数学实验第四次实验报告

计算机系　计 43　2014011330　黄家晖

2017 年 4 月 14 日

1　实验目的

- 掌握 MATLAB 优化工具箱的基本使用方法，对不同算法进行分析和比较；
- 练习使用无约束方法建立和求解实际问题模型。

2　计算题

2.1　CH7-T6 非线性最小二乘拟合问题

2.1.1　算法设计

由于题目中所给的函数 $f(x,t)$ 为非线性函数，因此使用 MATLAB 中的 `lsqcurvefit`函数进行求解。针对 GN 和 LM 两种求解算法，可以通过 `LevenbergMarquardt`的 `on`和 `off`参数进行控制。

2.1.2　MATLAB 程序

注意：运行下面的代码需要 MATLAB 2011 及以下的版本，否则不包含 GN 解法。

```
%% Math Exp Homework 4 7-T6
% Non linear least square curve fit.

%% Data definition
y = [0.844, 0.908, 0.932, 0.936, 0.925, 0.908, 0.881, 0.850, 0.818, 0.784, ...
    0.751, 0.718, 0.685, 0.658, 0.628, 0.603, 0.580, 0.558, 0.538, 0.522, ...
    0.506, 0.490, 0.478, 0.467, 0.457, 0.448, 0.438, 0.431, 0.424, 0.420, ...
    0.414, 0.411, 0.406];
t = 10 * ((1:33) - 1);

%% Solve
x0 = [0.5, 1.5, -1, 0.01, 0.02];
fopt = optimset('levenberg-marquardt', 'off', 'MaxIter', 10000, ...
    'MaxFunEvals', 1000000);
[x, norm, res, flag, out] = lsqcurvefit(@fitfun, x0, t, y, [], [], fopt);

%% Draw
scale = 0:340;
yval = fitfun(x, scale);
figure;
hold on;
plot(scale, yval);
scatter(t, y, 'x');
```

```
legend('Fitted Curve', 'Dataset');

%% Random Test.
iters = [];
for i = 1:500
    rand_noise = 0.1 * rand(1, length(y)) - 0.05;
    y_new = y + rand_noise;
    [x, norm, res, flag, out] = lsqcurvefit(@fitfun, x0, t, y_new, [], [], fopt);
    iters = [iters; out.iterations];
end
ecdf(iters);
```

其中用到的函数 `fitfun`如下：

```
function y = fitfun( x, t )
    y = x(1) + x(2) * exp(-x(4) * t) + x(3) * exp(-x(5) * t);
end
```

2.1.3 计算结果与分析

使用 GN 方法求解的结果如下，总共迭代次数为 9.

$$x_1 = 0.3754, x_2 = 1.9358, x_3 = -1.4647, x_4 = 0.0129, x_5 = 0.0221$$

使用 LM 方法求解的结果如下，总共迭代次数为 35.

$$x_1 = 0.3754, x_2 = 1.9358, x_3 = -1.4647, x_4 = 0.0129, x_5 = 0.0221$$

对比二者，可以看出求解出的参数差别很小；就迭代次数上来看，GN 方法使用的迭代次数更小，此时 GN 算法更优。由于二者的参数很接近，所以将上述参数两种方法的曲线和数据点都画在一张图 1中，检测拟合的准确性。

下面针对 LM 方法，观察 y_i 震荡时，迭代收敛的次数是否会变化。重复 500 次，划出迭代次数的 CDF 如图 2所示。

根据图中可以看出，LM 算法迭代次数大多数在 50 轮以下（50% 左右），而最高迭代次数会达到 300 轮左右，能够看出 LM 算法会在数据产生微小变动的时候产生迭代次数的变化，这组数据有一定的病态程度。

