组　　号	Z08
实验题目	优化问题
队员姓名	1. 宋昭琰
	2. 徐意
	3. 李宗琳
	4. 向飞宇

目　录

1 实验一：求 Schaffer 函数的最小值

问题描述：求解以下函数的最小值.

$$f(x,y) = 0.5 + \frac{\sin^2\sqrt{(x^2+y^2)} - 0.5}{[1+0.001(x^2+y^2)]^2}$$

1.1 使用 fminunc 求解函数的最小值

求函数的最小值需要使用 fminunc 函数，而 fminunc 函数的参数可以通过传入一个结构体的方法来确定. 本题中，我们使用了不同的参数对这个问题进行求解，对于步长和精度，我们使用了 MatLab 的默认值，而对于算法，我们分别采用了：

- 信赖域算法（大规模算法）；
- 最速下降法（中小规模算法）；
- BFGS 方法（中小规模算法）；
- DFP 方法（中小规模算法）.

然后，通过设定不同的初值对问题进行求解，使用 exitflag 参数查看函数返回值，output 参数查看函数运行情况. 以下是我们的求解结果.

当初值为 $[1,1]$ 时，所得结果如下：

求解方法	最优解	最优解时的函数值	迭代次数
信赖域法	$[-0.5937\cdot10^{-8}, -0.5937\cdot10^{-8}]$	$5.5511\cdot10^{-17}$	4
最速下降法	$[-0.7982\cdot10^{-10}, -0.7982\cdot10^{-10}]$	0	3
BFGS	$[-0.5937\cdot10^{-8}, -0.5937\cdot10^{-8}]$	$5.5511\cdot10^{-17}$	4
DFP	$[-0.6387\cdot10^{-8}, -0.6387\cdot10^{-8}]$	$5.5511\cdot10^{-17}$	4

表 1 初值为 $[1,1]$ 时的求解结果

当初值为 $[2,4]$ 时，所得结果如下：

求解方法	最优解	最优解时的函数值	迭代次数
信赖域法	$[1.4036, 2.8071]$	0.0097	4
最速下降法	$[1.4036, 2.8071]$	0.0097	3
BFGS	$[1.4036, 2.8071]$	0.0097	4
DFP	$[1.4036, 2.8071]$	0.0097	4

表 2 初值为 $[2, 4]$ 时的求解结果

当初值为 $[5, 15]$ 时，所得结果如下：

求解方法	最优解	最优解时的函数值	迭代次数
信赖域法	$[4.963314.8899]$	0.1782	3
最速下降法	$[4.963314.8899]$	0.1782	9
BFGS	$[4.963314.8899]$	0.1782	3
DFP	$[4.963314.8899]$	0.1782	3

表 3 初值为 $[5, 15]$ 时的求解结果

由以上的求解结果可以看出，当初值越靠近 $[0, 0]$ 时，所求的最小值越接近 0（这也是函数的全局最小值）. 而初值越远离 $[0, 0]$ 时，无论使用什么算法，其误差都是很大的.

1.2 对结果的分析

实验一的函数求解结果对初值非常敏感，下面我们从函数的角度讨论一下这个问题.

首先做极坐标换元 $x = r\cos\theta, y = r\sin\theta, r \in \mathbb{R}, \theta \in [0, 2\pi]$. 于是原来的函数可以化为：

$$f(x, y) = g(r) = 0.5 + \frac{\sin^2 r - 0.5}{(1 + 0.001r^2)^2} = 0.5 - \frac{\cos 2r}{2(1 + 0.001r^2)^2}$$

对 r 求导，可得：

$$g'(r) = \frac{\sin 2r}{(1 + 0.001r^2)^2} + \frac{\cos 2r \cdot 0.002r}{(1 + 0.001r^2)^3}$$

