统计推断在数模转换系统中的应用

组号 59　姓名 方斯瑞 学号 5130309265，姓名 颜铭贤 学号 5130309731

摘要：为了研究对产品监测模块传感器优化、降低成本，我们进行采样进行拟合。预设条件提供给我们的数据资料有 469 组样本，而每个样本有 51 个观测点(采样点)。而为了完整测定这些数值，需要耗时费力费钱。如果我们能找出一种方式，能减少观测点数量，却又不至于过度拉大标准差，既可提高功效，又能降低成本。想解决这个问题，我们必须找出一个优化采样的算法。在初始探讨阶段，我们查询了解了几个方法，希望最终能在后期的实践操作中能以更小的工作量取得更大的成果。

关键词：插值法，占空比，函数关系，优化采样，样条插值，遗传算法

ABSTRACT:

We write this article for the reason that we want to introduce the appliance of the statistical inference in analog-to-digital conversion. We analyze lots of statistical data by measuring the D-U relation of Single-chip microcomputer. For the sake of calibrating the system fully, we find the section points out by the calculation of the arc string distance first, then characterize the whole system characteristics by using the 7 feature points found by using genetic algorithm, after that was the comparison which involves different interpolation(fitting). We also try to reduce the number of the feature points, like 6 or 5, and make analysis in the article. Meanwhile, we rectify some part of the Genetic algorithm.

Key words: Arc string distance, Polynomial fitting, Segmentation interpolation, Genetic algorithm

1.引言

传感器对整体产品占有极其重要的地位，其监测到的数据与其后续针对数据处理的反应机制，决定了整体产品的核心，实不可或缺。而对于传感器测定度准值的成本控制，更是重要的影响了产品整体的质量与稳定性。而在实际应用中，由于工业生产的产品的标准化非完美性，我们在使用生产商提供的资料时，总会同实际使用上有一定误差，因此我们要从产品函数特性分布信息，了解实际得到的数据与理想传感器间的差距。

而依照课程提供给我们的数据看来(469 组样本，每组 51 个观测点)，我们不可能一个个拟合每组数据来寻找最佳的传感器曲线。所以我们的课题在于选择合适的拟合方法后，运用遗传算法等对采样方式进行优化筛选。通过对本课题的研究，我们能够了解目前受欢迎的选点方法、遗传算法，并且对数据处理有更深的理解。

2.拟合或插值方法的研究

我们将数组数据输入 origin 划出拟合曲线，通过查看很多组拟合曲线，发现每个系统的关系曲线形状有极大相似性，所以我们决定对于每一组数据采用统一的方法得出曲线方程式来表示其函数关系，并进行得分评价作为标准(或成本估计)。首先，我们必须承认提供给我们的数据的传感器都是能正常运作的，那么我们的拟合工作才有意义。每一次拟合都有评分标准(成本计算)，对于每个点，由其观察值与加权值有如下的加权(评价函数)：

$$s_{i,j}=\begin{cases}0 & \text{if } \left|\hat{y}_{i,j}-y_{i,j}\right|\le 0.5\\ 0.5 & \text{if } 0.5<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 1\\ 1.5 & \text{if } 1<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 2\\ 6 & \text{if } 2<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 3\\ 12 & \text{if } 3<\left|\hat{y}_{i,j}-y_{i,j}\right|\le 5\\ 25 & \text{if } \left|\hat{y}_{i,j}-y_{i,j}\right|>5\end{cases}$$

图 2-1 误差成本计算公式[3]

而成本计算亦须考虑观测成本、误差成本

$$S_i=\sum_{j=1}^{51}s_{i,j}+q\cdot n_i$$

图 2-2 总成本计算公式(首项为误差成本，次项为测定成本,其中 q=12)[3]

2.1 多项式拟合

2.1.1 概述

实验所提供的数据曲线如下图所示绝大部分都很接近一个多项式曲线。所以我们尝试用多项式拟合曲线。

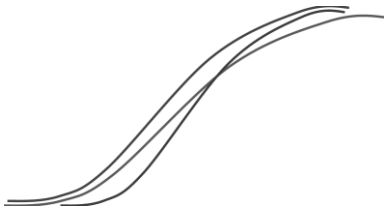

图 2-3 实验数据拟合曲线

2.1.2 高次拟合结果

图为 18 点 17 次拟合图，经过每一个点，但曲线已经特别扭曲，而实验中 53 个数据，扭曲程度会更高，这种拟合虽然拟合误差最小，但是很值得怀疑。故放弃了这一方法。

