统计推断在数模转换系统中的应用

组号 71　　姓名 侯晨 学号 5140309064, 姓名 李曦 学号 5140309053

摘要：运用Matlab软件对传感器进行校准定标。在分析比较多项式拟合、插值拟合等不同拟合方式得到的样本数据曲线后，选用光滑样条插值拟合样本。通过遗传算法，对于样本曲线的选点进行优化，降低校准定标的总成本。

关键字：校准定标，插值拟合，遗传算法

1 引言

本课题是工业生产中的常遇到的产品定标问题。

在工业生产中，我们往往需要通过传感器测量相关的参数，但是传感器使用种种原理将带测量转换为直接可观测的量的过程中往往要经过一系列的非电信号到电信号转换与机械传动，因此被检测物理量与直接测出量之间绝大数时候并不呈线性，当非线性带来的误差不能被接受的时候就需要重新定标。

然而对于器件一致性差，样本容量大的情况，传统的密集选点法并不能高效地完成校准定标的工作，并且在测量中将付出极大的成本，这就要求我们寻求更为优化的方法完成校准定标的工作。

2 样本拟合

2.1 数据分析

随意选取 4 组数据作出散点图图 1，我们可以看出数据点有以下几个特征：

- Y 取值随 X 取值的增大而单调递增；
- X 取值在[5.0,10.0]区间内，Y 取值在[0,100]区间内；
- 不同个体的特性曲线形态相似但两两相异；
- 特性曲线按斜率变化大致可以区分为首段、中段、尾段三部分，中段的平均斜率小于首段和尾段；
- 首段、中段、尾段单独来看都不是完全线性的，且不同个体的弯曲形态有随机性差异；
- 不同个体的中段起点位置、终点位置有随机性差异。

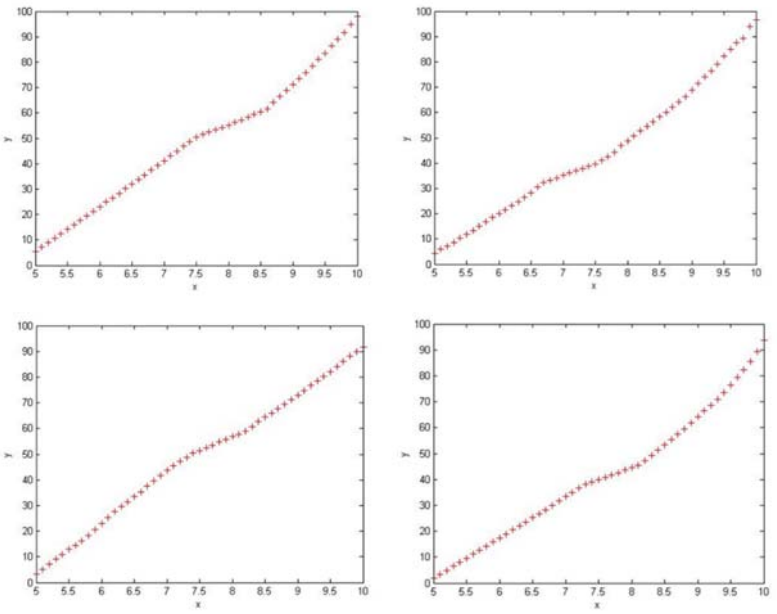
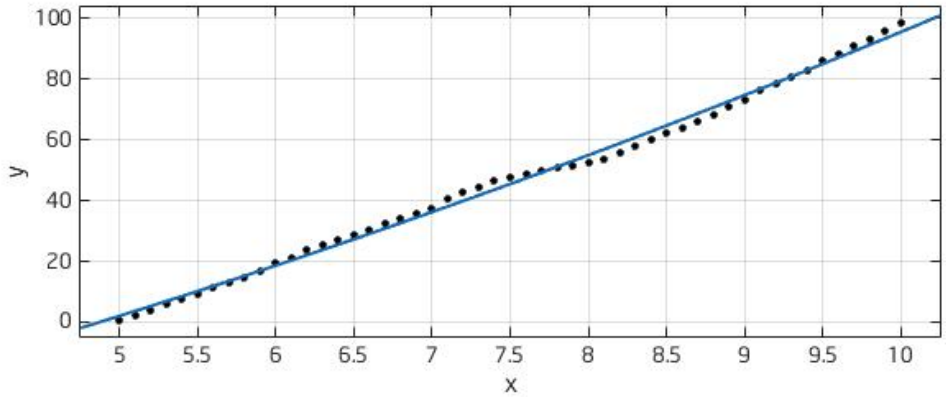


