大连理工大学机器学习课程

遗传算法求解优化问题

Optimization problems via Genetic Algorithm

学　院（系）：经济管理学院

专　　业：信息管理与信息系统

学 生 姓 名：韩可可 姜玥 刘一诺 高宇

学　　号：201605060 201605057

201605017 201605023

指 导 教 师：李先能

完 成 日 期：2019.6.30

大连理工大学

Dalian University of Technology

目录

1 工作概述

1.1 工作目标

使用 Python 编程，实现遗传算法，并求解 6 个 Benchmark 优化问题。

1.2 工作内容

团队首先进行遗传算法理论的学习与原理理解，应用 PYTHON 进行编程，实现遗传算法，对各个函数板块进行封装。包括：算法初始化，对二进制数据进行解码，设定列表格式的群体以及参数的上界和下界，计算适应度（其适应度越大越好），采用竞争选择和轮盘赌两个方法进行选择，并指定的交叉方式为单位点交叉，变异时根据变异的概率随机选择基因进行取反操作，遍历所有个体的适应度，保留其中最优个体来进行进化。

其次，我们将封装好的函数进行实现，使用传入参数列表计算目标函数，保证其维度 D>10，并通过主函数来调用各种封装函数，迭代次数为 5000，但会发现其基本在 100 代内能收敛到最优值。

接着，结合函数以及 for 循环可实现调参，并将其输出为 dataframe 和折线图的形式，最终得到最优参数。

最后，利用另一种演化计算的方法——粒子群算法，求解上述的 6 个优化问题，并于遗传算法进行比较。

2 主要代码解释

2.1 编码函数

编码函数的大致思路如下：通过列表存储群体（包含大量个体），同时每个个体用列表的形式存储染色体信息，一个个体有多个染色体，每个染色体的是由二进制形式表示。染色体通过随机生成的基因片段组成，整个过程通过三个 for 循环语句便可以实现。

编码函数参数列表及其含义：population_size 种群大小，chrom_num 每个个体包含的染色体个数，chrom_length 染色体长度，seed 随机数种子可以保证结果可以复现。

def encoding_f(population_size,chrom_num,chrom_length,seed=12345):

2.2 解码函数

解码函数的大致思路如下：通过将二进制转化为十进制，利用精度、参数上下界来确定每个染色体的具体数值,主要通过一个循环语句以及计算公式实现。

主要的计算公式为：t = [round(lower_bound + i * (upper_bound - lower_bound) /(math.pow(2, len(individuals_list[0])) - 1),ndigits=ndigits) for i in t

编码函数参数列表及其含义：population_init 群体，列表格式，lower_bound:参数下界，upper_bound:参数上界，accuracy:参数精度。

def decoding_f(population_list, lower_bound, upper_bound, accuracy):

2.3 适应度函数

这里包括如何计算二进制应该覆盖的范围，根据参数范围和精度要求进行求解，主要利用 binary_range = 1/accuracy*(upper_bound-lower_bound)和 for 循环实现。

适应度函数的表示通过调用解码函数实现，为了方便后面轮盘选择，把适应度全部转换为正数，方法所有元素-Min(适应度)+1。

适应度函数参数列表及其含义：population_list 群体列表 object_fun:目标函数，lower_bound 参数范围下界，upper_bound 参数范围下界 accuracy:精度，即参数精确到小数点之后几位。

def fitness_f(population_list,object_fun,lower_bound,upper_bound, accuracy):

2.4 选择函数

选择函数分为两个函数，即竞争选择和轮盘赌选择，竞争选择的大致思路如下：random.sample()函数进行随机抽取 10 个样本，max()函数在 10 个样本中选择最大的一个当作父代；轮盘赌选择的大致思路如下：轮盘赌选择又称为概率选择，即适应度/sum（适应度）的形式计算概率，利用 for 循环+判断进行带概率的选择。

编码函数参数列表及其含义：population_list 种群列表，population_fit_list 适应度数值，choose_type:选择方式，目前支持竞争(tournament)和轮盘赌(wheel)，其中竞争抽取的样本数为 10。

def choose_f(population_list,population_fit_list, choose_type):

2.5 交叉函数

染色体交叉的方法为单点交叉。遍历个体所有的染色体组，判断是否交叉，具体为取出一组染色体，如果两个染色体不一样，并且随机数小于指定的概率，则进行交叉。随后，在每个染色体上随机选择一个位点，进行交换即可。

交叉函数参数列表及其含义：individual1 要交叉的个体，individual2 要交叉的个体，cross_prob 交叉的概率。

def cross_f(individual1,individual2, cross_prob):

2.6 变异概率

变异方法为二进制变异。生成随机数 p，用来判断是否变异，如果 p 小于指定变异

概率，那么对这个个体进行变异，随机选择一个基因或者位点，如果该基因为 0 则变异为 1，如果为 1 变异为 0。

交叉函数参数列表及其含义：individual:个体，list 格式，需要遍历，mutate_prob:变异概率。

def mutate(individual, mutate_prob):

2.7 保留最优个体

这里写了 keep_best_individual_f 函数，for 循环遍历了所有个体的适应度，判断下一代适应度是否更高，适应度越大，越可能为最优解，最终保留适应度最大的个体信息。

该函数参数列表及其含义：best_individual:字典类型，分别取个体，适应度，代际 ，fitness_list 适应度，变异之后重新计算，generation 代际数据输入，表示最优个体出现在哪一代 。

def keep_best_individual_f (best_individual,fitness_list,population_list,generation):

2.8 进化

调用以上各个函数，进化的逻辑是不停的选择、交叉、变异，最终生成最后的样本。输入群体、进化代数、交叉概率、变异概率，根据种群大小，进化(population_size-1)/2 次，先计算上一代群体的适应度，进行不停的选择、交叉、变异，对个体进行比较输出最终个体和群体。

该函数参数列表及其含义: population_list 上一代群体,cross_prob 交叉概率，mutate_prob 变异概率，population_fit_list 适应度集合，choose_type 选择的方式，object_fun 目标函数，lower_bound 参数下界，upper_bound 参数上界，accuracy 精度，best_individual 最优的个体，choose_type 选择的方法，generation 代际，表示这是第几代的进化。

def evolve_f(population_list,cross_prob,mutate_prob,choose_type,
object_fun,lower_bound,upper_bound,best_individual,generation,accuracy):

2.9 主函数

- 初始化群体，设置相关参数：

确认染色体长度、进行初始化、设置代际、迭代次数、群体大小、每个个体染色体条数、交叉概率、变异概率、适应度集合、选择方式、目标函数、参数上下界、精度。

- 不停迭代和演化，直至达到指定代数：

计算适应度、演化迭代并输出结果。

2.10 公式求解

1) 改写公式为-sum([i*i for i in parm_list])，调用上述函数，设置相应参数，传入参数列表，计算目标函数；

2) 改写公式为-sum([(i+1)*parm_list[i]**2 for i in range(len(parm_list))])，调用上述函数，传入参数列表，计算目标函数；

3) 改写公式为-sum([math.floor(i+0.5)**2 for i in parm_list])，用到了向下取整（比自己小的最大整数），调用上述函数，传入参数列表，计算目标函数；

4) 改写公式为-sum([(i+1)*parm_list[i]**4 for i in range(len(parm_list))])-random_value，调用上述函数，传入参数列表，这里要多设置一个 random 变量，计算目标函数；

5) 改写公式为 math.cos(x1)*math.cos(x2)*math.exp(-(x1-math.pi)**2-(x2-math.pi)**2)，调用上述函数，传入参数列表，计算目标函数；

6) 改写公式为-(0.26*sum([i**2 for i in parm_list])-0.48*parm_list[0]*parm_list[1])，调用上述函数，传入参数列表，计算目标函数；

3 调参结果分析与对比

由大作业内容可知，需要调整的参数依次是：Population size（种群大小）、不同的二进制长度、不同的 crossover 和 mutation probability（交叉率和变异率）、不同的选择方式（轮盘赌选择、竞争选择），我们通过调用主函数以及 for 循环语句完成参数的调整。

其中，根据经验以及文献，我们依次给出每个变量的调参范围以及调参间隔：Population size（种群大小）在[100,2000]之间，每隔 50 为一个单位；交叉率在[0.6,0.99]之间，每 0.5 一个单位；变异率在[0.01-0.2]之间，每 0.01 一个单位；选择方式为轮盘赌选择和竞争选择；二进制长度利用精度来进行调整参数，范围在[0.001,0.0001]之间。此外，都一些固定参数，即迭代次数为 5000，维度 D 为 10。

调参思路如下：

- 通过 def object_fun(parm_list)函数设置各目标函数，其中 parm_list 传入参数列表，用来计算目标函数，参数为十进制的真实值，最终返回各个目标函数。
- 设置一些固定参数，它们依次是：generation_size=5000，population_size=100，chrom_num = 10，cross_prob=0.9，mutate_prob = 0.1，choose_type='tournament' or choose_type='wheel'，object_fun=object_fun，lower_bound=-100，upper_bound=100，accuracy=0.001 ，is_print=False。
- 利用 for 循环以及调用主函数实行参数调整循环，输出每次调参的最终结果以及误差。
- 将上述结果转化为 dataframe 数据框和折线图的形式进行可视化。

3.1 函数一

$$f_1(x) = \sum_{i=1}^{D} x_i^2 \qquad x\epsilon[-100,100]$$

✧ population_size，调参范围[100,2000]，每 50 为一个 step。

表 3.1 第一个函数种群大小迭代情况

	tune_parma	result	generation	param
0	100	2081.351566	195	[0.001, -1.472, 0.0, -32.605, 0.001, -23.112, ...
1	150	1230.585380	4950	[-25.953, -12.716, -4.692, 0.056, 0.001, 0.004...
2	200	534.323000	150	[0.001, -12.819, 0.0, -0.603, 0.0, -3.977, -18...
3	250	12.847784	335	[0.0, -0.106, 0.001, -0.476, 0.0, -0.239, -3.5...
4	300	11.333621	996	[0.001, -0.593, 0.0, -2.735, 0.0, -0.977, -1.5...
5	350	0.156583	97	[0.0, 0.0, 0.001, 0.0, 0.001, 0.001, 0.0, -0.3...
6	400	40.259096	126	[0.0, -0.565, 0.001, -0.415, 0.0, -6.302, -0.2...
7	450	0.081763	360	[0.001, 0.0, -0.06, -0.196, -0.199, -0.012, 0....

8	500	259.737566	994	[0.001, -0.403, 0.0, -0.013, 0.001, -3.516, -1...
9	550	825.497470	325	[0.0, -12.503, 0.001, -25.098, 0.0, -6.266, -0...
10	600	51.326147	562	[0.0, 0.001, -0.098, 0.0, 0.0, 0.001, 0.0, 0.0...
11	650	0.009610	1089	[0.0, 0.001, -0.0, 0.0, 0.001, 0.0, 0.0, 0.0, ...
12	700	41.026950	160	[0.0, 0.0, -6.257, 0.001, 0.0, 0.0, 0.0, 0.0, ...
13	750	0.022007	323	[0.0, 0.001, -0.1, 0.001, 0.0, 0.0, 0.0, 0.0, ...
14	800	14.961475	81	[0.0, 0.0, 0.001, -0.007, 0.0, -3.516, 0.0, 0....
15	850	0.842609	361	[0.001, 0.001, 0.0, -0.004, 0.0, -0.392, 0.001...
16	900	0.621131	43	[0.0, 0.001, 0.0, -0.001, 0.001, -0.782, 0.0, ...
17	950	3.818118	769	[0.0, 0.001, -0.0, 0.0, 0.0, 0.0, 0.001, 0.0, ...
18	1000	0.611524	45	[0.0, 0.0, 0.0, -0.0, 0.0, -0.782, 0.0, 0.0, -...
19	1050	0.000002	2261	[0.0, 0.0, -0.0, 0.0, 0.0, 0.001, 0.0, 0.001, ...
20	1100	0.671411	128	[0.0, 0.001, 0.0, -0.202, 0.001, -0.794, 0.0, ...
21	1150	39.062518	246	[0.001, 0.0, -6.25, 0.001, 0.0, 0.0, 0.0, 0.0,...
22	1200	0.000008	42	[0.0, 0.0, -0.002, 0.0, 0.001, 0.0, 0.0, 0.001...
23	1250	2.443139	155	[0.0, 0.0, 0.001, -1.563, 0.0, -0.0, 0.0, 0.0,...
24	1300	0.000001	45	[0.0, 0.001, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0...
25	1350	0.611526	41	[0.0, 0.001, 0.0, -0.0, 0.001, -0.782, 0.0, 0....
26	1400	0.000002	39	[0.0, 0.001, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0...
27	1450	0.000001	40	[0.0, 0.0, -0.0, 0.0, 0.0, 0.001, 0.0, 0.0, -0...
28	1500	0.000008	146	[0.0, 0.0, -0.002, 0.0, 0.0, 0.0, 0.0, 0.0, -0...
29	1550	0.000000	42	[0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.0...
30	1600	0.000000	42	[0.0, -0.0, -0.0, -0.0, -0.0, 0.0, -0.0, 0.0, ...
31	1650	0.000001	42	[0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.0...
32	1700	0.039042	47	[0.0, -0.001, -0.025, -0.0, -0.0, 0.0, -0.196,...
33	1750	0.000001	42	[-0.0, -0.001, -0.0, -0.0, 0.0, 0.0, -0.0, 0.0...
34	1800	0.000001	46	[0.0, -0.0, -0.0, -0.0, -0.0, 0.0, -0.001, 0.0...
35	1850	0.152881	59	[0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.3...
36	1900	0.000002	45	[0.0, -0.0, -0.001, -0.001, -0.0, 0.0, -0.0, 0...
37	1950	0.000004	46	[-0.0, 0.002, -0.0, 0.0, -0.0, -0.0, 0.0, 0.0,...