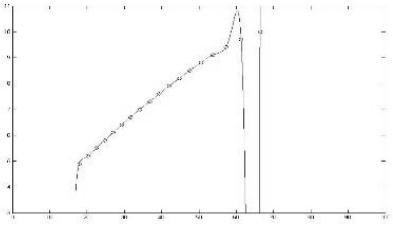

图 2-4 多项式拟合函数曲线

2.2 线性插值

2.2.1 概述

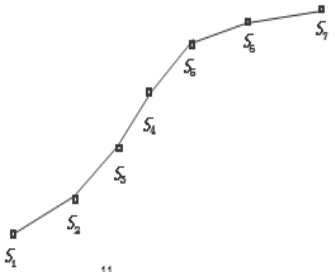

图 2-5 线性插值方法概述

假定观测了 7 组数值（7 个数据点）

假设我们已知坐标(x0,y0)与(x1,y1),要得到[x0,x1]区间内某一位置 x 在直线上的 y 值。

我们得到（y-y0）(x1-x0)=(y1-y0)(x-x0)。假设方程两边的值为 α，那么这个值就是插值系数—从 x0 到 x 的距离与从 x0 到 x1 距离的比值。由于 x 值已知，所以可以从公式得到 α 的值：α=(x-x0)/(x1-x0)，同样 α=(y-y0)/(y1-y0)。这样，在代数上就可以表示成为： y = (1- α)y0 + αy1，或者 y = y0 + α(y1 - y0)，这样通过 α 就可以直接得到 y。

2.2.2 Matlab 程序实现

根据《课程设计课题和要求》的传感器特性来看，线性插值对于可能是不完全线性的实验数据而言，线性插值对其会造成较大误差，故同样放弃了这一方法、

2.3 三次样条插值法

2.3.1 概述

假设在[a,b]上测量有 n-1 个点，在[a,b]上划分。Δ:a=x0<x1<….<xn-1<xn=b，则三次样条插值法所得的函数 S(x)具有如下特征[1]：

1、在每个小区间[xj,xj+1]上是三次多项式。

2、在每个节点 S(xj)=yj。

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(xj-0)=S(xj+0); S’(xj-0)=S’(xj+0); S’’(xj-0)=S’’(xj+0)。共 3n-3 个条件；除此之外，由插值可得 n+1 个条件，仍需两个，这两个条件有边值得出：

1、给定断点 x0,xn 处的一阶导数值

2、给定断点 x0,xn 处的二阶导数值（特别的 S’’(x0+0)=0, S’’(xn+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’’(x0+0)= S’’(xn+0)，S’(x0+0) =S’(xn+0)。

这样求出的三次曲线比 2.1 中的多项式拟合应该要精确许多。

2.3.1 三次样条插值法相比于七点法的优点

三次样条插值法相比于七点法对所测量得到的信息的应用足够充分，最后所得的曲线必过每个测量点，在每两个测量点之间都有自己的三次多项式。七点法最后所得的式子不一定过测量点。即用三次样条插值法所得的拟合方程，更加贴近真实的测量结果，同时也没有丧失多阶导数连续性。不过时间长效率低是一个不足。[2]

2.3.2 Matlab 程序实现

类似于线性，Matlab 中也提供了三次样条插值函数 spline，只需要调用语句

FitY = interp1（DataX，DataY，FitX，‘spline’）即可，其中 DataX 和 DataY 分别是待拟合数据，FitY 是根据拟合出的函数代入 FitX 值而得到的函数值。

