

第 1 章

粒子群优化算法原理及其 MATLAB 实现

1.1 粒子群优化算法的基本原理

粒子群优化 (Particle Swarm Optimization, PSO) 算法由 James Kennedy 等人于 1995 年提出, 提出该算法的灵感来自动物界中鸟群、兽群和鱼群等的迁移、捕食等群体活动。在群体活动中, 每一个个体都会受益于所有成员在这个过程中发现和累积的经验, 因此粒子群优化算法属于进化计算的一种。

粒子群优化算法原理简单, 在内存需求和计算速度方面的成本较低, 是一种能够优化非线性和多维问题的算法。该算法的基本概念是构造一群粒子, 粒子群在其周围的空间 (也就是问题空间) 中移动, 寻找它们的目标点。

1.1.1 初始化阶段

粒子群优化算法自被提出以来, 已经产生了多种变体, 但其核心概念是在问题空间中初始化一群粒子, 粒子通过特定算法进行随机移动, 并通过适应度函数评估它们的移动质量。假设目标空间 (Dim 维) 随机产生了由 nPop 个粒子组成的一个种群, 则该种群应表示为 nPop×Dim 维的向量, 第 i 个粒子表示为

$$\mathbf{X}_i = (x_{i1}, x_{i2}, \dots, x_{i\text{Dim}}), i=1, 2, \dots, \text{nPop} \quad (1-1)$$

假设目标空间的上下边界用 [UpB, LoB] 表示, 则 $\mathbf{X}_i$ 的初始化位置可以通过式 (1-2) 产生。

$$x_{ij} = \text{rand}() \times (\text{UpB} - \text{LoB}) + \text{LoB}, j=1, 2, \dots, \text{Dim} \quad (1-2)$$

1.1.2 位置和速度的更新

粒子群通过初始化阶段产生随机解, 然后通过迭代找到最优解。在每一次迭代中, 粒子通过跟踪自身经验和团体经验来更新自己, 具体算法如下。

$$\mathbf{v}_i = w \times \mathbf{v}_i + c_1 \times \text{rand}() \times (\mathbf{pbest}_i - \mathbf{x}_i) + c_2 \times \text{rand}() \times (\mathbf{gbest} - \mathbf{x}_i) \quad (1-3)$$

$$w = w_{\max} - t \times \left(\frac{w_{\max} - w_{\min}}{\text{maxIt}} \right) \quad (1-4)$$

$$\mathbf{x}_i = \mathbf{x}_i + \mathbf{v}_i \quad (1-5)$$

其中, $\mathbf{v}_i$ 是第 i 个粒子的移动速度, $\mathbf{x}_i$ 是第 i 个粒子当前的位置, c_1 、 c_2 为学习因子,

通常设置为 $c_1=c_2=2$ 。 $\mathbf{pbest}_i$ 为第 i 个粒子的最优解, $\mathbf{gbest}$ 是整个种群的最优解。 w 为惯性权重, 是一个线性下降的非负数, 当 w 较大时, 全局搜索能力较强; 当 w 较小时, 局部寻优能力较强。 $w_{\max}$ 是初始惯性权重, $w_{\min}$ 是迭代至最大时的惯性权重, 一般设置为 $w_{\max}=0.9$, $w_{\min}=0.4$ 。

式 (1-3) 由 3 个部分组成, 第一部分 $w \times \mathbf{v}_i$ 表示粒子受上一次速度和惯性的影响, 第二部分 $c_1 \times \text{rand}() \times (\mathbf{pbest}_i - \mathbf{x}_i)$ 表示粒子受自身经验的影响, 第三部分 $c_2 \times \text{rand}() \times (\mathbf{gbest}_i - \mathbf{x}_i)$ 反映了粒子群的经验共享, 这 3 个部分决定了粒子的下一次运动情况。

1.2 算法流程图

粒子群优化算法流程描述如下。

- (1) 随机初始化每个粒子的速度。
- (2) 计算适应度值并更新全局最优位置和局部最优位置。
- (3) 根据式 (1-3) 和式 (1-5) 计算 $\mathbf{v}_{i,d}$ 和 $\mathbf{x}_{i,d}$, 更新粒子的速度和位置。
- (4) 计算适应度值并更新全局最优位置和局部最优位置。
- (5) 判断是否满足终止条件, 若满足则跳出循环, 否则返回步骤 (3)。
- (6) 输出最优位置及最优适应度值。

