《计算智能导论》

课程报告

（2020-2021 学年秋学期）

粒子群优化求解
Schaffer F6 函数优化问题

班级：09011804

学号：2018302068

姓名：赵敏琨

2020 年 12 月

一、问题描述

目标函数为 Schaffer F6 函数，其函数表达式如下：

$$f(x_1,x_2)=0.5+\frac{\sin^2\sqrt{x_1^2+x_2^2}-0.5}{(1+0.001(x_1^2+x_2^2))^2}$$

优化问题描述为：

$$\begin{aligned}&\min\ f(x_1,x_2)\\&s.t.\quad -100\le x_1\le 100\\&\qquad -100\le x_2\le 100\end{aligned}$$

连续函数的优化问题，可以用粒子群算法求解。

二、方法设计

1. 最小化问题，**适应函数取为目标函数**，编写目标函数 OptimOBJ(x)；
2. **绘制目标函数图像**，编写绘图函数 DrawOptimOBJ()；
3. 注意到优化问题带有约束条件，所以需要用带约束的粒子群优化算法解决，**采用“惩罚值”的办法来处理约束**，约束条件写在算法主文件中；
4. **粒子群初始化：**

- **随机生成初始种群位置 pop_x(i,j)**：须满足位置限制 $-200\le x_i\le 200,(i=1,2)$；
- **随机生成初始种群速度 pop_v(i,j)**：须满足速度限制 $-2\le v_i\le 2,(i=1,2)$；
- **采用全局邻域拓扑（星状拓扑）**：每个粒子可以与其他所有粒子通信，所有其他粒子都是该粒子的近邻；
- **初始化个体历史最佳位置（个体极值点）pbest 和个体历史最佳适应度（个体极值）fitness_pbest**：随机初始化后不满足约束个体的 fitness_pbest 置为 $\frac{PV}{10^8}=100$，这么做使得初始化种群时实际惩罚值较小，达到扩大搜索空间的目的；
- **初始化群体历史最佳位置（局部极值点）gbest 和群体历史最佳适应度（局部极值）fitness_gbest**：先在随机初始种群中比较一遍，得到随机初始种群的群体历史最佳位置；

5. **粒子群迭代更新：**

- **更新速度并对速度进行边界处理：**

速度更新公式为

$$v_i = c_1 \cdot \left(-\frac{1}{ger} \cdot iter + 1 \right) \cdot v_i + c_2 \cdot rand() \cdot (pbest_i - x_i) + c_3 \cdot rand() \cdot (gbest_i - x_i)$$

其中，c_2为自我学习因子，c_3为群体学习因子。c_1为惯性权重，ger为最大迭代次数，$iter$为当前迭代次数，相当于惯性权重因子叠加线性函数，使之随迭代次数增加而减小。因为初始阶段较大的惯性权重可以扩展算法在搜索空间中的探索范围；末尾阶段较小的惯性权重可以加强算法的局部搜索能力。

速度边界处理为

$$v_{ij} > V_{\max} \Rightarrow v_{ij} = V_{\max}$$
$$v_{ij} < V_{\min} \Rightarrow v_{ij} = V_{\min}$$

- **更新位置并对位置进行边界处理：**

位置更新公式为

$$x_i = x_i + v_i$$

位置边界处理为

$$x_{ij} > X_{\max} \Rightarrow x_{ij} = X_{\max}$$
$$x_{ij} < X_{\min} \Rightarrow x_{ij} = X_{\min}$$

- **进行约束条件判断并计算新种群各个个体的适应度：**

满足约束的个体适应度取为适应函数（目标函数）值，

不满足约束的个体适应度置为$\frac{PV}{1+1.05^{-iter+\frac{ger}{3}}}$，相当于进行适应值比例变换，迭代时原始惩罚值叠加类 Sigmoid 函数，使实际惩罚值随着代际增大而增大。因为迭代开始时，较小的惩罚值可以抑制竞争，扩大搜索范围；迭代后期时，较大的惩罚值可以鼓励竞争，加快收敛速度。