2.2 CH7-T7 经济学中的生产函数曲线

2.2.1 算法设计

为了使用线性最小二乘拟合，首先要将待拟合的曲线进行线性化，对于式：

$$Q(K, L) = aK^{\alpha}L^{\beta}$$

两边取对数，得到：

$$\log Q = \alpha \log K + \beta \log L + \log a$$

与拟合函数进行比较，取基函数 $\phi_0 = \log K, \phi_1 = \log L, \phi_2 = 1$，构造：

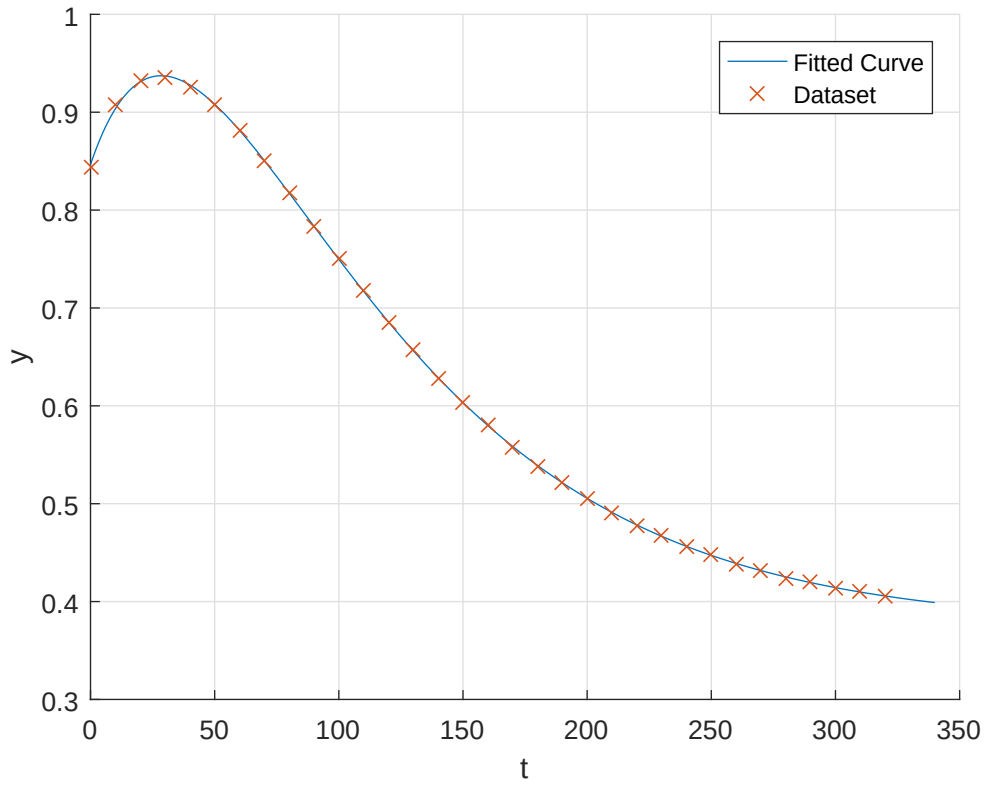

图 1: 函数拟合结果，由于两种方法结果很接近，因此合并成一条曲线

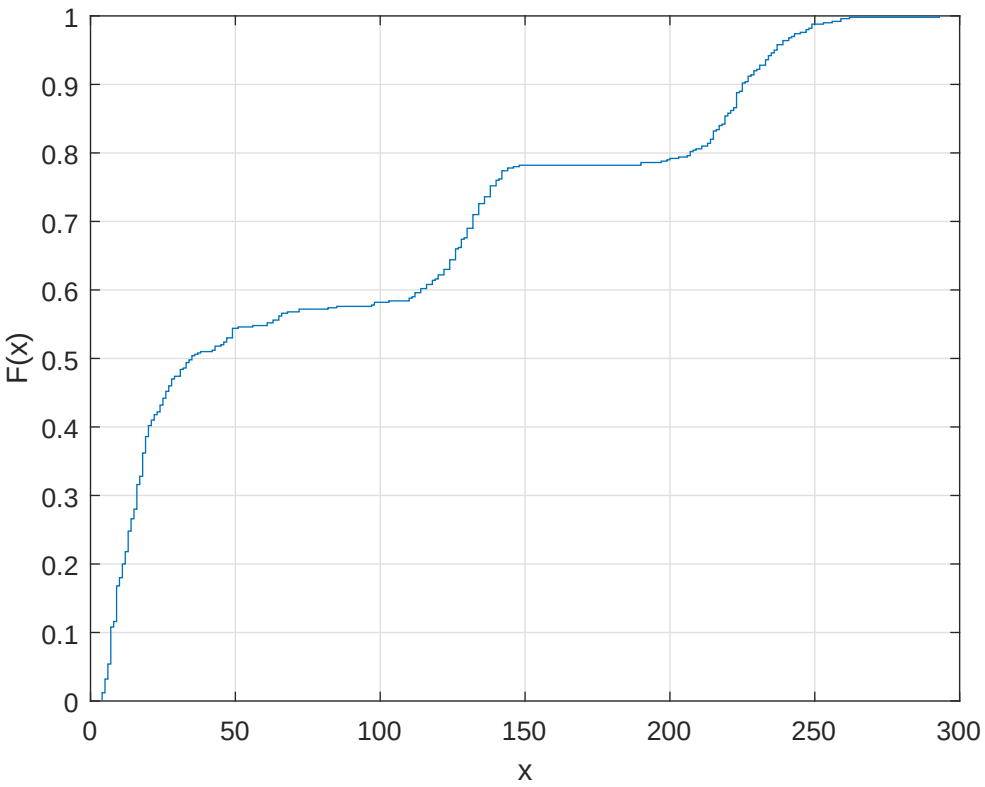

图 2: LM 算法迭代次数的累积分布函数

$$\Phi = \begin{bmatrix} \log K_1 & \log L_1 & 1 \\ \log K_2 & \log L_2 & 1 \\ \vdots & \vdots & \vdots \\ \log K_n & \log L_n & 1 \end{bmatrix} \tag{1}$$

$$\mathbf{b} = (\alpha, \beta, \log a)^T$$

$$\mathbf{y} = (\log Q_1, \log Q_2, \cdots, \log Q_n)^T$$

即求解：

$$\Phi \mathbf{b} = \mathbf{y}$$

对于上述超定方程组，在 MATLAB 中直接使用左除操作符，就可以得到最小二乘解。

而对于非线性最小二乘拟合，则通过 MATLAB 中自带的 `lsqcurvefit`函数进行优化求解，初始值的选取为 $\alpha = 0.5, \beta = 0.5, a = 0.5$。

2.2.2 MATLAB 程序

```
%% Math Exp Homework 4 7-T7
% Fit a Cobb-Douglas curve

%% Input data
Q = [0.7171, 0.8964, 1.0202, 1.1962, 1.4928, 1.6909, 1.8548, 2.1618, ...
    2.6638, 3.4634, 4.6759, 5.8478, 6.7885, 7.4463, 7.8345, 8.2068, ...
    9.9468, 9.7315, 10.4791]';