粒子群优化算法流程图如图 1-1 所示。

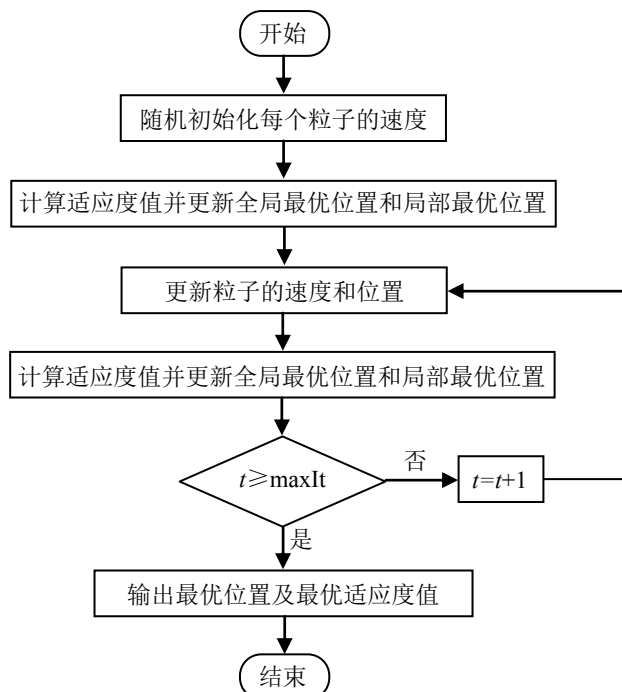


图 1-1 粒子群优化算法流程图

1.3 粒子群优化算法的 MATLAB 实现

粒子群优化算法的代码如下。

```
%%-----粒子群优化算法-PSO.m-----%%
%%输入: nPop, Dim, UpB, LoB, maxIt, F_Obj
% nPop: 粒子数量
% Dim: 目标空间的维度
% UpB: 目标空间的上边界
% LoB: 目标空间的下边界
% maxIt: 算法的最大迭代次数
% F_Obj: 适应度函数接口
%%输出: Best, CNVG
% Best: 记录全部迭代完成后的最优位置 (Best.Pos) 和最优适应度值 (Best.Fit)
% CNVG: 记录每次迭代的最优适应度值, 用于绘制迭代过程中的适应度变化曲线
%%其他
% X: 种群结构体, 记录种群所有成员的位置 (X.Pos) 和当前位置对应的适应度值 (X.Fit)
%%-----%%
function [Best,CNVG]=PSO(nPop, Dim, UpB, LoB, maxIt, F_Obj)

disp('PSO is now tackling your problem')
tic % 记录运行时间
%% 初始化参数
Vmax=6;
wMax=0.9;
wMin=0.4;
c1=2;
c2=2;

%% 初始化粒子的速度
vel=zeros(nPop,Dim);
%% 初始化种群
X=initialization(nPop, Dim, UpB, LoB);
% 个体获得的最优适应度值和最优位置
for i=1:nPop
    X(i).BestP=zeros(1, Dim);
    X(i).BestF=inf;
end
% 种群获得的最优适应度值和最优位置
Best.Pos=zeros(1,Dim);
Best.Fit=inf;
CNVG=zeros(1,maxIt);

%% 开始迭代
for l=1:maxIt % 迭代次数
    % 检查边界
    for i=1:nPop
```

```

        X(i).Pos=min(X(i).Pos,UpB);
        X(i).Pos=max(X(i).Pos,LoB);
    end
    for i=1: nPop
        % 计算每个粒子的目标函数
        X(i).Fit=F_Obj(X(i).Pos);

        if X(i).BestF>X(i).Fit
            X(i).BestF=X(i).Fit;
            X(i).BestP=X(i).Pos;
        end

        if Best.Fit>X(i).Fit
            Best.Fit=X(i).Fit;
            Best.Pos=X(i).Pos;
        end
    end
    % 更新 PSO 的 w
    w=wMax-l*((wMax-wMin)/maxIt);
    % 更新粒子的速度和位置
    for i=1:nPop
        for j=1:Dim
            vel(i,j)=w*vel(i,j)+c1*rand()*(X(i).BestP(j)-X(i).Pos(j))+c2*rand()*(Best.Pos(j)-X(i).Pos(j));
            if(vel(i,j)>Vmax)
                vel(i,j)=Vmax;
            end
            if(vel(i,j)< -Vmax)
                vel(i,j)= -Vmax;
            end
            X(i).Pos(j)=X(i).Pos(j)+vel(i,j);
        end
    end
    CNVG(l)= Best.Fit;
end
toc
end