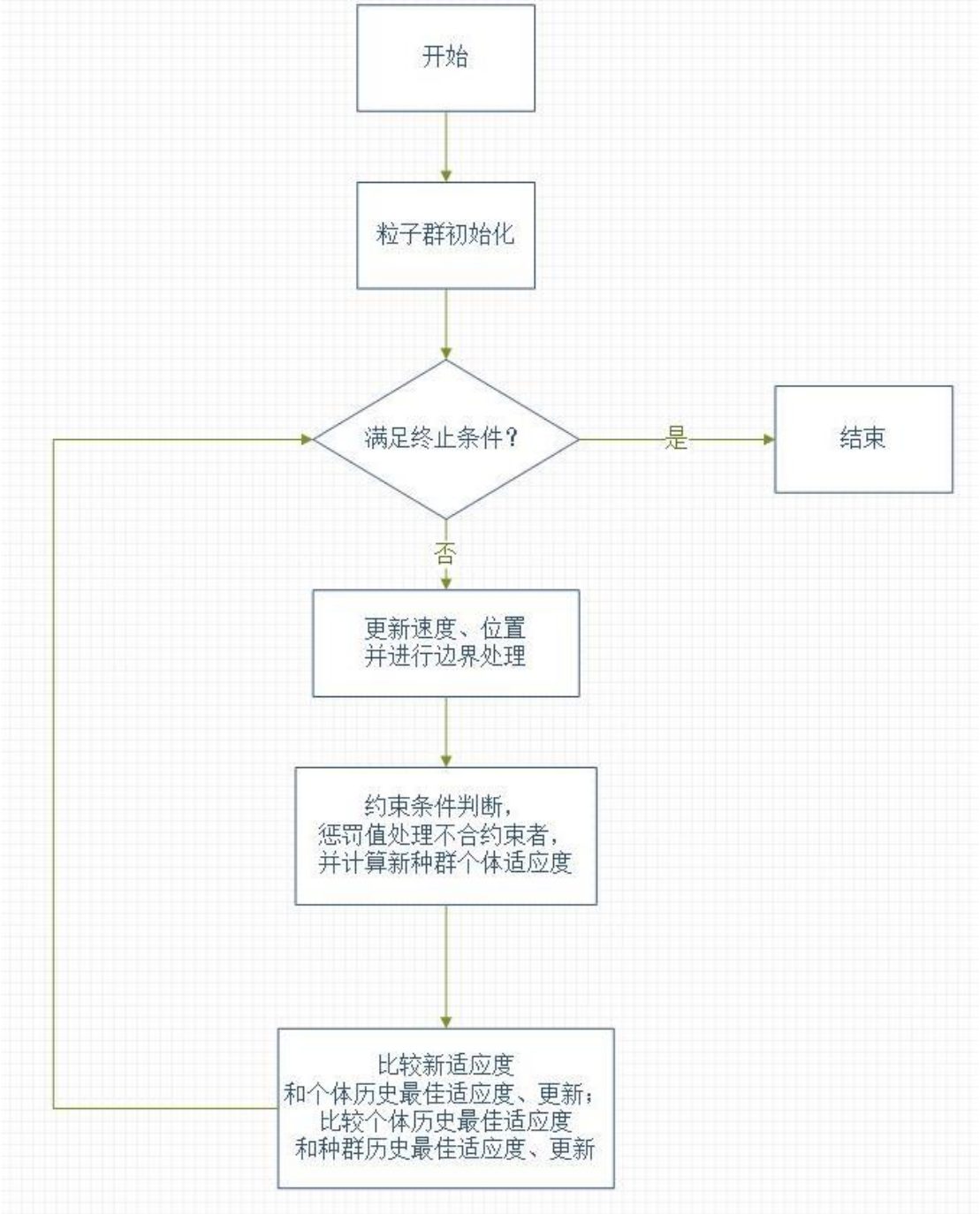
- **比较新适应度与个体历史最佳适应度**：最小化问题，如果更小则更新个体历史最佳适应度；
- **比较个体历史最佳适应度与种群历史最佳适应度**：最小化问题，如果更小则更新种群历史最佳适应度；
- **达到最大迭代次数，停止迭代**；

6. **迭代结果输出：**

- **输出迭代过程群体历史最佳适应度（局部极值）变化情况图**；
- **输出目标函数最优值与最优值点**；

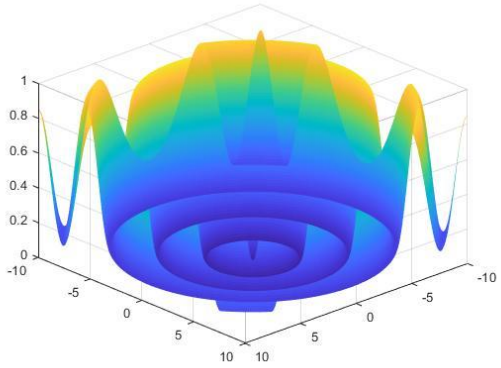
算法流程图如下：

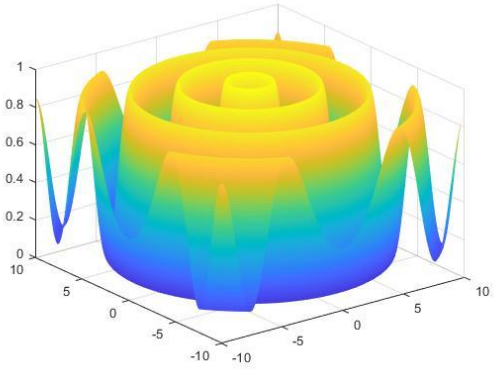
带约束的PSO算法流程图



三、实验分析

作目标函数图像得：





可以看出， $\min f(x_1, x_2)$ 的全局最优解在(0, 0)处，在全局最优解附近，有较多的局部最优解。

算法运行结果：

1. 第一组运行

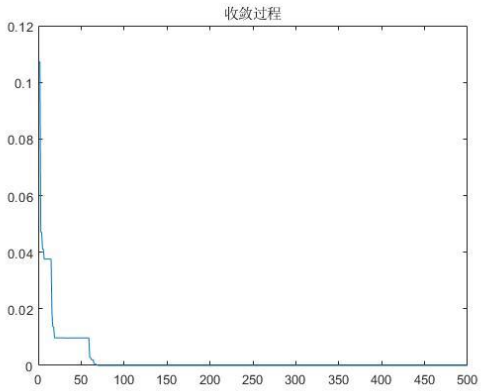
种群参数设置为：

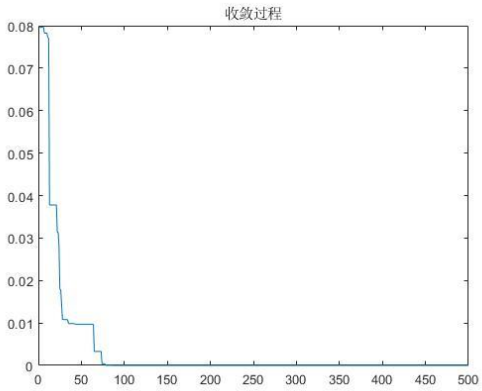
序号	初始种群个数	最大迭代次数	位置限制		速度限制		原始惯性权重	自我学习因子	群体学习因子
	sizepop	*ger*	X_{max}	X_{min}	V_{max}	V_{min}	c_1	c_2	c_3
1	**500**	500	200	-200	2	-2	0.5	2	2
2	**200**	500	200	-200	2	-2	0.5	2	2
3	**100**	500	200	-200	2	-2	0.5	2	2
4	**50**	500	200	-200	2	-2	0.5	2	2
5	**10**	500	200	-200	2	-2	0.5	2	2