其导数为 0 的点满足：

$$(1 + 0.001r^2) \cdot \sin 2r = 0.002r \cdot \cos 2r$$

由于 $\cos 2r = 0$ 时无法满足上式，继续化简可得：

$$\tan 2r = \frac{0.002r}{1 + 0.001r^2} \tag{1}$$

任何满足方程（1）的点 $\boldsymbol{r}^*$，都可能成为 Schaffer 函数的极小值，而方程（1）的解如下图所示：

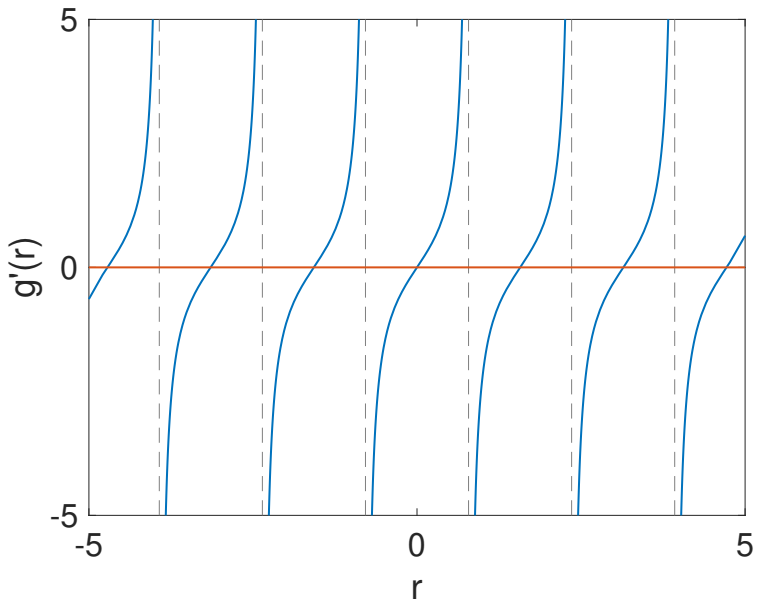

图 1 方程（1）的解

由上图可以看出，方程（1）的解是周期性的，这对应着更多的 $[x, y]$ 点对，使得函数取极小（极大）值. 所以使用最速下降等算法时，很可能陷入一个局部最优，这样就得不到全局最优解了.

而 Schaffer 函数的全局最优解可以这样算出，由于：

$$\cos 2r \leqslant 1, \quad (1 + 0.001r^2)^2 \geqslant 1.$$

可得：

$$\frac{\cos 2r}{2(1 + 0.001r^2)^2} \leqslant 0.5$$

故 $\min g(r) = 0$，此时对应的最优解为 $[0, 0]$，这与实验结果相符.

1.3 实验一的代码

```
f = @(x) 0.5 + ((sin(sqrt(x(1).^2 + x(2).^2))).^2 - 0.5) ./ (1 +
↪  0.001.*(x(1).^2 + x(2).^2)).^2;
x0 = [5 15];
```

```
options = optimset;
options.LargeScale = 'on';
options.Algorithm = 'trust-region';
[x1, fval1, exitflag1, output1] = fminunc(f, x0, options);

options = optimset;
options.LargeScale = 'off';
options.HessUpdate = 'steepdesc';
options.LineSearchType = 'quadratic';
[x2, fval2, exitflag2, output2] = fminunc(f, x0, options);

options = optimset;
options.LargeScale = 'off';
options.HessUpdate = 'bfgs';
options.LineSearchType = 'quadratic';
[x3, fval3, exitflag3, output3] = fminunc(f, x0, options);

options = optimset;
options.HessUpdate='dfp';
options.LineSearchType = 'quadratic';
[x4, fval4, exitflag4, output4] = fminunc(f, x0, options);
```

2 实验二：求解非线性约束的最优问题

问题描述：求解以下非线性约束的最优问题.

$$\min f(x) = -x_1x_2x_3$$

$$\text{s.t.} \begin{cases} -x_1 - 2x_2 - 2x_3 \leq 0 \\ x_1 + x_2 + 2x_3 \leq 72 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

2.1 利用 fmincon 函数求解非线性约束的最优化问题

思路分析：我们利用 fmincon 函数来求解该方程，该方程只有线性约束且只有不等式约束，故我们很容易确定函数中线性不等式约束的系数矩阵 A 与对应约束的最大值向量

$\boldsymbol{b}$:

$$A = \begin{bmatrix} -1 & -2 & -2 \\ 1 & 2 & 2 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 0 \\ 72 \end{bmatrix}$$

由于 x_1, x_2, x_3 均有非负限制，需要将求解的下界向量 ub 设置为 $[0,0,0]^T$.

而对于初始点的选择，我们从方程容易看到 $\boldsymbol{x_0} = [0,0,0]^T$ 这一点满足约束条件，具有可行性并且所有可行解对应的函数值都不劣于该点所对应的函数值，故我们取 $[0,0,0]^T$ 为初始点.

由于函数性质较好，我们使用 fmincon 函数的默认设置（算法为内点法，Hessian 矩阵使用 BFGS 方法近似，误差与步长为默认设置）解得全局最优解为 $[24,12,12]^T$，此时函数的最小值为 -3456.

2.2 对结果的分析

此节从理论角度对结果进行分析，并说明上面实验结果的正确性.

首先,$x_1, x_2, x_3 > 0$,所以 $-x_1-2x_2-2x_3 \leqslant 0$ 的条件自然满足. 由 $x_1+2x_2+2x_3 \leqslant 72$,与均值不等式可得:

$$\sqrt[3]{x_1 \cdot 2x_2 \cdot 2x_3} \leqslant \frac{x_1 + 2x_2 + 2x_3}{3} \leqslant 24$$

化简后即可得 $x_1x_2x_3 \leqslant 3456$. 即 $-x_1x_2x_3 \geqslant -3456$. 等号当且仅当 $x_1 = 2x_2 = 2x_3$,且 $x_1 + 2x_2 + 2x_3 = 72$ 取得，即有:

$$x_1 = 24, \quad x_2 = 12, \quad x_3 = 12$$

这与实验结果相符，说明实验结果正确.

2.3 问题二的代码

```
[x,fval] = fmincon(@(x)-x(1).*x(2).*x(3), [0; 0; 0], [-1, -2, -2; 1, 2,
↪  2], [0; 72], [], [], [0; 0; 0], [])
```

3 实验三：椭圆轨道的确定问题

问题描述：根据测量小行星的坐标数据，给出小行星的运动轨道. 假定小行星绕太阳转动，所测量的数据如下:

x	5.764	6.286	6.759	7.168	7.480
y	0.648	1.202	1.823	2.526	3.360

3.1 利用椭圆的第一定义确定椭圆轨道

定义 3.1 （椭圆的第一定义） 任意一点到两定点距离为定值的点所构成的图形是椭圆，两定点称为椭圆的焦点，定值称为椭圆的长轴.