K = [0.0910, 0.2543, 0.3121, 0.3792, 0.4754, 0.4410, 0.4517, 0.5595, ...
    0.8080, 1.3072, 1.7042, 2.0019, 2.2914, 2.4941, 2.8406, 2.9854, ...
    3.2918, 3.7314, 4.3500]';
L = [4.8179, 4.9873, 5.1282, 5.2783, 5.4334, 5.5329, 6.4749, 6.5491, ...
    6.6152, 6.6808, 6.7455, 6.8065, 6.8950, 6.9820, 7.0637, 7.1394, ...
    7.2085, 7.3025, 7.3740]';

%% Use linear least square fitting
% log(Q) = alpha * log(K) + beta * log(L) + log(a)
phi = [log(K), log(L), ones(19, 1)];
y = log(Q);
beta = phi \ y;
[beta, norm] = lsqlin(phi, y, [], [], [], [], [-100, 0, 0], [100, 1 1]);
fprintf('Linear: Alpha = %f, Beta = %f, a = %f\n', beta(1), beta(2), exp(beta(3)));

%% Use Non-Linear fitting
% Q = aK^alphaL^beta
x0 = [0.5, 0.5, 0.5];
[x, norm, res] = lsqcurvefit(@cdfun, x0, [K, L]', Q');
fprintf('Non-Linear: Alpha = %f, Beta = %f, a = %f\n', x(2), x(3), x(1));

%% Plot curve.
figure;
hold on;
[X, Y] = meshgrid(0:0.1:5, 4:0.1:8);
al = x(2); %beta(1);
be = x(3); %beta(2);
```

```
a = x(1); %exp(beta(3));
Z = a .* (X .^ al) .* (Y .^ be);
mesh(X, Y, Z);
scatter3(K, L, Q, 'r')
xlabel('K');
ylabel('L');
zlabel('Q');
```