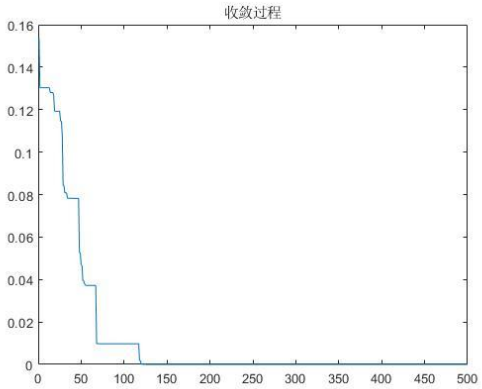
结果：

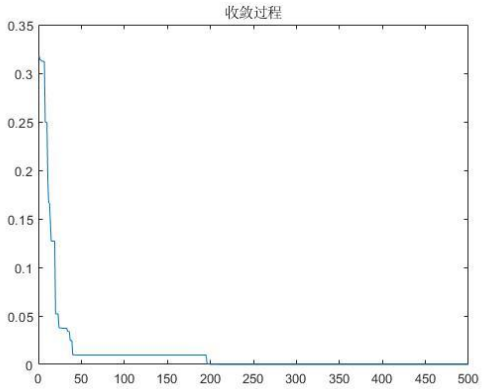
序号	目标函数值	变量取值	
	$f(x_1, x_2)$	x_1	x_2
1	0	-0.000000	0.000000
2	0	-0.000000	-0.000000
3	0	0.000000	0.000000
4	0	-0.000000	0.000000
5	0.00033385	-0.005411	0.017444

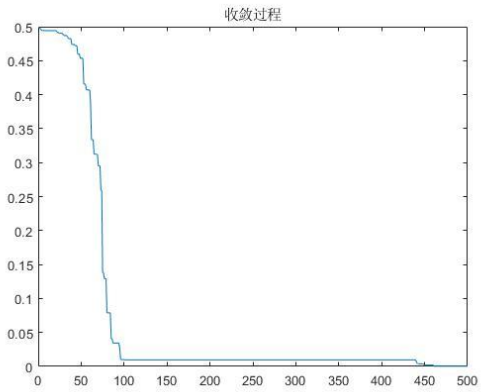
收敛过程：











由第一组的 5 次运行结果可以看出，初始种群个数越小，算法迭代收敛效果越差。综合试验结果，取初始种群个数为 $sizepop=100$；

2. 第二组运行

种群参数设置为：

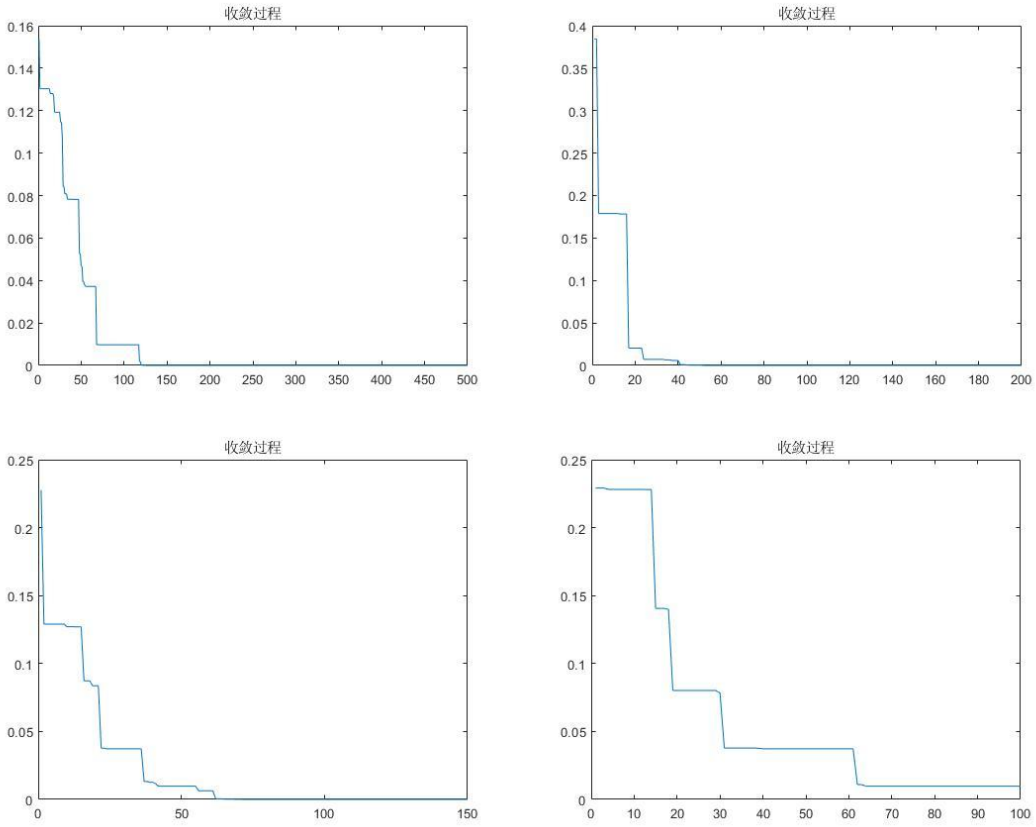
序号	初始种群个数	最大迭代次数	位置限制		速度限制		原始惯性权重	自我学习因子	群体学习因子
	sizepop	*ger*	X_{max}	X_{min}	V_{max}	V_{min}	c_1	c_2	c_3
1	100	**500**	200	-200	2	-2	0.5	2	2
2	100	**200**	200	-200	2	-2	0.5	2	2

