

参考他人报告或代码的申明

统计推断课程，2015 年秋季学期第 30 组，成员：朱廷杰学号 5140309252，林顺达学号 5140309254，在报告编写过程中，以下方面参考了往届报告，现列表说明：

主要参考项目	说明
代码方面	《统计推断在数模模数转换中的应用》，刘昊，2013 年秋季学期，组号 03 在该组报告附录提供的程序代码基础上，进行了少量修改。

除了以上注明的参考内容，和报告中列出的引用文献注解，本报告其他部分都不包含任何其他个人或集体已经发表或撰写过的作品成果。

统计推断在数模转换系统中的应用

组号：30 小组成员：朱廷杰 5140309252 林顺达 5140309254

摘要：本报告以上海交通大学工程实践与科技创新 3A 的实验结果研究数据，探究统计推断在数模转换系统中的应用。在实际的工作过程中，如果我们选择对每个样品都测定 51 个数据，工作量和成本无疑十分巨大。因此，我们选择了使用插值与拟合的方法，通过取样本中的特定的点进行拟合，这条拟合曲线可以较为精确的反映原本的函数曲线，并且成本也在可以接受的范围之内。报告以遗传算法为基本思路，采用了三次样条插值和立方插值的方式，最终得出了比较令人满意的取点方式。

关键词：遗传算法，适应度，交叉，变异，插值方式

1 引言

本课程要研究的问题是为某种产品内部的一个检测模块寻求校准工序的优化方案。在实际生产的过程中，以本课程的样品为例，如果每一个样品都进行 51 次取点，由于时间和成本过于巨大，实际上是不可能实现的。因此，在实际的生产生活中，由于样品的函数具有相似性，所以往往采取用一定数目的观测点，通过观测点来反映具体函数的方法来进行校准。