由开普勒定律可得小行星的运动轨道为椭圆，且太阳为其轨道的一个焦点. 太阳所在点坐标为 $O(0,0)$，设另一个焦点为 $F(x_0,y_0)$，椭圆的半长轴为 a. 由椭圆的第一定义可得：

$$CO+CF=2a, C \text{ 在椭圆上}$$

假设 C 点坐标为 (x,y)，则可将上式转换为：

$$\sqrt{x^2+y^2}+\sqrt{(x-x_0)^2+(y-y_0)^2}=2a \tag{2}$$

这个方程有 3 个未知量，需要至少 3 个数据点才能求出椭圆的方程，而本题中有 5 个数据点，所以可以利用类似最小二乘法的思路，确定最优的椭圆轨道，使这些数据点离椭圆的距离最小. 从方便计算的角度上说，方程（2）适合于构造优化函数，假设五个数据点的坐标分别为 $(x_i,y_i), i=1,2,\cdots 5$，则优化函数为：

$$\min_{x_0\in\mathbb{R},y_0\in\mathbb{R},a\in\mathbb{R}^+} G(x_0,y_0,a)=\sum_{i=1}^{5}\left|\sqrt{x_i^2+y_i^2}+\sqrt{(x_i-x_0)^2+(y_i-y_0)^2}-2a\right|$$

对于以上的优化函数，我们使用 MatLab 中的 fmincon 函数求出其最小值，由于 $(0,0,0)$ 在函数定义域内，且该函数的最小值也在其附近（根据测量点的数据估算），我们将其作为搜索的初值. 然后根据函数的定义域，我们设定了变量的下界 $(-\infty,-\infty,0)$ 与上界 $(+\infty,+\infty,+\infty)$. 由于这个问题不需要进行其它约束条件，其余部分可以置空值. 对于其它设置，我们在算法上保留了 MatLab 内置的内点法（interior-point），函数下降的阈值设定为 10^{-12}，其余条件保持默认. 最终得出的结果表示，MatLab 找到了局部最小值，且其值为 0.0016，这样的结果是可以接受的.

与此同时，在函数取得局部最小值时，有 $x_0=6.3942$，$y_0=4.8191$，$a=5.0094$.

然后，我们能够根据以上参数绘制椭圆的图像，我们使用了 MatLab 中的 ellipse1 函数生成绘制椭圆所需的数据点，这个函数需要椭圆中心的坐标 (x_c,y_c)，椭圆的半长轴 a，偏心率 e 以及旋转角 θ 绘制一个椭圆. 首先，椭圆的中心为两个焦点连线的中点，即有：

$$x_c=\frac{x_0+0}{2}=\frac{x_0}{2},\quad y_c=\frac{y_0+0}{2}=\frac{y_0}{2}$$

而半长轴已经求出，焦距可以根据两个焦点给出，所以偏心率为：

$$e=\frac{c}{a}=\frac{\sqrt{x_0^2+y_0^2}}{2a}$$

椭圆的旋转角可以由两个焦点给出：

$$\theta=\arctan\left(\frac{y_0}{x_0}\right),\quad 0\leqslant\theta<\pi$$

由以上公式，可以绘制出椭圆的图像如下：

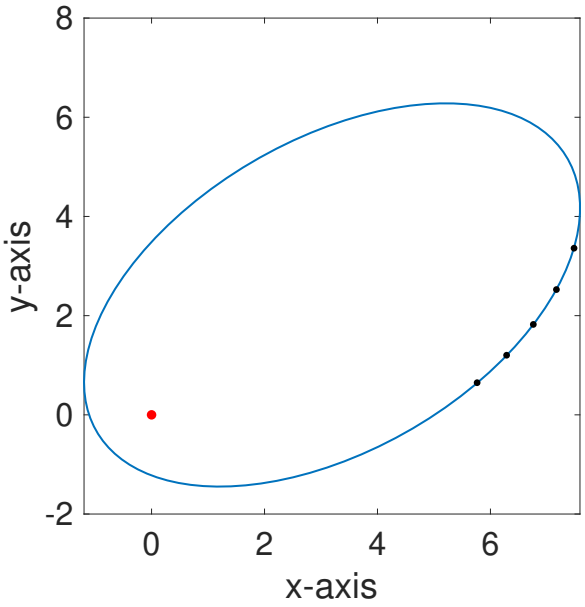

图 2 利用椭圆的第一定义确定椭圆轨道图像

3.2 利用二次曲线的定义确定轨道

以下利用二次曲线的定义确定轨道，首先二次曲线的一般方程如下：

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$$

此处可以先假设 $A = 1$，即不考虑焦点的情况下，使用 5 个参数已经可以确定这条轨道，而所测量的数据中已经有 5 个点. 即有：

$$x_i^2 + Bx_i^2y_i^2 + Cy_i^2 + Dx_i + Ey_i + F = 0, \quad i = 1, 2, \cdots, 5$$