```

粒子群优化算法的种群初始化代码如下。

```

%%-----种群初始化函数-initialization.m-----%%
% nPop: 粒子数量
% Dim: 目标空间的维度
% UpB: 目标空间的上边界
% LoB: 目标空间的下边界
% X.Pos 为种群的初始化位置, X.Fit 为其适应度值
function X=initialization(nPop,Dim,UpB,LoB)
    for i=1:nPop
        X(i).Pos=rand(1,Dim).*(UpB-LoB)+LoB; % 初始化个体的位置
        X(i).Fit=inf; % 初始化个体的适应度值
    end
end

```

1.4 粒子群优化算法的应用案例

1.4.1 求解单峰函数极值问题

问题描述：计算函数 $f(x) = \sum_{i=1}^n x_i^2$ ($-100 \leq x_i \leq 100$) 的最小值，其中 x_i 的维度为 30。

以 $f(x_1, x_2)$ 为例，函数的搜索曲面如图 1-2 所示。

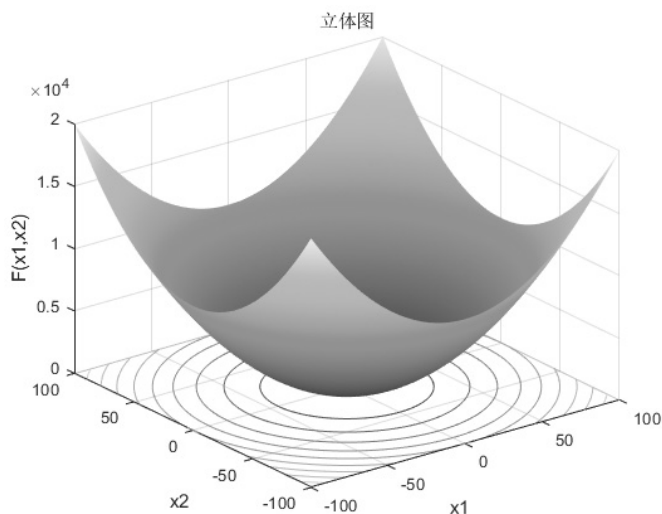


图 1-2 $f(x_1, x_2)$ 的搜索曲面

绘图代码如下。

```
%%-----绘制  $f(x_1, x_2)$  的搜索曲面-Fun_plot.m-----%%
x1 = -100:1:100;
x2 = -100:1:100;
L1 = length(x1);
L2 = length(x2);
for i = 1:L1
    for j = 1:L2
        f(i,j) = x1(i).^2+x2(j).^2;
    end
end
figure
surf(x1,x2,f,'LineStyle','none');           % 绘制搜索曲面
title('立体图');                           % 标题
xlabel('x1');                               % x 轴标注
ylabel('x2');                               % y 轴标注
zlabel('F(x1,x2)');                         % z 轴标注
```

函数 $f(x)$ 为单峰函数，意味着只有一个极值点，即当 $x=(0,0,\cdots,0)$ 时，产生理论最小值 $f(0,0,\cdots,0)=0$ 。通过仿真模拟，将该函数问题转换为适应度函数代码，具体代码如下。

```

%%-----适应度函数-fitness.m-----%%
function Fit = fitness(xi)
    %  $x_i$  为输入的一个个体，维度为[1,Dim]
    % Fit 为输出的适应度值
    Fit = sum(xi.^2);
end