其中用到的函数 `cdfun`如下：

```
function y = cdfun( x, t )
    y = x(1) * (t(1,:) .^ x(2)) .* (t(2,:) .^ x(3));
end
```

2.2.3 计算结果与分析

使用线性最小二乘拟合计算出的结果为：

$$\alpha = 0.66, \beta = 1.27, a = 0.31$$

最终的二次曲面如图 3所示。其中，红色的圆圈代表原有的数据点。可以看出，大部分点都落在了曲线上。但是实际上，题目中给出了约束条件 $0 < \alpha, \beta < 1$，上述参数中的 β 并不符合约束条件，因此尝试采用 `lsqlin`函数进行进行带约束的拟合，得出结果为：

$$\alpha = 0.78, \beta = 0.29, a = 1.91$$

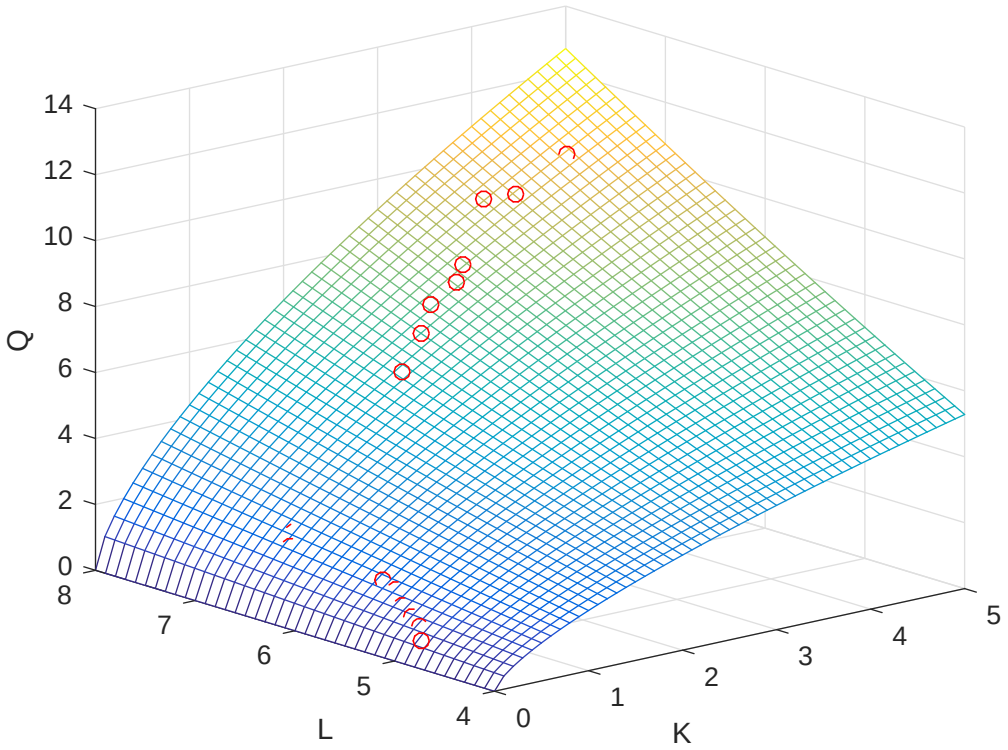

图 3: 线性最小二乘拟合结果

使用非线性最小二乘拟合计算出的结果为：

$$\alpha = 0.77, \beta = 0.73, a = 0.83$$

非线性最小二乘拟合对应的二次曲面如图 4所示：