这样可以得到：

$$\begin{cases} B = -1.1199 \\ C = 1.1301 \\ D = -5.3238 \\ E = -0.7047 \\ F = 1.6279 \end{cases}$$

这条曲线的方程为：

$$x^2 - 1.1199xy + 1.1301y^2 - 5.3238x - 0.7047y + 1.6279 = 0$$

根据以上的数据, 由于这条二次曲线是隐函数, 所以可以使用 MatLab 中的 fimplicit 函数作出这条二次曲线. 最后得出的结果为:

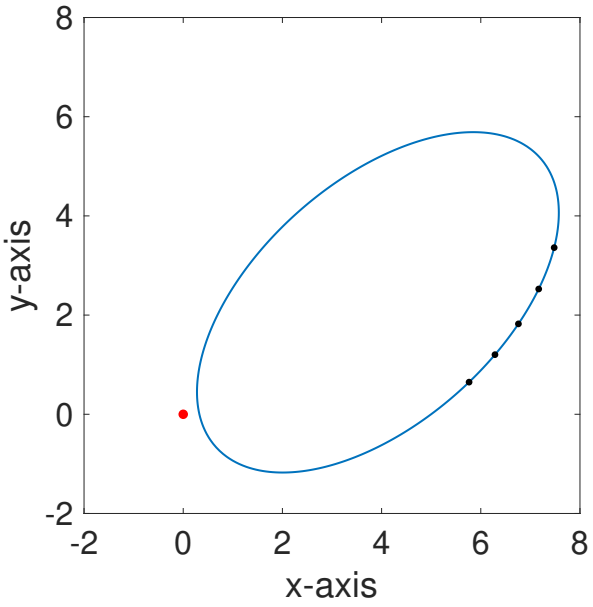

图 3 利用二次曲线的定义确定轨道图像

但是使用这个方法没有利用到焦点的性质，可以发现，焦点并不在椭圆以内，所以这个方法存在一定的问题.

3.3 轨道的误差分析

对于小行星的坐标测量不可能完全精确，所以需要进行误差分析. 由于我们无法得知小行星的精确坐标，我们对测量数据加入了一个随机误差项，它表现在对测量数据随机乘上 $[0.9995, 1.0005]$ 之间的一个数. 然后将这组随机生成的数据作为小行星的“真实坐标”，将算出的两条轨道进行比较，以下是求解结果:

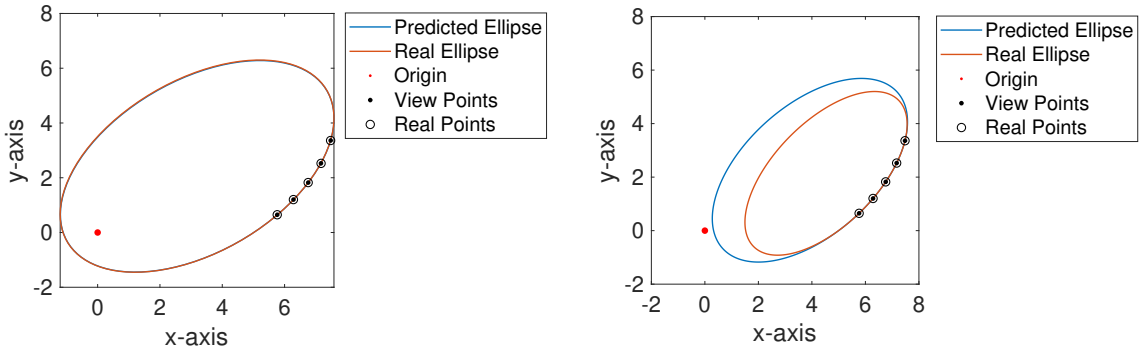

图 4 方法一（左）与方法二（右）的误差分析

考虑到对于星体的测量，0.05% 的绝对误差已经很小，但是在这个误差下，使用方法二（二次曲线的定义）计算的结果之间的差距仍然很大；而使用方法一（椭圆的第一定义）计算的结果之间的差距几乎可以忽略.

使用二次曲线的定义会造成巨大的误差，这是因为我们使用了 5 个数据点才得出了轨道的方程，并没有能够用于修正误差的多余数据；反观使用椭圆定义的方法，这个方法使用 3 个数据点就能计算出结果，而我们拥有 5 个数据点，所以能综合数据，得到一条更符合这五个点的轨道. 同时，轨道上的五个测量点距离非常近，使用二次曲线的定义就是在解一个线性方程组，只要有一点误差，就可能影响方程的解，甚至可能得不出一条椭圆轨道.