3	100	**150**	200	-200	2	-2	0.5	2	2
4	100	**100**	200	-200	2	-2	0.5	2	2

结果：

序号	目标函数值	变量取值	
	$f(x_1,x_2)$	x_1	x_2
1	0	0.000000	0.000000
2	0	0.000000	-0.000000
3	0	-0.000000	-0.000000
4	0.0097159	-2.849679	-1.315073

收敛过程：



由第二组的 4 次运行结果可以看出，最大迭代次数越小，算法迭代收敛效果越差；第 4 次运行更是收敛到了局部最优解。综合试验结果，取最大迭代次数为 $ger = 200$；

3. 第三组运行

种群参数设置为：

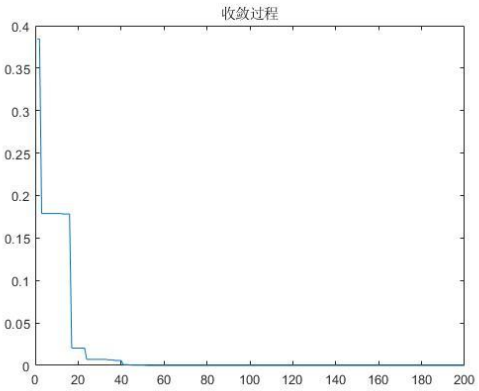
序号	初始种群个数	最大迭代次数	位置限制		速度限制		原始惯性权重	自我学习因子	群体学习因子
	sizepop	*ger*	X_{max}	X_{min}	V_{max}	V_{min}	c_1	c_2	c_3

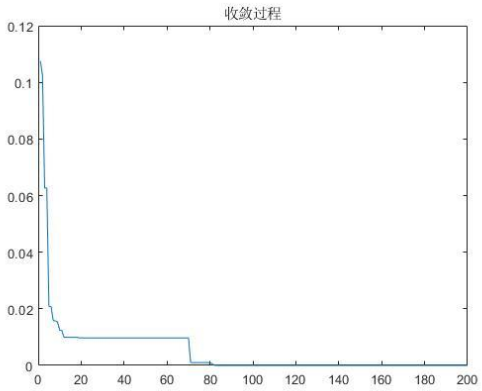
1	100	200	**200**	**-200**	**2**	**-2**	0.5	2	2
2	100	200	**100**	**-100**	**1**	**-1**	0.5	2	2

结果：

序号	目标函数值	变量取值	
	$f(x_1,x_2)$	x_1	x_2
1	0	0.000000	-0.000000
2	0	-0.000000	0.000000

收敛过程：





由第三组 2 次运行结果可以看出，第 1 次运行的收敛代数更早，可能是由于位置限制和速度限制相对宽松，而不满足约束条件个体直接赋予惩罚值，使得初始搜索空间更广，同时更容易也更快收敛到全局最优解。综合试验结果，取

$X_{\max}=200, X_{\min}=-200$
$V_{\max}=2, V_{\min}=-2$ 。

4. 第四组运行

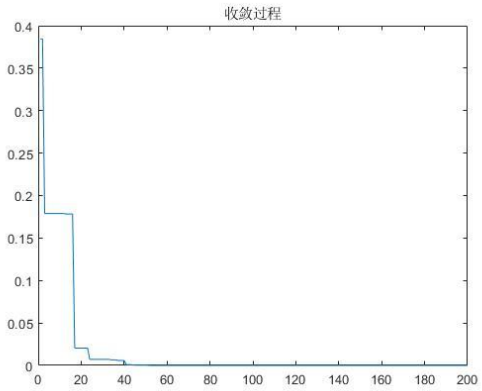
种群参数设置为：

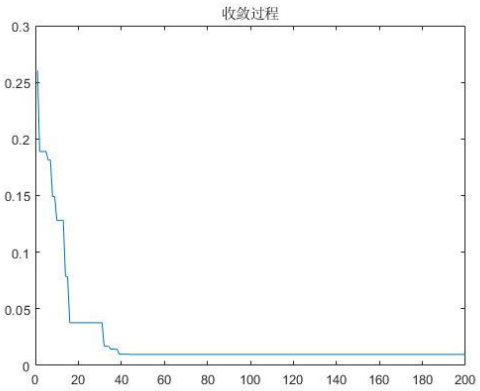
序号	初始种群个数	最大迭代次数	位置限制		速度限制		原始惯性权重	自我学习因子	群体学习因子
	sizepop	*ger*	X_{max}	X_{min}	V_{max}	V_{min}	c_1	c_2	c_3
1	100	200	200	-200	2	-2	**0.5**	2	2
2	100	200	200	-200	2	-2	**0.8**	2	2
3	100	200	200	-200	2	-2	**0.3**	2	2