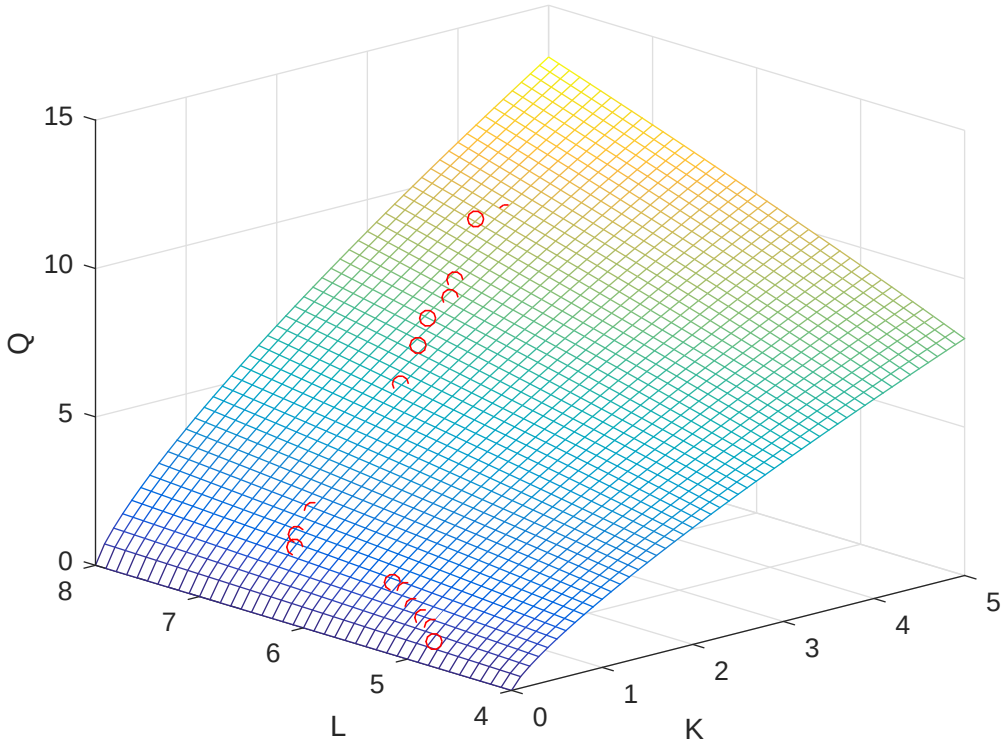

图 4: 非线性最小二乘拟合结果

线性拟合的 norm为 0.3870，非线性拟合的 norm为 2.7488。线性拟合的效果更好。

在 Cobb-Douglas 函数的各个参数中，α 代表资金投入对于总产值重要性系数（或弹性系数），即随着更多资金的投入，总产值增加的线性程度。如果 $\alpha > 1$，则说明随着资金投入的增加，总产值以更快的速度增加。同理，β 则代表劳动力投入对于总产值增加的重要性程度。通常而言，资金的投入和劳动力的投入可能是相互制约的关系，通过确定 α 和 β 以及他们的相对关系，则可以得出增加哪一项的投入更能提高总产值，获得最大的收益。

3 应用题

3.1 CH7-5 原子位置关系

3.1.1 模型建立

建立原子位置的平面直角坐标系，并假设第 1 个原子的位置在原点 $(0,0)$ 处，这种假定并不影响各个原子的相对位置关系。设第 i 个原子（$i = 2, 3, \cdots, 25$）在坐标系中的位置为 (x_i, y_i)，第 i 个原子和第 j 个原子的距离为 d_{ij}。问题可以转化为求得最优的坐标集合 (x_i, y_i)，从而使得题目中给的相对距离关系尽量满足。