2 数据样本分析

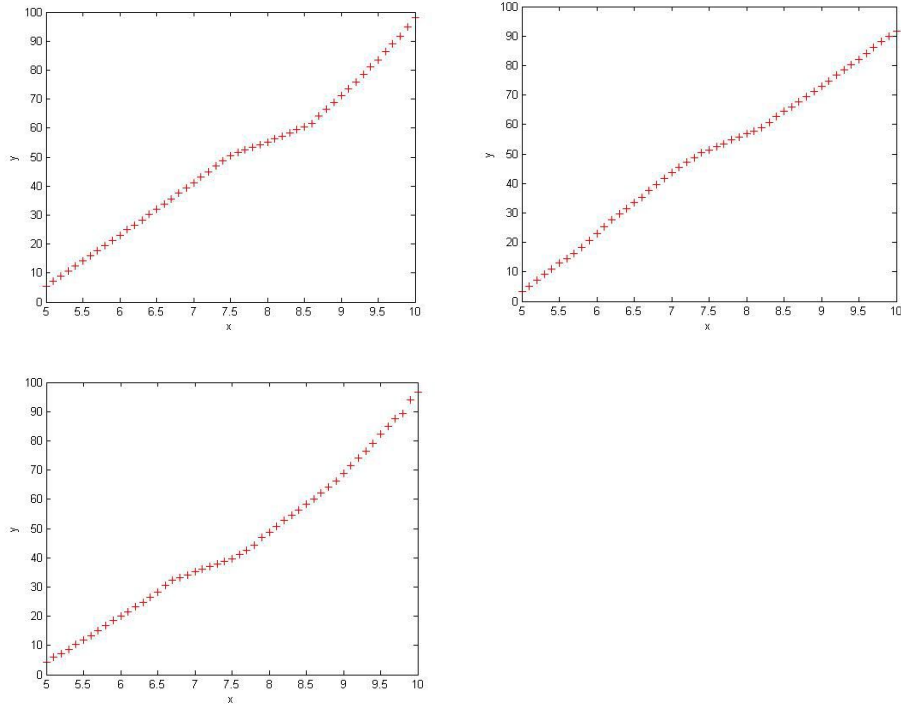


图 2-1 3 个数据样本的 x-y 曲线

图 2-1 给出了三组数据的 x-y 曲线，尽管曲线不是完全的一致，但是基本都满足如下规律：

我们发现图像大致分为三个部分，5-7,8.5-10 区间的弯曲度比较大，7-8.5 区间的弯曲度比较小。

3 评价方式的构建

为了解决这个问题，我们需要从数据中寻找几个数据点。而我们所选取的数据点，必然要满足在评价函数下是最优解，因此我们定义如下的误差成本函数：

$$s_{i,j} = \begin{cases} 0 & \text{if } |\hat{y}_{i,j} - y_{i,j}| \leq 0.4 \\ 0.1 & \text{if } 0.4 < |\hat{y}_{i,j} - y_{i,j}| \leq 0.6 \\ 0.7 & \text{if } 0.6 < |\hat{y}_{i,j} - y_{i,j}| \leq 0.8 \\ 0.9 & \text{if } 0.8 < |\hat{y}_{i,j} - y_{i,j}| \leq 1 \\ 1.5 & \text{if } 1 < |\hat{y}_{i,j} - y_{i,j}| \leq 2 \\ 6 & \text{if } 2 < |\hat{y}_{i,j} - y_{i,j}| \leq 3 \\ 12 & \text{if } 3 < |\hat{y}_{i,j} - y_{i,j}| \leq 5 \\ 25 & \text{if } |\hat{y}_{i,j} - y_{i,j}| > 5 \end{cases} \quad (3-1)$$

下标 i 代表样品序号，下标 j 代表观测点序号。

并且观测点的个数也会引起一部分的成本，我们定义对某一样品 i 的定标成本如下：

$$S_i = \sum_{j=1}^{51} s_{i,j} + 12N_i \quad (3-2)$$

所以我们得出定标方案的总成本为：

$$C = \frac{1}{M} \sum_{i=1}^M S_i \quad (3-3)$$

接下来我们所需要做的就是寻找成本最低的定标方案。

4 遗传算法的设计

为了获得最佳的取点方式，我们采用了遗传算法。遗传算法是模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型，是一种通过模拟自然进化过程搜索最优解的方法。他的基本过程包含：初始化，计算适应度，选择，交叉，变异，迭代，终止。

4.1 初始化与终止

初始化是指生成初始种群，而终止则是满足条件后种群停止进化。在本实验中，初始种群采用随机生成的方式，以具有 7 个基因位的基因为例：首先头尾的 1 和 51 这两个观测点需要保证；然后随机生成 5 个 2~50 之间的整数作为剩下的 5 位基因。重复这个过程 N 次，就能得到一个包含 N 个个体的种群作为初始种群。

对于个体基因的存储方式，我们没有选择二进制数组，而是考虑采用了整数数组。这里是觉得采用二进制数组浪费的空间比较大，数组里绝大部分基因位都是 0。而且就本课题而言，整数数组在变异方面要优于二进制数组，具体原因会在变异环节阐述。

整数数组存储基因示例：[1, 6, 15, 27, 34, 44, 51] 就表示取点方案为第 1、5、15、27、34、44、51 个观测点的个体。

在实验中，我们发现迭代到后期的话种群适应度收敛，所以我们设置了 100 代的循环代数，当达到 100 次之后进化就终止了。

4.2 适应度的计算

公式 (3-2) 和 (3-3) 给出了每一代中个体的成本与这一代平均成本的计算方法。每个个体代表了一种取点方案，对每种取点方案，通过对所给的 400 组数据进行拟合和成本计算，可以计算出这种方案的平均定标成本。平均定标成本越低，说明这个方案越优质，反映在遗传算法中就是该个体的适应度比较高。所以可以选择平均定标成本的倒数作为每个个体的适应度。

4.3 选择

选择过程就是模拟优胜劣汰的过程，适应度高的个体有更大的机会被留下。对所有个体的适应度求和，取每个个体的适应度与适应度总和的比值作为该个体的选择概率，将个体选择概率依次累加后得到 [0, 1] 内递增的累计概率，通过在 [0, 1] 内生成随机数得到子代个体。为了获得更好的效果，这里人为地保证前一代最优的个体能够被选择留下。

