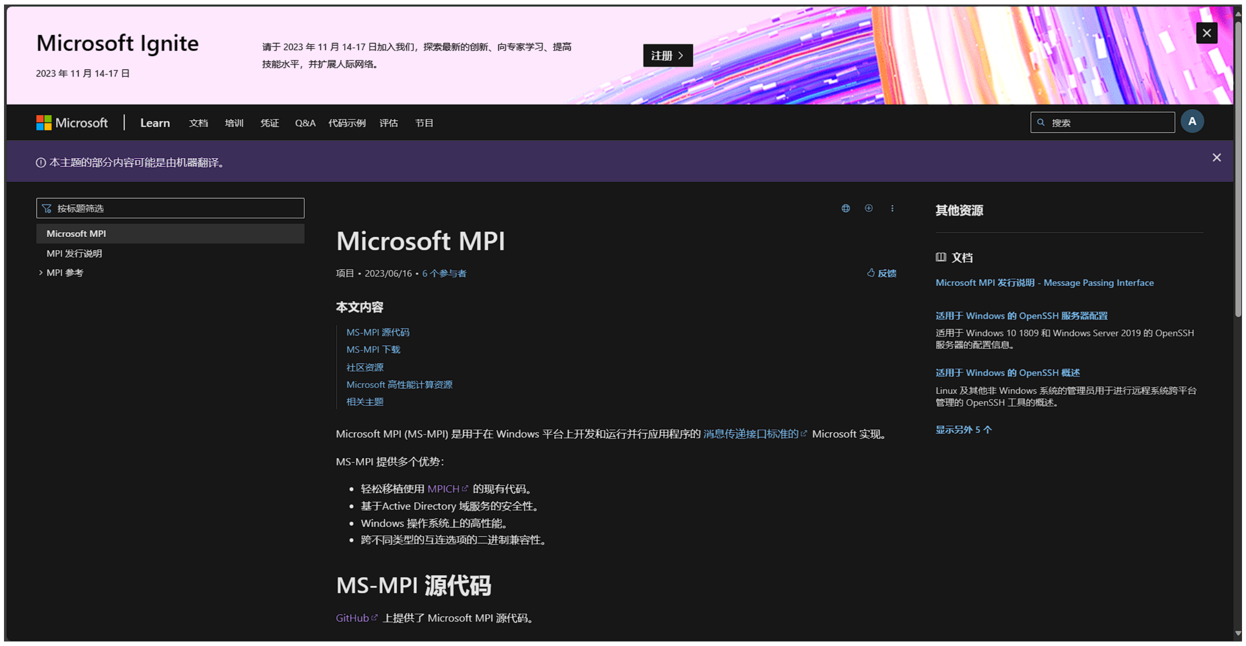
```
from mpi4py import MPI
```

安装相应的包

对于Windows：

额外前往Microsoft MPI 下载 MPI



ⓘ 本主题的部分内容可能是由机器翻译。

Microsoft MPI

项目 • 2023/06/16 • 6 个参与者

本文内容

- MS-MPI 源代码
- MS-MPI 下载
- 社区资源
- Microsoft 高性能计算资源
- 相关主题

Microsoft MPI (MS-MPI) 是用于在 Windows 平台上开发和运行并行应用程序的 消息传递接口标准的 Microsoft 实现。

MS-MPI 提供多个优势：

- 轻松移植使用 MPICH 的现有代码。
- 基于Active Directory 域服务的安全性。
- Windows 操作系统上的高性能。
- 跨不同类型的互连选项的二进制兼容性。

