintf('小误差概率 pp：%f\n',pp);

%%%%%%%%%%%%%%%%%%
%检验  预测精度 ycd

% fprintf('   pp>0.95  或 C<0.35  为 1 级(好)\n   pp>0.80  或 C<0.50  为 2 级(合格)\n   pp>0.70  或 C<0.65  为 3 级(勉强合格)\n   pp<=0.70 或 C>=0.65 为 4 级(不合格)\n\n',C);
if pp>0.95
    pd=1;
%     fprintf('   因 pp>0.95 且 ');
end
if pp<=0.95&pp>0.80
    pd=2;
%     fprintf('   因 pp<=0.95&pp>0.80 且 ');
end
```

```
if pp<=0.80&pp>0.70
    pd=3;
%     fprintf('   因 pp<=0.80&pp>0.70 且 ');
end
if pp<=0.70
    pd=4;
%     fprintf('   因 pp<=0.70 且 ');
end

pd1=0;
if C<0.35
    pd1=1;
%     fprintf('C<0.35');
end
if C>=0.35&C<0.50
    pd1=2;
%     fprintf('C>=0.35&C<0.50');
end
if C>=0.50&C<0.65
    pd1=3;
%     fprintf('C>=0.50&C<0.65');
end
if C>=0.65
    pd1=4;
%     fprintf('C>=0.65');
end

pdd=max(pd,pd1);

if pdd==1
    pddstr='1 级（好）';
%     fprintf(' 故根据经验，预测精度为 1 级（好）\n\n');
end
if pdd==2
    pddstr='2 级（合格）' ;
%     fprintf(' 故根据经验，预测精度为 2 级（合格）\n\n');
end
if pdd==3
    pddstr='3 级（勉强合格）';
%     fprintf(' 故根据经验，预测精度为 3 级（勉强合格）\n\n');
end
if pdd==4
    pddstr='4 级（不合格）';
%     fprintf(' 故根据经验，预测精度为 4 级（不合格）\n\n');
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                  %%
%%            三、预测              %%
%%                                  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%8 模型经验合格后可用于预测，预测公式为：X0(i+1)=Xx(i+1)-Xx(i)
a1=size(IIi);
n1=a1(2);
X0X=ones(1,n1);
for i=1:n1
    X0X(i)=xua*(exp((-a)*(IIi(i)))-exp((-a)*(IIi(i)-1)));
%     fprintf('预测结果：第%d 个值(原数据的第%d 个序列号)的预测结果为：
%f\n',IIi(i),IIi(i)+1,X0X(i));
end
px=[1:n];
% plot(px,x,'-k',IIi+1,X0X,'-*r');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                  %%
%%        三、预测值精度评估        %%
%%                                  %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%计算出的模型值为
xg=ones(1,n);
xg(1)=x(1);
for i=2:n
    xg(i)=xua*(exp((-a)*(i-1))-exp((-a)*(i-2)));
end
% fprintf('计算出的模型值为：');
% fprintf('%f ',xg);
% fprintf('\n');

%残差分别为
eg=x-xg;
% fprintf('残差分别为：');
% fprintf('%f ',eg);
% fprintf('\n');
```

```
Q=inv(B'*B);
Q11=Q(1,1);
Q12=Q(1,2);
Q21=Q(1,2);
Q22=Q(2,2);

q0=((eg*eg')/(n-1))^0.5;

q0q=ones(1,n);
for k=0:n-1

q0q(k+1)=((a*k*x(1)-x(1)-k*u)^2*Q11+Q22+2*(a*k*x(1)-x(1)-k*u)*Q12)^0.5*exp(-a*k
)*q0;
end
% fprintf('\n');
% for i=1:n
% fprintf('原数据的第%d 个序列号的预测值为：%f±%f\n',i,x(i),q0q(i));
% end
% fprintf('\n');
%=======================
AUA=[a;u;ua;xua];
XEG=[xg;eg];
PPC=[pp;C];
%=======================
%所以预测值为
XIi=0;
if isempty(IIi)
    XIi='-';
    X0X='-';
    Q0Q='-';
    return;
end
fprintf('');
Q0Q=ones(1,n1);
for k=1:n1