4.4 交叉和变异

交叉的过程是模拟自然界中的基因重组，其目的就是保持基因的丰富性，通过交叉的过程，可以保证样品的种群始终保持一定的大小。变异是模拟自然界中的基因突变的过程，这是产生新的基因的唯一方法。突变在该算法的意义是在开始选取种群时，我们有一定几率会把最优解排除在种群外。通过变异，我们可以保证种群中会有新的样品补充进来，保证了结果的准确性。

交叉之前，通过之前规定的交叉概率求得参与交叉的个体数，再由随机数生成需要参加交叉的个体以及交叉点的位置，然后交换两组基因交叉点之后的基因位。如基因为[1, 3, 11, 45, 51]和[1, 6, 9, 25, 51]的个体，交叉点为 30，则交叉后得到的基因为[1, 3, 11, 51]和[1, 6, 9, 25, 45, 51]。由于本课题的最优解不仅取决于取点的位置，而且还和取点的个数有关，这么交叉有几率可以改变个体基因位的数量，即取点的个数，所以这么设计可以同时完成对取点数量的研究。

对于每位基因，通过生成的[0, 1]之间的随机数与变异概率的比较决定其是否变异，若变异则随机在突变范围内改变其值，即若原基因位为 30，突变范围为 ± 1 ，则变异后可能的值为 29, 30, 31。由于在迭代很多代之后，往往当前的最优解与实际最优解的差别可能只是在某些基因位上相差 ± 1 ，所以这种变异方式相较于二进制存储法使用的等位基因突变，可以更精确地“寻找”最优解。

4.4 迭代

重复上述的选择、交叉、变异过程，种群的整体适应度会越来越高，逐渐收敛，此时得到的适应度最高的个体就接近所要求的最优解。

5 遗传算法的参数选择

遗传算法实际使用效果与参数的选择有很大的关系，程序完成后，我们用不同的参数试运行了数次，对于参数的选择有了一个初步的认识。

5.1 初始基因基因位数量

初始基因基因位数量就是初始的取点数量。我们采用了以下方法决定这个参数的大小：在迭代、交叉的过程中取点数量会收敛到最优的取点数量，根据试运行的结果直接选择最优的取点数量作为初始的取点数量。

5.2 种群个体数

种群个体数决定每一代的个体数量。这个参数设置大的话，开始时可以提供丰富的基因型供选择交叉，但是种群的整体适应度会收敛得比较慢，而且每一代迭代的时间会比较久。我们试过了 50,100,200 这三种种群个体数，个体数 50 的情况迭代一代大概 20 秒，个体数 100 的情况迭代一代大概 45 秒，个体数 200 的情况迭代一代大概 90 秒。这三种个体数最后得到的结果都差不多，考虑到时间成本，所以我们最后选择了 50 的个体数。

5.3 循环代数

迭代到后期的话种群适应度收敛，每一代的变化就比较小了，所以循环代数不用设置得很大，这里选择 100 作为循环代数。

5.4 变异概率和变异区间

变异概率和变异区间设置比较大的话有利于快速寻找优质解，但是种群收敛得就比较慢，而且波动会比较大。变异区间设置小的话有利于迭代后期精确寻找最优解，这里我们是选择了小的变异区间 ± 1 。

5.5 拟合方式

我们先后试验了两种拟合方式，分别是三次样条插值和立方插值。

三次样条插值的定义如下：

假设在[a,b]上测量有 $n-1$ 个点，在[a,b]上划分。 $\Delta: a=x_0 < x_1 < \dots < x_{n-1} < x_n=b$ ，则三次样条

插值法所得的函数 $S(x)$ 具有如下特征：

- (1) 在每个小区间 $[x_j, x_{j+1}]$ 上是三次多项式。
- (2) 在每个节点 $S(x_j)=y_j$ 。

(3) $S(x)$ 在 $[a, b]$ 二阶导数连续。

因为 $S(x)$ 在 $[a, b]$ 上二阶导数连续, 所以在每个节点, 由连续性得 $S(x_j-0)=S(x_j+0)$; $S'(x_j-0)=S'(x_j+0)$; $S''(x_j-0)=S''(x_j+0)$ 。共 $3n-3$ 个条件; 除此之外, 由插值得得 $n+1$ 个条件, 仍需两个, 这两个条件有边值得出:

(1) 给定断点 x_0, x_n 处的一阶导数值

(2) 给定断点 x_0, x_n 处的二阶导数值 (特别的 $S''(x_0+0)=0, S''(x_n+0)=0$ 称为自然边

值条件)

(3) 周期性便捷条件: 即以 $b-a$ 为周期, $S''(x_0+0)=S''(x_n+0)$, $S'(x_0+0)=S'(x_n+0)$ 。

这样求出的三次曲线比 2.4.1 中的多项式拟合应该要精确许多。

立方插值的定义如下:

样条曲线 $S(x)$ 是一个分段定义的公式。给定 $n+1$ 个数据点, 共有 n 个区间, 三次样条方程满足以下条件:

a. 在每个分段区间 $[x_i, x_{i+1}]$ ($i = 0, 1, \dots, n-1, x$ 递增), $S(x) = S_i(x)$ 都是一个三次多项式。

b. 满足 $S(x_i) = y_i$ ($i = 0, 1, \dots, n$)

c. $S(x)$, 导数 $S'(x)$, 二阶导数 $S''(x)$ 在 $[a, b]$ 区间都是连续的, 即 $S(x)$ 曲线是光滑的。

所以 n 个三次多项式分段可以写作:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \quad i = 0, 1, \dots, n-1$$

其中 a_i, b_i, c_i, d_i 代表 $4n$ 个未知系数。

两种插值方法的区别在于边界条件的不同, 在我们使用的 matlab 软件中, 三次样条插值是默认为非扭结边界, 使两端点的三阶导与这两端点的邻近点的三阶导相等。二立方插值有定义可知, 只是满足边界二阶导数连续。选择三次样条插值 (spline) 时, 最优解在 96~97 附近, 选择立方插值 (cubic) 时, 最优解在 89~90 附近。

6 对参数选择的尝试

在分析的过程中, 我们就种群规模这个参数进行讨论, 为了观察出这个参数对结果的影响, 我们根据数据做分别做出种群 50 迭代 200 次, 种群 100 迭代 200 次, 种群 200 迭代 200 次的图像。

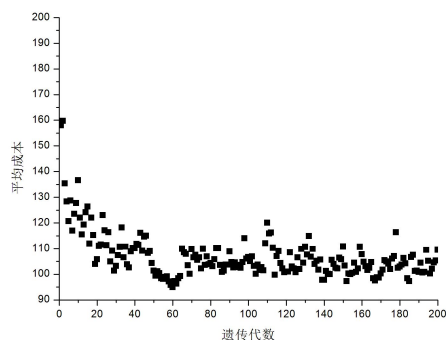


图 6-1 种群 50 迭代 200 代

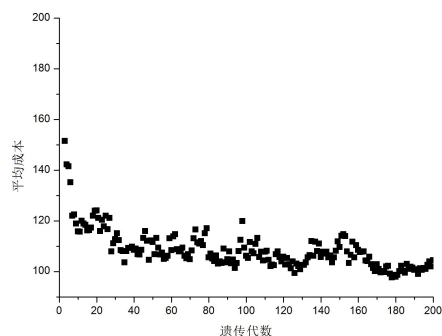


图 6-2 种群 100 迭代 200 代

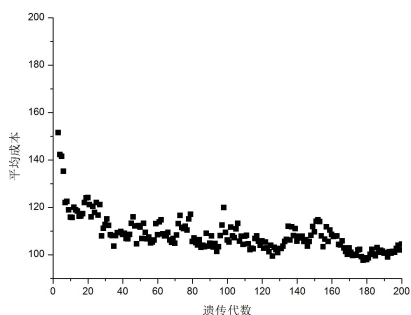


图 6-3 种群 200 迭代 200 代

通过分析比较，我们得出结论，在遗传代数足够大的情况下，例如图片中的 200 代，种群数的大小对收敛的结果没有显著的影响。但是种群数越大，收敛的速度越快。例如如图 6-1 在 100 代之前依然存在着巨大的波动。而在图 6-2, 6-3 中，平均成本的数值在 0-40 代是极速下降，迅速在 110 左右稳定。