设题目中所有给定距离关系的点对坐标 (i, j) 的集合为 M，以满足条件的欧氏距离作为误差衡量标准，给出下列无约束优化目标：

$$\min \Sigma_{(i,j)\in M}[(x_i - x_j)^2 + (y_i - y_j)^2 - d_{ij}^2]^2$$

最终，各个原子的位置关系就可以通过直角坐标系上的坐标进行表示了。

3.1.2 算法设计

根据题目中所给数据，这是一个 48 变量的优化问题，应尝试采用 MATLAB 工具箱中的 `fminunc`工具求解，并使用 BFGS, DFP 和最速下降法进行比较，取使目标函数最小的变量值作为最终结果。此外，还可以调整收敛的精度，进一步提高准确性。

3.1.3 MATLAB 程序

```
%% Math Exp Homework 4 7-T5
% Compute the distance of atoms.

%% Define optimize goal
% Data Matrix
global dm;
dm = [4, 1, 0.9607; 12, 1, 0.4399; 13, 1, 0.8143;
    17, 1, 1.3765; 21, 1, 1.2722; 5, 2, 0.5294;
    16, 2, 0.6144; 17, 2, 0.3766; 25, 2, 0.6893;
    5, 3, 0.9488; 20, 3, 0.8000; 21, 3, 1.1090;
    24, 3, 1.1432; 5, 4, 0.4758; 12, 4, 1.3402;
    24, 4, 0.7006; 8, 6, 0.4945; 13, 6, 1.0559;
    19, 6, 0.6810; 25, 6, 0.3587; 8, 7, 0.3351;
    14, 7, 0.2878; 16, 7, 1.1346; 20, 7, 0.3870;
    21, 7, 0.7511; 14, 8, 0.4439; 18, 8, 0.8363;
    13, 9, 0.3208; 15, 9, 0.1574; 22, 9, 1.2736;
    11, 10, 0.5781; 13, 10, 0.9254; 19, 10, 0.6401;
    20, 10, 0.2467; 22, 10, 0.4727; 18, 11, 1.3840;
    25, 11, 0.4366; 15, 12, 1.0307; 17, 12, 1.3904;
    15, 13, 0.5725; 19, 13, 0.7660; 15, 14, 0.4394;
    16, 14, 1.0952; 20, 16, 1.0422; 23, 16, 1.8255;
    18, 17, 1.4325; 19, 17, 1.0851; 20, 19, 0.4995;
    23, 19, 1.2277; 24, 19, 1.1271; 23, 21, 0.7060;
    23, 22, 0.8052];

%% Start solution
% Initial Condition
%x0 = ones(48, 1);
x0 = -1 + 2 * rand(48, 1);
fopt = optimset('HessUpdate', 'bfgs', 'MaxFunEvals', 1000000, ...
    'MaxIter', 10000);
[xres, vres, exitr, outr] = fminunc(@atomdis, x0, fopt);
```

其中用到的函数 `atomdis`如下：

```
function y = atomdis( x )
    global dm;
    y = 0;
    for t = 1:length(dm)
        i = dm(t, 1);
        j = dm(t, 2);
        dij = dm(t, 3);
```

```
        if i == 1
            x_i = 0;
            y_i = 0;
        else
            x_i = x(2 * i - 3);
            y_i = x(2 * i - 2);
        end
        if j == 1
            x_j = 0;
            y_j = 0;
        else
            x_j = x(2 * j - 3);
            y_j = x(2 * j - 2);
        end
        y = y + ((x_i - x_j)^2 + (y_i - y_j)^2 - dij * dij) ^ 2;
    end
end
```

3.1.4 计算结果

对于初始值，首先选取尝试了全为 1 的初值，最大迭代次数选为 10^4，最大方程求值次数选为 10^6。经过比较，各个搜索方向的性能如表 1所示，表中标识为 -代表迭代次数超出了预定范围，算法提前终止。