3.遗传算法（GA）

3.1 概述

为了选点我们查询了遗传算法相关资料，我们整理基本运算过程如下：

A. 赋值：对遗传算法的执行参数赋值。参数例如进化代数计数器(常设成 t=0)、种群规模、变量个数、交叉概率以及最大进化代数(常设成 T)。

B. 初始化：建立区域描述器(依照条件订出变量取值范围)、随机生成 M 个个体作为初始群体 P(0)。

C. 选择：执行比例选择算子进行选择操作。选择算子的是为了为遗传运算的核心价值建立基础：把优化的个体直接遗传到下一代或通过配对交叉产生新的个体再遗传到下一代。选择操作是建立在群体中个体的适应度评估基础上的。

D. 交叉：按交叉概率对交叉算子执行交叉操作。把两个爸爸代个体的部分结构加以替换重组而生成新个体的操作。遗传算法的核心就是交叉算子。

E. 变异：将概率执行离散变异操作。即是对群体中的个体串的某些基因座上的基因值作变动。群体 P(t)经过选择、交叉、变异运算之后得到下一代群体 P(t 1)

F. 检核：判断是否满足遗传运算的终止进化代数(我们要定义终止代码)，不满足则返回运算，满足则输出运算结果。[1]

3.2 遗传算法的实现

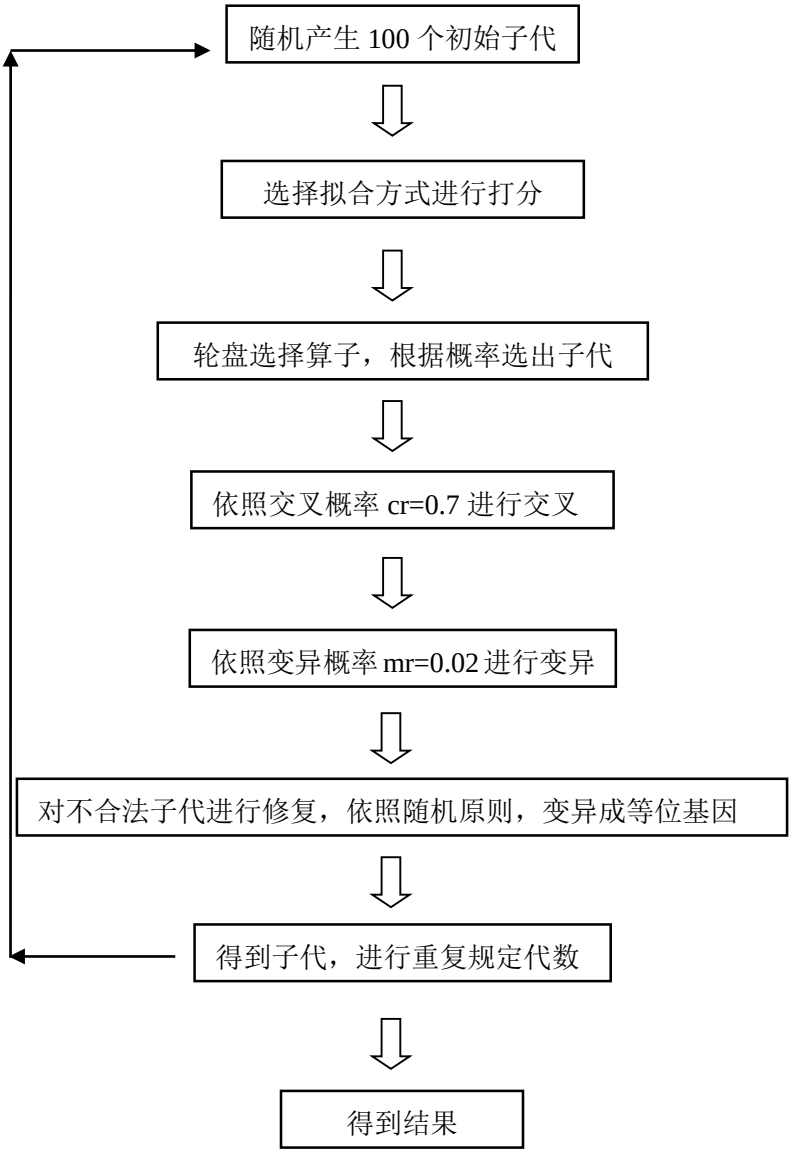

图 3-1 遗传算法流程简图

3.3 影响遗传算法效率或结果的几个因素

3.3.1 起始种群生成方式

对于不同的起始种群生成方式，对之后的遗传选择过程会产生一定的影响。本例采用的是完全随机方式，使用元整函数 round 和随机函数 rand 对初始种群进行处理，生成 100 个完全随机的基因序列作为初始种群进行适应度计算。

3.3.2 适应度计算公式

不同的适应度计算公式对接下来的基因选择、交叉互换、基因变异过程都存在影响。本例采用的是对误差成本取负数次幂 α 的方法计算各代各基因间的适应度。分别取误差成本均值的倒数（-1 次幂）、-3/2 次幂，平方的倒数（-2 次幂）作为适应度函数来计算。