MS-MPI 源代码

GitHub 上提供了 Microsoft MPI 源代码。

其他资源

文档

Microsoft MPI 发行说明 - Message Passing Interface

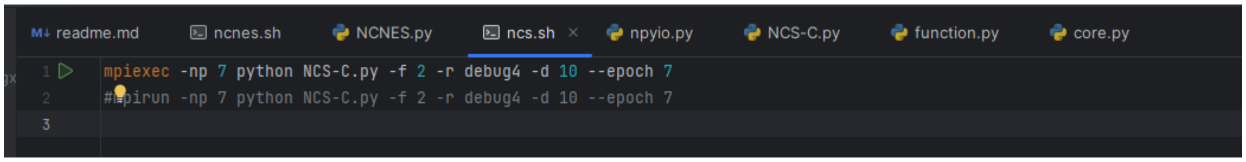
适用于 Windows 的 OpenSSH 服务器配置

适用于 Windows 10 1809 和 Windows Server 2019 的 OpenSSH 服务器的配置信息。

适用于 Windows 的 OpenSSH 概述

Linux 及其他非 Windows 系统的管理员用于进行远程系统跨平台管理的 OpenSSH 工具的概述。

显示另外 5 个

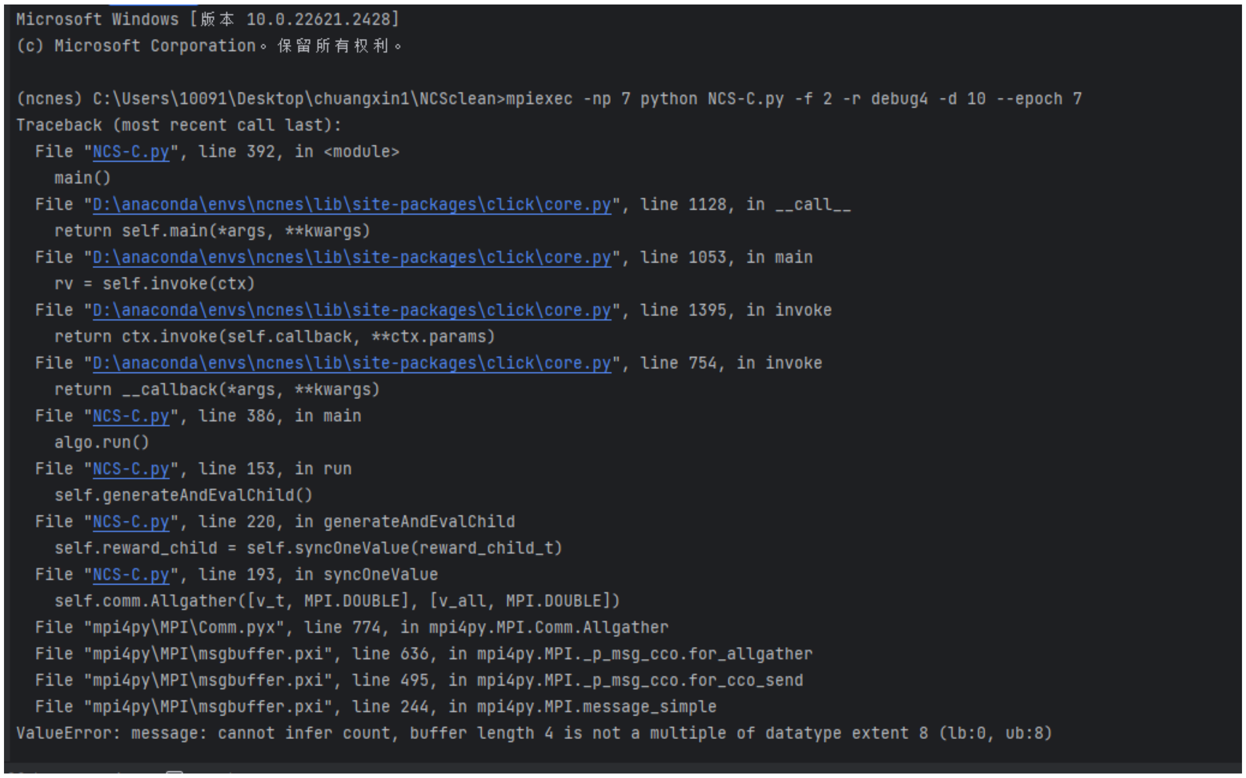


```
mpiexec -np 7 python NCS-C.py -f 2 -r debug4 -d 10 --epoch 7
#mpirun -np 7 python NCS-C.py -f 2 -r debug4 -d 10 --epoch 7