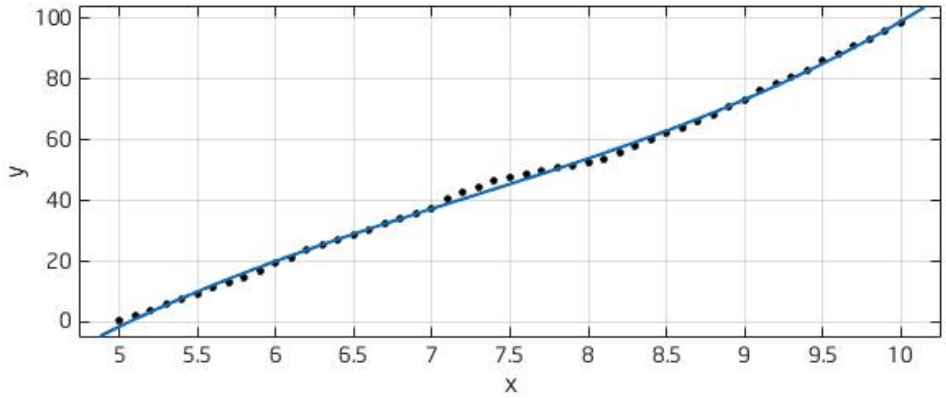
图 1.4 个不同样本的数据散点图

2.2 拟合方法选取

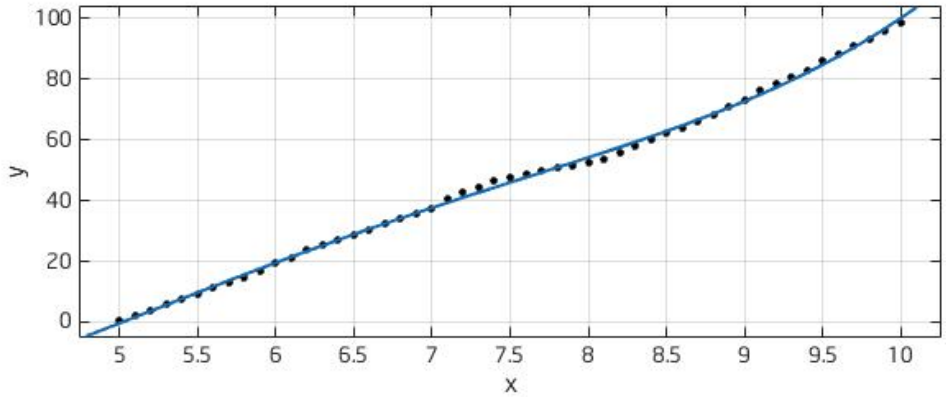
2.2.1 多项式拟合



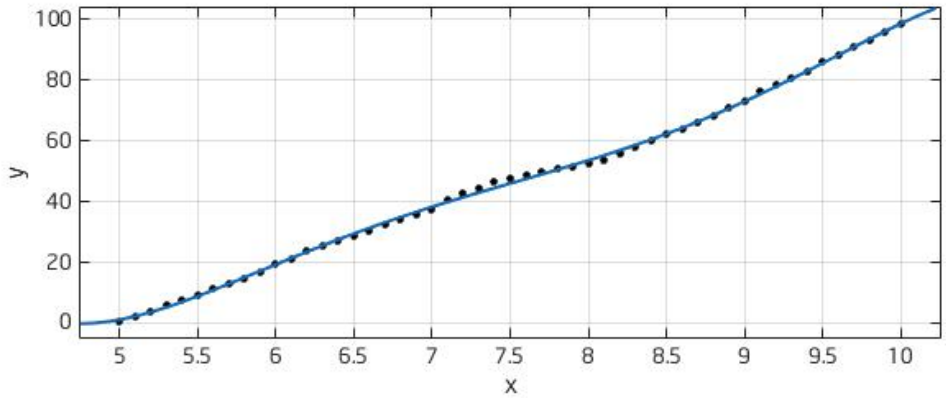
2 次多项式拟合曲线　SSE(和方差) = 168.8818



3 次多项式拟合曲线　SSE(和方差) = 57.0402



4 次多项式拟合曲线　SSE(和方差) = 47.1996

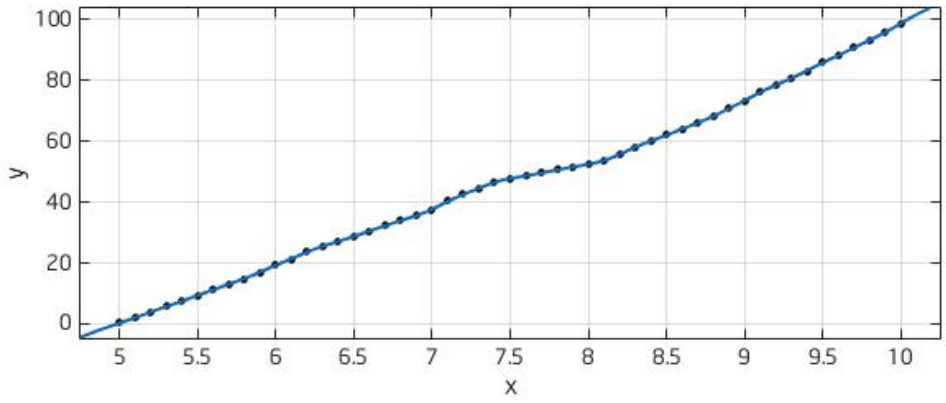


5 次多项式拟合曲线　SSE(和方差) = 28.3419

可以看出，随着多项式阶数的升高，拟合效果在不断变好。

我们再来尝试光滑样条插值（smooth spline）

2.2.2 光滑样条插值



样条插值光滑参数（smooth parameter）p = 0.9998889

SSE(和方差) = 0.6229

可以看出，光滑样条插值的拟合效果远远好于多项式拟合，因此拟合方法我们采用光滑样条插值法，光滑参数 p 设定为 0.999。

3 特征点的选取

3.1 特征点选取及其优化的必要性

由于实验给出的数据共有 51 组，出于时间空间的考虑，我们不可能选取所有的样本进行分析，因此必须从 51 个样本中选取一定数量的特征点进行分析。

特征点的选取依然有优化的必要。假如选取 6 组特征点。那么 C_{51}^{6}的量级则达到了百万，显然穷举法及其耗费时间与空间，这与我们优化的思想相悖。而最好的方法是采用启发式算法对特征点进行选取以达到较好的最终效果和较好的优化效果。