3.3.3 基因选择方式

基因选择步骤是遗传算法的核心部分，不同的基因选择方法影响基因优化的速度，最直接体现在结果收敛速度的快慢。本例采用轮盘赌方式选择下一代基因，适应度越高的子代越有可能留存到下一代，具体步骤为：

1. 根据适应度函数，计算 fsum = Σfitness 和 pi = fi./Σfitness；
2. 用 rand 得到随机数，再计算轮盘定标值 s=round(rand*chromlength；
3. 求满足 Σpi >= s 的最小正整数 i 选中，成为新一代中新的基因
4. 重复 2,3，直到产生新的子代。

3.3.4 交叉互换概率和变异概率

仅依靠基因选择，得到的子代仅仅是起始种群中适应度较高的基因，但可能会漏掉因起始种群随机选择的其他适应度更佳的基因，通过交叉互换和变异过程能产生新的子代基因，并在下一代的循环中根据基因选择的结果判定其好坏。选取合适的交叉概率和变异概率能更快的找到更优秀的子代，本例中，交叉概率取 0.7，变异概率取 0.02，使用随机函数 rand 产生不同大小的 0~1 的小数，通过判断其于交叉概率和变异概率的大小关系而决定是否发生交叉互换或基因变异。

4.不同拟合参数对最终结果的影响

4.1 参数组合 A　交叉概率=0.7，变异概率=0.02，适应度计算公式次幂 α=-1

表4-1 组合A 300代点遗传代数与最高分值

1	11	21	31	41	51	61	71	81	91
209.8454	144.468	132.7505	108.1493	110.3507	112.2964	112.9041	103.0032	105.0885	97.14286
101	111	121	131	141	151	161	171	181	191
98.14179	98.40192	97.15458	96.08422	100.0586	96.08422	96.29638	100.1461	96.08422	102.5672
201	211	221	231	241	251	261	271	281	291
107.049	99.04371	103.1482	96.96055	98.13646	97.7516	94.28145	94.20362	95.92537	94.28145