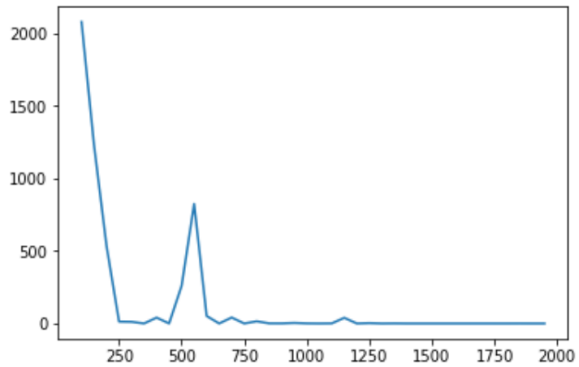

图 3.1 第一个函数种群大小迭代情况

由图表可知，当种群大小在 1250 之前时，每次寻找到的最小值波动较大，一方面是因为种群大小过小，无法保证找出的父代较强，另一方面，遗传算法同样存在局部最优的情况，但当种群数量为 1300 时，实现了精度在 0.01 以上的最小值，因此，种群数量的大小为 1300。

✧ cross_prob 交叉率，调参范围[0.6,0.99]，每 0.05 是一个 step。

表 3.2 第一个函数交叉率迭代情况

	tune_parma	result	generation	param
0	0.60	2.606004	178	[0.0, 0.0, 0.001, -0.025, 0.0, -0.0, 0.0, 0.0,...
1	0.65	2.771057	190	[0.0, 0.0, 0.0, -1.661, 0.0, -0.11, 0.0, 0.0, ...
2	0.70	2.453534	376	[0.0, 0.0, -0.098, 0.0, 0.0, 0.0, 0.0, 0.0, -1...
3	0.75	0.000170	1942	[0.0, 0.0, 0.0, -0.0, 0.0, -0.013, 0.0, 0.001,...
4	0.80	0.000270	361	[0.0, 0.001, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0...
5	0.85	2.482012	150	[0.0, 0.001, 0.0, -0.196, 0.0, 0.0, 0.001, -0....
6	0.90	0.000001	45	[0.0, 0.001, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0...
7	0.95	0.000001	213	[0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, 0.001, -0...

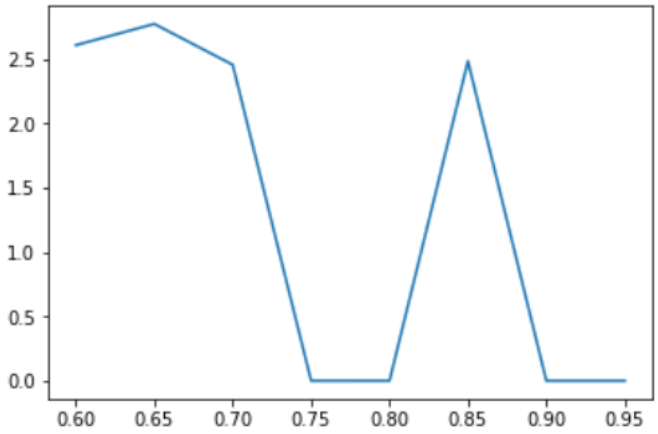

图 3.2 第一个函数交叉率迭代情况

由图表可知，当交叉率在 0.9 之前时，每次寻找到的最小值波动较大，这可能是因为交叉率过小，个体的变化不太，仍然存在较弱的基因，但当变异率为 0.9 时，实现了精度在 0.01 以上的最小值，因此，交叉率的大小为 0.9 即可。

✧ mutate_prob 变异率，调参范围[0.001,0.2]，每 0.02 是一个 step。

表 3.3 第一个函数变异率迭代情况

	tune_parma	result	generation	param
0	0.20	0.000046	48	[0.0, -0.0, 0.003, 0.0, -0.0, -0.0, 0.006, -0....

1	0.18	0.157623	46	[0.0, 0.0, 0.0, -0.397, 0.0, -0.002, 0.0, 0.0,...
2	0.16	0.000000	1130	[0.0, 0.0, 0.0, -0.0, 0.0, -0.0, 0.0, 0.0, -0....
3	0.14	0.000038	66	[0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.001, 0.001, ...
4	0.12	0.000009	303	[0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.0...
5	0.10	0.000001	45	[0.0, 0.001, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0...
6	0.08	0.000177	43	[0.0, 0.001, 0.001, -0.0, 0.001, -0.0, 0.001, ...
7	0.06	0.000081	493	[0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.0...
8	0.04	0.000050	1583	[0.0, 0.0, 0.001, -0.0, 0.002, -0.0, 0.0, 0.00...
9	0.02	0.000009	4254	[0.0, 0.0, 0.002, -0.0, 0.0, -0.0, 0.0, 0.0, -...

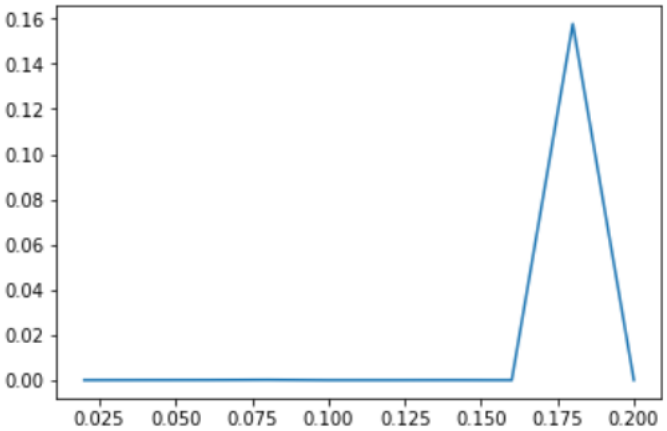

图 3.3 第一个函数变异率迭代情况

由图表可知，当交叉率在 0.15-0.2 之间时，寻找到的最小值波动较大，这可能是因为变异率过大，个体的优异基因发生突变的数量增大，但当变异率为 0.1 时左右时，便实现了精度在 0.01 以上的最小值，因此，交叉率的大小为 0.1 即可。

✧ accuracy 精度，来表示二进制长度，调参范围[0.0001,0.001]，每 0.0001 为一个 step。

表 3.4 第一个函数精度迭代情况

	tune_parma	**result**	**generation**	**param**
0	0.0010	0.000001	45	[0.0, 0.001, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0...
1	0.0009	0.000001	45	[0.0, 0.001, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0...
2	0.0008	0.000001	45	[0.0, 0.001, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0...
3	0.0007	0.000004	44	[0.001, -0.0, 0.0, -0.0, 0.0, -0.0, 0.001, -0....
4	0.0006	0.000004	44	[0.001, -0.0, 0.0, -0.0, 0.0, -0.0, 0.001, -0....
5	0.0005	0.000004	44	[0.001, -0.0, 0.0, -0.0, 0.0, -0.0, 0.001, -0....
6	0.0004	0.000004	44	[0.001, -0.0, 0.0, -0.0, 0.0, -0.0, 0.001, -0....
7	0.0003	2.633876	299	[-0.391, -0.195, 0.0, 0.0, 0.0, -0.001, 0.0, 0...
8	0.0002	2.633876	299	[-0.391, -0.195, 0.0, 0.0, 0.0, -0.001, 0.0, 0...

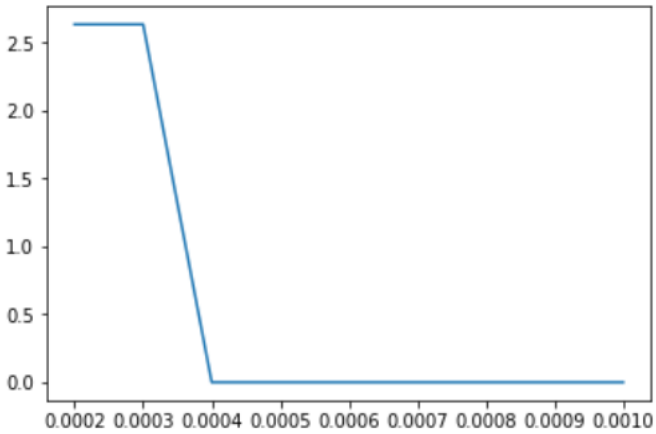

图 3.4 第一个函数精度迭代情况

由图表可知，当精度在 0.0002-0.0004 时，最优值情况不显著且不稳定，但当精度在 0.0005-0.001 附近时，都可以收敛到最优的情况，这说明精度（二进制）的设置情况对最终最优值的求解影响不大，只要设置合理即可，因此我们取 0.001 即可。

✧ 选择方式，轮盘赌选择及竞争选择两种方式

竞争选择：

前几次的迭代情况：

```
第-1次迭代 最佳拟合值:-67042.195349 最佳个体所在代际:-1
第0次迭代 最佳拟合值:-6224.555097 最佳个体所在代际:0
第1次迭代 最佳拟合值:-3921.6821969999996 最佳个体所在代际:1
第2次迭代 最佳拟合值:-3130.570935 最佳个体所在代际:2
第3次迭代 最佳拟合值:-2131.474924 最佳个体所在代际:3
第4次迭代 最佳拟合值:-1491.00174 最佳个体所在代际:4
第5次迭代 最佳拟合值:-993.249499 最佳个体所在代际:5
第6次迭代 最佳拟合值:-499.41758899999996 最佳个体所在代际:6
第7次迭代 最佳拟合值:-228.40584999999996 最佳个体所在代际:7
第8次迭代 最佳拟合值:-172.801602 最佳个体所在代际:8
第9次迭代 最佳拟合值:-80.45719199999999 最佳个体所在代际:9
第10次迭代 最佳拟合值:-47.143784 最佳个体所在代际:10
```

后几次的迭代情况：

```
第4986次迭代 最佳拟合值:-1e-06 最佳个体所在代际:45
第4987次迭代 最佳拟合值:-1e-06 最佳个体所在代际:45
第4988次迭代 最佳拟合值:-1e-06 最佳个体所在代际:45
第4989次迭代 最佳拟合值:-1e-06 最佳个体所在代际:45
第4990次迭代 最佳拟合值:-1e-06 最佳个体所在代际:45
第4991次迭代 最佳拟合值:-1e-06 最佳个体所在代际:45
第4992次迭代 最佳拟合值:-1e-06 最佳个体所在代际:45
第4993次迭代 最佳拟合值:-1e-06 最佳个体所在代际:45
第4994次迭代 最佳拟合值:-1e-06 最佳个体所在代际:45
第4995次迭代 最佳拟合值:-1e-06 最佳个体所在代际:45
第4996次迭代 最佳拟合值:-1e-06 最佳个体所在代际:45
第4997次迭代 最佳拟合值:-1e-06 最佳个体所在代际:45
第4998次迭代 最佳拟合值:-1e-06 最佳个体所在代际:45
第4999次迭代 最佳拟合值:-1e-06 最佳个体所在代际:45
```

最终的结果为：最优结果 1e-06 迭代次数为 45，最优个体:[0.0, 0.001, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.0, -0.0]

轮盘赌选择：

前几次的迭代情况：

```
第-1次迭代 最佳拟合值:-67042.195349 最佳个体所在代际:-1
第0次迭代 最佳拟合值:-11203.458340000001 最佳个体所在代际:0
第1次迭代 最佳拟合值:-10502.118272999998 最佳个体所在代际:1
第2次迭代 最佳拟合值:-8400.167417 最佳个体所在代际:2
第3次迭代 最佳拟合值:-6683.1543200000015 最佳个体所在代际:3
第4次迭代 最佳拟合值:-6267.162868 最佳个体所在代际:4
第5次迭代 最佳拟合值:-6169.088233 最佳个体所在代际:5
第6次迭代 最佳拟合值:-4790.817962000002 最佳个体所在代际:6
第7次迭代 最佳拟合值:-4058.360218 最佳个体所在代际:7
第8次迭代 最佳拟合值:-4058.360218 最佳个体所在代际:7
第9次迭代 最佳拟合值:-4058.360218 最佳个体所在代际:7
第10次迭代 最佳拟合值:-3945.1689620000006 最佳个体所在代际:10
```

后几次的迭代情况(1000 代时)：

```
第986次迭代 最佳拟合值:-0.006893000000000001 最佳个体所在代际:58
第987次迭代 最佳拟合值:-0.006893000000000001 最佳个体所在代际:58
第988次迭代 最佳拟合值:-0.006893000000000001 最佳个体所在代际:58
第989次迭代 最佳拟合值:-0.006893000000000001 最佳个体所在代际:58
第990次迭代 最佳拟合值:-0.006893000000000001 最佳个体所在代际:58
第991次迭代 最佳拟合值:-0.006893000000000001 最佳个体所在代际:58
第992次迭代 最佳拟合值:-0.006893000000000001 最佳个体所在代际:58
第993次迭代 最佳拟合值:-0.006893000000000001 最佳个体所在代际:58
第994次迭代 最佳拟合值:-0.006893000000000001 最佳个体所在代际:58
第995次迭代 最佳拟合值:-0.006893000000000001 最佳个体所在代际:58
第996次迭代 最佳拟合值:-0.006893000000000001 最佳个体所在代际:58
第997次迭代 最佳拟合值:-0.006893000000000001 最佳个体所在代际:58
第998次迭代 最佳拟合值:-0.006893000000000001 最佳个体所在代际:58
第999次迭代 最佳拟合值:-0.006893000000000001 最佳个体所在代际:58
```

最优结果: 0.006893000000000001 最优个体: [-0.082, 0.013]

通过上去的比较可知，竞争选择和轮盘赌选择均能求得最优值，但是相比于轮盘赌选择，竞争选择迭代的速度很快，竞争选择迭代次数为 45 次，而轮盘赌为 58 次并且竞争选择能够更好的保持多样性，因此竞争选择为较优的选择方式。

✧ 最优参数下的求解情况

最优状况下的参数情况：

In [61]:
```
# 迭代次数000次
generation_size=5000
# 群体大小
population_size=1300
# 染色体个数
chrom_num = 10
cross_prob=0.9
mutate_prob = 0.1
choose_type='tournament'
#choose_type='wheel'
object_fun=object_fun
lower_bound=-100
upper_bound=100
accuracy=0.001
is_print=True
```

最终结果为：1e-06，无限接近于最优值 0。

3.2 函数二

$$f_2(x) = \sum_{i=1}^{D} i\, x_i^2 \qquad x\epsilon[-10,10]$$

✧ population_size，调参范围[100,2000]，每 1000 为一个 step。

表 3.5 第二个函数种群大小迭代情况

	tune_parma	result	generation	param
0	100	222.317470	941	[0.001, -4.147, -3.204, -1.305, 0.001, -5.0, 0...
1	200	21.548152	720	[0.0, -2.74, -0.313, -0.055, 0.0, -0.054, 0.0,...
2	300	17.152717	1649	[0.002, -0.005, -0.627, -0.078, 0.0, 0.008, -0...
3	400	0.491324	621	[0.008, -0.313, -0.004, -0.002, 0.004, 0.0, -0...
4	500	1.979669	823	[-1.407, 0.001, 0.001, -0.001, -0.001, 0.001, ...
5	600	0.000134	73	[-0.0, 0.001, 0.002, -0.005, -0.002, 0.0, 0.0,...
6	700	0.000137	230	[-0.002, 0.002, 0.0, -0.0, -0.005, 0.0, -0.0, ...
7	800	0.003052	156	[-0.0, -0.039, 0.0, 0.0, 0.0, 0.0, -0.0, -0.0,...
8	900	0.000050	34	[0.0, -0.005, -0.0, 0.0, -0.0, -0.0, 0.0, 0.0,...
9	1000	0.553461	32	[0.003, 0.001, -0.02, -0.0, -0.0, 0.0, 0.0, 0....
10	1100	0.000008	36	[-0.0, 0.0, 0.0, -0.0, -0.0, 0.0, 0.0, 0.001, ...
11	1200	0.000038	2759	[-0.0, 0.001, 0.0, -0.003, -0.0, 0.0, 0.0, 0.0...
12	1300	0.000005	34	[0.0, -0.0, 0.0, 0.0, -0.001, -0.0, -0.0, 0.0,...
13	1400	0.000000	38	[-0.0, 0.0, 0.0, -0.0, -0.0, 0.0, 0.0, 0.0, 0....
14	1500	0.000000	39	[0.0, 0.0, -0.0, -0.0, -0.0, 0.0, 0.0, 0.0, -0...
15	1600	1.861411	37	[0.0, -0.0, -0.0, -0.0, 0.0, 0.0, -0.0, -0.0, ...
16	1700	0.000000	34	[0.0, -0.0, 0.0, 0.0, -0.0, -0.0, -0.0, 0.0, -...
17	1800	0.000000	31	[0.0, 0.0, 0.0, 0.0, 0.0, -0.0, -0.0, -0.0, -0...
18	1900	0.881721	33	[0.0, -0.0, 0.0, 0.0, -0.0, -0.0, -0.0, 0.0, -...