	迭代次数	目标函数调用次数	函数值
BFGS	156	8477	0.0482
DFP	-	-	0.092
最速下降法	710	104664	2.6499

表 1: 初始值为全 1 时各个搜索方向算法的性能比较

之后又对表现良好的 BFGS 算法取了 $[-1, 1]$ 之间的随机初值，尝试 1000 次，画出了方程求值次数与求得的函数最优值的分布图如图 5所示。

最终使用 BFGS 算法取一个目标函数值最小（$f_{min} = 0.014$）的 x 作为输出。最终求出的各个原子的最优坐标如表 2所示，作图如图 6所示。

3.1.5 结果分析

根据上述使用不同方法和不同初值进行分析比较，可以看出在求解本题的过程中，使用 BFGS 公式的拟牛顿法能够以较少的迭代次数和函数求值次数得到较优的函数值，相比于 DFP 公式和最速下降法表现更好。在初值的选取方面，能够看出初值对于方法的函数求值次数和最终找到的是否为全局最优解关系**很密切**。在图 5中可以清楚地看到这一点。在选择某些初始值时，能够求得全局的最优值，而有些初始值则可能误导算法进入局部最优解并停止迭代。不过大体上来说，随着方程求值次数的增加，得到全局最优解的概率也相应增加，如果算法此次运行的时间足够长，则有较大的把握认为此次收敛的结果是全局最优解。

4 收获与建议

通过这次的实验，我对 MATLAB 中提供求解无约束优化问题的程序以及求解方法理解更加深刻，通过实际编程、画图的方式观察了方程求解的结果，这是书本上无法学到的知识。