最终选点结果 第 300 代 Answer = [2 11 21 30 42 50]　Cost = 94.0853

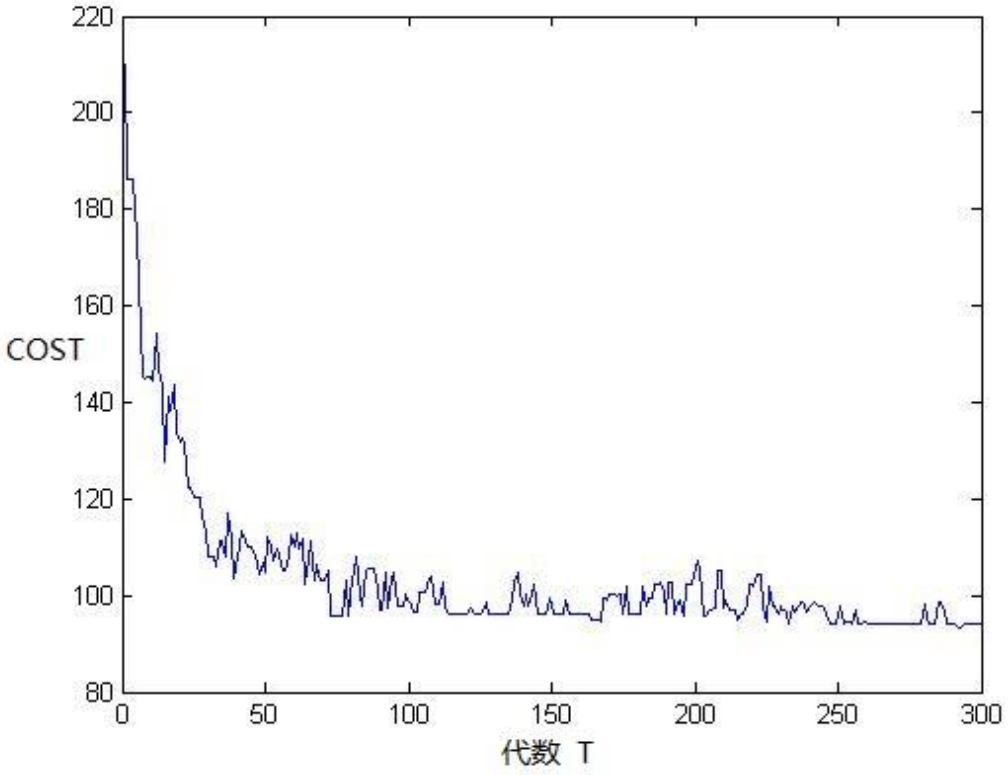

图 4-1 参数组合 A 下 Cost 最大值与代数 T 的关系

4.1.1 结果分析：

用 Matlab 对结果进行分析表明，在大约 100 代的时候误差成本开始收敛，大致收敛在 95~100 区间内。但随后由于交叉互换和变异等情况出现，Cost 值发生波动，直到约 250 代之后开始重新收敛，最终在至 94~95 区间内完成收敛，相信若增大代数 T，可以得到更加稳定收敛的结果。

总体来说，参数组合 A 收敛速度一般，受交叉互换和基因变异的结果影响波动较大，但收敛结果较好，收敛后比较稳定。

4.2 参数组合 B 交叉概率=0.7，变异概率=0.02，适应度计算公式次幂 α=-2

表4-2 组合B 300代点遗传代数与最高分值

1	11	21	31	41	51	61	71	81	91
218.6674	138.4328	113.7655	97.26546	99.70469	97.45522	99.70469	96.99041	96.28571	99.70469
101	111	121	131	141	151	161	171	181	191
99.70469	96.99041	99.70469	99.70469	96.99041	99.70469	96.99041	96.96162	96.99041	96.96162
201	211	221	231	241	251	261	271	281	291
96.96162	96.96162	96.96162	96.32836	96.32836	94.76439	96.96162	96.96162	96.96162	96.96162

最终选点结果第 300 代 Answer = [2 8 21 31 43 49] Cost = 94.83689

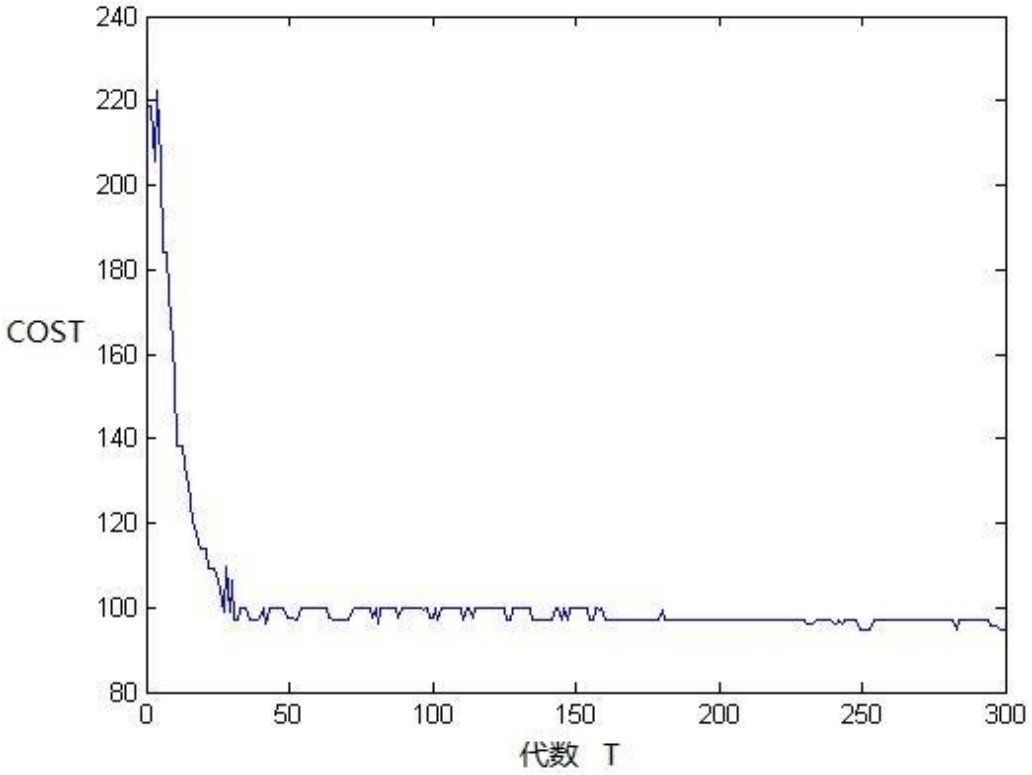

图 4-2 参数组合 B 下 Cost 最大值与代数 T 的关系

4.2.1 结果分析：

从曲线看出，当 α=-2 时，Cost 值收敛非常快速，在 50 代前就收敛到 95-100 区间内，随后在约 150~250 代间 Cost 基本保持在 98，最后在 300 代前产生了更优的子代，Cost 降低到 94-95 区间内。