3.4 实验三的代码

```
% data-based
x = [5.764 6.286 6.759 7.168 7.480];
y = [0.648 1.202 1.823 2.526 3.360];

% randomize for a real value
randomizex = 0.9995 + 0.001 * rand(1, 5);
randomizey = 0.9995 + 0.001 * rand(1, 5);
x_real = x .* randomizex;
y_real = y .* randomizey;

result = linear_equation_solution(x, y);
real_result = linear_equation_solution(x_real, y_real);
figure(1);
fimplicit(@(x, y) x.^2 + result(1)*x.*y + result(2).*y.^2 + result(3)*x
↪  + ...
                        result(4)*y + result(5), [-2 8], 'linewidth',
                        ↪  1);
hold on;
fimplicit(@(x, y) x.^2 + real_result(1)*x.*y + real_result(2).*y.^2 +
↪  real_result(3)*x + ...
                        real_result(4)*y + real_result(5), [-2 8],
                        ↪  'linewidth', 1);
hold on;
scatter(0, 0, 200, '.r');
```

```
hold on;
plot(x, y, 'k.', 'MarkerSize', 10);
hold on;
plot(x_real, y_real, 'ko');
hold off;
set(gca, 'fontsize', 16);
xlabel('x-axis');
ylabel('y-axis');
axis square;
legend(["Predicted Ellipse"; "Real Ellipse"; "Origin"; "View Points";
↪  "Real Points"], 'Location','NorthEastOutside');

[ea, eb] = elliptic_solution(x, y);
[ea_real, eb_real] = elliptic_solution(x_real, y_real);
figure(2);
plot(ea, eb, 'linewidth', 1);
hold on;
plot(ea_real, eb_real, 'linewidth', 1);
hold on;
scatter(0, 0, 200, '.r');
hold on;
plot(x, y, 'k.', 'MarkerSize', 10);
hold on;
plot(x_real, y_real, 'ko');
hold off;
set(gca, 'fontsize', 16);
xlabel('x-axis');
ylabel('y-axis');
axis square;
legend(["Predicted Ellipse"; "Real Ellipse"; "Origin"; "View Points";
↪  "Real Points"], 'Location','NorthEastOutside');

function result = linear_equation_solution(x, y)
    % Solving problem using linear equation
    A = [x.*y; y.^2; x; y; ones(1, 5)]';
```

```
    b = transpose(-x.^2);
    result = A \ b;
end

function [ea, eb] = elliptic_solution(x, y)
    % Solving problem using optimization
    f = @(t) sum(abs(sqrt(x.^2 + y.^2) + sqrt((x - t(1)).^2 + (y -
    ↪  t(2)).^2) - 2 .* t(3)));
    options = optimset('fmincon');
    options.TolFun = 1e-12;
    options.TolProjCGAbs = 1e-12;
    options.TolCon = 1e-12;
    options.Display = 'off';
    [result2, ~, ~] = fmincon(f, [0 0 0], [], [], [], [], [-inf -inf 0],
    ↪  [inf inf inf], [], options);
    x0 = result2(1); y0 = result2(2);
    xc = x0 / 2; yc = y0 / 2;
    a = result2(3);
    c = sqrt(x0.^2 + y0.^2) / 2;
    ecc = c / a;
    offset = rad2deg(atan2(y0, x0));
    [ea, eb] = ellipse1(xc, yc, [a, ecc], offset);
end
```

4 实验四：模拟盲人下山的迭代寻优算法

问题描述：以 $f(x_1, x_2) = 8x_1^2 + 9x_2^2 - 8x_1x_2 - 12x_1 - 6x_2$ 为例来模拟一座山峰，解决盲人下山的最优化问题 $\min_{x_1,x_2\in\mathbb{R}} f(x_1, x_2)$.

4.1 情景解读

问题对盲人下山情景做了以下描述：

> 由于盲人看不见山势的变化，因此他只能根据脚下的变化选择一个前进的方向，然后做探测性移动. 很自然，这一方向应是下降方向，盲人沿着该方向探测移动，走到山在该方向的最低点并停止在该点，然后在新的位置重新寻

找前进方向，继续进行探测性移动. 按这种方式，可以期待盲人最终达到某一个山谷的最低点.

可以知道，盲人本身可以感知到下降方向，并且在理想情况下沿着方向直线行走；当行走时感觉到不能再下降时，盲人会停下来在该点找到另外一条下降的方向，继续直线行走. 我们将之归纳为数学语言便是：

- 每一个位置下一步的方向应该是下降方向；
- 确定好方向后，盲人会一直沿着这个方向移动，直到该方向上的函数达到最小值，停止原方向的移动，确定新的方向.

4.2 设计迭代寻优算法

1. 初始化：最大迭代次数 $\max\_\text{iter} = 1000$，精度 $\text{eps} = 1e-3$.
2. 随机选取一个点作为初始点，计作 $\boldsymbol{x_0}$，令方向 $\boldsymbol{d} = -\nabla f(\boldsymbol{x_0})$，迭代次数 $\text{iter} = 1$.
3. 当 $\text{iter} < \max\_\text{iter}$ 时，计算 $\min f(\boldsymbol{x_0} + h\boldsymbol{d})$，得到目标最小值和 h_0. 令 $\boldsymbol{x} = \boldsymbol{x_0} + h_0\boldsymbol{d}$，$\text{iter} = \text{iter} + 1$；当 $\text{iter} \geq \max\_\text{iter}$ 时，转到 5.
4. 当 $||\boldsymbol{x} - \boldsymbol{x_0}|| \geq \text{eps}$，转到 3；否则转到 5.
5. 得到最终结果 $\boldsymbol{x}^*$.

4.3 设计可视化

为了展现路径图，我们将二元函数用等高线的形式绘制出每个点的高度. 为了体现初始点的随机性，我们采用交互式命令 **ginput** 在图像上随机选取一个，设为 $\boldsymbol{x_0}$；利用以上算法算出来此后每一个 $\boldsymbol{x}$，并且用线段连接.