我们的程序中使用的是种群数 50 迭代 100 代，由上图的结论得知在 100 代时对结果并无显著影响，而且经过我们的测试，个体数 50 的情况迭代一代大概 20 秒，个体数 100 的情况迭代一代大概 45 秒，个体数 200 的情况迭代一代大概 90 秒。考虑到时间成本和精确度，50 的种群数是完全合理的。

7 参数选择和程序运行结果

经过前面对参数选择的分析和尝试，最终我们程序的参数选定为：

初始基因基因位数量： 6

种群个体数： 50

循环代数： 100

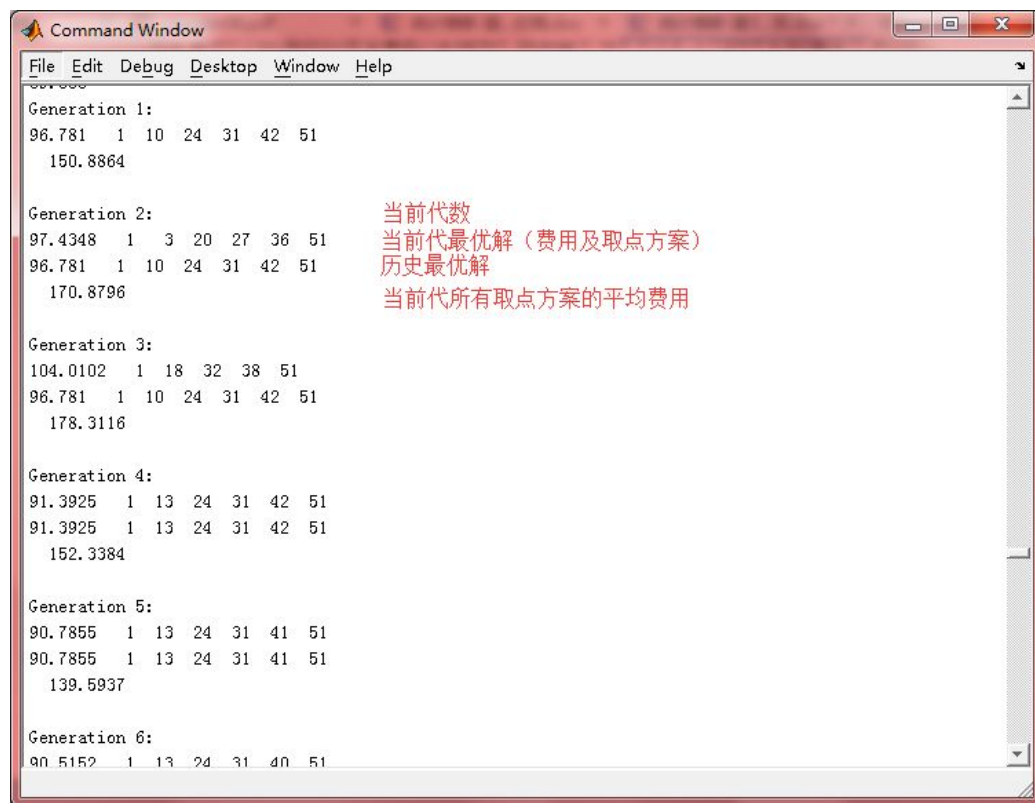
交叉概率： 0.5

变异概率： 0.1

变异区间： ± 1

拟合方式分别取立方插值和三次样条插值

程序运行效果图：



```
Command Window
File Edit Debug Desktop Window Help

Generation 1:
96.781 1 10 24 31 42 51
150.8864

Generation 2:
97.4348 1 3 20 27 36 51
96.781 1 10 24 31 42 51
170.8796

Generation 3:
104.0102 1 18 32 38 51
96.781 1 10 24 31 42 51
178.3116

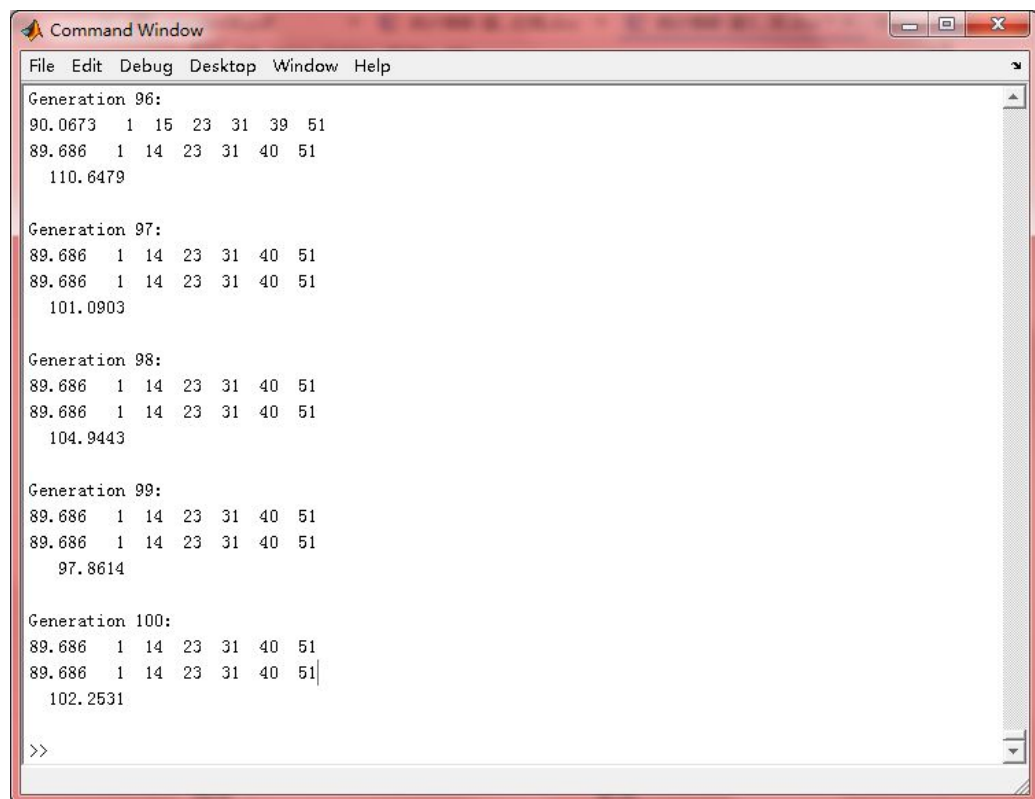
Generation 4:
91.3925 1 13 24 31 42 51
91.3925 1 13 24 31 42 51
152.3384

Generation 5:
90.7855 1 13 24 31 41 51
90.7855 1 13 24 31 41 51
139.5937

Generation 6:
90.5152 1 13 24 31 40 51
```

当前代数
当前代最优解 (费用及取点方案)
历史最优解
当前代所有取点方案的平均费用

迭代 100 代大约花费了半个小时, 程序最后的运行结果为 (立方插值)：



```
Command Window
File Edit Debug Desktop Window Help

Generation 96:
90.0673 1 15 23 31 39 51
89.686 1 14 23 31 40 51
110.6479

Generation 97:
89.686 1 14 23 31 40 51
89.686 1 14 23 31 40 51
101.0903

Generation 98:
89.686 1 14 23 31 40 51
89.686 1 14 23 31 40 51
104.9443

Generation 99:
89.686 1 14 23 31 40 51
89.686 1 14 23 31 40 51
97.8614

Generation 100:
89.686 1 14 23 31 40 51
89.686 1 14 23 31 40 51
102.2531

>>
```