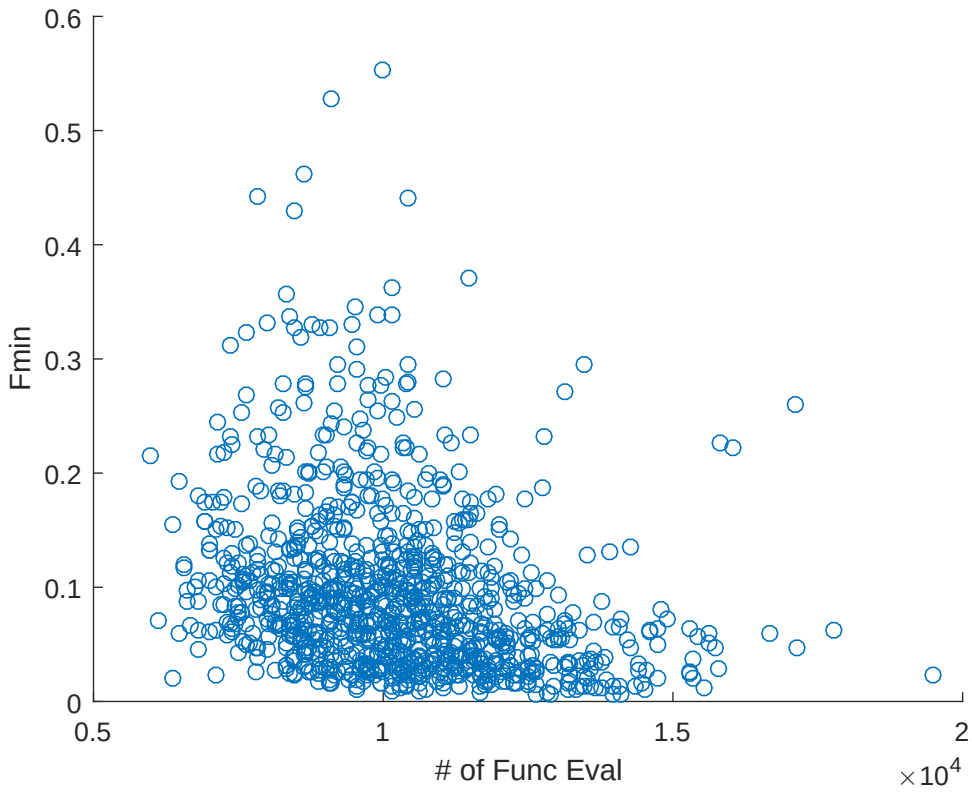

图 5: 随机初始值下 BFGS 公式方程求值次数与求得的函数最优值的分布图

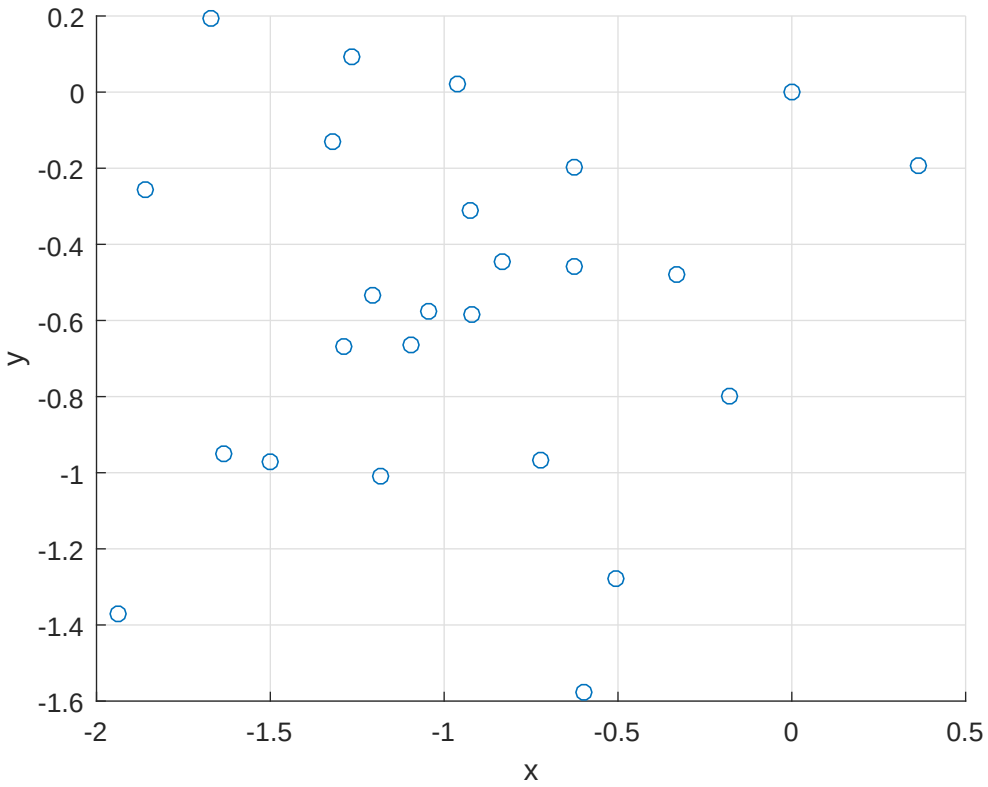

图 6: 各个原子的坐标

原子编号	x	y	原子编号	x	y
1	0	0	14	-1.0441	-0.5761
2	-0.7216	-0.9689	15	-0.6241	-0.4595
3	-1.6344	-0.9497	16	-0.5999	-1.5758
4	-0.9639	0.0203	17	-0.5067	-1.2784
5	-0.8314	-0.4446	18	-1.9362	-1.3703
6	-1.2055	-0.5329	19	-0.6268	-0.199
7	-1.2868	-0.6687	20	-0.92	-0.5859
8	-1.1821	-1.0084	21	-1.2642	0.0937
9	-0.3292	-0.4788	22	-1.4983	-0.9713
10	-1.097	-0.6632	23	-1.8593	-0.2552
11	-1.3216	-0.1303	24	-1.6705	0.1943
12	0.3648	-0.1939	25	-0.924	-0.3101
13	-0.1791	-0.8008			

表 2: 各个原子的位置

同时，在做上机实验的过程中，我对 MATLAB 这款软件的使用也更加熟练了。希望在之后的课堂上老师能够当堂进行相关的技巧演示并给出题目的分步解答。