```

其中， x_i 为输入的一个个体，适应度函数就是问题模型，Fit 为 Fun_Plot 函数的返回结果，称为适应度值。

假设粒子数量 nPop=30，目标空间的维度 Dim=30。仿真过程可以理解为将 30 个粒子随机放置在(-100,100)的目标空间中，粒子通过粒子群优化算法在有限的迭代次数（maxIt）内更新位置，并将每次更新的位置传入 Fun_Plot 函数获取适应度值，直到找到一个粒子的位置可以使 Fun_Plot 函数获得或接近理论最优解，即完成求解过程。求解该问题的主函数代码如下。

```

%%-----主函数-main.m-----%%
clc; % 清屏
clear all; % 清除所有变量
close all; % 关闭所有窗口
% 参数设置
nPop = 30; % 粒子数量
Dim = 30; % 目标空间的维度
UpB = 500; % 目标空间的上边界
LoB = -500; % 目标空间的下边界
maxIt = 50; % 算法的最大迭代次数
F_Obj = @(x)fitness(x); % 设置适应度函数
% 利用粒子群优化算法求解问题
[Best,CNVG] = PSO(nPop, Dim, UpB, LoB, maxIt,F_Obj);
% 绘制迭代曲线
figure
plot(CNVG,'r-', 'linewidth',2); % 绘制收敛曲线
axis tight; % 坐标轴显示范围为紧凑型
box on; % 加边框
grid on; % 添加网格
title('粒子群优化算法收敛曲线') % 添加标题
xlabel('迭代次数') % 添加 x 轴标注
ylabel('适应度值') % 添加 y 轴标注
disp(['求解得到的最优解为: ',num2str(Best.Pos)]);
disp(['最优解对应的函数值为: Fit=',num2str(Best.Fit)])

```

运行主函数代码（main.m），输出结果包括粒子群优化算法运行时间、迭代次数内获得的最优适应度值（最优解），以及最优解对应的参数值 x 。

```

PSO is now tackling your problem
历时 0.077771 秒。
求解得到的最优解为: 0.32507      -0.20194      -1.1672      0.48224      -0.74691
1.6895      -0.88796      0.46347      -1.0674      -2.964      -0.34373      2.2045
-0.53536      0.67817      -0.87304      0.040216      0.18783      2.6606      2.0384
0.58452      2.0494      0.84039      0.85256      2.184      -2.7893      0.31233

```

0.66858 2.4544 0.24293 0.94409

最优解对应的函数值为: Fit=59.883

将代码返回的 CNVG 绘制成图 1-3 所示的收敛曲线, 可以直观地看到每次迭代适应度值的变化情况。

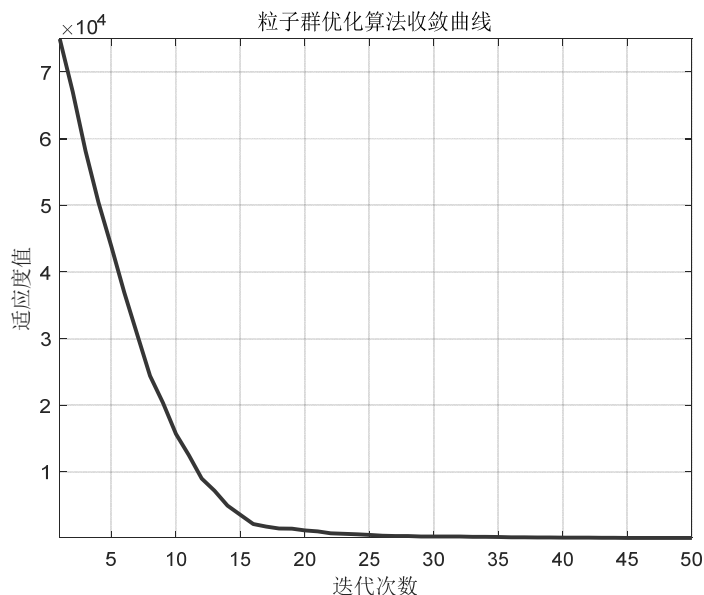


图 1-3 程序运行结果

1.4.2 求解多峰函数极值问题

问题描述: 计算函数 $f(x) = \sum_{i=1}^n -x_i \sin \sqrt{|x_i|}$ ($-500 \leq x_i \leq 500$) 的最小值, 其中 x_i 的维度为 30。以 $f(x_1, x_2)$ 为例, 函数的搜索曲面如图 1-4 所示。