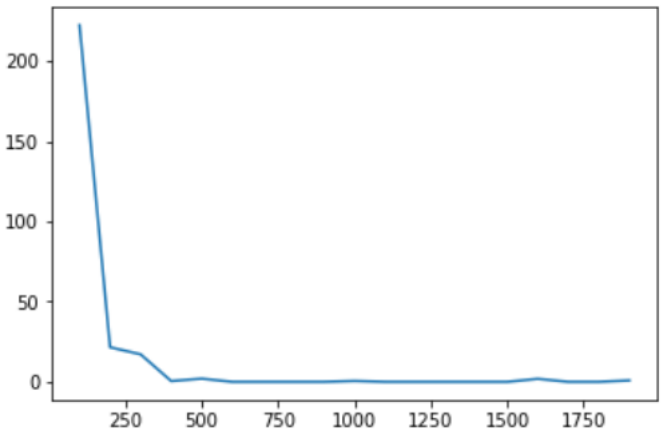

图 3.5 第二个函数种群大小迭代情况

由图表可知，当种群大小在 1200 之前时，每次寻找到的最小值波动较大，一方面

是因为种群大小过小，无法保证找出的父代较强，另一方面，遗传算法同样存在局部最优的情况，但当种群数量为 1400 时，实现了精度在 0.01 以上的最小值，因此，种群数量的大小为 1400。

✧ cross_prob 交叉率，调参范围[0.6,0.99]，每 0.05 为一个 step。

表 3.6 第二个函数交叉率迭代情况

	tune_parma	result	generation	param
0	0.60	0.067529	35	[0.0, -0.01, 0.0, 0.0, -0.002, -0.0, -0.098, 0...
1	0.65	0.000017	35	[0.002, -0.0, 0.001, 0.0, -0.0, -0.0, -0.0, 0....
2	0.70	0.000184	33	[0.003, -0.0, 0.0, 0.0, -0.0, -0.0, -0.005, 0....
3	0.75	0.036504	38	[0.0, -0.0, 0.0, 0.0, -0.0, -0.078, -0.0, 0.0,...
4	0.80	0.000243	32	[0.0, -0.01, 0.0, 0.0, -0.0, -0.0, -0.001, 0.0...
5	0.85	0.000000	32	[0.0, 0.0, -0.0, -0.0, -0.0, 0.0, 0.0, 0.0, -0...
6	0.90	0.000000	38	[-0.0, 0.0, 0.0, -0.0, -0.0, 0.0, 0.0, 0.0, 0....
7	0.95	0.587826	2852	[0.0, 0.001, 0.0, 0.0, 0.0, -0.313, -0.0, -0.0...

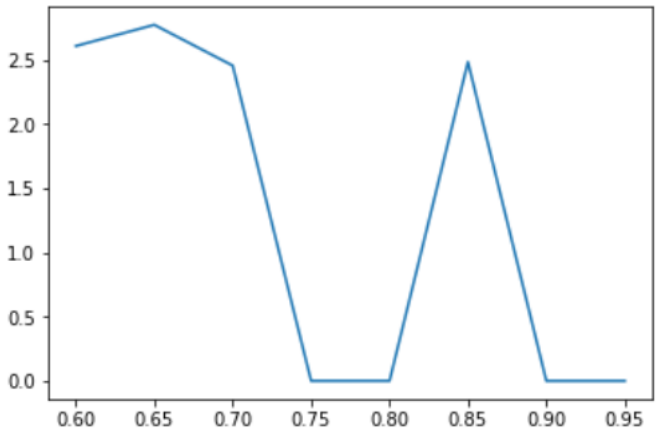

图 3.6 第二个函数交叉率迭代情况

由图表可知，当交叉率在 0.9 之前时，每次寻找到的最小值波动较大，这可能是因为交叉率过小，个体的变化不太，仍然存在较弱的基因，但当变异率为 0.9 时，实现了精度在 0.01 以上的最小值，因此，交叉率的大小为 0.9 即可。

✧ mutate_prob 变异率，调参范围[0.001,0.2]，每 0.02 一个 step。

表 3.7 第二个函数变异率迭代情况

	tune_parma	result	generation	param
0	0.20	0.000008	37	[0.001, -0.0, -0.0, -0.0, -0.0, 0.0, 0.001, 0....
1	0.18	0.000000	33	[-0.0, -0.0, 0.0, 0.0, 0.0, 0.0, -0.0, -0.0, 0...

2	0.16	0.000000	38	[-0.0, -0.0, 0.0, 0.0, -0.0, -0.0, 0.0, 0.0, -...
3	0.14	1.171875	33	[0.0, -0.0, -0.625, -0.0, -0.0, 0.0, 0.0, 0.0,...
4	0.12	0.000000	902	[0.0, 0.0, -0.0, -0.0, -0.0, 0.0, 0.0, 0.0, -0...
5	0.10	0.000005	34	[0.0, -0.0, 0.0, 0.0, -0.001, -0.0, -0.0, 0.0,...
6	0.08	1.075157	34	[0.0, -0.625, -0.313, -0.0, 0.0, 0.0, -0.0, -0...
7	0.06	0.587838	32	[0.0, 0.002, 0.0, 0.002, 0.0, -0.313, -0.0, -0...
8	0.04	0.000049	38	[-0.0, 0.0, -0.0, -0.001, -0.003, 0.0, -0.0, 0...
9	0.02	0.000000	35	[0.0, 0.0, 0.0, 0.0, 0.0, -0.0, -0.0, -0.0, -0...

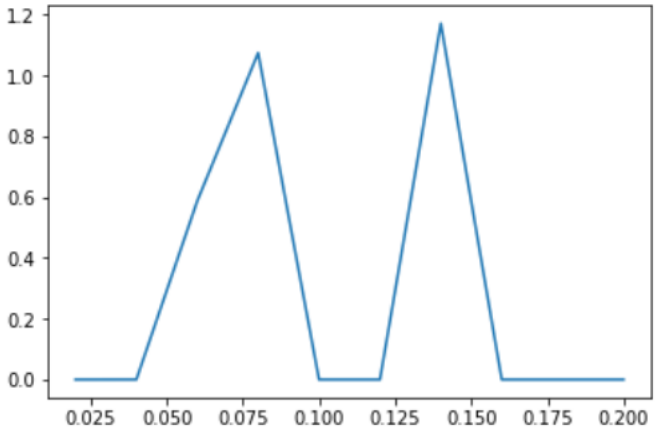

图 3.7 第二个函数变异率迭代情况

由图表可知，当变异率在 0.02-0.1 之间时，寻找到的最小值波动较大，这可能是因为变异率过大，个体的优异基因发生突变的数量增大，但当变异率为 0.14 时，其收敛的数值与最优解相差太大，因此我们取[0.1,0.2]范围内，除 0.14 的变异率都可，它们都实现了精度在 0.01 以上的最小值，因此，交叉率的大小为 0.1 即可。

✧ accuracy 精度，来表示二进制长度，调参范围[0.0001,0.001]，每 0.001 为一个 step。

表 3.8 第二个函数精度迭代情况

	tune_parma	**result**	**generation**	**param**
0	0.0010	0.049325	47	[0.001, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0...
1	0.0009	0.049325	47	[0.001, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0...
2	0.0008	0.049325	47	[0.001, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0...
3	0.0007	241.302959	45	[0.0, -0.012, 0.001, -0.098, 0.0, -6.25, 0.006...
4	0.0006	241.302959	45	[0.0, -0.012, 0.001, -0.098, 0.0, -6.25, 0.006...
5	0.0005	241.302959	45	[0.0, -0.012, 0.001, -0.098, 0.0, -6.25, 0.006...
6	0.0004	241.302959	45	[0.0, -0.012, 0.001, -0.098, 0.0, -6.25, 0.006...
7	0.0003	0.000667	463	[-0.025, -0.001, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0...
8	0.0002	0.000667	463	[-0.025, -0.001, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0...

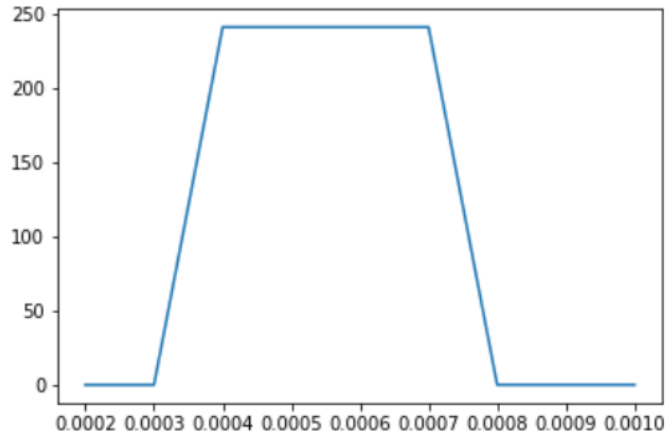

图 3.8 第二个函数精度迭代情况

由图表可知，当精度在 0.0002-0.0008 时，最优值情况不显著且不稳定，但当精度在 0.001 附近时，都可以收敛到最优的情况，这说明精度（二进制）的设置情况对最终最优值的求解影响不大，只要设置合理即可，因此我们取 0.001 即可。

✧ 选择方式，轮盘赌选择及竞争选择两种方式

竞争选择：

前几次的迭代情况：

```
第-1次迭代 最佳拟合值:-67042.195349 最佳个体所在代际:-1
第0次迭代 最佳拟合值:-6547.282398000001 最佳个体所在代际:0
第1次迭代 最佳拟合值:-4338.310015 最佳个体所在代际:1
第2次迭代 最佳拟合值:-2941.7323109999998 最佳个体所在代际:2
第3次迭代 最佳拟合值:-2123.588406 最佳个体所在代际:3
第4次迭代 最佳拟合值:-815.616833 最佳个体所在代际:4
第5次迭代 最佳拟合值:-667.0540649999998 最佳个体所在代际:5
第6次迭代 最佳拟合值:-567.950449 最佳个体所在代际:6
第7次迭代 最佳拟合值:-324.08494099999996 最佳个体所在代际:7
第8次迭代 最佳拟合值:-251.57088600000003 最佳个体所在代际:8
第9次迭代 最佳拟合值:-180.791158 最佳个体所在代际:9
第10次迭代 最佳拟合值:-110.86972800000001 最佳个体所在代际:10
第11次迭代 最佳拟合值:-68.81759 最佳个体所在代际:11
第12次迭代 最佳拟合值:-53.783632 最佳个体所在代际:12
第13次迭代 最佳拟合值:-14.785293 最佳个体所在代际:13
```

后几次的迭代情况：

```
第4986次迭代 最佳拟合值:-2e-06 最佳个体所在代际:39
第4987次迭代 最佳拟合值:-2e-06 最佳个体所在代际:39
第4988次迭代 最佳拟合值:-2e-06 最佳个体所在代际:39
第4989次迭代 最佳拟合值:-2e-06 最佳个体所在代际:39
第4990次迭代 最佳拟合值:-2e-06 最佳个体所在代际:39
第4991次迭代 最佳拟合值:-2e-06 最佳个体所在代际:39
第4992次迭代 最佳拟合值:-2e-06 最佳个体所在代际:39
第4993次迭代 最佳拟合值:-2e-06 最佳个体所在代际:39
第4994次迭代 最佳拟合值:-2e-06 最佳个体所在代际:39
第4995次迭代 最佳拟合值:-2e-06 最佳个体所在代际:39
第4996次迭代 最佳拟合值:-2e-06 最佳个体所在代际:39
第4997次迭代 最佳拟合值:-2e-06 最佳个体所在代际:39
第4998次迭代 最佳拟合值:-2e-06 最佳个体所在代际:39
第4999次迭代 最佳拟合值:-2e-06 最佳个体所在代际:39
```

最终的结果为：最优结果:2e-06 ；迭代次数 39；最优个体: [0.0, 0.001, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.0, -0.001]