3.2 遗传算法

3.2.1 遗传算法介绍

遗传算法（Genetic Algorithm）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。遗传算法是从

代表问题可能潜在的解集的一个种群(population)开始的,而一个种群则由经过基因(gene)编码的一定数目的个体(individual)组成。每个个体实际上是染色体(chromosome)带有特征的实体。染色体作为遗传物质的主要载体，即多个基因的集合，其内部表现（即基因型）是某种基因组合，它决定了个体的形状的外部表现，如黑头发的特征是由染色体中控制这一特征的某种基因组合决定的。因此，在一开始需要实现从表现型到基因型的映射即编码工作。由于仿照基因编码的工作很复杂，我们往往进行简化，如二进制编码，初代种群产生之后，按照适者生存和优胜劣汰的原理，逐代（generation）演化产生出越来越好的近似解，在每一代，根据问题域中个体的适应度（fitness）大小选择（selection）个体，并借助于自然遗传学的遗传算子（genetic operators）进行组合交叉（crossover）和变异（mutation），产生出代表新的解集的种群。这个过程将导致种群像自然进化一样的后生代种群比前代更加适应于环境，末代种群中的最优个体经过解码（decoding），可以作为问题近似最优解。

3.2.2 算法实现

(1)个体编码

所给数据之中 x 的取值为[5.0, 5.1, ……, 10.0]共 51 个点。将 51 个值编码为 6 位的 2 进制编码（$2^5 = 32$，$2^6 = 64$），因此 6 位可以完全表示 51 个 x 取值的情况。而在交叉时的越界情况在之后讨论。