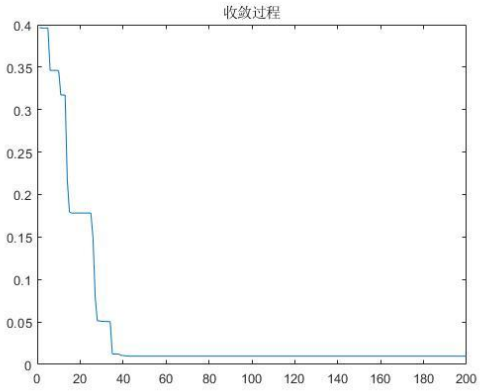
结果：

序号	目标函数值	变量取值	
	$f(x_1,x_2)$	x_1	x_2
1	0	0.000000	-0.000000
2	0.0097159	-0.921900	-3.000031
3	0.0097159	-3.117048	0.366191

收敛过程：







由第四组 3 次运行结果可以看出，原始惯性权重过大或过小都会导致算法收敛效果变差。综合试验结果，取原始惯性权重 $c_1 = 0.5$ 。

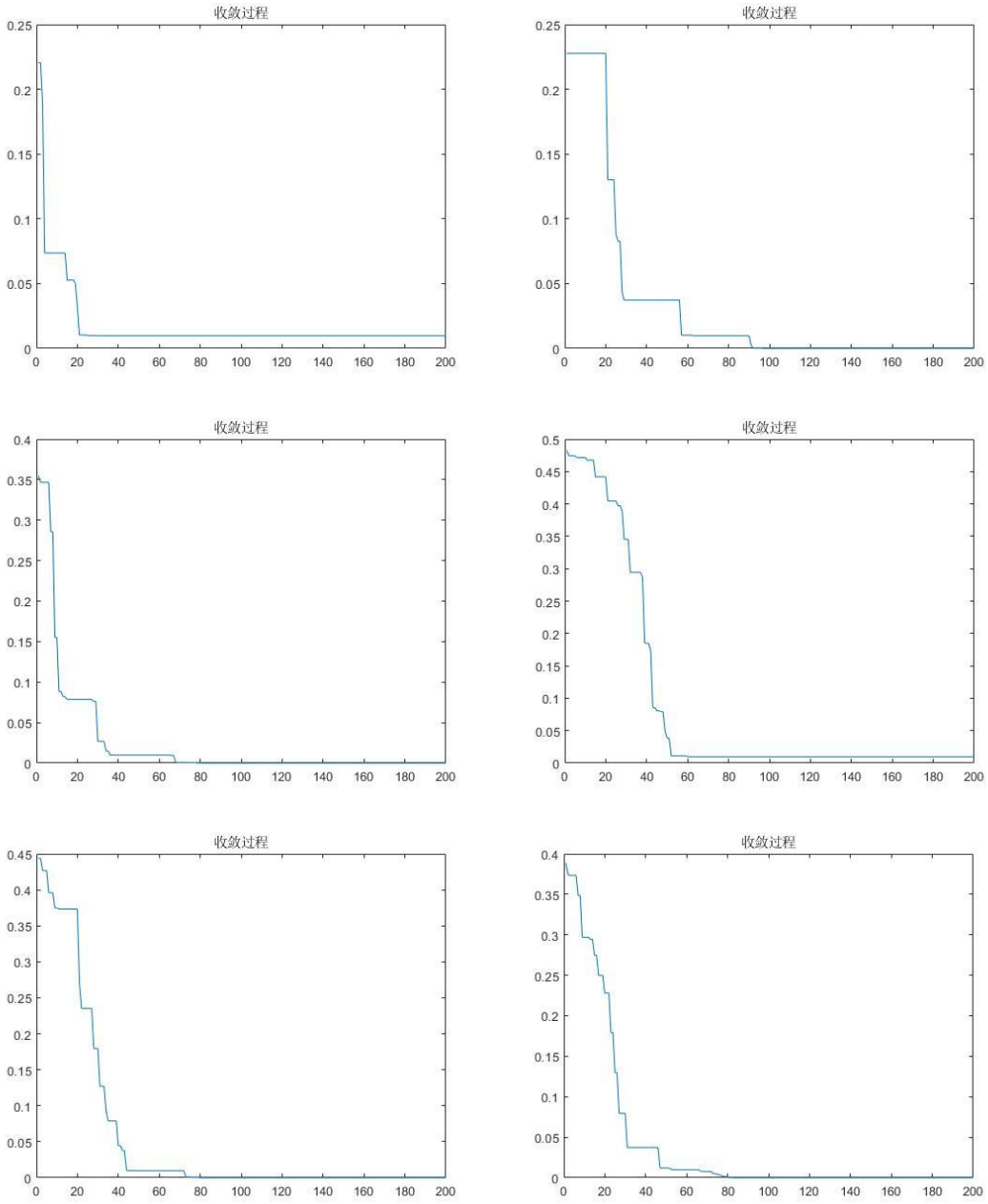
5. 第五组运行

种群参数设置为：

初始种群个数	最大迭代次数	位置限制		速度限制		原始惯性权重	自我学习因子	群体学习因子
sizepop	*ger*	X_{max}	X_{min}	V_{max}	V_{min}	c_1	c_2	c_3
100	200	200	-200	2	-2	0.5	**2**	**2**

结果：

序号	目标函数值	变量取值	
	$f(x_1, x_2)$	x_1	x_2
1	0.0097159	-1.249995	-2.878819
2	0	0.000000	-0.000000
3	0	0.000000	-0.000000
4	0.0097159	1.208008	2.896688
5	0	0.000000	-0.000000
6	0	-0.000000	0.000000