Q0Q(k)=((a*IIi(k)*x(1)-x(1)-IIi(k)*u)^2*Q11+Q22+2*(a*IIi(k)*x(1)-x(1)-IIi(k)*u)
*Q12)^0.5*exp(-a*IIi(k))*q0;
%      fprintf('预测结果：第%d 个值(原数据的第%d 个序列号)的预测值为：%f±
%f\n',IIi(k),IIi(k)+1,X0X(k),Q0Q(k));
end
```

四　遗传算法源代码

说明: fga.m 为遗传算法的主程序; 采用二进制Gray编码,采用基于轮盘赌法的非线性排名选择, 均匀交叉,变异操作,而且还引入了倒位操作!

```
function
[BestPop,Trace]=fga(FUN,LB,UB,eranum,popsize,pCross,pMutation,pInversion,options)
% [BestPop,Trace]=fmaxga(FUN,LB,UB,eranum,popsize,pcross,pmutation)
% Finds a maximum of a function of several variables.
% fmaxga solves problems of the form:
%      max F(X) subject to: LB <= X <= UB
% BestPop       - 最优的群体即为最优的染色体群
% Trace         - 最佳染色体所对应的目标函数值
% FUN           - 目标函数
% LB            - 自变量下限
% UB            - 自变量上限
% eranum        - 种群的代数,取 100--1000(默认 200)
% popsize       - 每一代种群的规模; 此可取 50--200(默认 100)
% pcross        - 交叉概率,一般取 0.5--0.85 之间较好(默认 0.8)
% pmutation     - 初始变异概率,一般取 0.05-0.2 之间较好(默认 0.1)
% pInversion    - 倒位概率,一般取 0.05－0.3 之间较好(默认 0.2)
% options       - 1*2 矩阵,options(1)=0 二进制编码(默认 0),option(1)~=0
%十进制编码,option(2)设定求解精度(默认 1e-4)
-------------------------------------------------------------------

T1=clock;
if nargin<3, error('FMAXGA requires at least three input arguments'); end
if nargin==3,
eranum=200;popsize=100;pCross=0.8;pMutation=0.1;pInversion=0.15;options=[0 1e-4];end
if nargin==4,
popsize=100;pCross=0.8;pMutation=0.1;pInversion=0.15;options=[0 1e-4];end
if nargin==5, pCross=0.8;pMutation=0.1;pInversion=0.15;options=[0 1e-4];end
if nargin==6, pMutation=0.1;pInversion=0.15;options=[0 1e-4];end
if nargin==7, pInversion=0.15;options=[0 1e-4];end
if find((LB-UB)>0)
   error('数据输入错误,请重新输入(LB<UB):');
end
s=sprintf('程序运行需要约%.4f 秒钟时间,请稍等......',(eranum*popsize/1000));
disp(s);