(2)初始群体生成

通过随机数产生最初的一代群体。种群数量设定为 200，每个个体的选点随机生成。

(3)计算适应度

将该代群体得到的 x，y 的值作为原始数据利用三次样条插值函数 spline()得到 y 关于 x 的函数，由此得到每个样本的定标。

(4)计算选择结果

由每个个体的适应度确定其存活概率，由该概率模拟出被选择的个体，作为交叉产生下一代的亲代。

(5)交叉

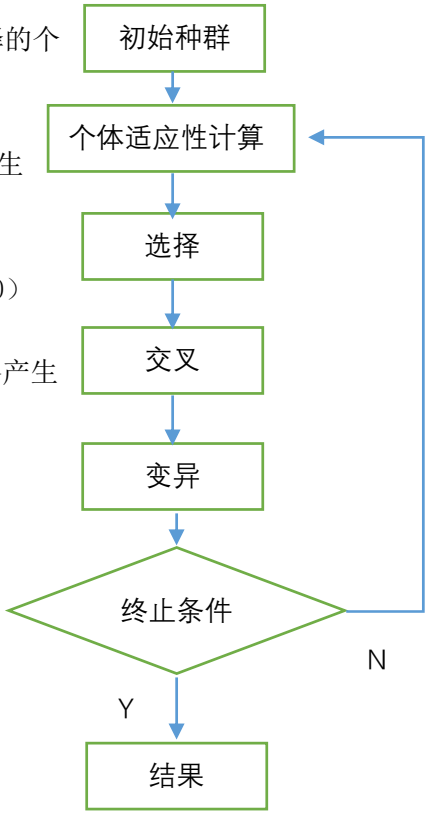
让被选择的结果中个体随机两两配对，随机确定交叉位点，产生子代。

(6)变异

以小概率 0.001 让产生的子代某一位点发生变异（0->1, 1->0）

(7)替换（特殊情况处理）

将交叉变异中出现的越界个体以 1~51 中的随机数替代，最终产生的子代作为下一代生成的亲代。



遗传算法图示

附录

程序代码

1) 主程序

```
%main.m
data = csvread('20150915dataform.csv');
nums = 200;
point = 6;
times = 200;
n = finit(nums, point);
G = 1;

for i = 1 : times
    y = fgety(data, n, nums, point);
    x = fntox(n, nums, point);
    ys = fspline(x, y, nums);
    C = favgcost(data, ys, nums, point);
    index = fmini(C, nums);
    fprint(x, C, index, G, point);
    live = flive(C, n, nums, point);
    cho = fchoose(live, nums);
    eend = fend(cho, n, nums, point);
    b = fntob(eend, nums, point);
    cross = fcross(b, nums, point);
    muta = fmuta(cross, nums, point);
    n = fbton(muta, nums, point);
    n = freplace(n, nums, point);
    G = G + 1;
end
```

2) 种群初始化

```
%finit.m
%create a series of different points chosen
function f=finit(nums, point)
n = zeros(nums, point);
for i = 1 : nums
    for j = 1 : point
        randn = randi(51);
        if j == 1
            n(i, j) = randn;
            continue;
        end
        sameflag = 1;
        while sameflag == 1
```

```
            sameflag = 0;
            for k = 1 : j - 1
                if n(i, k) == randn
                    randn = randi(51);
                    sameflag = 1;
                    break;
                end
            end
        end
        n(i, j) = randn;
    end
end
f = n;
end
```

3) 由初始化所得整数矩阵找到数据表中所给y值

```
%fgety.m
function f=fgety(data, n, nums, point)
y = zeros(nums, point, 400);
for i = 1 : nums
    for  j = 1 : point
        p = n(i, j);
        for k = 1 : 400
            y(i, j, k) = data(2 * k, p);
        end
    end
end
f = y;
end
```

4) 将整数矩阵转化为对应x值的矩阵

```
%fgety.m
function f=fgety(data, n, nums, point)
y = zeros(nums, point, 400);
for i = 1 : nums
    for  j = 1 : point
        p = n(i, j);
        for k = 1 : 400
            y(i, j, k) = data(2 * k, p);
        end
    end
end
f = y;
end
```

5) 根据x与y的数据值利用三次样条插值进行定标

```
%fspline.m
function f=fspline(x, y, nums)
ys = zeros(nums, 400, 51);
xd = zeros(1, 51);
for i = 1 : 51
    xd(i) = (i - 1) * 0.1 + 5.0;
end
for i = 1 : nums
    for j = 1 : 400
        pp = spline(x(i, : ), y(i, : , j));
        t = ppval(pp, xd);
        for k = 1 : 51
            ys(i, j, k) = t(k);
        end
    end
end
f = ys;
end
```

6) 利用题目所给的公式以及之前定标所得的y估计值，数据中y的实际值来计算每个个体的平均成本

```
%favgcost.m
%average cost
function f=favgcost(data, ys, nums, point)
C = zeros(1, nums);
for i = 1 : nums
    ss = 0;
    S = 0;
    for j = 1 : 400
        for k = 1 : 51
            d = abs(data(2 * j, k) - ys(i, j, k));
            if d <= 0.4
                ss = 0;
            end
            if 0.4 < d && d <= 0.6
                ss = 0.1;
            end
            if 0.6 < d && d <= 0.8
                ss = 0.7;
            end
            if 0.8 < d && d <= 1
                ss = 0.9;
```

```
            end
            if 1 < d && d <=2
                ss = 1.5;
            end
            if 2 < d && d <= 3
                ss = 6;
            end
            if 3 < d && d <= 5
                ss = 12;
            end
            if d > 5
                ss = 25;
            end
            S = S + ss;
        end
        S = S + 12 * point;
    end
    C(1, i) = S / 400;
end
f = C;
end
```