三次样条插值：

```

Command Window
File Edit Debug Desktop Window Help
Generation 96:
96.5113 1 9 20 27 35 44 51
96.5113 1 9 20 27 35 44 51
116.9548

Generation 97:
96.5113 1 9 20 27 35 44 51
96.5113 1 9 20 27 35 44 51
113.7408

Generation 98:
96.5113 1 9 20 27 35 44 51
96.5113 1 9 20 27 35 44 51
114.3201

Generation 99:
96.5113 1 9 20 27 35 44 51
96.5113 1 9 20 27 35 44 51
109.5597

Generation 100:
96.5113 1 9 20 27 35 44 51
96.5113 1 9 20 27 35 44 51
110.8252

>>

```

7 结论

通过三次样条插值，我们得出如下最优解：

成本				取点			
96.51	1	9	20	27	35	44	51

通过立方插值，可以得出如下最优解：

成本				取点			
89.686	1	14	23	31	40	51	

因此，我们找到的最佳取点方式是：

成本				取点			
89.686	1	14	23	31	40	51	

插值方式为立方插值。

8 参考文献

- [1] 上海交大电子工程系·统计推断在数模转换系统中的应用课程讲义 [EB/OL].<ftp://202.120.39.248>.
- [2] 词条“三次插值拟合”，“遗传算法”. 百度文库.
- [3] 三次样条插值 (Cubic Spline Interpolation) 及代码实现 (C 语言). 百度文库：
http://wenku.baidu.com/link?url=3SIFsezlWfhS_mexZR9e-hbCTkxVQX9nivQLdQTBMcM

ohKx8TkglXBPwPpinwWVSNsmKduNO0S45bUHMmAHcZrGVSSdCJQPLTbZ1OM1rk
1S.

附录:

遗传算法代码:

```
function f=geneticalgorithm()
    global popsize;popsize=51;%群体大小
    global N;N=50;%种群数
    global pc;pc=0.5;%交叉概率
    global generation;generation=100;%循环代数
    global pm;pm=0.1;%变异概率
    global NumOfDatas;NumOfDatas=400;%数据组数
    A=csvread('20150915dataform.csv');
    cost=zeros(1,N);%每组基因的平均标定费用
    groupfitness=zeros(1,N);%种群适应度

    %生成初始种群 group
    for i=1:N
        group{i}=[1,popsize,ceil(rand(1,4)*49+1)];
        group{i}=unique(group{i});
        group{i}=sort(group{i});
        %group{i} %测试代码
        cost(i)=avgcost(A,group{i});
        groupfitness(i)=1/cost(i); %种群适应度
    end

    [bestvalue,index]=min(cost);
    bestgene=group{index};

    disp('Generation 1:');
    disp([num2str(bestvalue),' ',num2str(bestgene)]);
    disp(sum(cost)/N);

    for g=2:generation
        sumfitness=sum(groupfitness(:));
        groupfitness=groupfitness/sumfitness;
        groupfitness=cumsum(groupfitness);

        %自然选择得到下一代
        newgroup{1}=bestgene; %保证最优基因存在
```

```

for i=2:N
    k=rand(1,1);
    if(k<=groupfitness(1))
        newgroup{i}=group{1};
    end
    for j=1:N-1
        if(groupfitness(j)<k && k<=groupfitness(j+1))
            newgroup{i}=group{j+1};
        end
    end
end

%基因重组
for i=1:N
    cut=rand(1)*popsize;
    change1=ceil(rand()*N);
    change2=ceil(rand()*N);
    gene1=newgroup{change1};
    gene2=newgroup{change2};
    newgroup{change1}=[gene1(find(gene1<=cut)),gene2(find(gene2>cut))];
    newgroup{change2}=[gene2(find(gene2<=cut)),gene1(find(gene1>cut))];
end

%变异
for i=1:N
    for j=2:size(newgroup{i},2)-1
        if rand(1)<=pm
            move=floor(rand(1)*3)-1; %取样点在±1 的范围突变
            if (newgroup{i}(j)+move>=1 && newgroup{i}(j)+move<=51)
                newgroup{i}(j)=newgroup{i}(j)+move;
            end
        end
    end
    newgroup{i}=unique(newgroup{i});
    newgroup{i}=sort(newgroup{i});
end

%更新种群
cost=zeros(1,N);
groupfitness=zeros(1,N);
for i=1:N
    cost(i)=avgcost(A,newgroup{i});
    groupfitness(i)=1/cost(i);
end