轮盘赌选择：

前几次的迭代情况：

```
第-1次迭代 最佳拟合值:-366951.950681 最佳个体所在代际:-1
第0次迭代 最佳拟合值:-35040.03719999999 最佳个体所在代际:0
第1次迭代 最佳拟合值:-34523.527356 最佳个体所在代际:1
第2次迭代 最佳拟合值:-34523.527356 最佳个体所在代际:1
第3次迭代 最佳拟合值:-33990.325770999996 最佳个体所在代际:3
第4次迭代 最佳拟合值:-33742.131375000004 最佳个体所在代际:4
第5次迭代 最佳拟合值:-33742.131375000004 最佳个体所在代际:4
第6次迭代 最佳拟合值:-23505.573275000002 最佳个体所在代际:6
第7次迭代 最佳拟合值:-23505.573275000002 最佳个体所在代际:6
第8次迭代 最佳拟合值:-21777.287919 最佳个体所在代际:8
第9次迭代 最佳拟合值:-15567.162913 最佳个体所在代际:9
第10次迭代 最佳拟合值:-15567.162913 最佳个体所在代际:9
第11次迭代 最佳拟合值:-15567.162913 最佳个体所在代际:9
第12次迭代 最佳拟合值:-15567.162913 最佳个体所在代际:9
第13次迭代 最佳拟合值:-15567.162913 最佳个体所在代际:9
第14次迭代 最佳拟合值:-15567.162913 最佳个体所在代际:9
第15次迭代 最佳拟合值:-15567.162913 最佳个体所在代际:9
第16次迭代 最佳拟合值:-15567.162913 最佳个体所在代际:9
```

后几次的迭代情况：

```
第981次迭代 最佳拟合值:-0.0 最佳个体所在代际:72
第982次迭代 最佳拟合值:-0.0 最佳个体所在代际:72
第983次迭代 最佳拟合值:-0.0 最佳个体所在代际:72
第984次迭代 最佳拟合值:-0.0 最佳个体所在代际:72
第985次迭代 最佳拟合值:-0.0 最佳个体所在代际:72
第986次迭代 最佳拟合值:-0.0 最佳个体所在代际:72
第987次迭代 最佳拟合值:-0.0 最佳个体所在代际:72
第988次迭代 最佳拟合值:-0.0 最佳个体所在代际:72
第989次迭代 最佳拟合值:-0.0 最佳个体所在代际:72
第990次迭代 最佳拟合值:-0.0 最佳个体所在代际:72
第991次迭代 最佳拟合值:-0.0 最佳个体所在代际:72
第992次迭代 最佳拟合值:-0.0 最佳个体所在代际:72
第993次迭代 最佳拟合值:-0.0 最佳个体所在代际:72
第994次迭代 最佳拟合值:-0.0 最佳个体所在代际:72
第995次迭代 最佳拟合值:-0.0 最佳个体所在代际:72
第996次迭代 最佳拟合值:-0.0 最佳个体所在代际:72
```

最优结果: 0.0 最优个体: [0.0, 0.0]

通过上去的比较可知，竞争选择和轮盘赌选择均能求得最优值，但是相比于轮盘赌选择，竞争选择迭代的速度很快，竞争选择迭代次数为 39 次，而轮盘赌为 72 次，并且竞争选择能够更好的保持多样性，因此竞争选择为较优的选择方式。

✧ 最优参数下的求解情况

最优状况下的参数情况：

In [38]:
```
# 迭代次数000次
generation_size=5000
# 群体大小
population_size=1400
# 染色体个数
chrom_num = 10
cross_prob=0.9
mutate_prob = 0.1
choose_type='tournament'
#choose_type='wheel'
object_fun=object_fun
lower_bound=-100
upper_bound=100
accuracy=0.001
is_print=True
```

最终结果为：1e-06，无限接近于最优值 0。

3.3 函数三

$$f_3(x)=\sum_{i=1}^{D}(|x_i+0.5|)^2 \quad x\epsilon[-100,100]$$

✧ population_size，调参范围[100,2000] ，每 50 是一个 step。

表 3.9 第三个函数种群大小迭代情况

	tune_parma	result	generation	param
0	100	1924	46	[0.431, -4.499, 0.265, -32.116, 0.117, -20.391...
1	150	1233	67	[-25.189, -13.115, -3.496, 0.363, 0.019, 0.04,...
2	200	413	27	[0.317, -6.455, 0.232, -0.432, 0.422, -4.326, ...
3	250	21	23	[0.001, -1.85, 0.196, -0.408, 0.239, -1.464, -...
4	300	65	29	[0.297, -0.158, 0.353, -1.447, 0.246, -0.145, ...
5	350	4	34	[0.325, 0.152, 0.496, 0.354, 0.46, 0.37, 0.403...
6	400	339	30	[0.242, -13.067, 0.322, -13.421, 0.106, -1.066...
7	450	2	32	[0.202, 0.494, -0.47, -1.429, -1.027, -0.405, ...
8	500	425	506	[0.198, -0.488, 0.221, -0.146, 0.148, -13.268,...
9	550	205	30	[0.3, 0.065, -6.374, 0.328, 0.063, 0.129, 0.27...
10	600	52	146	[0.199, 0.171, -0.384, 0.074, 0.209, 0.014, 0....
11	650	41	26	[0.276, 0.138, -6.493, 0.396, 0.306, 0.085, 0....
12	700	10	20	[0.412, -3.483, 0.167, 0.126, 0.345, 0.28, -0....
13	750	2	93	[0.227, 0.051, -0.646, 0.495, 0.221, 0.228, 0....
14	800	80	31	[0.215, 0.199, 0.175, -8.101, 0.184, -4.34, 0....
15	850	6	27	[0.275, 0.015, 0.274, -1.767, 0.182, -1.371, 0...
16	900	21	26	[0.0, 0.089, 0.061, -1.867, 0.073, -4.219, 0.2...
17	950	10	66	[0.451, 0.016, 0.266, -0.244, 0.172, -3.393, 0...

18	1000	2	22	[0.235, 0.153, 0.126, -0.425, 0.014, -1.387, 0...
19	1050	4	18	[0.438, 0.15, -2.34, 0.099, 0.216, 0.171, 0.21...
20	1100	1	139	[0.004, 0.428, 0.038, -0.339, 0.211, -0.463, 0...
21	1150	0	22	[0.221, 0.051, -0.158, 0.061, 0.244, 0.032, 0....
22	1200	0	26	[0.304, 0.165, -0.219, 0.257, -0.288, -0.31, 0...

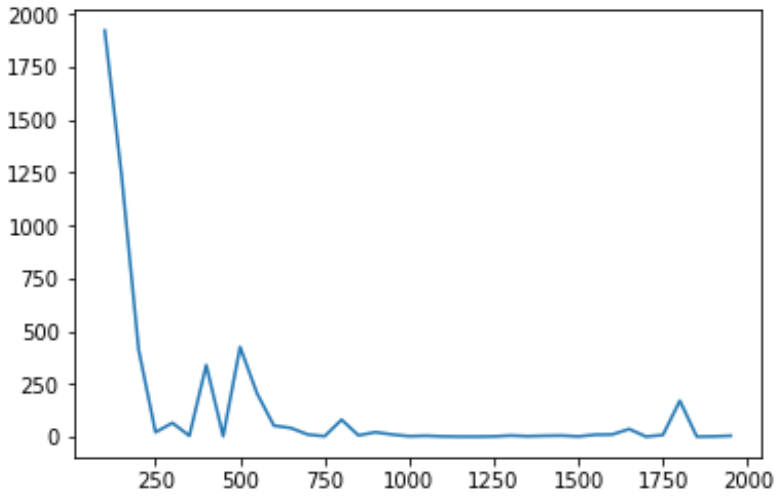

图 3.9 第三个函数种群大小迭代情况

由图表可知，当种群大小为 1850 之前会经历一些波动，可能原因是种群数目不够多，未能找到较优父代，此外遗传算法也存在局部最优的情况，但当种群数量 1850 时，实现了精度在 0.01 以上的最小值，因此，种群数量的大小为 1850。

✧ cross_prob 交叉率，调参范围[0.6,0.99] ，每 0.05 是一个 step。

表 3.10 第三个函数交叉率迭代情况

	tune_parma	result	generation	param
0	0.600	0	6	[-0.489, -0.44, -0.04, -0.164]
1	0.625	0	9	[0.119, 0.182, -0.103, -0.4]
2	0.650	1	7	[0.107, -0.084, -1.308, 0.351]
3	0.675	0	11	[0.459, -0.475, 0.126, 0.159]
4	0.700	0	8	[0.262, -0.475, -0.349, -0.428]
5	0.725	0	9	[0.019, -0.434, -0.241, 0.364]
6	0.750	0	10	[0.116, -0.459, -0.094, -0.328]
7	0.775	0	7	[0.119, -0.447, -0.268, 0.251]
8	0.800	0	9	[0.348, -0.085, -0.057, -0.476]
9	0.825	0	7	[0.069, 0.492, -0.264, 0.309]
10	0.850	0	8	[0.119, -0.488, -0.219, 0.334]
11	0.875	0	6	[0.119, -0.172, -0.084, -0.077]
12	0.900	0	8	[0.448, -0.434, -0.461, -0.324]

13	0.925	0	4	[0.253, -0.473, -0.264, -0.231]
14	0.950	1	5	[0.207, -0.622, -0.23, -0.107]
15	0.975	0	6	[0.171, -0.083, -0.036, -0.093]

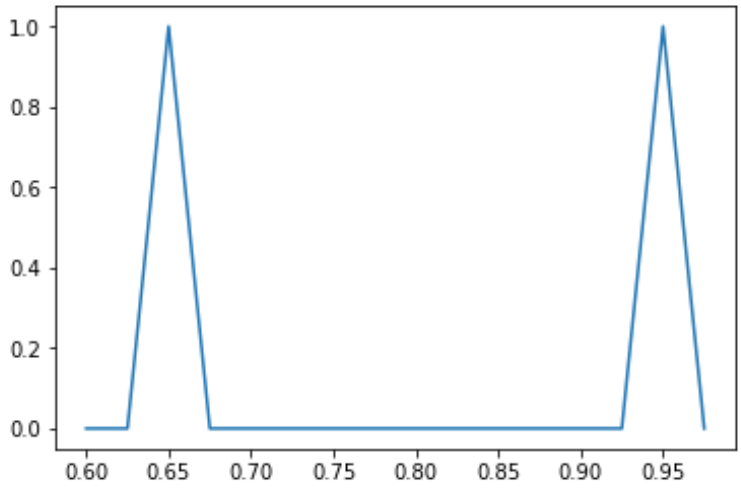

图 3.10 第三个函数交叉率迭代情况

由图表可知，当交叉率在 0.7 之前和 0.9 以后时，每次寻找到的最优值波动较大，这可能是因为交叉率过小，个体的变化不太，仍然存在较弱的基因，但当变异率为 0.9 时，实现了精度在 0.01 以上的最小值，因此，交叉率的大小为 0.9 即可。

✧ mutate_prob 变异率，调参范围[0.001,0.2]，，每 0.01 是一个 step。

表 3.11 第三个函数变异率迭代情况

	tune_parma	result	generation	param
32	0.0001	0	2	[0.219, -0.206]
33	0.0002	0	1	[0.219, -0.206]
34	0.0003	0	1	[0.219, -0.206]
35	0.0004	0	1	[0.219, -0.206]
36	0.0005	0	1	[0.219, -0.206]
37	0.0006	0	1	[0.219, -0.206]
38	0.0007	0	1	[0.219, -0.206]
39	0.0008	0	1	[0.219, -0.206]
40	0.0009	0	1	[0.219, -0.206]

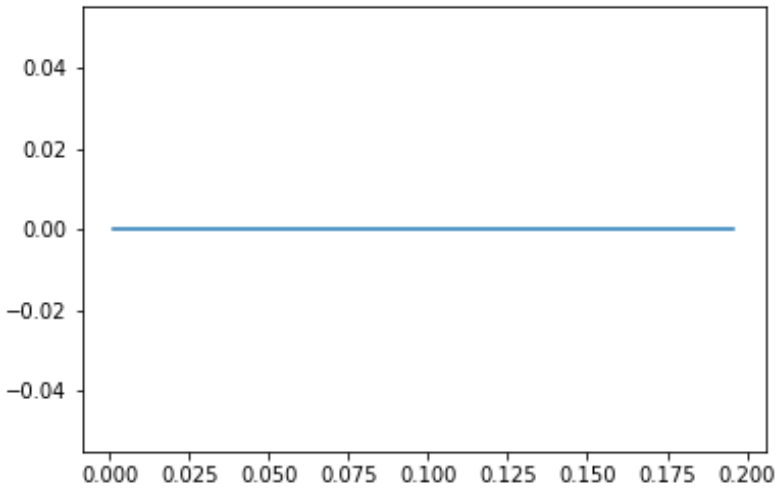

图 3.11 第三个函数变异率迭代情况

此情况下该参数比较特别，即时将参数范围设置足够大、步长足够小，也未能寻找到最优解，于是打算在更改其他参数初始值的情况下对该参数迭优调参，经过多次试验，发现当 population_size 设置为 1100，chrom_num = 2 时能收敛到最优值，因为染色体数量一般要大于变量的数量，所以设置为 2 也比较合理，由结果显示说明，变异率参数在此情况下影响不大。

✧ accuracy 精度，来表示二进制长度，调参范围[0.0001,0.001]，每 0.001 是一个 step。

表 3.12 第三个函数精度迭代情况

	tune_parma	result	generation	param
0	0.0001	0	1	[0.0741, 0.431, 0.3916, -0.3796]
1	0.0002	0	2	[-0.166, -0.383, 0.349, -0.366]
2	0.0003	0	3	[-0.166, -0.383, 0.349, -0.366]
3	0.0004	0	4	[0.302, 0.392, 0.294, 0.07]
4	0.0005	0	5	[0.302, 0.392, 0.294, 0.07]
5	0.0006	0	6	[0.302, 0.392, 0.294, 0.07]
6	0.0007	0	7	[0.302, 0.392, 0.294, 0.07]
7	0.0008	0	8	[0.448, -0.434, -0.461, -0.324]
8	0.0009	0	9	[0.448, -0.434, -0.461, -0.324]

