数学实验 · 实验报告5

计算机系　计01　容逸朗　2020010869

1　7-5 原子位置

1.1　问题分析

题目给出了 52 对原子相互之间的距离，要求求出每个原子的位置关系，为此可以利用勾股定理列出对应的方程，又注意到方程的个数比未知数多，因此需要利用无约束优化并求解。

1.2　模型假设与建立

可以假设第 i 个原子 m_i 的位置是 (x_i, y_i) ，那么原子及其距离组成的三元组 (m_{a_i}, m_{b_i}, d_i) 对应等式为：

$$(x_{a_i} - x_{b_i})^2 + (y_{a_i} - y_{b_i})^2 = d_i^2, \quad i = 1, 2, \dots, 52 \tag{1}$$

显然，由 (1) 组成的方程组共有 52 条方程，超过未知数的个数： $25 \times 2 = 50$ ，因此这个方程组是超定方程组，为此可以利用无约束优化来解决问题，也就是我们只需要求：

$$\underset{x_1, y_1, \dots, x_{25}, y_{25}}{argmin} \sum_{i=1}^{52} ((x_{a_i} - x_{b_i})^2 + (y_{a_i} - y_{b_i})^2 - d_i^2)^2 \tag{2}$$

不失一般性，可以假设 $(x_{25}, y_{25}) = (0, 0)$ ，这样可以限制解生成的位置而不影响结果。（这时变量个数变为 48 ）

1.3　算法设计

可以利用 matlab 的 `fminunc` 函数解无约束优化问题，同时利用优化工具箱的 bfgs, dfp 和最速下降法做搜索，便可得到较为理想的结果。

1.4　程序

```matlab
format short g;

% 数据
a = [4, 1, 0.9607; 5, 4, 0.4758; 18, 8, 0.8363; 15, 13, 0.5725;
     12, 1, 0.4399; 12, 4, 1.3402; 13, 9, 0.3208; 19, 13, 0.7660;
     13, 1, 0.8143; 24, 4, 0.7006; 15, 9, 0.1574; 15, 14, 0.4394;
     17, 1, 1.3765; 8, 6, 0.4945; 22, 9, 1.2736; 16, 14, 1.0952;
     21, 1, 1.2722; 13, 6, 1.0559; 11, 10, 0.5781; 20, 16, 1.0422;
     5, 2, 0.5294; 19, 6, 0.6810; 13, 10, 0.9254; 23, 16, 1.8255;
     16, 2, 0.6144; 25, 6, 0.3587; 19, 10, 0.6401; 18, 17, 1.4325;
     17, 2, 0.3766; 8, 7, 0.3351; 20, 10, 0.2467; 19, 17, 1.0851;
     25, 2, 0.6893; 14, 7, 0.2878; 22, 10, 0.4727; 20, 19, 0.4995;
     5, 3, 0.9488; 16, 7, 1.1346; 18, 11, 1.3840; 23, 19, 1.2277;
     20, 3, 0.8000; 20, 7, 0.3870; 25, 11, 0.4366; 24, 19, 1.1271;
     21, 3, 1.1090; 21, 7, 0.7511; 15, 12, 1.0307; 23, 21, 0.7060;
```

```
     24, 3, 1.1432; 14, 8, 0.4439; 17, 12, 1.3904; 23, 22, 0.8052];

% 初始值
mi = 1;
s0 = zeros(48, 1);

for i = 1: 100
    % 生成 [-1, 1] 的数
    x0 = 2 * rand(48, 1) - 1;
    % 优化项
    opt = optimset('HessUpdate', 'bfgs', 'MaxFunEvals', 1000000, 'MaxIter', 50000);
    % 求解
    [x, fv, ef, out] = fminunc(@(x)chem(x, a), x0, opt);

    % 比当前最小值更优
    if fv < mi
        mi = fv;
        s0 = x;
    end
end

% 按点的方法输出结果
s = zeros(25, 3);
for i = 1: 24
    s(i, 1) = i;
    s(i, 2) = s0(2 * i - 1);
    s(i, 3) = s0(2 * i);
end
s(25, 1) = 25;
s
mi

% 画图
hold on
plot(0, 0, 'bo')
for i = 1: 24
    plot(x(2 * i - 1), x(2 * i), 'bo')
end
title("原子位置图")
xlabel("x")
ylabel("y")

function y = chem(x, a)
    y = 0;
```

```
    for i = 1: length(a)
        % 取出原子1，原子2及其距离
        m1 = a(i, 1);
        m2 = a(i, 2);
        dis = a(i, 3);

        % 确定原子1的位置
        x1 = 0;
        y1 = 0;
        if m1 ~= 25
            x1 = x(2 * m1 - 1);
            y1 = x(2 * m1);
        end

        % 确定原子2的位置
        x2 = 0;
        y2 = 0;
        if m2 ~= 25
            x2 = x(2 * m2 - 1);
            y2 = x(2 * m2);
        end

        y = y + ((x1 - x2) ^ 2 + (y1 - y2) ^ 2 - dis ^ 2) ^ 2;
    end
end
```