7) 返回平均成本最小的个体的下标

```
%fmini.m
function f=fmini(C, nums)
index = 1;
for i = 2 : nums
    if(C(index) > C(i))
        index = i;
    end
end
f = index;
end
```

8) 以一定格式打印出本代成本最小的个体，及其所选点

```
%fprint.m
function fprint(x, C, index, G, point)
fprintf('\nG = %d, minimum average cost is %f', G, C(index));
fprintf('\nPoint choose:');
for i = 1 : point
    fprintf('%.1f, ', x(index, i));
end
end
```

9) 以每个个体的平均成本作为参数计算每个个体的存活概率

```
%flive.m
%clculate the probablity of living
function f=flive(C, n, nums, point)
prob = zeros(1, nums);
live = zeros(1, nums);
for i = 1 : nums
    prob(i) = 10000 / (C(i) ^ 3);
end
%abandon the same choosen-points
for i = 1 : nums
    for j = 1 : i - 1
        flag = 1;
        if prob(i) == prob(j)
            for k = 1 : point
                if n(i, k) ~= n(j, k)
                    flag = 0;
                    break;
                end
            end
            if flag == 1
                prob(i) = 0;
            end
        end
    end
end
sum = 0;
for i = 1 : nums
    sum = sum + prob(i);
end
live(1) = prob(1);
for i = 2 : nums
    live(i) = prob(i) / sum + live(i - 1);
end
f = live;
end
```

10) 由每个个体的存活概率来计算出本代被选中的个体

```
%fchoose.m
function f=fchoose(live, nums)
cho = zeros(1, nums);
for i = 1 : nums
    c = randi(nums) / nums;
    for j = 1 : nums
```

```
        if c <= live(j);
            cho(i) = j;
            break;
        end
        if j == nums
            cho(i) = j;
        end
    end
end
f = cho;
end
```

11)利用上一个函数得出的结果整理出选择后的种群

```
%fend.m
%get an array of the num/point being chosen
function f=fend(cho, n, nums, point)
eend = zeros(nums, point);
for i = 1 : nums
    for j = 1 : point
        eend(i, j) = n(cho(i), j);
    end
end
f = eend;
end
```

12)交叉产生子代

```
%fcross.m

function f=fcross(b, nums, point)
cross = zeros(nums, point, 6);
rd = randperm(nums);
for i = 1 : 2 : nums
    for j = 1 : point
        r = randi(6);
        for k = 1 : r
            cross(i, j, k) = b(rd(i), j, k);
            cross(i + 1, j, k) = b(rd(i + 1), j, k);
        end
        for k = r + 1 : 6
            cross(i, j, k) = b(rd(i + 1), j, k);
            cross(i + 1, j, k) = b(rd(i), j, k);
        end
    end
end
```

```
f = cross;
end
```

13)让子代以一定的概率产生变异

```
%fmuta.m
function f=fmuta(cross, nums, point)
for i = 1 : nums
    for j = 1 : point
        prob = randi(1000);
        if prob == 1
            p = randi(6);
            if cross(i, j, p) == 1
                cross(i, j, p) = 0;
            else
                cross(i, j, p) = 1;
            end
        end
    end
end
f = cross;
end
```

14)将交叉或变异中产生的越界个体以及重复个体用1~51的随机数替代

```
%freplace.m
%delete the number over limit
function f=freplace(n, nums, point)
for i = 1 : nums
    for j = 1 : point
        if n(i, j) > 51 || n(i, j) < 1
            n(i, j) = randi(51);
        end
    end
end
for i = 1 : nums
        sameflag = 1;
        while sameflag == 1
            sameflag = 0;
            for j = 2 : point
                for k = 1 : j - 1
                    if n(i, j) == n(i, k)
                        sameflag = 1;
                        n(i, k) = randi(51);
                        continue;
                    end
```

```
            end
        end
    end
f = n;
end
```

15)将二进制矩阵转化回整数矩阵

```
%fbton.m
%binary = number
function f=fbton(b, nums, point)
n = zeros(nums, point);
for i = 1 : nums
    for j = 1 : point
        number = 0;
        for k = 6 : -1 : 1
            number = number * 2 + b(i, j , k);
        end
        n(i, j) = number;
    end
end
f = n;
end
```

16)将整数矩阵编码成对应的二进制矩阵

```
%fntob.m
%number to binary
function f=fntob(n, nums, point)
b = zeros(nums, point, 6);
for i = 1 : nums
    for j = 1 : point
        number = n(i, j);
        for k = 6 : -1 : 1
            b(i, j, k) = mod(number, 2);
            number = floor(number / 2);
        end
    end
end
f = b;
end
```