global m n NewPop children1 children2 VarNum
```

```
bounds=[LB;UB]';bits=[];VarNum=size(bounds,1);
precision=options(2);%由求解精度确定二进制编码长度
bits=ceil(log2((bounds(:,2)-bounds(:,1))'./precision));%由设定精度划分区间
[Pop]=InitPopGray(popsize,bits);%初始化种群
[m,n]=size(Pop);
NewPop=zeros(m,n);
children1=zeros(1,n);
children2=zeros(1,n);
pm0=pMutation;
BestPop=zeros(eranum,n);%分配初始解空间 BestPop,Trace
Trace=zeros(eranum,length(bits)+1);
i=1;
while i<=eranum
    for j=1:m
        value(j)=feval(FUN(1,:),(b2f(Pop(j,:),bounds,bits)));%计算适应度
    end
    [MaxValue,Index]=max(value);
    BestPop(i,:)=Pop(Index,:);
    Trace(i,1)=MaxValue;
    Trace(i,(2:length(bits)+1))=b2f(BestPop(i,:),bounds,bits);
    [selectpop]=NonlinearRankSelect(FUN,Pop,bounds,bits);%非线性排名选择
[CrossOverPop]=CrossOver(selectpop,pCross,round(unidrnd(eranum-i)/eranum));
%采用多点交叉和均匀交叉，且逐步增大均匀交叉的概率
    %round(unidrnd(eranum-i)/eranum)
    [MutationPop]=Mutation(CrossOverPop,pMutation,VarNum);%变异
    [InversionPop]=Inversion(MutationPop,pInversion);%倒位
    Pop=InversionPop;%更新
pMutation=pm0+(i^4)*(pCross/3-pm0)/(eranum^4);
%随着种群向前进化，逐步增大变异率至 1/2 交叉率
    p(i)=pMutation;
    i=i+1;
end
t=1:eranum;
plot(t,Trace(:,1)');
title('函数优化的遗传算法');xlabel('进化世代数(eranum)');ylabel('每一代最优适应度(maxfitness)');
[MaxFval,I]=max(Trace(:,1));
X=Trace(I,(2:length(bits)+1));
hold on; plot(I,MaxFval,'*');
text(I+5,MaxFval,['FMAX=' num2str(MaxFval)]);
```

```
str1=sprintf('进化到 %d 代 ,自变量为 %s 时,得本次求解的最优值 %f\n 对应染色体是：%s',I,num2str(X),MaxFval,num2str(BestPop(I,:)));
disp(str1);
%figure(2);plot(t,p);%绘制变异值增大过程
T2=clock;
elapsed_time=T2-T1;
if elapsed_time(6)<0
    elapsed_time(6)=elapsed_time(6)+60;
elapsed_time(5)=elapsed_time(5)-1;
end
if elapsed_time(5)<0
    elapsed_time(5)=elapsed_time(5)+60;elapsed_time(4)=elapsed_time(4)-1;
end %像这种程序当然不考虑运行上小时啦
str2=sprintf('程序运行耗时 %d 小时 %d 分钟 %.4f 秒
',elapsed_time(4),elapsed_time(5),elapsed_time(6));
disp(str2);

%初始化种群
%采用二进制 Gray 编码,其目的是为了克服二进制编码的 Hamming 悬崖缺点
function [initpop]=InitPopGray(popsize,bits)
len=sum(bits);
initpop=zeros(popsize,len);%The whole zero encoding individual
for i=2:popsize-1
    pop=round(rand(1,len));
    pop=mod(([0 pop]+[pop 0]),2);
    %i=1 时,b(1)=a(1);i>1 时,b(i)=mod(a(i-1)+a(i),2)
    %其中原二进制串:a(1)a(2)...a(n),Gray 串:b(1)b(2)...b(n)
    initpop(i,:)=pop(1:end-1);
end
initpop(popsize,:)=ones(1,len);%The whole one encoding individual

%解码

function [fval] = b2f(bval,bounds,bits)
% fval   - 表征各变量的十进制数
% bval   - 表征各变量的二进制编码串
% bounds - 各变量的取值范围
% bits   - 各变量的二进制编码长度
scale=(bounds(:,2)-bounds(:,1))'./(2.^bits-1); %The range of the variables
numV=size(bounds,1);
cs=[0 cumsum(bits)];
for i=1:numV
a=bval((cs(i)+1):cs(i+1));
```