1.5 程序结果

若选取所有点的初始位置为 $(0,0)$ ，那么算法不收敛，无解。因此取所有点的初始位置为 $(1,1)$ ，此时算法收敛。采用不同的搜索方法有如下结果：

搜索方法	迭代次数	函数调用次数	函数值	附注
BFGS	172	9604	0.0489	
DFP	1194	58947	0.0478	①
最速下降法	4943	726670	4.0074	①

表1 搜索算法对比

① 由于迭代步长小于预设值，算法提前终止

由此可知，BFGS 在此问题下的性能较 DFP 和最速下降法更优，因此可以针对 BFGS 方法做进一步的讨论，这里选择初始值为 $[-1,1]$ 之间的随机数，在随机 100 次的情况下得到 $f_{min} = 0.010882$ ，此时原子位置如下图示：

从上面的结果可知，BFGS 在此问题下的性能较 DFP 和最速下降法更优（迭代次数、函数调用次数更少），对于本题而言，数据的初值对结果的影响相当巨大，极端情况下会导致算法不收敛。为了克服数据初值造成的影响，可以多次随机取初值，这样可以得到相对更优的结果。

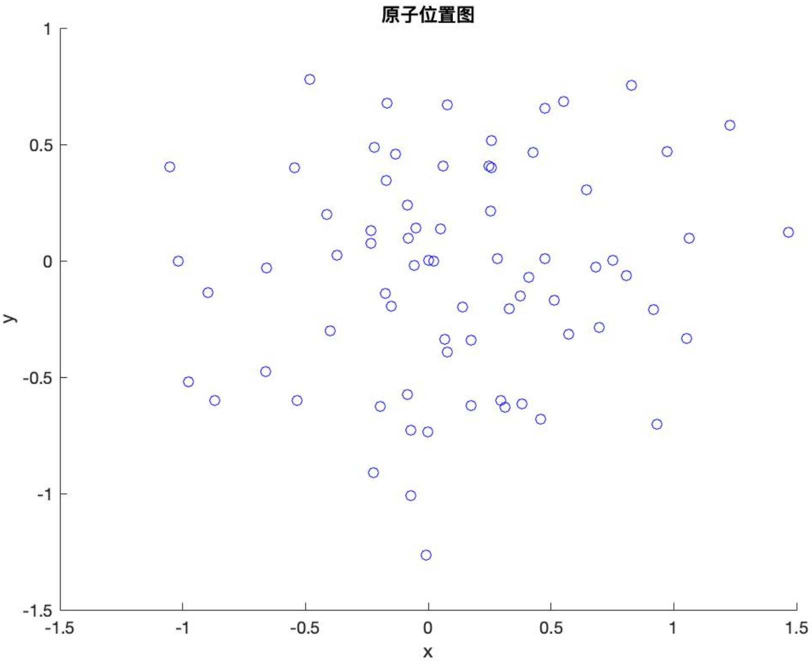

图1　原子在平面中的位置

每个原子的具体位置如下：

ID	x	y	ID	x	y	ID	x	y
1	0.040771	−0.79556	10	−0.15032	0.20641	19	−0.080436	−0.44505
2	−0.69042	−0.073031	11	0.42061	0.14911	20	−0.029684	0.024804
3	0.00089614	0.82034	12	−0.27307	−1.1048	21	0.88645	0.15361
4	0.20394	0.14979	13	0.6589	−0.25901	22	−0.3916	0.65705
5	−0.17166	−0.11516	14	0.28377	−0.54697	23	0.41551	0.68007
6	−0.29445	0.19501	15	0.12279	−0.15688	24	0.8999	0.11286
7	0.25017	−0.25099	16	−0.80244	−0.67823	25	0.0000	0.0000
8	−0.076976	−0.26367	17	−1.0447	0.051837			
9	0.3419	−0.3762	18	−0.18585	−1.0944			