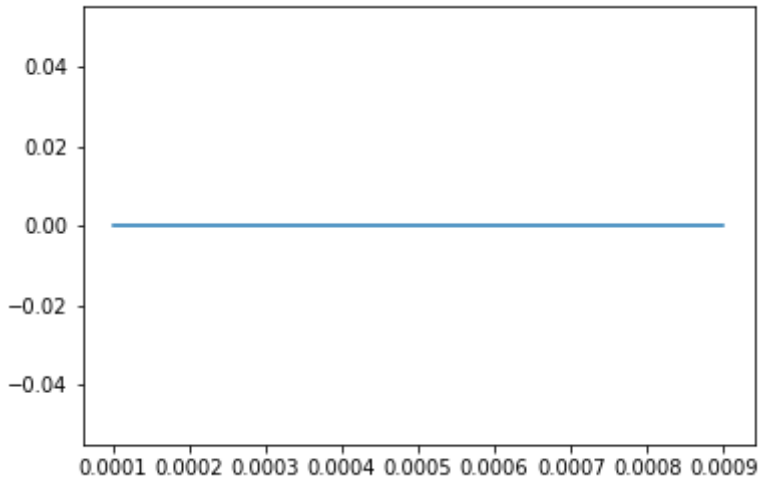

图 3.12 第三个函数精度迭代情况

此情况下该参数无论是 0.0001 更大的在范围内的值都可得出最优值 0，说明精度在此情况下影响不大。

✧ 选择方式，轮盘赌选择及竞争选择两种方式

竞争选择

迭代情况：

```
第4990次迭代 最佳拟合值:0 最佳个体所在代际:8
第4991次迭代 最佳拟合值:0 最佳个体所在代际:8
第4992次迭代 最佳拟合值:0 最佳个体所在代际:8
第4993次迭代 最佳拟合值:0 最佳个体所在代际:8
第4994次迭代 最佳拟合值:0 最佳个体所在代际:8
第4995次迭代 最佳拟合值:0 最佳个体所在代际:8
第4996次迭代 最佳拟合值:0 最佳个体所在代际:8
第4997次迭代 最佳拟合值:0 最佳个体所在代际:8
第4998次迭代 最佳拟合值:0 最佳个体所在代际:8
第4999次迭代 最佳拟合值:0 最佳个体所在代际:8
```

最终的结果为：最优结果 0 最优个体: [0.083, 0.293, -0.313, -0.367, 0.473, 0.103, 0.232, -0.33, -0.135, 0.216]

轮盘赌选择

迭代情况：

```
第4992次迭代 最佳拟合值:0 最佳个体所在代际:10
第4993次迭代 最佳拟合值:0 最佳个体所在代际:10
第4994次迭代 最佳拟合值:0 最佳个体所在代际:10
第4995次迭代 最佳拟合值:0 最佳个体所在代际:10
第4996次迭代 最佳拟合值:0 最佳个体所在代际:10
第4997次迭代 最佳拟合值:0 最佳个体所在代际:10
第4998次迭代 最佳拟合值:0 最佳个体所在代际:10
第4999次迭代 最佳拟合值:0 最佳个体所在代际:10
```

通过比较两种选择方式可知，竞争选择和轮盘赌选择均能求得最优值，但是相比于轮盘赌选择，竞争选择迭代的速度更快，并且竞争选择能够更好的保持多样性，因此竞

争选择为较优的选择方式。

✧ 最优参数下的求解情况

最优状况下的参数情况：

```
generation_size=5000
# 群体大小
population_size=1850
# 染色体个数
chrom_num = 10
cross_prob=0.9
mutate_prob = 0.02
choose_type='tournament'
object_fun=object_fun
lower_bound=-100
upper_bound=100
accuracy=0.001
```

最终结果为：0。

3.4 函数四

$$f_4(x)=\sum_{i=1}^{D} i\,x_i^4+random[0,1) \quad x\epsilon[-1.28，1.28]$$

✧ population_size，调参范围[100,2000]，，每 50 是一个 step。

表 3.13 第四个函数种群大小迭代情况

	tune_parma	result	generation	param
25	1350	0.4179321	41	[0.0, 0.001, 0.0, -0.0, 0.001, -0.782, 0.0, 0....
26	1400	0.4166987	39	[0.0, 0.001, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0...
27	1450	0.4166198	40	[0.0, 0.0, -0.0, 0.0, 0.0, 0.001, 0.0, 0.0, -0...
28	1500	0.4206749	146	[0.0, 0.0, -0.002, 0.0, 0.0, 0.0, 0.0, 0.0, -0...
29	1550	0.4169475	42	[0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.0...
30	1600	0.4166989	42	[0.0, -0.0, -0.0, -0.0, -0.0, 0.0, -0.0, 0.0, ...
31	1650	0.4166198	42	[0.0, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.0...

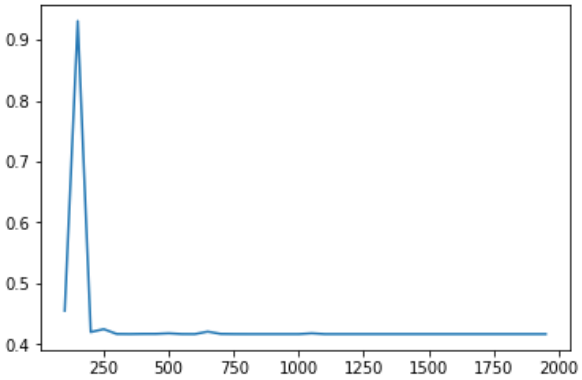

图 3.13 第四个函数种群大小迭代情况

由图表可知，当种群大小在 1100 之前时，每次寻找到的最小值波动较大，一方面是因为种群大小过小，无法保证找出的父代较强，另一方面，遗传算法同样存在局部最优的情况，但当种群数量为 1100 时，实现了精度在 0.01 以上的最小值，因此，种群数量的大小为 1100。

✧ cross_prob 交叉率，调参范围[0.6,0.99] ，每 0.05 是一个 step。

表 3.14 第四个函数交叉率迭代情况

	tune_parma	result	generation	param
5	0.680	0.914488	19	[-0.725, -0.505, -0.002, -0.202, 0.03, -0.32, ...
6	0.696	1.067652	15	[-0.277, -0.168, -0.641, -0.225, 0.081, -0.381...
7	0.712	0.937356	15	[-0.09, -0.185, -0.64, -0.161, 0.003, -0.211, ...
8	0.728	1.110555	15	[-0.493, -0.18, -0.64, -0.22, 0.028, -0.376, -...
9	0.744	0.455178	16	[-0.252, -0.108, -0.15, -0.237, 0.0, -0.06, 0....
10	0.760	0.425488	23	[-0.173, 0.041, 0.03, -0.043, 0.003, -0.178, -...
11	0.776	0.486511	16	[-0.194, 0.094, 0.344, -0.138, 0.081, -0.064, ..

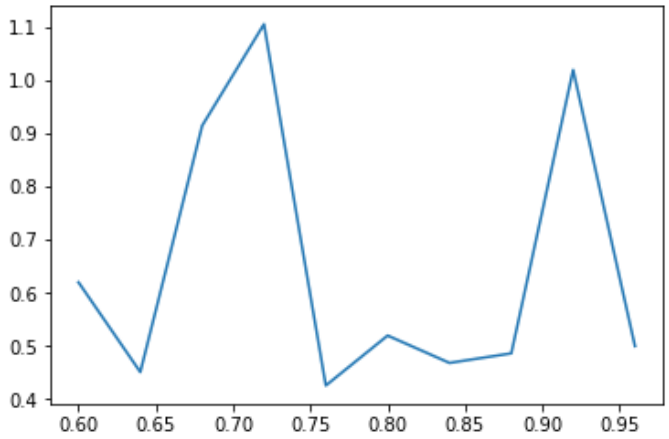

图 3.14 第四个函数交叉率迭代情况

由图表可知，当交叉率在 0.75 之前和之后时，每次寻找到的最小值波动非常大，这可能是因为交叉率过小，个体的变化不太，仍然存在较弱的基因，但当变异率为 0.75 时，实现了精度在 0.01 以上的最小值，因此，交叉率的大小为 0.75 即可。

✧ mutate_prob 变异率，调参范围[0.001,0.2] ，每 0.025 是一个 step。

表 3.15 第四个函数变异率迭代情况

	tune_parma	result	generation	param
0	0.600	0.619918	15	[0.02, -0.187, -0.154, -0.213, 0.025, -0.349, ...
1	0.616	1.025163	12	[0.045, -0.165, -0.642, -0.203, 0.03, -0.344, ...

2	0.632	0.927981	13	[0.035, -0.187, -0.64, -0.181, 0.103, -0.041, ...
3	0.648	0.430294	19	[-0.039, -0.088, 0.003, 0.009, 0.002, 0.057, -...
4	0.824	0.427591	20	[0.067, -0.082, -0.027, -0.225, 0.001, -0.02, ...
5	0.840	0.468070	15	[0.326, -0.01, -0.32, -0.167, 0.101, -0.161, -...
6	0.856	0.418361	17	[0.063, 0.022, -0.004, 0.083, 0.021, -0.062, 0...
7	0.824	0.427591	20	[0.067, -0.082, -0.027, -0.225, 0.001, -0.02, ...

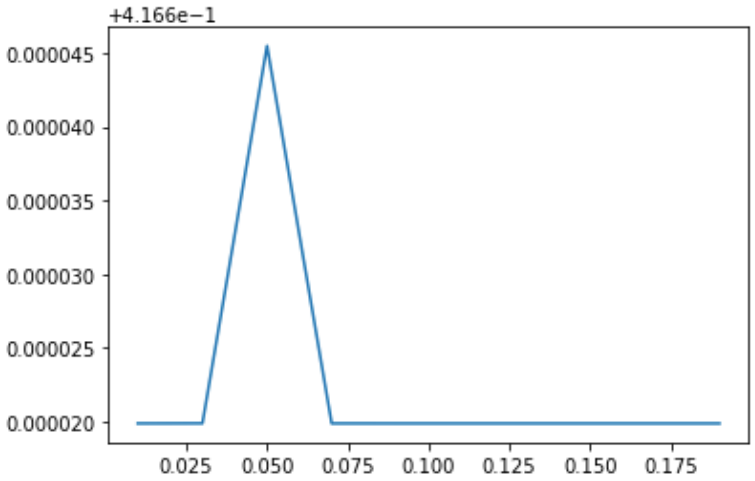

图 3.15 第四个函数变异率迭代情况

由图表可知，当变异率在 0.025 之后时，寻找到的最小值波动较大，但在变异率大于 0.075 时又取得了较好效果，但根据经验变异率不能超过 0.05，因此，变异率的大小为 0.02 即可。

✧ accuracy 精度，来表示二进制长度，调参范围[0.001,0.01]，，每 0.001 是一个 step。

表 3.16 第四个函数精度迭代情况

	tune_parma	**result**	**generation**	**param**
0	0.001	0.454891	18	[-0.124, -0.065, 0.325, 0.083, 0.022, 0.058, -...
1	0.002	0.424511	20	[0.09, -0.0, 0.0, 0.0, 0.0, -0.19, 0.01, 0.03,...
2	0.003	0.813691	12	[-0.0, 0.05, -0.33, -0.09, 0.02, -0.21, 0.2, 0...
3	0.004	0.813691	12	[-0.0, 0.05, -0.33, -0.09, 0.02, -0.21, 0.2, 0...
4	0.005	0.813691	12	[-0.0, 0.05, -0.33, -0.09, 0.02, -0.21, 0.2, 0...
5	0.006	0.439979	12	[0.18, -0.32, 0.08, -0.12, -0.04, 0.0, 0.05, -...
6	0.007	0.439979	12	[0.18, -0.32, 0.08, -0.12, -0.04, 0.0, 0.05, -...
7	0.008	0.439979	12	[0.18, -0.32, 0.08, -0.12, -0.04, 0.0, 0.05, -...
8	0.009	0.439979	12	[0.18, -0.32, 0.08, -0.12, -0.04, 0.0, 0.05, -...

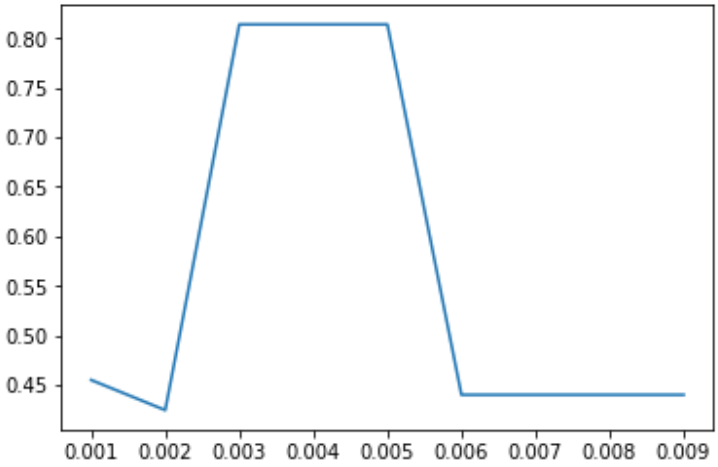

图 3.16 第四个函数精度迭代情况

由图表可知，当精度在 0.002-0.006 时，最优值情况非常不稳定，但当精度大于 0.006 时便可以收敛到最优的情况，说明在此情况下精度大于 0.006 时可以得到较好情况。

✧ 选择方式，轮盘赌选择及竞争选择两种方式

竞争选择

迭代情况：

```
第1505次迭代 最佳拟合值:-0.46656705215164945 最佳个体所在代际:85
第1506次迭代 最佳拟合值:-0.46656705215164945 最佳个体所在代际:85
第1507次迭代 最佳拟合值:-0.46656705215164945 最佳个体所在代际:85
第1508次迭代 最佳拟合值:-0.46656705215164945 最佳个体所在代际:85
第1509次迭代 最佳拟合值:-0.46656705215164945 最佳个体所在代际:85
第1510次迭代 最佳拟合值:-0.46656705215164945 最佳个体所在代际:85
第1511次迭代 最佳拟合值:-0.46656705215164945 最佳个体所在代际:85
第1512次迭代 最佳拟合值:-0.46656705215164945 最佳个体所在代际:85
```

最终的结果为：最优结果：0，最优个体: [0.245, 0.32, -0.319, -0.364, 0.389，-0.0687, 0.0201, 0.0053, 0.0064，0.0003]