由第五组 6 次运行结果，$c_2=2, c_3=2$ 时收敛到全局最优解的比率为 $\frac{4}{6}\times 100\%=66.67\%$。

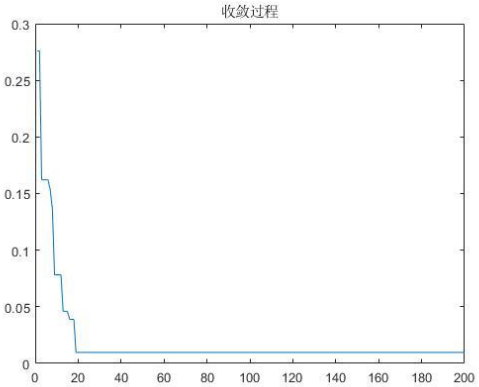
6. 第六组运行

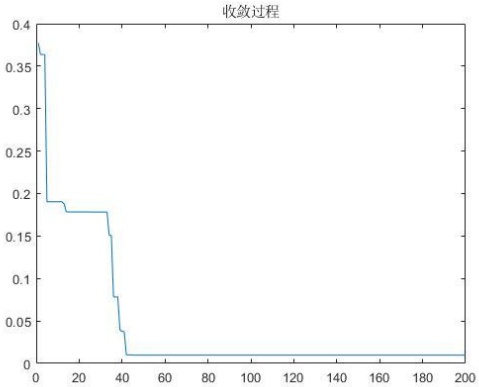
种群参数设置为：

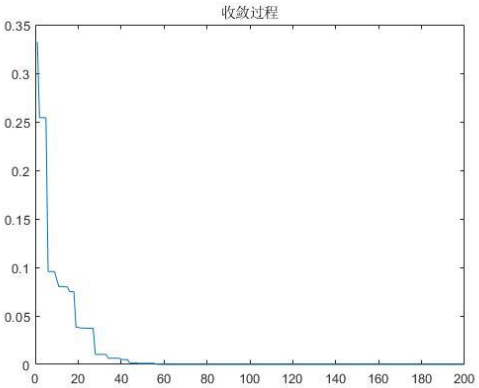
初始种群个数	最大迭代次数	位置限制		速度限制		原始惯性权重	自我学习因子	群体学习因子
sizepop	*ger*	X_{max}	X_{min}	V_{max}	V_{min}	c_1	c_2	c_3
100	200	200	-200	2	-2	0.5	**1.5**	**2.5**

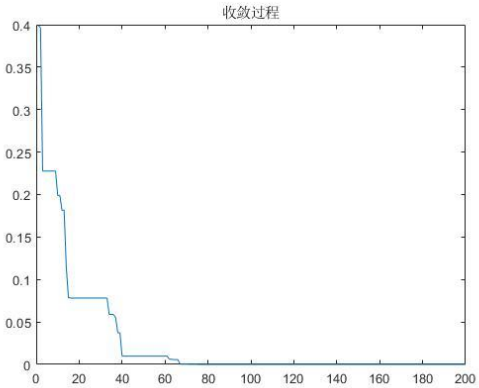
结果：

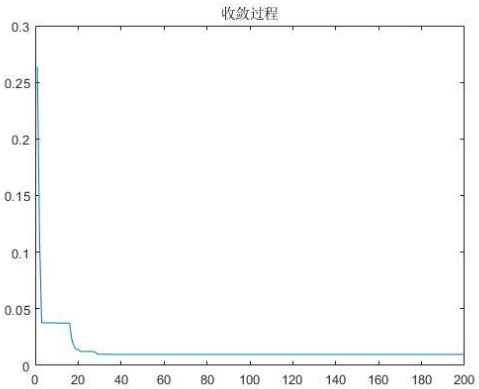
序号	目标函数值	变量取值	
	$f(x_1,x_2)$	x_1	x_2
1	0.0097159	2.492585	1.907120
2	0.0097159	2.282352	2.154288
3	0	0.000000	0.000000
4	0	-0.000000	-0.000000
5	0.0097159	2.274975	2.162077
6	0.0097159	-1.813240	2.561689