```
fval(i)=sum(2.^(size(a,2)-1:-1:0).*a)*scale(i)+bounds(i,1);
end

selectprob=fit/sum(fit);%计算各个体相对适应度(0,1)
q=max(selectprob);%选择最优的概率
x=zeros(m,2);
x(:,1)=[m:-1:1]';
[y x(:,2)]=sort(selectprob);
r=q/(1-(1-q)^m);%标准分布基值
newfit(x(:,2))=r*(1-q).^(x(:,1)-1);%生成选择概率
newfit=cumsum(newfit);%计算各选择概率之和
rNums=sort(rand(m,1));
fitIn=1;newIn=1;
while newIn<=m
    if rNums(newIn)<newfit(fitIn)
        selectpop(newIn,:)=pop(fitIn,:);
        newIn=newIn+1;
    else
        fitIn=fitIn+1;
    end
end

%交叉操作
function [NewPop]=CrossOver(OldPop,pCross,opts)
%OldPop 为父代种群，pcross 为交叉概率
global m n NewPop
r=rand(1,m);
y1=find(r<pCross);
y2=find(r>=pCross);
len=length(y1);
if len>2&mod(len,2)==1%如果用来进行交叉的染色体的条数为奇数，将其调整为偶数
    y2(length(y2)+1)=y1(len);
    y1(len)=[];
end
if length(y1)>=2
   for i=0:2:length(y1)-2
       if opts==0
           [NewPop(y1(i+1),:),NewPop(y1(i+2),:)]=EqualCrossOver(OldPop(y1(i+1),:),OldPop(y1(i+2),:));
       else
           [NewPop(y1(i+1),:),NewPop(y1(i+2),:)]=MultiPointCross(OldPop(y1(i+1),:),OldPop(y1(i+2),:));
       end
```

```
   end
end
NewPop(y2,:)=OldPop(y2,:);

%采用均匀交叉
function [children1,children2]=EqualCrossOver(parent1,parent2)

global n children1 children2
hidecode=round(rand(1,n));%随机生成掩码
crossposition=find(hidecode==1);
holdposition=find(hidecode==0);
children1(crossposition)=parent1(crossposition);%掩码为 1，父 1 为子 1 提供基因
children1(holdposition)=parent2(holdposition);%掩码为 0，父 2 为子 1 提供基因
children2(crossposition)=parent2(crossposition);%掩码为 1，父 2 为子 2 提供基因
children2(holdposition)=parent1(holdposition);%掩码为 0，父 1 为子 2 提供基因

%采用多点交叉，交叉点数由变量数决定

function [Children1,Children2]=MultiPointCross(Parent1,Parent2)

global n Children1 Children2 VarNum
Children1=Parent1;
Children2=Parent2;
Points=sort(unidrnd(n,1,2*VarNum));
for i=1:VarNum
    Children1(Points(2*i-1):Points(2*i))=Parent2(Points(2*i-1):Points
(2*i));
    Children2(Points(2*i-1):Points(2*i))=Parent1(Points(2*i-1):Points
(2*i));
end

%变异操作
function [NewPop]=Mutation(OldPop,pMutation,VarNum)

global m n NewPop
r=rand(1,m);
position=find(r<=pMutation);
len=length(position);
if len>=1
   for i=1:len
       k=unidrnd(n,1,VarNum); %设置变异点数，一般设置 1 点
```

```
        for j=1:length(k)
            if OldPop(position(i),k(j))==1
               OldPop(position(i),k(j))=0;
            else
               OldPop(position(i),k(j))=1;
            end
        end
    end
end
NewPop=OldPop;

%倒位操作

function [NewPop]=Inversion(OldPop,pInversion)

global m n NewPop
NewPop=OldPop;
r=rand(1,m);
PopIn=find(r<=pInversion);
len=length(PopIn);
if len>=1
    for i=1:len
       d=sort(unidrnd(n,1,2));
       if d(1)~=1&d(2)~=n
          NewPop(PopIn(i),1:d(1)-1)=OldPop(PopIn(i),1:d(1)-1);
          NewPop(PopIn(i),d(1):d(2))=OldPop(PopIn(i),d(2):-1:d(1));
          NewPop(PopIn(i),d(2)+1:n)=OldPop(PopIn(i),d(2)+1:n);
       end
    end
end
```