轮盘赌选择

迭代情况：

```
第986次迭代 最佳拟合值:-0.42440148463508254 最佳个体所在代际:98
第987次迭代 最佳拟合值:-0.42440148463508254 最佳个体所在代际:98
第988次迭代 最佳拟合值:-0.42440148463508254 最佳个体所在代际:98
第989次迭代 最佳拟合值:-0.42440148463508254 最佳个体所在代际:98
第990次迭代 最佳拟合值:-0.42440148463508254 最佳个体所在代际:98
第991次迭代 最佳拟合值:-0.42440148463508254 最佳个体所在代际:98
第992次迭代 最佳拟合值:-0.42440148463508254 最佳个体所在代际:98
第993次迭代 最佳拟合值:-0.42440148463508254 最佳个体所在代际:98
第994次迭代 最佳拟合值:-0.42440148463508254 最佳个体所在代际:98
第995次迭代 最佳拟合值:-0.42440148463508254 最佳个体所在代际:98
第996次迭代 最佳拟合值:-0.42440148463508254 最佳个体所在代际:98
第997次迭代 最佳拟合值:-0.42440148463508254 最佳个体所在代际:98
第998次迭代 最佳拟合值:-0.42440148463508254 最佳个体所在代际:98
第999次迭代 最佳拟合值:-0.42440148463508254 最佳个体所在代际:98
```

最优结果: 0.42440148463508254

最优个体: [0.0475, -0.0687, 0.0201, 0.0003, -0.1983, 0.0066, 0.0053, 0.0064, 0.0006, 0.0038]

通过上述的比较可知，竞争选择和轮盘赌选择均能求得最优值，但是相比于轮盘赌选择，竞争选择迭代的速度很快，并且竞争选择能够更好的保持多样性，因此竞争选择为较优的选择方式。

✧ 最优参数下的求解情况

最优状况下的参数情况：

```
generation_size=5000
# 群体大小
population_size=1100
# 染色体个数
chrom_num = 10
cross_prob=0.7
mutate_prob = 0.02
choose_type='tournament'
object_fun=object_fun
lower_bound=-1.28
upper_bound=1.28
accuracy=0.001
```

最终结果为：0

3.5 函数五

$$f_5(x) = -\cos(x_1)\cos(x_2)\exp(-(x_1-\pi)^2-(x_2-\pi)^2) \quad x\epsilon[-100,100]$$

✧ population_size，调参范围[100,2000]

表 3.17 第五个函数种群大小迭代情况

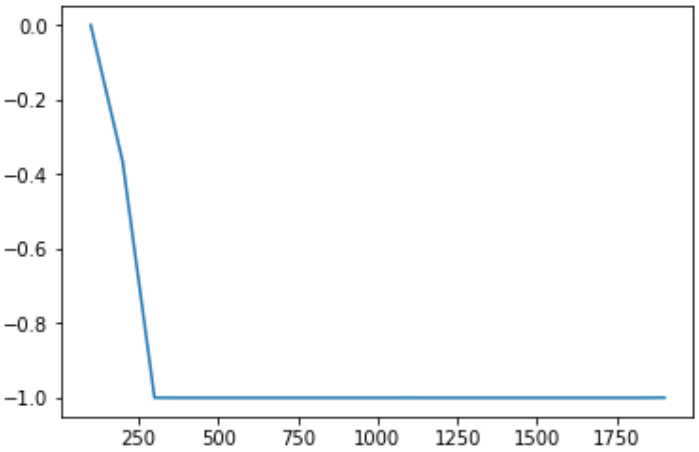

图 3.17 第五个函数种群大小迭代情况

由图表可知，当种群大小在 260 之前时，每次寻找到的最小值波动较大，一方面是因为种群大小过小，无法保证找出的父代较强，另一方面，遗传算法同样存在局部最优的情况，但当种群数量大于 260 时，实现了精度在 0.01 以上的最小值，因此，种群数量的大小可设置为 1000。

✧ cross_prob 交叉率，调参范围[0.6,0.99],，每 0.05 是一个 step。

表 3.18 第五个函数交叉率迭代情况

	tune_parma	result	generation	param
0	0.60	-0.999991	14	[3.144, 3.141, -40.33, -12.278, 25.33, -65.588...
1	0.65	-0.999953	11	[3.136, 3.141, -40.343, -57.269, -17.877, -99....
2	0.70	-0.999999	15	[3.142, 3.141, -40.208, -62.287, -18.103, -94....
3	0.75	-0.999968	12	[3.137, 3.141, -39.573, -7.69, 82.484, -59.631...
4	0.80	-0.999991	16	[3.144, 3.141, -40.55, -70.19, -17.893, -99.19...
5	0.85	-0.999587	9	[3.125, 3.141, -40.555, -57.669, 82.092, -85.8...
6	0.90	-1.000000	12	[3.142, 3.142, -39.868, -20.187, 56.964, -61.5...
7	0.95	-0.999481	9	[3.142, 3.123, -46.508, -58.365, -5.792, -94.2...

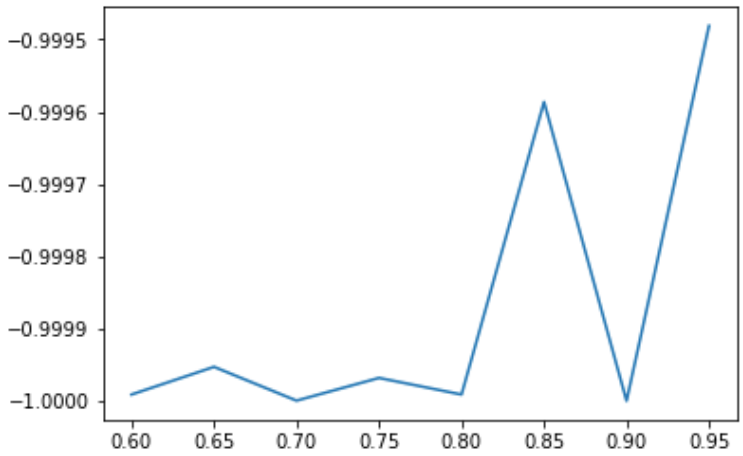

图 3.18 第五个函数交叉率迭代情况

由图表可知，当交叉率在 0.80-0.90 时最小值波动较大，这可能是因为交叉率过大时个体的变化比较大，使得个体原有的优秀基因没能得到保留，但当变异率为 0.7 或 0.9 时，实现了精度在 0.01 以上的最小值，但经过最终比较发现设置为 0.9 时效果更好，因此，交叉率的大小为 0.9 即可。

✧ mutate_prob 变异率，调参范围[0.001,0.2],，每 0.02 是一个 step。

表 3.19 第五个函数变异率迭代情况

	tune_parma	result	generation	param
0	0.20	-0.933684	14	[3.144, 2.928, -46.64, -57.256, -18.3, -72.252...
1	0.18	-0.999991	16	[3.144, 3.141, -90.358, -85.825, 75.008, -94.9...
2	0.16	-0.999587	11	[3.125, 3.141, -90.343, -22.881, 81.723, -73.1...
3	0.14	-0.999999	19	[3.142, 3.141, -40.561, -82.799, -42.965, -97....
4	0.12	-0.999983	12	[3.144, 3.144, -96.606, -57.69, -17.909, -95.1...
5	0.10	-0.999847	15	[3.136, 3.15, -43.466, -87.437, 6.651, -97.618...
6	0.08	-0.999980	14	[3.142, 3.138, -99.757, -82.712, -17.965, -96....

7	0.06	-0.999980	16	[3.142, 3.138, -43.515, -10.84, 81.329, -59.72...
8	0.04	-0.921828	14	[3.142, 2.909, -46.615, -73.289, 31.701, -60.5...
9	0.02	-0.999980	18	[3.142, 3.138, -90.382, -87.399, -67.964, -96....

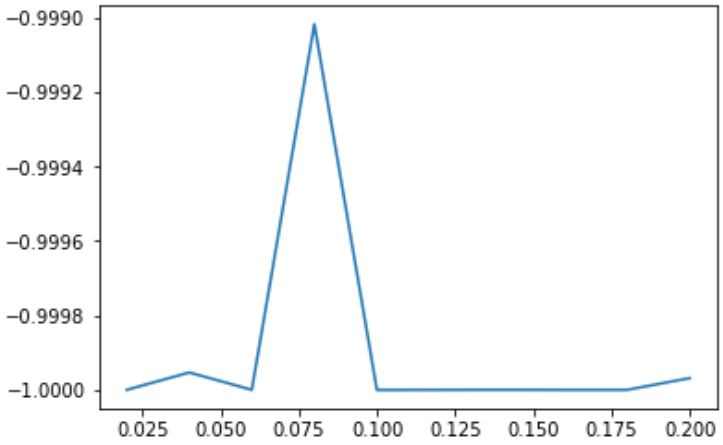

图 3.19 第五个函数变异率迭代情况

由图表可知，当变异率在 0.025 之后时，寻找到的最小值波动较大，但在变异率大于 0.075 时又取得了较好效果，但根据经验变异率不能超过 0.05，因此，变异率的大小为 0.02 即可。

✧ accuracy 精度，来表示二进制长度，调参范围[0.001,0.01]，，每 0.001 是一个 step。

表 3.20 第五个函数精度迭代情况

	tune_parma	result	generation	param
0	0.0001	-1.000000	12	[3.142, 3.142, -39.868, -20.187, 56.964, -61.5...
1	0.0002	-1.000000	12	[3.142, 3.142, -39.868, -20.187, 56.964, -61.5...
2	0.0003	-1.000000	12	[3.142, 3.142, -39.868, -20.187, 56.964, -61.5...
3	0.0004	-0.997400	13	[3.1, 3.144, 7.795, 38.126, 4.889, 9.851, -98....
4	0.0005	-0.997400	13	[3.1, 3.144, 7.795, 38.126, 4.889, 9.851, -98....
5	0.0006	-0.997400	13	[3.1, 3.144, 7.795, 38.126, 4.889, 9.851, -98....
6	0.0007	-0.997400	13	[3.1, 3.144, 7.795, 38.126, 4.889, 9.851, -98....
7	0.0008	-0.000079	9	[4.981, 1.238, 91.848, -97.162, -83.201, -99.0...
8	0.0009	-0.000079	9	[4.981, 1.238, 91.848, -97.162, -83.201, -99.0...

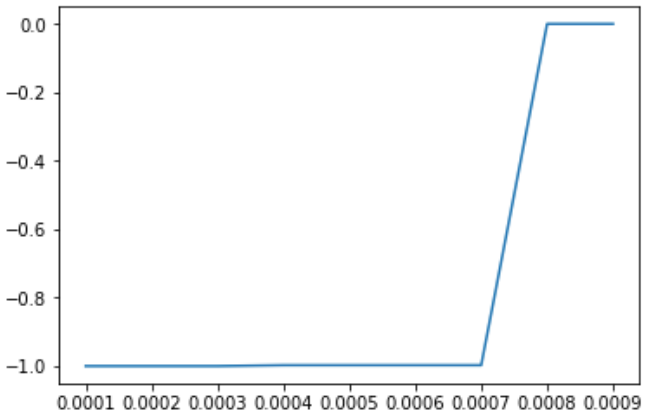

图 3.20 第五个函数精度迭代情况

由图表可知，当精度在大于 0.007 时，最优值情况非常不稳定，但当精度小于 0.007 时便可以收敛到最优的情况，说明在此情况下精度小于 0.007 时可以得到较好情况。

✧ 选择方式，轮盘赌选择及竞争选择两种方式

竞争选择迭代情况：

```
第4989次迭代 最佳拟合值:0.9995870402557842 最佳个体所在代际:12
第4990次迭代 最佳拟合值:0.9995870402557842 最佳个体所在代际:12
第4991次迭代 最佳拟合值:0.9995870402557842 最佳个体所在代际:12
第4992次迭代 最佳拟合值:0.9995870402557842 最佳个体所在代际:12
第4993次迭代 最佳拟合值:0.9995870402557842 最佳个体所在代际:12
第4994次迭代 最佳拟合值:0.9995870402557842 最佳个体所在代际:12
第4995次迭代 最佳拟合值:0.9995870402557842 最佳个体所在代际:12
第4996次迭代 最佳拟合值:0.9995870402557842 最佳个体所在代际:12
第4997次迭代 最佳拟合值:0.9995870402557842 最佳个体所在代际:12
第4998次迭代 最佳拟合值:0.9995870402557842 最佳个体所在代际:12
第4999次迭代 最佳拟合值:0.9995870402557842 最佳个体所在代际:12
```

最优结果:-0.9999995022068257，最优个体: [3.142, 3.142]

轮盘赌选择迭代情况：

```
第92次迭代 最佳拟合值:0.9999995022068257 最佳个体所在代际:10
第93次迭代 最佳拟合值:0.9999995022068257 最佳个体所在代际:10
第94次迭代 最佳拟合值:0.9999995022068257 最佳个体所在代际:10
第95次迭代 最佳拟合值:0.9999995022068257 最佳个体所在代际:10
第96次迭代 最佳拟合值:0.9999995022068257 最佳个体所在代际:10
第97次迭代 最佳拟合值:0.9999995022068257 最佳个体所在代际:10
第98次迭代 最佳拟合值:0.9999995022068257 最佳个体所在代际:10
第99次迭代 最佳拟合值:0.9999995022068257 最佳个体所在代际:10
最优结果： -0.9999995022068257
```

最优结果: -0.9999995022068257，最优个体: [3.142, 3.142]

通过上述的比较可知，竞争选择和轮盘赌选择均能求得最优值，但是相比于轮盘赌选择，竞争选择迭代的速度很快，并且竞争选择能够更好的保持多样性，因此竞争选择为较优的选择方式。

✧ 最优参数下的求解情况

最优状况下的参数情况：

```
generation_size=5000
population_size=5000
chrom_num = 2
cross_prob=0.9
mutate_prob = 0.02
choose_type='tournament'
object_fun=object_fun
lower_bound=-100
upper_bound=100
accuracy=0.001
```

最终结果为：-0.999999999，无限趋近于-1

3.6 函数六

$$f_6(x) = 0.26(x_1{}^2 + x_2{}^2) - 0.48x_1x_2 \quad x\epsilon[-10,10]$$

✧ population_size，调参范围[100,2000]，每 100 为一个 step。

表 3.21 第六个函数种群大小迭代情况

	tune_parma	result	generation	param
0	100	1124.387559	4649	[-25.5, -23.437, -6.367, 0.001, -33.268, 0.0, ...
1	200	235.868841	54	[-14.219, -13.898, -25.0, 0.004, 0.019, 0.0, -...
2	300	271.307144	1122	[0.0, 0.001, -12.525, -25.94, -14.485, -0.473,...
3	400	1.475611	693	[0.196, 0.177, -1.564, -0.013, -0.006, -1.688,...
4	500	3.629990	39	[-7.814, -7.233, -0.032, -1.575, -0.002, -0.06...
5	600	0.675329	194	[0.001, 0.001, -0.0, 0.0, 0.0, 0.001, 0.0, 0.0...
6	700	1.512437	465	[5.768, 6.25, -0.0, 0.0, 0.0, 0.001, 0.001, 0....
7	800	0.635424	595	[0.001, 0.0, 0.0, -0.031, 0.0, -0.0, 0.001, 0....
8	900	45.819811	2239	[0.0, 0.001, -12.5, 0.0, 0.0, 0.0, 0.0, 0.0, -...
9	1000	6.644950	44	[12.5, 11.538, 0.0, -0.0, 0.0, -0.0, 0.001, 0....
10	1100	0.000371	2558	[0.09, 0.098, 0.0, -0.0, 0.0, -0.001, 0.002, 0...
11	1200	2.539072	40	[0.0, 0.0, 0.0, -0.006, 0.0, -0.0, 0.0, 0.0, -...
12	1300	0.001478	212	[0.196, 0.181, -0.0, 0.0, 0.001, 0.0, 0.0, 0.0...
13	1400	1.368355	46	[-2.885, -3.125, 0.0, 0.0, -0.0, -1.954, 0.0, ...
14	1500	24.673868	44	[23.047, 25.0, 0.0, -1.563, 0.0, -0.002, 0.0, ...
15	1600	1.502405	258	[-5.769, -6.25, 0.002, 0.0, 0.0, -0.0, 0.0, -0...
16	1700	0.000002	171	[0.0, -0.002, 0.001, 0.0, 0.0, -0.0, -0.0, -0....
17	1800	0.000003	44	[0.003, 0.003, 0.0, -0.0, 0.0, -0.001, 0.0, 0....
18	1900	0.039913	42	[0.0, -0.001, -0.0, -0.391, -0.0, 0.0, -0.025,...