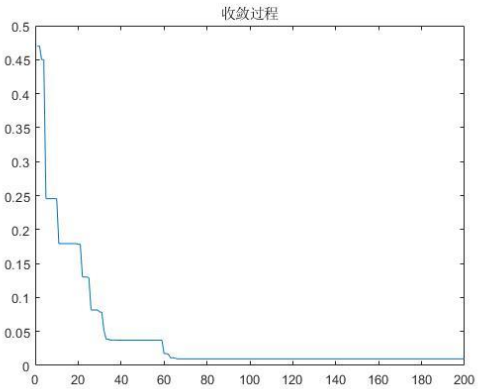












由第六组 6 次运行结果，$c_2=1.5, c_3=2.5$ 时收敛到全局最优解的比率为

$\frac{2}{6}\times 100\% = 33.33\%$ 。

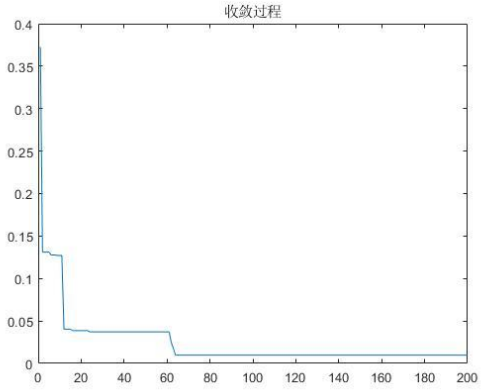
7. 第七组运行

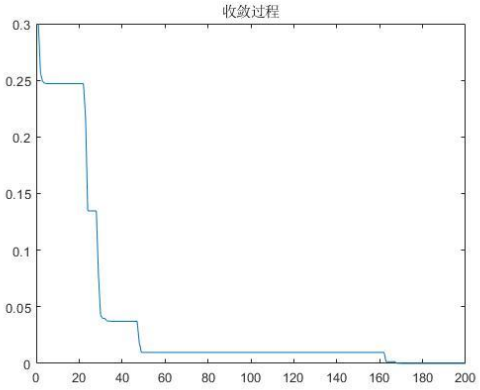
种群参数设置为：

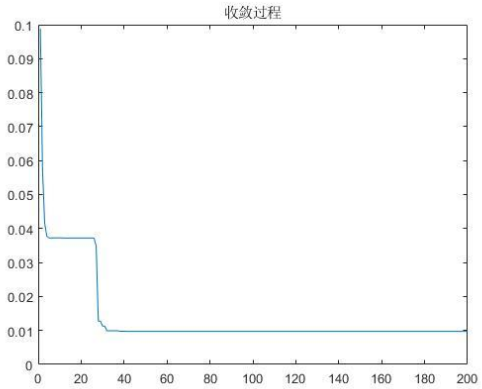
初始种群个数	最大迭代次数	位置限制		速度限制		原始惯性权重	自我学习因子	群体学习因子
sizepop	*ger*	X_{max}	X_{min}	V_{max}	V_{min}	c_1	c_2	c_3
100	200	200	-200	2	-2	0.5	**2.5**	**1.5**

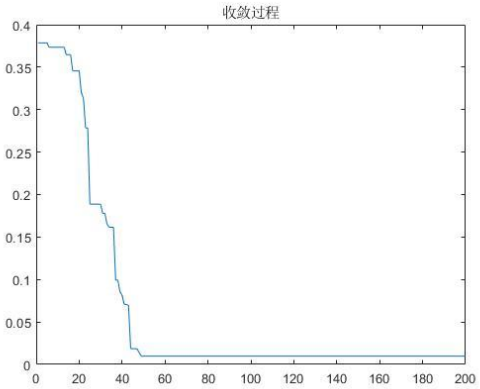
结果：

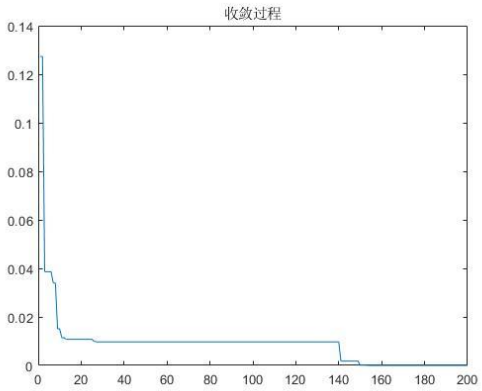
序号	目标函数值	变量取值	
	$f(x_1,x_2)$	x_1	x_2
1	0.0097159	2.139311	-2.296396
2	0	-0.000000	-0.000000
3	0.0097159	2.856536	-1.300111
4	0.0097159	-2.984244	0.971788
5	0	0.000000	-0.000000
6	0	-0.000000	-0.000000

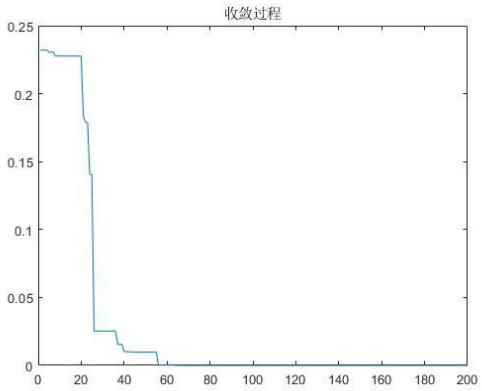












由第七组 6 次运行结果，$c_2=2.5, c_3=1.5$ 时收敛到全局最优解的比率为 $\frac{3}{6}\times 100\%=50.00\%$。

综合第五组、第六组、第七组试验，取自我学习因子 $c_2=2$，群体学习因子 $c_3=2$。

8. 所以最终整定的种群参数设置为：

初始种群个数	最大迭代次数	位置限制		速度限制		原始惯性权重	自我学习因子	群体学习因子
sizepop	*ger*	X_{max}	X_{min}	V_{max}	V_{min}	c_1	c_2	c_3
100	200	200	-200	2	-2	0.5	2	2

此时针对 Schaffer F6 函数优化问题，带约束的粒子群优化算法能较好地收敛到全局最优解。

附录：程序源代码

1. 目标函数文件 OptimOBJ.m

```
function objval = OptimOBJ(x)
[row,col]=size(x);
if row ~= 2
    error('输入的参数错误');
end
if col > 1
    error('输入的参数错误');
end
x1=x(1,1);
x2=x(2,1);
```

```
temp=(x1)^2+(x2)^2;
objval = 0.5 + ((sin(sqrt(temp)))^2-0.5) / (1+0.001*temp)^2;
```

2. 绘图函数文件 DrawOptimOBJ.m

```
function DrawOptimOBJ()
x=[-10:0.05:10];
y=x;
[X,Y]=meshgrid(x,y);
[row,col]=size(X);
for l=1:col
    for h=1:row
        z(h,l)=OptimOBJ([X(h,l);Y(h,l)]);
    end
end
surf(X,Y,z);
shading interp
```