利用下文代码，随机给出初始点，可以得到若干个路径图：

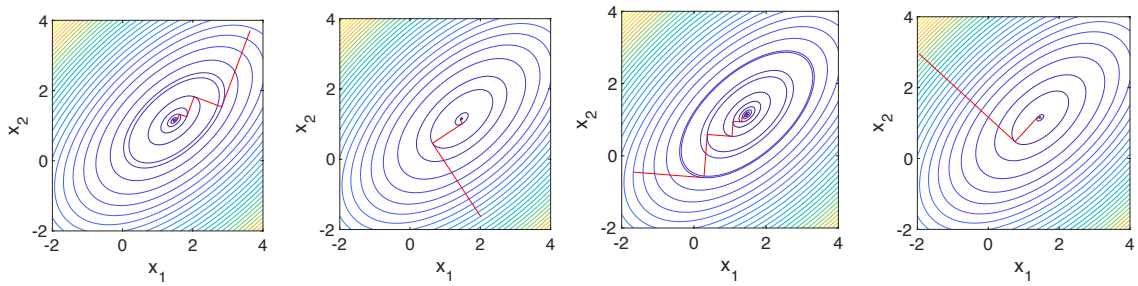

图 5 不同初始点的路径

以上的下山实验方法对应于优化理论中的**最速下降法**，而从上面取不同的点的实验结果可以看出，无论所选的点的位置如何，相邻两次下降的方向是垂直的. 这一点可以通过计算先后两点的梯度得出. 而对于不同的点，迭代次数是不同的，由于本题中所求的函数性质较好，我们选取的初始点都能够到达最优解. 但是对于性质不好的函数，初始点的选取就决定了实验的结果好坏.

4.4 实验四的代码

```
clf;
syms x1 x2 h; % symbolic values, h represents step.
max_iter = 1000; % maximum iteration
eps = 1e-3; % precisiion
func = 8 * x1^2 + 9 * x2^2 - 10 * x1 * x2 - 12 * x1 - 6*x2;
a = -2; b = 4;
xmin = a; xmax = b; ymin = a; ymax = b;
X1 = linspace(xmin, xmax, 100);
X2 = linspace(ymin, ymax, 100);
[xx, yy] = meshgrid(X1, X2);
zz = 8 * xx.^2 + 9 * yy.^2 - 10 * xx .* yy - 12 * xx - 6 * yy;
contour(xx, yy, zz, 30);
axis([xmin xmax ymin ymax]);
hold on;
axis equal;
plot(1.468085, 1.148936, 'o'); % minimum value
v = [x1 x2];
df = gradient(func, v);

[x0(1), x0(2)] = ginput(1); % randomize initial points
plot(x0(1), x0(2), 'r', 'MarkerSize',10);
hold on;
fun = matlabFunction(func);
iter=1;

while iter <= max_iter
    df0 = subs(df, v, x0);
    d = -df0; % fastest descend for the direction
    fun0 = fun(x0(1) + h * d(1), x0(2) + h * d(2));
    [h0, fval] = fminbnd(matlabFunction(fun0), 0, 1000); % calculate
    ↪ minimum
    temp = x0;
    x0(1) = x0(1) + h0 * d(1);
    x0(2) = x0(2) + h0 * d(2);
```

```
    x = [temp(1) x0(1)];
    y = [temp(2) x0(2)];
    H_line2 = plot(x,y);
    contour(xx, yy, zz, [fval fval],'-'),
    set(H_line2, 'color', 'red', 'linewidth', 2); % draw plot
    iter = iter + 1;
    if norm(x0 - temp) < eps
        break; % break iteration if two points are very near
    end
end
```

附录 A 使用含精英选择的多种群遗传算法对实验一进行改进

从实验一的求解结果可以看出，所得函数最小值非常依赖于初值的选取，而使用遗传算法则可以大幅降低初值对实验结果的影响. 首先介绍遗传算法的相关概念.

A.1 遗传算法的基本概念

遗传算法（Genetic Algorithm）是一种进化算法，其基本原理是仿效生物界中的自然选择原理：即通过把问题参数编码为染色体，再通过迭代的方式进行选择，交叉，变异，种群间交流等运算交换染色体的信息，不断生成更符合优化条件的个体. 这是一个全局优化算法，理论上能够通过不断迭代得到最优解. 本文遗传算法采用谢菲尔德大学的遗传算法工具箱（GA Toolbox, gatbx）与自己编写的遗传算法框架（ezga），以下为问题的求解过程：

- 种群的初始化：产生一个包含 50 个个体的种群，种群的初始个体通过 gatbx 中的 crtrp 函数产生. 此时需要给定生成种群的上下界，虽然在实际情况下，本题中没有上下界，但是我们设定其值为 $[-50, 50]$，这样已经足够有代表性.
- 适应度函数：该个体的适应度可以通过 gatbx 中的 ranking 函数进行计算，其实质是将所有个体所对应的函数值映射到 $[0, 1]$ 区间中，即：

$$FV = \frac{1}{M-m}(M - 0.5 - \frac{\sin^2\sqrt{(x^2+y^2)} - 0.5}{[1+0.001(x^2+y^2)]^2})$$

其中 M 为当前个体中最大的函数值，m 为当前个体中最小的函数值，通过上述公式可以将取值区间平移至 $[0, 1]$，且由计算公式可见，个体适应度越接近 1，则个体越优秀.