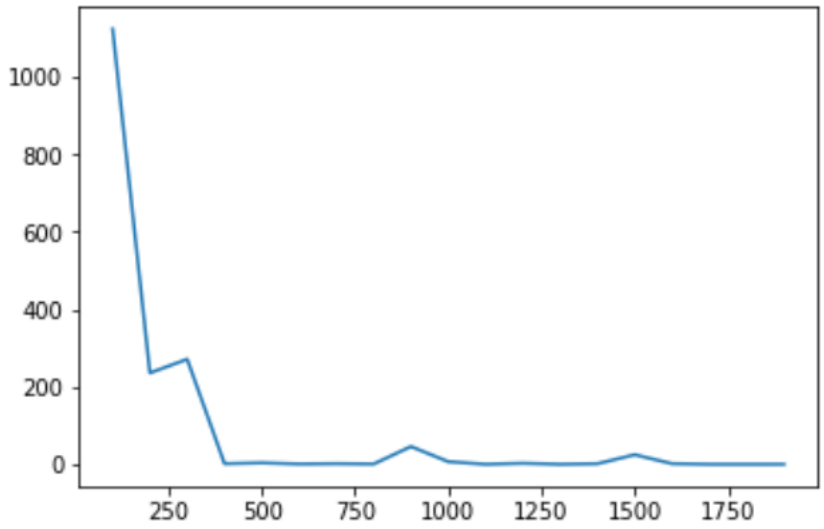

图 3.21 第六个函数种群大小迭代情况

由图表可知，函数 2 中的种群大小对函数最优值的求解有较大影响，当种群大小设置为[100,1700]之间时，都有小幅度的波动，即这块期间内得到局部最优解的概率较大，但是当种群大小为 1700 之后时，实现了精度在 0.01 以上的最小值，因此，种群数量的大小为 1700。

✧ cross_prob 交叉率，调参范围[0.6,0.99]，每 0.05 为一个 step。

表 3.22 第六个函数交叉率迭代情况

	tune_parma	result	generation	param
0	0.60	2.800182e-02	1061	[-0.157, -0.145, 0.0, -0.0, -0.078, -0.313, -0...
1	0.65	3.000000e-07	51	[0.001, 0.001, -0.0, -0.0, -0.001, 0.0, 0.0, 0...
2	0.70	2.547314e-02	35	[0.002, 0.002, -0.0, -0.002, -0.0, 0.0, 0.0, 0...
3	0.75	1.640340e-03	39	[0.036, 0.039, 0.0, 0.0, 0.0, -0.0, -0.078, -0...
4	0.80	1.585700e-03	36	[0.01, 0.009, 0.0, 0.0, -0.0, -0.078, 0.0, 0.0...
5	0.85	2.547298e-02	32	[0.0, -0.001, 0.0, 0.001, -0.0, -0.313, -0.001...
6	0.90	0.000000e+00	35	[0.0, 0.0, -0.0, -0.0, -0.0, 0.0, 0.0, 0.0, -0...
7	0.95	0.000000e+00	30	[0.0, -0.0, 0.0, 0.0, -0.0, -0.0, -0.0, 0.0, -...

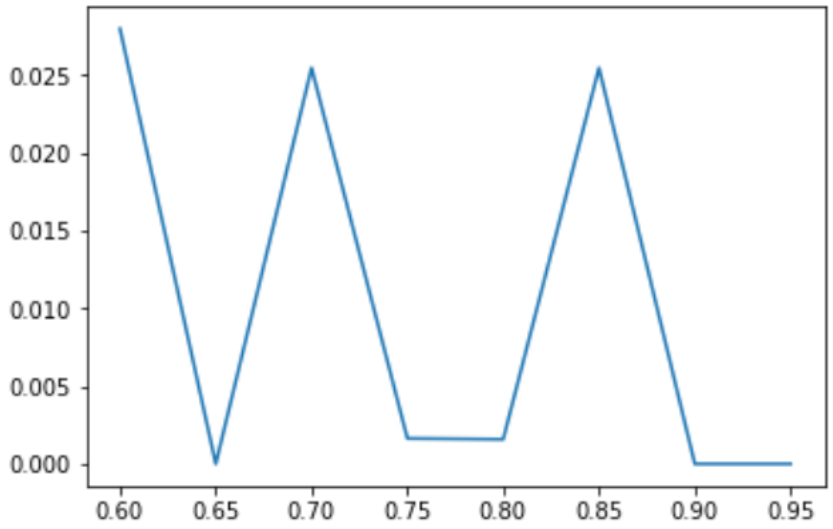

图 3.22 第六个函数交叉率迭代情况

由图表可知，当交叉率在 0.9 之前时，每次寻找到的最小值波动较大，这可能是因为交叉率过小，个体的变化不太，仍然存在较弱的基因，但当变异率为 0.9 时，实现了精度在 0.01 以上的最小值，因此，交叉率的大小为 0.9 即可。

✧ mutate_prob 变异率，调参范围[0.001,0.2],每 0.02 为一个 step。

表 3.23 第六个函数变异率迭代情况

	tune_parma	result	generation	param
0	0.20	9.615406e-01	34	[5.0, 4.615, -0.0, -0.0, -0.002, 0.0, 0.0, 0.0...
1	0.18	2.640000e-06	35	[0.001, 0.001, -0.003, -0.0, -0.001, 0.0, 0.0,...
2	0.16	1.260000e-06	42	[0.005, 0.005, -0.0, -0.0, 0.0, 0.0, 0.0, 0.00...
3	0.14	3.000000e-07	35	[0.001, 0.001, -0.0, -0.0, -0.001, 0.0, 0.0, 0...
4	0.12	2.760000e-06	35	[0.002, 0.002, 0.0, 0.0, 0.0, -0.003, -0.0, -0...
5	0.10	1.000000e-06	39	[0.005, 0.005, 0.0, 0.0, 0.0, -0.0, -0.0, -0.0...
6	0.08	2.600000e-07	39	[0.0, 0.0, -0.0, -0.0, -0.001, 0.0, 0.0, 0.0, ...
7	0.06	6.010476e-02	34	[1.151, 1.25, -0.005, -0.0, -0.0, 0.0, 0.0, 0....
8	0.04	4.200000e-07	36	[0.002, 0.002, -0.0, -0.0, -0.0, 0.0, 0.0, 0.0...
9	0.02	9.615388e-01	32	[5.0, 4.615, -0.0, -0.0, -0.0, 0.0, 0.0, 0.0, ...

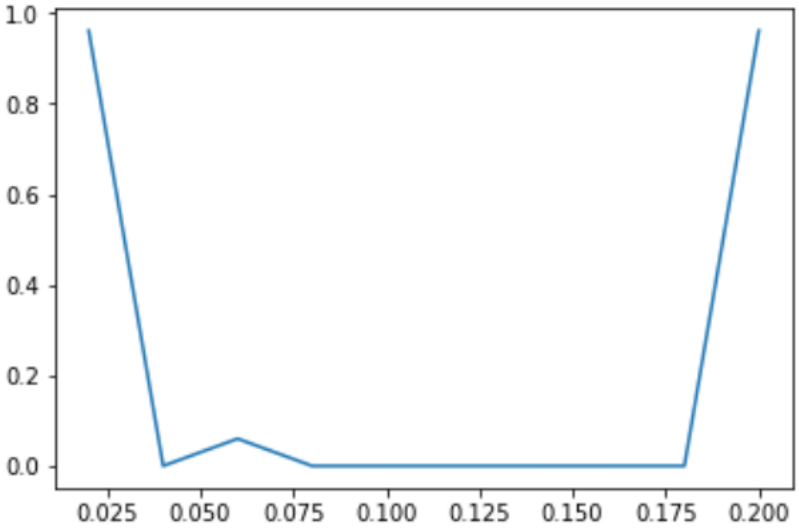

图 3.23 第六个函数变异率迭代情况

由图表可知，当变异率在[0.025,0.075]之间时，寻找到的最小值波动较大，这可能是因为变异率过小，个体的优异基因发生突变的数量很小，没有将劣质基因转换为强大的基因，但当变异率为 0.2 时，其收敛的数值与最优解相差太大，因此我们取[0.1,0.175]范围内的变异率，它们都实现了精度在 0.01 以上的最小值，因此，交叉率的大小为 0.1 即可。

✧ accuracy 精度，来表示二进制长度，调参范围[0.0001,0.001]，每 0.0001 为一个 step。

表 3.24 第六个函数精度迭代情况

	tune_parma	result	generation	param
0	0.0010	1.368355	46	[-2.885, -3.125, 0.0, 0.0, -0.0, -1.954, 0.0, ...
1	0.0009	1.368355	46	[-2.885, -3.125, 0.0, 0.0, -0.0, -1.954, 0.0, ...
2	0.0008	1.368355	46	[-2.885, -3.125, 0.0, 0.0, -0.0, -1.954, 0.0, ...
3	0.0007	0.000370	106	[0.098, 0.09, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, -...
4	0.0006	0.000370	106	[0.098, 0.09, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, -...
5	0.0005	0.000370	106	[0.098, 0.09, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, -...
6	0.0004	0.000370	106	[0.098, 0.09, 0.0, -0.0, 0.0, 0.0, 0.0, 0.0, -...
7	0.0003	0.375601	246	[-2.885, -3.125, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0...
8	0.0002	0.375601	246	[-2.885, -3.125, 0.0, 0.0, 0.0, -0.0, 0.0, 0.0...

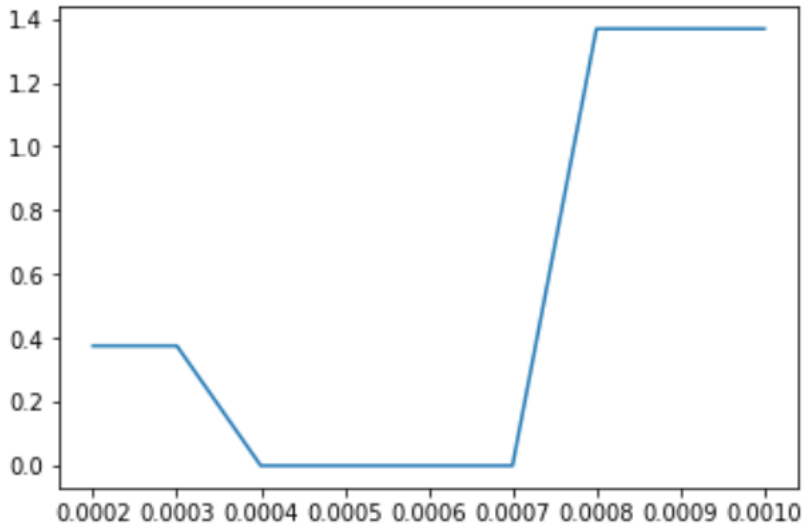

图 3.24 第六个函数精度迭代情况

由图表可知，当精度在 0.0004-0.007 时，最优值情况显著，都可以收敛到最优的情况，因此我们取 0.0005 为最终得精度。

✧ 选择方式，轮盘赌选择及竞争选择两种方式

竞争选择

前几次的迭代情况：

```
第-1次迭代 最佳拟合值:-21793.56327202 最佳个体所在代际:-1
第0次迭代 最佳拟合值:-1154.83642864 最佳个体所在代际:0
第1次迭代 最佳拟合值:-749.3022265400002 最佳个体所在代际:1
第2次迭代 最佳拟合值:-401.20906315999997 最佳个体所在代际:2
第3次迭代 最佳拟合值:-287.78229050000004 最佳个体所在代际:3
第4次迭代 最佳拟合值:-213.39542613999993 最佳个体所在代际:4
第5次迭代 最佳拟合值:-104.10402829999998 最佳个体所在代际:5
第6次迭代 最佳拟合值:-70.51391133999999 最佳个体所在代际:6
第7次迭代 最佳拟合值:-41.511835159999976 最佳个体所在代际:7
第8次迭代 最佳拟合值:-36.585144919999976 最佳个体所在代际:8
第9次迭代 最佳拟合值:-21.219303019999955 最佳个体所在代际:9
第10次迭代 最佳拟合值:-17.197227940000033 最佳个体所在代际:10
第11次迭代 最佳拟合值:-10.814652600000045 最佳个体所在代际:11
第12次迭代 最佳拟合值:-9.395538739999992 最佳个体所在代际:12
```