总体来说，收敛速度较参数组合 A 来说非常快，但是收敛的结果一般，需要更大的遗传代数才能得到同参数组合 A 差不多的结果。

4.3 参数组合 C 交叉概率=0.7，变异概率=0.02，适应度计算公式次幂 α=-3/2

表4-3 组合C 300代点遗传代数与最高分值

1	11	21	31	41	51	61	71	81	91
218.6674	140.8859	114.9488	106.4755	106.6503	105.4136	101.42	100.7591	99.37953	99.37953
101	111	121	131	141	151	161	171	181	191
99.20362	99.02559	100.0288	99.37953	99.64072	101.1642	101.613	99.64072	98.32836	99.23881
201	211	221	231	241	251	261	271	281	291
101.7772	100.5352	99.73987	100.3699	99.73987	98.83689	99.73987	99.51386	99.73987	99.73987

最终选点结果第 300 代 Answer = [2 4 18 26 36 47 50] Cost = 100.10874

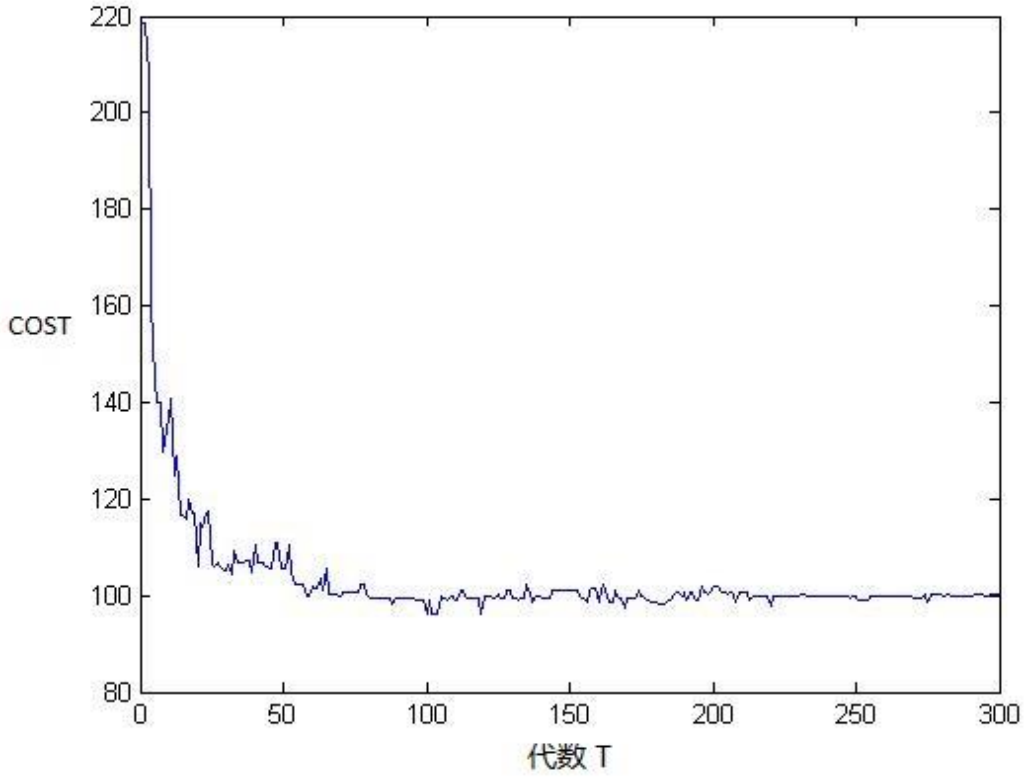

图 4-3 参数组合 C 下 Cost 最大值与代数 T 的关系

4.3.1 结果分析：

从曲线看出，当 α=-3/2 时，Cost 值收敛的速度略慢于参数组合 B 但明显快于参数组合 A，在约 100 代开始收敛于 95-105 区间内，随后约 200~300 代间 Cost 基本收敛于 100，最后选点方式得到的 Cost=100.1087。