表2　原子位置具体值

1.6　结果分析

从上面的结果可知，BFGS 在此问题下的性能较 DFP 和最速下降法更优（迭代次数、函数调用次数更少），对于本题而言，数据的初值对结果的影响相当巨大，极端情况下会导致算法不收敛。为了克服数据初值造成的影响，可以多次随机取初值，这样可以得到相对更优的结果。

1.7 结论

利用 BFGS 得到的值可见 1.5 节表 1 。

2 7-8 给药方案

2.1 问题分析

根据题干给出的模型做无约束优化，即可得到结果。

2.2 模型假设与建立

题干已经给出如下模型：

$$c(t) = \frac{d}{V}\frac{k_1}{k_1-k}(e^{-kt}-e^{-k_1 t}) = b\frac{k_1}{k_1-k}(e^{-kt}-e^{-k_1 t}) \tag{3}$$

这里 d 表示口服剂量，V 表示中心室容积，t 表示时间，$c(t)$ 表示 t 时刻下中心室血药浓度，同时方程还有 3 个未知参数 $k_1, k, b = \frac{d}{V}$ 。

为了计算最优解，我们可以将方程转化为一个无约束最优问题，也就是：

$$\underset{k_1,k,b}{argmin}\sum_{i=1}^{14}\left(c(t_i) - b\frac{k_1}{k_1-k}(e^{-kt_i}-e^{-k_1 t_i})\right)^2 \tag{4}$$

2.3 算法设计

可以利用 matlab 的 `fminunc` 函数解无约束优化问题，同时利用优化工具箱的 bfgs, dfp 和最速下降法做搜索，便可得到较为理想的结果。

2.4 程序

```
format short

a = [0.083, 10.9; 0.167, 21.1; 0.25, 27.3;
     0.50, 36.4; 0.75, 35.5; 1.0, 38.4;
     1.5, 34.8; 2.25, 24.2; 3.0, 23.6;
     4.0, 15.7; 6.0, 8.2; 8.0, 8.3;
     10.0, 2.2; 12.0, 1.8];

x0 = [1 2 3];
opt = optimset('HessUpdate', 'bfgs', 'MaxFunEvals', 1000000, 'MaxIter', 50000);
% 求解
[x, fv, ef, out] = fminunc(@(x)drug(x, a), x0, opt);

function y = drug(x, a)
     y = 0;
     b = x(1);
```

```
    k = x(2);
    k1 = x(3);
    for i = 1: length(a)
        t = a(i, 1);
        ct = a(i, 2);
        y = y + (ct - b * k1 * (exp(-k * t) - exp(-k1 * t))./(k1 - k))^ 2;
    end
end
```