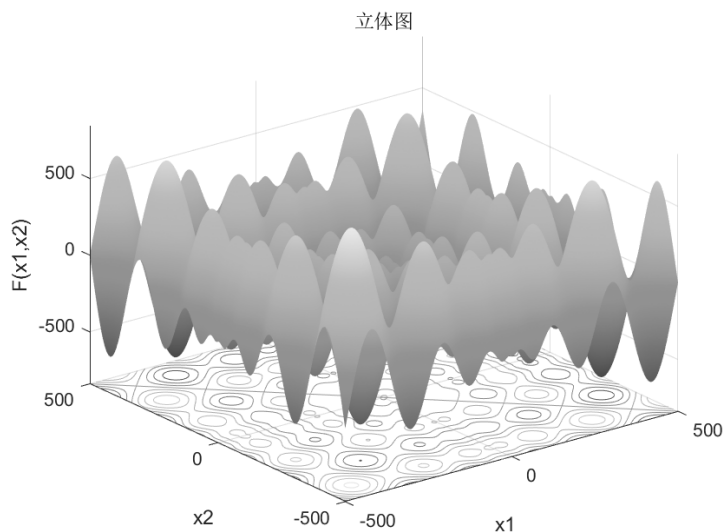


图 1-4 $f(x_1, x_2)$ 的搜索曲面

绘图代码如下。

```
%%-----绘制  $f(x_1, x_2)$  的搜索曲面-Fun_plot2.m-----%%
x1 = -500:1:500;
x2 = -500:1:500;
L1 = length(x1);
L2 = length(x2);
for i = 1:L1
    for j = 1:L2
        f(i,j) = -x1(i). * sin(sqrt(abs(x1(i)))) - x2(j). * sin(sqrt(abs(x2(j))));
    end
end
figure
surf(x1,x2,f,'LineStyle','none');           % 绘制搜索曲面
title('立体图');                             % 标题
xlabel('x1');                                 % x 轴标注
ylabel('x2');                                 % y 轴标注
zlabel('F(x1,x2)');                           % z 轴标注
```

函数 $f(x)$ 为多峰函数，意味着存在多个局部最优解，在仿真过程中容易陷入局部最优解而忽略全局最优解，该函数的理论最优解为 $-418.9829 \times \text{Dim} = -12569.487$ 。通过仿真模拟，将该函数问题转换为适应度函数代码，具体代码如下。

```
%%-----适应度函数-fitness2.m-----%%
function Fit = fitness2(xi)
    %  $x_i$  为输入的一个个体，维度为 [1,Dim]
    % Fit 为输出的适应度值
    Fit = sum(-xi.*sin(sqrt(abs(xi))));
end
```

假设粒子数量 $n\text{Pop}=30$ ，目标空间的维度 $\text{Dim}=30$ ，算法的最大迭代次数 $\text{maxIt}=50$ ，求解该问题的主函数代码如下。

```
%%-----主函数-main2.m-----%%
clc;                                           % 清屏
clear all;                                    % 清除所有变量
close all;                                    % 关闭所有窗口
% 参数设置
nPop = 30;                                    % 粒子数量
Dim = 30;                                     % 目标空间的维度
UpB = 500;                                    % 目标空间的上边界
LoB = -500;                                    % 目标空间的下边界
maxIt = 50;                                    % 算法的最大迭代次数
F_Obj = @(x)fitness2(x);                     % 设置适应度函数
% 利用粒子群优化算法求解问题
[Best,CNVG] = PSO(nPop,Dim,UpB,LoB,maxIt,F_Obj);
% 绘制迭代曲线
figure
plot(CNVG,'r-','linewidth',2);               % 绘制收敛曲线
```

```
axis tight; % 坐标轴显示范围为紧凑型
box on; % 加边框
grid on; % 添加网格
title('粒子群优化算法收敛曲线') % 添加标题
xlabel('迭代次数') % 添加 x 轴标注
ylabel('适应度值') % 添加 y 轴标注
disp(['求解得到的最优解为: ',num2str(Best.Pos)]);
disp(['最优解对应的函数值为: Fit=',num2str(Best.Fit)]);
```

运行主函数代码（main2.m），输出结果包括粒子群优化算法运行时间、迭代次数内获得的最优适应度值（最优解），以及最优解对应的参数值 x 。

```
PSO is now tackling your problem
历时 0.092645 秒。
求解得到的最优解为: 198.2975      383.9779      -58.81796      360.938      -73.80468
-112.391      -257.3094      295.7535      122.5256      201.2418      193.7742      -130.1101
-302.4958      364.8042      227.6558      337.1757      -250.0492      -341.2588      -264.2267
-311.6108      238.0601      -264.4758      63.09453      -306.1143      -299.9259      299.897
385.8314      -282.2044      170.1426      -66.69024
最优解对应的函数值为: Fit=-2730.991
```