3. 带约束的粒子群优化算法文件：PSO_with_Constraints.m

```
%% 清空环境
clc;clear;close all;
%% 作目标函数图像
 figure('Name','Schaffer F6 函数绘图');
 DrawOptimOBJ();
%% 目标函数与约束条件
%    objval = OptimOBJ(x)
%    nvars = 2   （变量个数）
%    约束条件为：
%    |x(1)| <= 100;
%    |x(2)| <= 100;
c1 = [1,0];
c2 = [0,1];
fun = @OptimOBJ;
cons1 = @(X)(abs(c1*X)<=100);
cons2 = @(X)(abs(c2*X)<=100);
%% 设置种群参数
dim = 2;                            % 空间维数=nvars
sizepop = 100;                      % 初始种群个数
ger = 200;                          % 最大迭代次数
xlimit_max = 200*ones(dim,1);       % 设置位置参数限制
xlimit_min = -200*ones(dim,1);
vlimit_max = 2*ones(dim,1);         % 设置速度限制
vlimit_min = -2*ones(dim,1);
```

```
c_1 = 0.5;                              % 原始惯性权重
c_2 = 2;                                % 自我学习因子
c_3 = 2;                                % 群体学习因子
PV = 10^10;                             % 原始惩罚值（Punishment Value）
%% 种群初始化
%  首先随机生成初始种群位置
%  然后随机生成初始种群速度
%  然后初始化个体历史最佳位置，以及个体历史最佳适应度
%  然后初始化群体历史最佳位置，以及群体历史最佳适应度
for i=1:dim
    for j=1:sizepop
        pop_x(i,j) = xlimit_min(i)+(xlimit_max(i) - xlimit_min(i))*rand;  % 初始种群
的位置
        pop_v(i,j) = vlimit_min(i)+(vlimit_max(i) - vlimit_min(i))*rand;  % 初始种群
的速度
    end
end
pbest = pop_x;                                  % 每个个体的历史最佳位置
for j=1:sizepop
    if cons1(pop_x(:,j))
        if cons2(pop_x(:,j))
            fitness_pbest(j) = fun(pop_x(:,j));     % 每个个体的历史最佳适应度
        else
            fitness_pbest(j) = PV/10^8;
        end
    else
        fitness_pbest(j) = PV/10^8;
    end
end
% 初始化种群时实际惩罚值较小，达到扩大搜索空间的目的
gbest = pop_x(:,1);                             % 种群的历史最佳位置
fitness_gbest = fitness_pbest(1);               % 种群的历史最佳适应度
for j=1:sizepop
    if fitness_pbest(j) < fitness_gbest         % 如果求最小值，则为<; 如果求最大值，
则为>;
        gbest = pop_x(:,j);                     % 记录种群的历史最佳位置
        fitness_gbest=fitness_pbest(j);
    end
end


%% 粒子群迭代更新
```

```
%     更新速度并对速度进行边界处理
%     更新位置并对位置进行边界处理
%     进行约束条件判断并计算新种群各个个体的适应度
%     新适应度与个体历史最佳适应度做比较
%     个体历史最佳适应度与种群历史最佳适应度做比较
%     再次循环或结束

iter = 1;                          %迭代次数
record = zeros(ger, 1);            % 记录器
while iter <= ger
    for j=1:sizepop
        %    更新速度并对速度进行边界处理
        %    原始惯性权重因子叠加线性函数，使实际惯性权重随迭代次数增加而减小
        pop_v(:,j)= c_1*(-1/ger*iter+1) * pop_v(:,j) +
c_2*rand*(pbest(:,j)-pop_x(:,j))+c_3*rand*(gbest-pop_x(:,j));% 速度更新
        for i=1:dim
            if  pop_v(i,j) > vlimit_max(i)
                pop_v(i,j) = vlimit_max(i);
            end
            if  pop_v(i,j) < vlimit_min(i)
                pop_v(i,j) = vlimit_min(i);
            end
        end

        %    更新位置并对位置进行边界处理
        pop_x(:,j) = pop_x(:,j) + pop_v(:,j);% 位置更新
        for i=1:dim
            if  pop_x(i,j) > xlimit_max(i)
                pop_x(i,j) = xlimit_max(i);
            end
            if  pop_x(i,j) < xlimit_min(i)
                pop_x(i,j) = xlimit_min(i);
            end
        end

        %    进行约束条件判断并计算新种群各个个体的适应度
        if cons1(pop_x(:,j))
            if cons2(pop_x(:,j))
                fitness_pop(j) = fun(pop_x(:,j));    % 当前个体的适应度
            else
                fitness_pop(j) = PV/(1+1.05^(-iter+ger/3));
            end
```

```
        else
            fitness_pop(j) = PV/(1+1.05^(-iter+ger/3));
        end
        % 进行适应值比例变换：
        % 迭代时原始惩罚值叠加类Sigmoid函数，相当于实际惩罚值随着代际增大而增大。
        % 迭代开始时，实际惩罚值小，抑制竞争，扩大搜索范围；
        % 迭代后期时，实际惩罚值大，鼓励竞争，加快收敛速度。

        %    新适应度与个体历史最佳适应度做比较
        if fitness_pop(j) < fitness_pbest(j)        % 如果求最小值，则为<; 如果求最大值，则为>;
            pbest(:,j) = pop_x(:,j);                % 更新个体历史最佳位置
            fitness_pbest(j) = fitness_pop(j);      % 更新个体历史最佳适应度
        end

        %    个体历史最佳适应度与种群历史最佳适应度做比较
        if fitness_pbest(j) < fitness_gbest         % 如果求最小值，则为<; 如果求最大值，则为>;
            gbest = pbest(:,j);                     % 更新群体历史最佳位置
            fitness_gbest=fitness_pbest(j);         % 更新群体历史最佳适应度
        end
    end

    record(iter) = fitness_gbest;                   % 最小值记录

    iter = iter+1;

end
%% 迭代结果输出
figure('Name','粒子群优化算法迭代过程');
plot(record);title('收敛过程'); hold on;
disp(['最优值：',num2str(fitness_gbest)]);
disp('变量取值：');
fprintf('%.6f\t',gbest);
```