2.5 计算结果

通过上面的程序可以得到如下结果：

$$b = 46.8275, \quad k = 0.2803, \quad k_1 = 3.6212, \quad f_{min} = 34.2317$$

2.6 结果分析

利用上面的数据生成对应的函数，其图像为：（红色星星为题给数据）

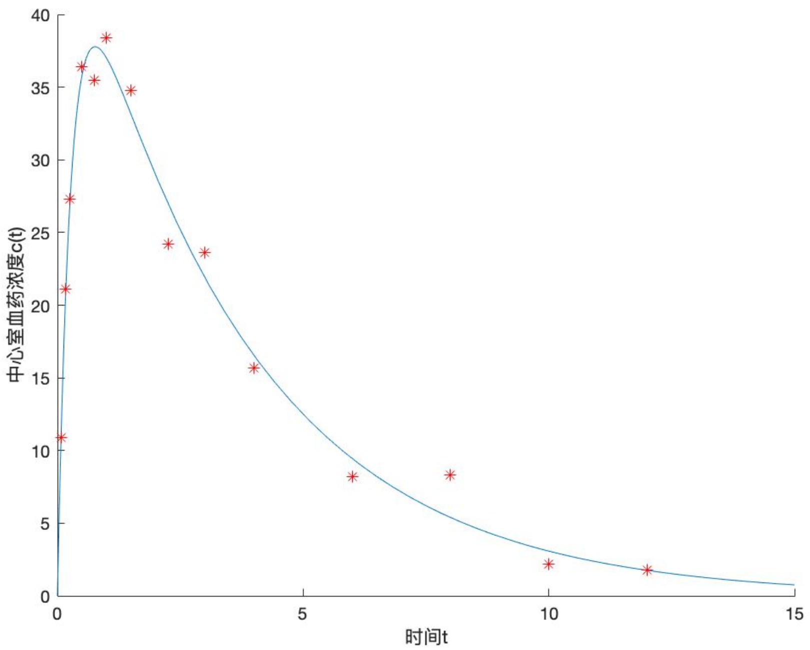

图2　时间–中心室血药浓度

可以看到，在患者用药的短时间内，血药浓度急速上升至峰值，然后浓度缓慢下降，这与实际理论相符。

2.7 结论

参数为 $b = 46.8275, k = 0.2803, k_1 = 3.6212$ 。

3 8-6 投资

3.1 问题分析

利用数学方法建立投资模型，然后利用线性规划解出结果即可。

3.2 模型假设与建立

假设证券 i 的信用等级为 l_i，到期年限为 t_i，到期税前收益为 p_i ，买入 x_i 万元的证券。除此之外，还需要：

1. 若其种类为政府或代办机构，则参数 $b_i = 1$ ，否则 $b_i = 0$ ；
2. 若其种类为市政，则课税参数 $c_i = 1$ ，否则 $c_i = 0.5$ ；

按照上述假设，总收益为：

$$z = \sum_i p_i \cdot c_i \cdot x_i \tag{5}$$

约束条件为：

1. 政府及代办机构的证券至少购入 A_1 万元：

$$\sum_i b_i \cdot x_i \geq A_1 \tag{6}$$

   变型可得：

$$-\sum_i b_i \cdot x_i \leq -A_1 \tag{7}$$

2. 所购证券平均信用不超过 A_2：

$$\frac{\sum_i l_i \cdot x_i}{\sum_i x_i} \leq A_2 \tag{8}$$

   变型可得：

$$\sum_i (l_i - A_2) \cdot x_i \leq 0 \tag{9}$$

3. 所购证券平均到期年限不超过 A_3：

$$\frac{\sum_i t_i \cdot x_i}{\sum_i x_i} \leq A_3 \tag{10}$$

   变型可得：

$$\sum_i (t_i - A_3) \cdot x_i \leq 0 \tag{11}$$

3.2.1 实例

对于问题 $8-6$ ，有数据如下：

i	c_i	l_i	t_i	p_i
1	1	2	9	4.3
2	0.5	2	15	5.4
3	0.5	1	4	5.0
4	0.5	1	3	4.4
5	1	5	2	4.5

表3　题干数据

除此以外，还有 $A_1 = 400, A_2 = 1.4, A_3 = 5$ 。

1. 对于第一小问，需要增加一个约束条件：

$$\sum_{i=1}^{5} x_i \le 1000 \tag{12}$$

2. 对于第二小问，需要增加一个约束条件并更改总收益方程：

$$\begin{aligned} k &\le 100 \\ z' &= z - 0.0275k \end{aligned} \tag{13}$$

3. 对于第三小问，在 1 的基础上更改：

$$\begin{aligned} &(1) \quad p_1 = 4.5\% \\ &(2) \quad p_3 = 4.8\% \end{aligned}$$

3.3　算法设计

直接利用 matlab 的 `linprog` 方法计算即可。

3.4　程序

3.4.1　第一、三小题

```
format short

% 模型参数
a1 = 400;
a2 = 1.4;
a3 = 5;
r1 = [0, -1, -1, -1, 0];
r2 = [2, 2, 1, 1, 5] - a2;
r3 = [9, 15, 4, 3, 2] - a3;
r4 = [1, 1, 1, 1, 1];

% 对应不等式
A = [r1; r2; r3; r4];
b = [-a1, 0, 0, 1000];

```

```
% 目标函数
c = -[4.3, 5.4 * 0.5, 5.0 * 0.5, 4.4 * 0.5, 4.5] * 0.01;

% 初值与解
v1 = zeros(5, 1);
opt = optimoptions(@linprog, 'Algorithm', 'interior-point');
[x, fv, ef, out, lag] = linprog(c, A, b, [], [], v1, [], [], opt);

% 结果
lag.ineqlin
lag.lower
lag.upper
x
-fv
```