总体来说，收敛速度适中，300 代之后得到的最终结果中选了 7 个点，相较于参数组合 A 和 B 的选点个数（6 个点）来说，也比 A 和 B 的最终收敛结果要大。

5.结论和改进意见

5.1 结论

经过不同 α 取值运行 300 代之后取点结果的比较，参数组合 A 和参数组合 B 的最终结果优于参数组合 C，综合考虑了各参数组合的三个方面 A）Cost 值最低 B）收敛速度 C）遗传稳定性，最终选择的取点方案为 X = [2 11 21 30 42 50] ,最终 Cost = 94.0853。

5.2 改进意见

考虑到程序更改参量再得到结果需要的时间较长，我们只对适应度计算公式次幂 α 使用了控制变量法，得到了关于 α 的三组数据并从中选取了最优解。对于 3.3 中提及得到几个因素都可能会影响最后得到的结果，在优化程序运行速度之后，可以对其他参数进行变量调整，综合多个不同参数的最终选点方案之后可能会得到更优的选点方案。

6.课程总结

前期的工作主要是广泛搜集资料、将课件与信息消化，并进行头脑风暴。首先我们先对选点方法进行了认识，并对贯穿整个课程的核心方法—遗传算法，给予了相较深入的探究。

而本课提最终目的便是研讨出最适当的拟合算法，因此我们对各式的拟合算法先行了广泛的认识了解，并列举出了几个我们认为可以适用的算法和算法的关键参数，再行进行参数选择的讨论。希冀之后借着编程技巧的加强以及与老师的讨论，可以让我们的算法更加完备。

后期藉由编程技巧与数据结构算法课程的学习，并且对 Matlab 介面的逐渐孰悉，运算程序逐步步上正轨。借着咨询同班同学、咨询老师，并不对程序运行结果不完美之处进行修正，思考更好的解决方案，最终报告时与老师互动的过程大致顺利，整体报告(小论文)也逐渐成形，算法也渐趋成熟。

7.参考文献

[1] Wikipedia – 遗传算法 [OL]
http://zh.wikipedia.org/zh/%E9%81%97%E4%BC%A0%E7%AE%97%E6%B3%95
[2] Wikipedia – 样条插值 [OL]
http://zh.wikipedia.org/wiki/%E6%A0%B7%E6%9D%A1%E6%8F%92%E5%80%BC
[3]上海交通大学电子工程系 统计推断在数模转换系统中的应用 课程讲义[OL]
ftp://202.120.39.248

8.程序附录

8.1 初始化种群函数 initpop.m

```
function Pop = initpop(popsize,chromlength)
global popsize; %种群大小
global chromlength;%染色体长度
global Pop;%当代种群

Pop = round (rand(popsize,chromlength));
```

8.2 误差成本计算函数 calcost.m

```
%y1 为实测值，y2 为定标后估读数

function cost = calcost(y1,y2)
cost = 0;
for i=1:length(y1)
    if (abs(y1(i)-y2(i)))<=0.5
        cost = cost+0;
    elseif(abs(y1(i)-y2(i)))>0.5 && (abs(y1(i)-y2(i)))<=1
        cost = cost+0.5;
```

```
    elseif(abs(y1(i)-y2(i)))>1 && (abs(y1(i)-y2(i)))<=2
        cost = cost+1.5;
    elseif(abs(y1(i)-y2(i)))>2 && (abs(y1(i)-y2(i)))<=3
        cost = cost+6;
    elseif(abs(y1(i)-y2(i)))>3 && (abs(y1(i)-y2(i)))<=5
        cost = cost+12;
    elseif(abs(y1(i)-y2(i)))>5
        cost = cost+25;
    end
end
```