将代码返回的 CNVG 绘制成图 1-5 所示的收敛曲线，可以直观地看到每次迭代适应度值的变化情况。

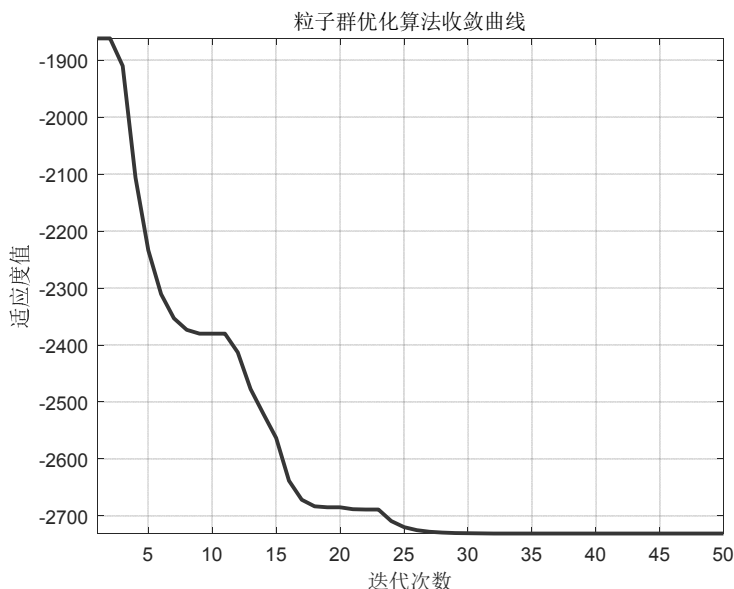


图 1-5 程序运行结果

1.4.3 拉力/压力弹簧设计问题

拉力/压力弹簧设计问题模型如图 1-6 所示。

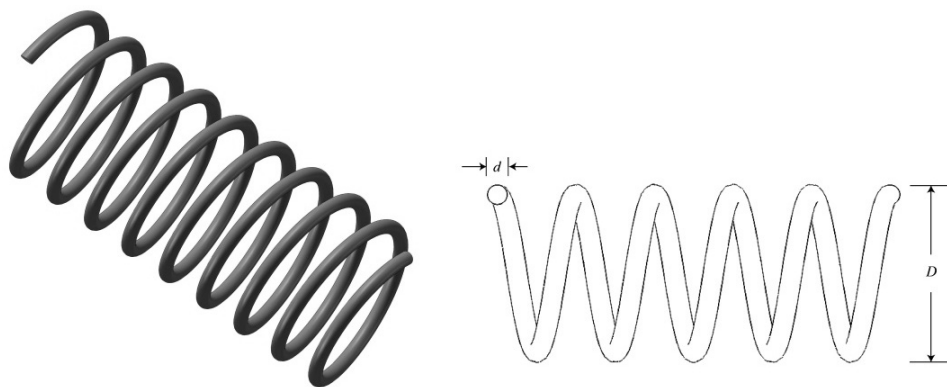


图 1-6 拉力/压力弹簧设计问题模型

问题描述：拉力/压力弹簧设计问题旨在通过优化算法找到弹簧直径（ d ）、平均线圈直径（ D ）以及有效线圈数（ N ）的最优值，并在最小偏差（ g_1 ）、剪切应力（ g_2 ）、冲击频率（ g_3 ）、外径限制（ g_4 ）4 种约束条件下降低弹簧的质量。

拉力/压力弹簧设计的数学模型描述如下。

设： $\mathbf{x} = [x_1 \ x_2 \ x_3] = [d \ D \ N]$

目标函数： $\min f(\mathbf{x}) = (x_3 + 2)x_2x_1^2$

约束条件：

$$g_1(\mathbf{x}) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \leq 0$$

$$g_2(\mathbf{x}) = \frac{4x_2^2 - x_1 x_2}{12566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} - 1 \leq 0$$

$$g_3(\mathbf{x}) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \leq 0$$

$$g_4(\mathbf{x}) = \frac{x_1 + x_2}{1.5} - 1 \leq 0$$

变量范围：

$$0.05 \leq x_1 \leq 2.00$$

$$0.25 \leq x_2 \leq 1.30$$

$$2.00 \leq x_3 \leq 15.00$$

拉力/压力弹簧设计问题的适应度函数代码如下。

```
%%-----适应度函数-fitness_Spring_Design.m-----%%
function Fit=fitness_Spring_Design(x)
% 惩罚系数
PCONST = 100000;
% 目标函数
Fit=(x(3)+2)*x(2)*(x(1)^2);
G1=1-((x(2)^3)*x(3))/(71785*x(1)^4);
```