```

```

[value,index]=min(cost);    %更新最优解

disp(['Generation ',num2str(g),':']);
disp([num2str(value),'    ',num2str(newgroup{index})]);

if value<=bestvalue
    bestvalue=value;
    bestgene=newgroup{index};
else
    newgroup{1}=bestgene;
    groupfitness(1)=1/bestvalue;
end
group=newgroup;

disp([num2str(bestvalue),'    ',num2str(bestgene)]);
disp(sum(cost)/N);
end
end

```

%每组基因的平均定标成本

```

function f=avgcost(A,array)
    global NumOfDatas; %数据组数
    global popsize;
    sum=0;
    chooseX=[];
    x=A(1,1:popsize);
    for i=1:size(array,2)
        chooseX=[chooseX,x(array(i))];
    end
    for i=1:NumOfDatas
        y=A(2*i,1:popsize);
        sum=sum+onedatacost(x,y,chooseX,array);
    end
    f=sum/NumOfDatas;
end

```

%每组数据的定标成本

```

function f=onedatacost(x,y,chooseX,array)
    global popsize;
    costOfPerCheck=12;    %每个抽样点的测定成本
    numOfPoint=size(array,2); %抽样点数目
    chooseY=[];
    for i=1:numOfPoint
        chooseY=[chooseY,y(array(i))];
    end

```

```

end
y2=interp1(chooseX,chooseY,x,'cubic'); %立方插值
cost=0; %定标成本
error=0; %误差

%计算误差成本和测定成本
for i=1:popsiz
    error=abs(y2(i)-y(i));
    if error<=0.4
        cost=cost+0;
    elseif error<=0.6
        cost=cost+0.1;
    elseif error<=0.8
        cost=cost+0.7;
    elseif error<=1;
        cost=cost+0.9;
    elseif error<=2;
        cost=cost+1.5;
    elseif error<=3;
        cost=cost+6;
    elseif error<=5;
        cost=cost+12;
    else
        cost=cost+25;
    end
end
cost=cost+numOfPoint*costOfPerCheck;
f=cost;
end

```

test_ur_answer 代码:

%%%%%%%%%% 答案检验程序 2015-11-04 %%%%%%%%%%

```

my_answer=[ 1,14,23,31,40,51 ];%把你的选点组合填写在此
my_answer_n=size(my_answer,2);

```

```

% 标准样本原始数据读入
minput=dlmread('20150915dataform.csv');
[M,N]=size(minput);
nsample=M/2; npoint=N;
x=zeros(nsample,npoint);
y0=zeros(nsample,npoint);
y1=zeros(nsample,npoint);
for i=1:nsample

```

```

        x(i,:)=minput(2*i-1,:);
        y0(i,:)=minput(2*i,:);
    end
    my_answer_gene=zeros(1,npoint);
    my_answer_gene(my_answer)=1;

% 定标计算
index_temp=logical(my_answer_gene);
x_optimal=x(:,index_temp);
y0_optimal=y0(:,index_temp);
for j=1:nsample
    % 请把你的定标计算方法写入函数 mycurvefitting
    y1(j,:)=mycurvefitting(x_optimal(j,:),y0_optimal(j,:));
end

% 成本计算
Q=12;
errabs=abs(y0-y1);

le0_4=(errabs<=0.4);
le0_6=(errabs<=0.6);
le0_8=(errabs<=0.8);
le1_0=(errabs<=1);
le2_0=(errabs<=2);
le3_0=(errabs<=3);
le5_0=(errabs<=5);
g5_0=(errabs>5);

sij=0.1*(le0_6-le0_4)+0.7*(le0_8-le0_6)+0.9*(le1_0-le0_8)+1.5*(le2_0-le1_0)+6*(le3_0-le2_0)+
12*(le5_0-le3_0)+25*g5_0;
si=sum(sij,2)+Q*ones(nsample,1)*my_answer_n;
cost=sum(si)/nsample;

% 显示结果
fprintf('\n 经计算，你的答案对应的总体成本为%5.2f\n',cost);

```