8.3 适应度函数计算函数 calfitness.m

```
function calfitness(Pop)
global DataX;%存放 X 列数据
global DataY;%存放 Y 列数据
global popsize;
global chromlength;
global sample%需要拟合的样本数据量;
global Fitness;%适应度结果
global Answer;%当代中最佳适应度取点方案

for i=1:popsize
    datasize = 0;
    for j=1:chromlength
        if (Pop(i,j)==1)
            datasize = datasize+1;
        end
    end
    TempCost = zeros(1,sample);
    for k=1:sample
        sr = 1;
        TempDataX = zeros(1,datasize);
        TempDataY = zeros(1,datasize);
        for l=1:chromlength
            if (Pop(i,l)==1)
                TempDataX(sr) = DataX(k,l);
                TempDataY(sr) = DataY(k,l);
                sr=sr+1;
            end
```

```
            end
            FitY = interp1(TempDataX,TempDataY,[5.0:0.1:10.0],'spline');
            TempCost(k) = calcost(FitY,DataY(k,:))+12*(sr-1);
        end
        costotal = sum(TempCost);
        costavg = costotal/sample;
        Fitness(i) = ((costavg)^(-3/2));

end
fitnessmax = max(Fitness);
for g=1:popsize
        if Fitness(g) == fitnessmax
            for h=1:chromlength
                Answer(h) = Pop(g,h);
            end
        end
end
```

8.4 遗传选择实现函数 selection.m

```
function [Pop] = selection(Pop,Fitness)
global popsize;
global chromlength;
global Fitness;
global Pop
global NewPop;%临时存放子代种群

FitnessSum = zeros(1,popsize);
fsum = sum(Fitness);

for i=1:popsize
        tempsum = 0;
        for m=1:i
            tempsum = tempsum + Fitness(m);
        end
        FitnessSum(i) = tempsum;
end

for i=1:popsize
        s = rand()*fsum;
        j = 1;
        while FitnessSum(j)<s;
            j = j+1;
```

```
    end
    for k=1:chromlength
        NewPop(i,k) = Pop(j,k);
    end
end

for m=1:popsize
    for n=1:chromlength
        Pop(m,n) = NewPop(m,n);
    end
end
```

8.5 基因交叉实现函数 crossover.m

```
function crossover(Pop,cr) %cr=CrossRate 为交叉概率
global popsize;
global chromlength;
global Pop;

for i=1:2:popsize
    if(rand<cr)
        cpoint = round(rand*chromlength);
        if or (cpoint == 0, cpoint == 1)
            continue;
        end
        for j = cpoint:chromlength
            temp = Pop(i,j);
            Pop(i,j) = Pop(i+1,j);
            Pop(i+1,j) = temp;
        end
    end
end
```

8.6 基因变异实现函数 mutation.m

```
function [Pop] = mutation(Pop,mr) %mr=MutationRate 为变异概率
global popsize;
global chromlength;
global Pop;

for i=1:popsize
    if((rand/10)<mr)
```

```
            mpoint = round(rand*chromlength);
            if mpoint == 0;
                continue;
            end
            if(Pop(i,mpoint)==1)
                Pop(i,mpoint)=0;
            else Pop(i,mpoint)=1;
            end
        end
end
```

8.7 主函数 main.m

```
global DataX;
global DataY;
global popsize;%种群大小
global chromlength;%基因长度
global NewPop;%子代种群
global Pop;%当代种群
global sample%需要拟合的样本数据量;
global Fitness;%适应度结果
global Answer;%当代中最佳适应度取点方案

c=0.7; %交叉概率
d=0.02;%变异概率
p=100;%种群大小
q=300 ;%进行基因选择代数


Data = dlmread('20141010dataform.csv');
sample = (size(Data,1)/2);
chromlength = size(Data,2);
DataX = zeros(sample,chromlength);
DataY = zeros(sample,chromlength);
for i =1:sample
    DataX(i,:) = Data(2*i-1,:);
    DataY(i,:) = Data(2*i,:);
end

cr = c;
mr = d;
popsize = p;
generation = q;
```

```
Pop = initpop(popsize,chromlength);
Fitness = zeros(1,popsize);
NewPop = zeros(popsize,chromlength);
Answer = zeros(1,chromlength);
BestCost = zeros(1,generation);
BestAnswer = zeros(generation,chromlength);
Final = zeros(1,chromlength);

for G=1:generation
    disp (G);
    calfitness(Pop);
    disp((max(Fitness))^(-2/3));
    BestCost(G) = ((max(Fitness))^(-2/3));
    selection(Pop,Fitness);
    crossover(Pop,cr);
    mutation(Pop,mr);
    disp(Answer);
    for j=1:chromlength
        BestAnswer(G,j) = Answer(j);
    end
end
x = [1:1:generation];
y = BestCost;
plot(x,y);
```