后几次的迭代情况：

```
第4985次迭代 最佳拟合值:-0.0 最佳个体所在代际:41
第4986次迭代 最佳拟合值:-0.0 最佳个体所在代际:41
第4987次迭代 最佳拟合值:-0.0 最佳个体所在代际:41
第4988次迭代 最佳拟合值:-0.0 最佳个体所在代际:41
第4989次迭代 最佳拟合值:-0.0 最佳个体所在代际:41
第4990次迭代 最佳拟合值:-0.0 最佳个体所在代际:41
第4991次迭代 最佳拟合值:-0.0 最佳个体所在代际:41
第4992次迭代 最佳拟合值:-0.0 最佳个体所在代际:41
第4993次迭代 最佳拟合值:-0.0 最佳个体所在代际:41
第4994次迭代 最佳拟合值:-0.0 最佳个体所在代际:41
第4995次迭代 最佳拟合值:-0.0 最佳个体所在代际:41
第4996次迭代 最佳拟合值:-0.0 最佳个体所在代际:41
第4997次迭代 最佳拟合值:-0.0 最佳个体所在代际:41
第4998次迭代 最佳拟合值:-0.0 最佳个体所在代际:41
第4999次迭代 最佳拟合值:-0.0 最佳个体所在代际:41
```

最终结果：最优结果 0.0；最优个体 [-0.0, 0.0, -0.0, -0.0, 0.0, 0.0, -0.0, -0.0, 0.0, -0.0]

轮盘赌选择

前几次的迭代情况：

```
第-1次迭代 最佳拟合值:-9572.05918826 最佳个体所在代际:-1
第0次迭代 最佳拟合值:-0.9987552399999998 最佳个体所在代际:0
第1次迭代 最佳拟合值:-0.9987552399999998 最佳个体所在代际:0
第2次迭代 最佳拟合值:-0.19214676000000008 最佳个体所在代际:2
第3次迭代 最佳拟合值:-0.19214676000000008 最佳个体所在代际:2
第4次迭代 最佳拟合值:-0.19214676000000008 最佳个体所在代际:2
第5次迭代 最佳拟合值:-0.017240660000000005 最佳个体所在代际:5
第6次迭代 最佳拟合值:-0.017240660000000005 最佳个体所在代际:5
第7次迭代 最佳拟合值:-0.017240660000000005 最佳个体所在代际:5
第8次迭代 最佳拟合值:-0.008269060000000005 最佳个体所在代际:8
第9次迭代 最佳拟合值:-0.008269060000000005 最佳个体所在代际:8
第10次迭代 最佳拟合值:-0.008138 最佳个体所在代际:10
第11次迭代 最佳拟合值:-0.008138 最佳个体所在代际:10
第12次迭代 最佳拟合值:-0.008138 最佳个体所在代际:10
第13次迭代 最佳拟合值:-0.008138 最佳个体所在代际:10
第14次迭代 最佳拟合值:-0.008138 最佳个体所在代际:10
第15次迭代 最佳拟合值:-0.008138 最佳个体所在代际:10
第16次迭代 最佳拟合值:-0.008138 最佳个体所在代际:10
第17次迭代 最佳拟合值:-0.008138 最佳个体所在代际:10
```

后几次的迭代情况：

```
第974次迭代 最佳拟合值:-0.0 最佳个体所在代际:146
第975次迭代 最佳拟合值:-0.0 最佳个体所在代际:146
第976次迭代 最佳拟合值:-0.0 最佳个体所在代际:146
第977次迭代 最佳拟合值:-0.0 最佳个体所在代际:146
第978次迭代 最佳拟合值:-0.0 最佳个体所在代际:146
第979次迭代 最佳拟合值:-0.0 最佳个体所在代际:146
第980次迭代 最佳拟合值:-0.0 最佳个体所在代际:146
第981次迭代 最佳拟合值:-0.0 最佳个体所在代际:146
第982次迭代 最佳拟合值:-0.0 最佳个体所在代际:146
第983次迭代 最佳拟合值:-0.0 最佳个体所在代际:146
第984次迭代 最佳拟合值:-0.0 最佳个体所在代际:146
第985次迭代 最佳拟合值:-0.0 最佳个体所在代际:146
第986次迭代 最佳拟合值:-0.0 最佳个体所在代际:146
第987次迭代 最佳拟合值:-0.0 最佳个体所在代际:146
第988次迭代 最佳拟合值:-0.0 最佳个体所在代际:146
第989次迭代 最佳拟合值:-0.0 最佳个体所在代际:146
第990次迭代 最佳拟合值:-0.0 最佳个体所在代际:146
第991次迭代 最佳拟合值:-0.0 最佳个体所在代际:146
```

最终结果 0.0。即通过上述显示，我们可以看出，无论在迭代代数还是在训练时间上看，竞争选择都是优于轮盘赌的。

✧ 最优参数下的求解情况

最优状况下的参数情况：

In [43]:
```
# 迭代次数000次
generation_size=5000
# 群体大小
population_size=1700
# 染色体个数
chrom_num = 2
cross_prob=0.9
mutate_prob = 0.1
choose_type='tournament'
#choose_type='wheel'
object_fun=object_fun
lower_bound=-100
upper_bound=100
accuracy=0.0005
is_print=True
```

最终结果为：0.0，无限接近于最优值 0。

4. 其他演化算法-粒子群算法

4.1 粒子群算法原理

粒子群算法（Particle swarm optimization，PSO）是模拟群体智能所建立起来的一种优化算法，主要用于解决最优化问题（optimization problems）。 PSO 算法将群体中的每个个体看作 n 维搜索空间的粒子，粒子有自己的位置和速度，粒子通过在群体中的合作与竞争产生的群体智能指导优化搜索。

举个例子说明一下：假设一群鸟在觅食，在觅食范围内，只在一个地方有食物，所有鸟儿都看不到食物，即不知道食物的具体位置。但是能闻到食物的味道，能知道食物距离自己是远是近。假设鸟与鸟之间能共享信息，即互相知道每个鸟离食物多远。那么最好的策略就是结合自己离食物最近的位置和鸟群中其他鸟距离食物最近的位置这 2 个因素综合考虑找到最好的搜索位置。

与遗传算法类似，PSO 也有几个核心概念：

- 粒子（particle）：一只鸟。类似于遗传算法中的个体。
- 种群（population）：一群鸟。类似于遗传算法中的种群。
- 位置（position）：一个粒子（鸟）当前所在的位置。
- 经验（best）：一个粒子（鸟）自身曾经离食物最近的位置。
- 速度（velocity ）：一个粒子（鸟）飞行的速度。
- 适应度（fitness）：一个粒子（鸟）距离食物的远近。与遗传算法中的适应度类似。

4.2 粒子群算法实现

粒子群算法原理流程图解：

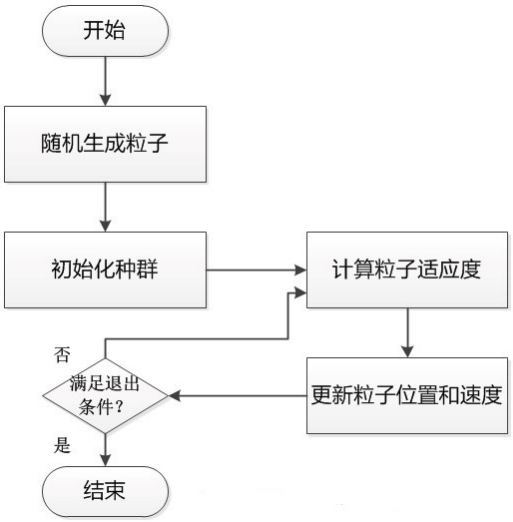

图 4.1 粒子群算法流程图

1）根据问题需要，随机生成粒子，粒子的数量可自行控制。
2）将粒子组成一个种群。这前 2 个过程一般合并在一起。
3）计算粒子适应度值。
4）更新种群中每个粒子的位置和速度。
5）满足退出条件就退出，不满足就转向步骤 3）。

Python 实现过程：

1）创建粒子类，这是用来解决粒子得存储问题，因为 python 中没有相应得数据结构，因此我们选择一个类来表示粒子结构，主要包括惯性权重、个体学习因子、社会学习因子、种群规模、最大迭代次数、速度极值、搜索极值即优化维度等。
2）创建 POS 类，这是粒子群算法得主体部分，包括初始化种群、速度更新、粒子位置更新、适应度更新的主要函数
2）与遗传算法类似，定义各个目标函数，并通过 test 函数进行测试。

4.3 遗传算法与粒子群算法的比较

以第一个函数为例，我们首先将粒子算法和遗传算法共有的参数，如迭代次数与种群大小设置为同样的数值，generations＝5000，population_size＝1300，结果如下图所示，很明显在 200 代以内其便达到了最优值 0。因此，我们通过不断地尝试，本着能找到最优值和尽可能快速收敛的原则，将它们定义为：generations＝200，population_size＝100

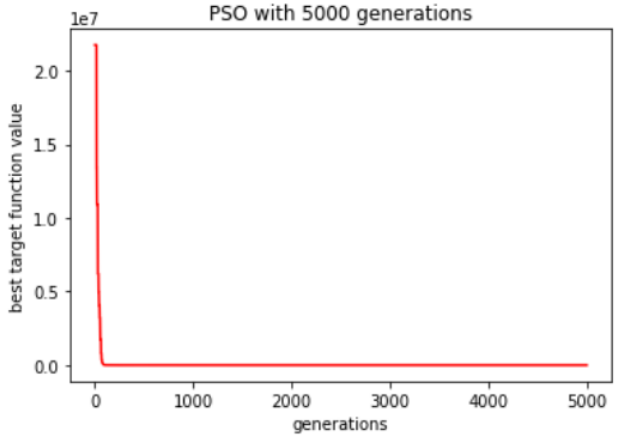

图 4.2 第一个函数 5000 代时情况

A. 第一个函数
遗传算法的情况：在第 45 次迭代处达到最佳拟合值 1e-06，收敛速度很快
粒子群算法情况：在第 80 代左右达到最佳拟合值 4.6073957063028955e-10
B. 第二个函数
遗传算法的情况：在第 39 次迭代处达到最佳拟合值 2e-06，收敛速度很快

粒子群算法情况：在第 17 代左右达到最佳拟合值 8.079954937589673e-12

C. 第三个函数

遗传算法的情况：在第 8 次迭代处达到最佳拟合值 0，收敛速度很快

粒子群算法情况：在第 100 代左右达到最佳拟合值 0

D. 第四个函数

遗传算法的情况：在第 10 次迭代处达到最佳拟合值 0，收敛速度很快

粒子群算法情况：在第 85 代左右达到最佳拟合值 0

E. 第五个函数

遗传算法的情况：在第 12 次迭代达到最佳拟合值-0.9999995022068257，收敛速度快

粒子群算法情况：在第 75 代左右达到最佳拟合值-1.0

F. 第六个函数

遗传算法的情况：在第 39 次迭代处达到最佳拟合值 2e-06，收敛速度很快

粒子群算法情况：在第 25 代左右达到最佳拟合值 3.2476788811615966e-10

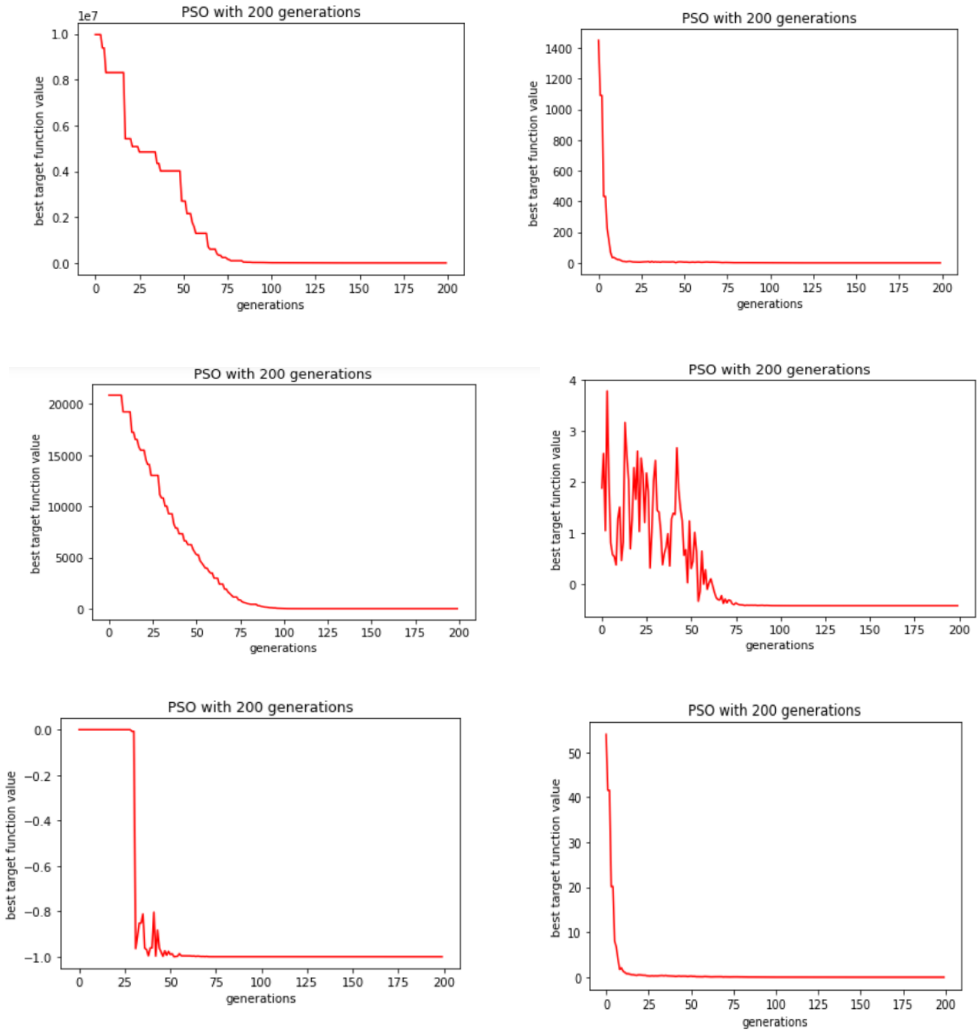

图 4.3 粒子群算法求解 6 个函数的情况

综上所述：在求解最优值问题上，不论是粒子群算法还是遗传算法，其都能很快收敛到最优值，但从迭代次数上来，遗传算法的迭代次数要明显小于粒子群算法。