- 选择操作：旧种群中的个体将以一定概率被选择进入新种群，进行下一轮进化. 本文将这个概率设为 0.95，采用轮盘赌的方式选择个体，个体被选中的概率与适应

度有关，个体适应度值越大，被选中的概率越大.

- 交叉操作：交叉概率设为 0.70，交叉方式选为 xovsp（单点交叉），即即被选中的两个亲代个体有 70% 的概率产生同时拥有双亲特征的子代个体.
- 变异操作：采用随机交换两个基因点的方式进行变异操作，使用 gatbx 中的 mutbga 函数实现，概率设为 0.10，即所有个体有 10% 的概率产生变异.
- 多种群优化处理：设定 10 个子种群，即每 5 个个体构成一个子种群，每 10 代将会进行种群间的交流，每个个体将有 20% 的概率进入其它种群，这样使得出现不同个体的概率增加，降低了早熟收敛的趋势.
- 精英策略：本文选择每个子种群中适应度最高的个体作为精英个体，它们将不参与选择，交叉，变异操作而直接成为下一代个体，这样使得优秀个体不会因为未被选择等情况而丢失.
- 非线性寻优策略：每 5 次迭代作为一次“大的进化”，将个体当前值作为寻优初值，使用 MatLab 中的 fmincon 函数进行非线性寻优，增大寻找到最优值的概率.

A.2 遗传算法的求解结果

使用遗传算法求解出最优解为 $[-0.1816 \cdot 10^{-8}, 0.2204 \cdot 10^{-8}]$，此时函数的最小值为 0（精度超过了 MatLab 能够表示的最小值），约经过了 12 次迭代就达到最优，以下是种群进化的图像:

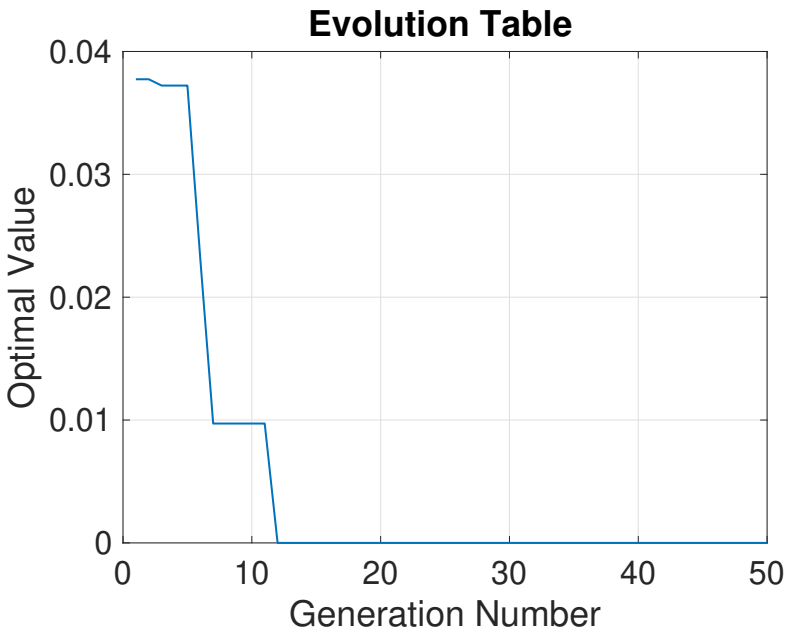

图 6 种群进化的图像

可以发现，遗传算法有效地解决了初值对结果的影响问题.