3.4.2 第二小题

```
format short

% 模型参数
a1 = 400;
a2 = 1.4;
a3 = 5;
r1 = [0, -1, -1, -1, 0, 0];
r2 = [2, 2, 1, 1, 5, a2] - a2;
r3 = [9, 15, 4, 3, 2, a3] - a3;
r4 = [1, 1, 1, 1, 1, -1];
r5 = [0, 0, 0, 0, 0, 1];

% 对应不等式
A = [r1; r2; r3; r4; r5];
b = [-a1, 0, 0, 1000, 100];

% 目标函数
c = -[[4.3, 5.4 * 0.5, 5.0 * 0.5, 4.4 * 0.5, 4.5] * 0.01, -0.0275];

% 初值与解
v1 = zeros(5, 1);
opt = optimoptions(@linprog, 'Algorithm', 'interior-point');
[x, fv, ef, out, lag] = linprog(c, A, b, [], [], v1, [], [], opt);

% 结果
lag.ineqlin
lag.lower
lag.upper
```

```
x
-fv
```

3.5 计算结果与分析

3.5.1 第 1 问

1. 调用 `lag.ineqlin`，得到不等式约束的拉格朗日乘子，其值分别为：$0, 0.0062, 0.0024, 0.0298$，这说明在第 1 个约束对结果不起作用；
2. 再调用 `x` 和 `lag.lower`，得到对应最优取值和下界约束的拉格朗日乘子，分别为：

变量	最优取值	下界约束条件
x_1	218.18	0
x_2	0	0.0302
x_3	736.36	0
x_4	0	0.0006
x_5	45.45	0

- 此时的最大收益为 29.8364 万元。

3.5.2 第 2 问

1. 调用 `lag.ineqlin`，得到不等式约束的拉格朗日乘子，其值分别为：$0, 0.0062, 0.0024, 0.0298, 0.0023$，这说明在第 1 个约束对结果不起作用；
2. 再调用 `x` 和 `lag.lower`，得到对应最优取值和下界约束的拉格朗日乘子，分别为：

变量	最优取值	下界约束条件
x_1	240.00	0
x_2	0	0.0302
x_3	810.00	0
x_4	0	0.0006
x_5	50.00	0
k	100.00	0

- 此时的最大收益为 30.07 万元。

3.5.3 第 3 问 - 更改 A 的税前收益为 4.5%

1. 调用 `lag.ineqlin`，得到不等式约束的拉格朗日乘子，其值分别为：$0, 0.0064, 0.0027, 0.0303$，这说明在第 1 个约束对结果不起作用；
2. 再调用 `x` 和 `lag.lower`，得到对应最优取值和下界约束的拉格朗日乘子，分别为：

变量	最优取值	下界约束条件
x_1	218.18	0
x_2	0	0.0344
x_3	736.36	0
x_4	0	0.0003
x_5	45.45	0

- 由此可知无需改变投资策略，此时的最大收益为 30.2727 万元。

3.5.4 第 3 问 - 更改 C 的税前收益为 4.8%

1. 调用 `lag.ineqlin` ，得到不等式约束的拉格朗日乘子，其值分别为：$0, 0.0064, 0.0024, 0.0294$ ，这说明在第 1 个约束对结果不起作用；
2. 再调用 `x` 和 `lag.lower`，得到对应最优取值和下界约束的拉格朗日乘子，分别为：

变量	最优取值	下界约束条件
x_1	336.00	0
x_2	0	0.0306
x_3	0	0.0004
x_4	648.00	0
x_5	16.00	0

- 由此知需要改变投资策略，从证券 ACE 转为购买证券 ADE ，此时的最大收益为 29.4240 万元。

3.6 结论

1. 若经理有 1000 万资金，应分别购入 218.18 万元，736.36 万元，45.45 万元的证券 A, C, E ，此时的最大收益为 29.8364 万元；
2. 若可借款 100 万元，则应借款 100 万，此时分别购入 240.00 万元，810.00 万元，50.00 万元的证券 A, C, E ，此时的最大收益为 30.07 万元；
3. 若证券 A 的税前收益为 4.5%，则无需改变投资策略；若证券 C 的税前收益为 4.8%，则需要改变投资策略为购入 336.00 万元，648.00 万元，16.00 万元的证券 A, D, E 。