A.3 遗传算法的相关代码

```
function [bestx, optimal] = ezga(fun, nvars, lb, ub)
    % CHANGE THE FUNCTION FIRST IF YOU WANT A MAXIMUM SEARCH

    % Configuration.Basics
    PopulationNumber = 50;
    MaxGeneration = 50;
    GenerationGap = 0.95;
    % Configuration.Selection
    SEL_Function = 'rws'; % rws or sus
    CrossoverProb = 0.7;
    % Configuration.Recombination
    REC_Function = 'xovsp'; % recdis or xovsp
    InsertOption = 1; % 0 for uniform choice, 1 for fitness value, 2
    ↪  for ratio value
    % Configuration.Mutation
    MUT_Function = 'mutbga'; % mutate or mut
    MutationProb = 0.1;
    % Configuration.Migration
    SUBPOP = 10;
    MigrationProb = 0.2;
    MigrationInterval = 10;
    % Configuration.NonlinearSearch
    EnableNlnrSearch = 0; % enable fmincon search here
    NonlinearInterval = 5;
    % Configuration.PlotFigure
    DrawEvolutionTable = 1;
    % Initialization
    tracer = zeros(nvars + 1, MaxGeneration);
    FieldDescriptor = [lb; ub];
    PopulationInfo = crtrp(PopulationNumber * SUBPOP, FieldDescriptor);
    % Optimization
    counter = 0;
    GlobalMaxFitnV = -inf * ones(SUBPOP, 1);
    X = PopulationInfo;
```

```
ObjectValue = zeros(size(X, 1), 1);
for i = 1:size(X, 1)
    ObjectValue(i) = fun(X(i, :)');
end

while counter < MaxGeneration
    FitnessValue = ranking(ObjectValue, 2, SUBPOP); % define fitness
    ↪ value
    [LocalMaxFitnV, LocalBestObjV, LocalBestIndividual] =
    ↪ eltselect(FitnessValue, PopulationInfo, ObjectValue,
    ↪ SUBPOP); % elitist selection
    SelectPopulation = select(SEL_Function, PopulationInfo,
    ↪ FitnessValue, GenerationGap, SUBPOP); % select population
    Recombination = recombin(REC_Function, SelectPopulation,
    ↪ CrossoverProb, SUBPOP); % recombine
    Mutation = mutate(MUT_Function, Recombination, FieldDescriptor,
    ↪ MutationProb, SUBPOP); % mutate
    X = Mutation;

    ObjectValueNext = zeros(size(X, 1), 1);
    for i = 1:size(X, 1)
        ObjectValueNext(i) = fun(X(i, :)'); % generate son
        ↪ generation values
    end

    % nonlinear optimization using current X value for a local
    ↪ optimization.
    if mod(counter, NonlinearInterval) == 0 && counter > 0 &&
    ↪ EnableNlnrSearch == 1
        tempX = zeros(size(X, 1), nvars);
        options = optimset();
        options.Display = 'off';
        for i = 1:size(X, 1)
            tempx = fmincon(fun, X(i, :)', [], [], [], [], lb, ub,
            ↪ [], options);
```

```
            tempX(i, :) = tempx';
        end
        X = tempX;
    end
    [PopulationInfo, ObjectValue] = ...
        reins(PopulationInfo, X, SUBPOP, InsertOption, ObjectValue,
        ↪ ObjectValueNext);  % reinsert son to father

    if (mod(counter,MigrationInterval) == 0)
        [PopulationInfo, ObjectValue] = ...
            migrate(PopulationInfo, SUBPOP, [MigrationProb, 1, 0],
            ↪ ObjectValue); % migration
    end

    [GlobalMaxFitnV, PopulationInfo, ObjectValue] = ...
        eltchange(PopulationInfo, ObjectValue, GlobalMaxFitnV,
        ↪ LocalMaxFitnV, ...
        LocalBestObjV, LocalBestIndividual, SUBPOP); % elitist
        ↪ substitution

    X = PopulationInfo;
    counter = counter + 1; % update counter
    [Optimal, Index] = min(ObjectValue); % get minimal index
    tracer(1:nvars, counter) = X(Index, 1:nvars); % get minimal in
    ↪ every generation
    tracer(end, counter) = Optimal; % get optimal solution in every
    ↪ generation
end
if DrawEvolutionTable == 1
    % Evolution Value
    figure(1);
    plot(1:MaxGeneration, tracer(end, :));
    grid on;
    xlabel('Generation Number');
    ylabel('Optimal Value');
```

```
        title('Evolution Table');
    end
    % Output Value
    optimal = tracer(end, end);
    bestx = tracer(1:nvars, end);
end

function [GlobalMaxFitnV, Chrom, ObjV] = eltchange(Chrom, ObjV,
↪   GlobalMaxFitnV, LocalMaxFitnV, LocalBestObjV, LocalBestIndividual,
↪   SUBPOP)
    % A function for ELiTe SELECTion in original GA
    if nargin == 6
        SUBPOP = 1;
    elseif nargin < 6

        error('Insufficient variables.')
    end

    col = size(Chrom, 1);
    NPop = col / SUBPOP;

    for i = 1:SUBPOP
        if LocalMaxFitnV(i) >= GlobalMaxFitnV(i)
            GlobalMaxFitnV(i) = LocalMaxFitnV(i);
            FitnVNew = ranking(ObjV(1 + (i - 1) * NPop: i * NPop), 2,
             ↪  SUBPOP);
            [~, LocalMinIndex] = min(FitnVNew);
            ObjV(LocalMinIndex) = LocalBestObjV(i);
            Chrom(LocalMinIndex, :) = LocalBestIndividual(i, :);
        end
    end
end

function [LocalMaxFitnV, LocalBestObjV, LocalBestIndividual] =
↪   eltselect(FitnV, Chrom, ObjV, SUBPOP)
```

```
    % A function for ELiTe SELECTion in original GA
    if nargin == 2
        SUBPOP = 1;
    elseif nargin == 1
        error('Insufficient variables.')
    end

    [col, row] = size(Chrom);
    NPop = col / SUBPOP;
    LocalMaxFitnV = zeros(SUBPOP, 1);
    LocalBestObjV = zeros(SUBPOP, 1);
    LocalBestIndividual = zeros(SUBPOP, row);
    for i = 1:SUBPOP
        [LocalMaxFitnV(i), LocalMaxIndex] = max(FitnV(1 + (i - 1) *
        ↪  NPop: i * NPop));
        LocalBestObjV(i) = ObjV(LocalMaxIndex);
        LocalBestIndividual(i, :) = Chrom(LocalMaxIndex, :);
    end
end
```