```

基于不同的系统。我们需要执行相应的代码, 他们本质上没有什么区别

mpiexec - > Windows

mpirun - > Linux



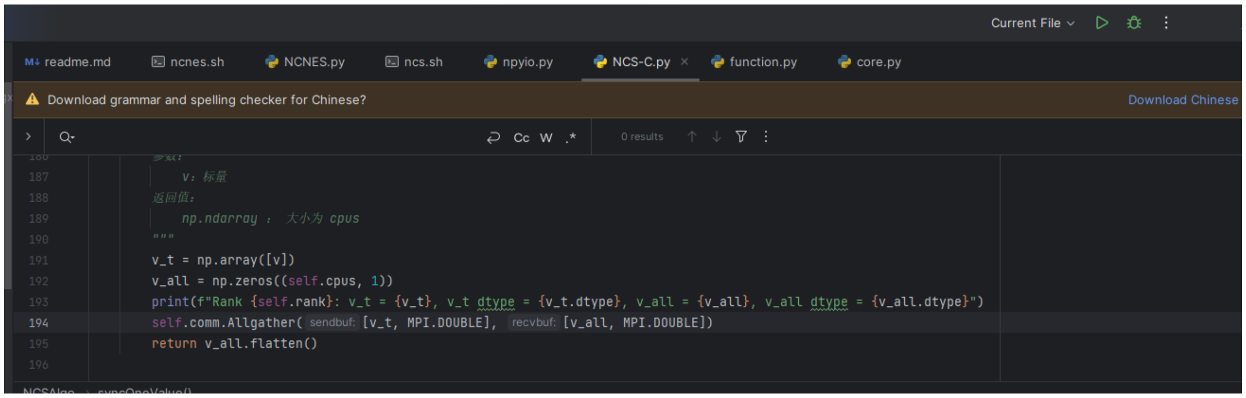
```
Microsoft Windows [版本 10.0.22621.2428]
(c) Microsoft Corporation。保留所有权利。

(ncnes) C:\Users\10091\Desktop\chuangxin1\NCSclean>mpiexec -np 7 python NCS-C.py -f 2 -r debug4 -d 10 --epoch 7
Traceback (most recent call last):
  File "NCS-C.py", line 392, in <module>
    main()
  File "D:\anaconda\envs\ncnes\lib\site-packages\click\core.py", line 1128, in __call__
    return self.main(*args, **kwargs)
  File "D:\anaconda\envs\ncnes\lib\site-packages\click\core.py", line 1053, in main
    rv = self.invoke(ctx)
  File "D:\anaconda\envs\ncnes\lib\site-packages\click\core.py", line 1395, in invoke
    return ctx.invoke(self.callback, **ctx.params)
  File "D:\anaconda\envs\ncnes\lib\site-packages\click\core.py", line 754, in invoke
    return __callback(*args, **kwargs)
  File "NCS-C.py", line 386, in main
    algo.run()
  File "NCS-C.py", line 153, in run
    self.generateAndEvalChild()
  File "NCS-C.py", line 220, in generateAndEvalChild
    self.reward_child = self.syncOneValue(reward_child_t)
  File "NCS-C.py", line 193, in syncOneValue
    self.comm.Allgather([v_t, MPI.DOUBLE], [v_all, MPI.DOUBLE])
  File "mpi4py\MPI\Comm.pyx", line 774, in mpi4py.MPI.Comm.Allgather
  File "mpi4py\MPI\msgbuffer.pxi", line 636, in mpi4py.MPI._p_msg_cco.for_allgather
  File "mpi4py\MPI\msgbuffer.pxi", line 495, in mpi4py.MPI._p_msg_cco.for_cco_send
  File "mpi4py\MPI\msgbuffer.pxi", line 244, in mpi4py.MPI.message_simple
ValueError: message: cannot infer count, buffer length 4 is not a multiple of datatype extent 8 (lb:0, ub:8)
```

在Windows环境下会出现以上的报错，在Linux 似乎没有

系统认为

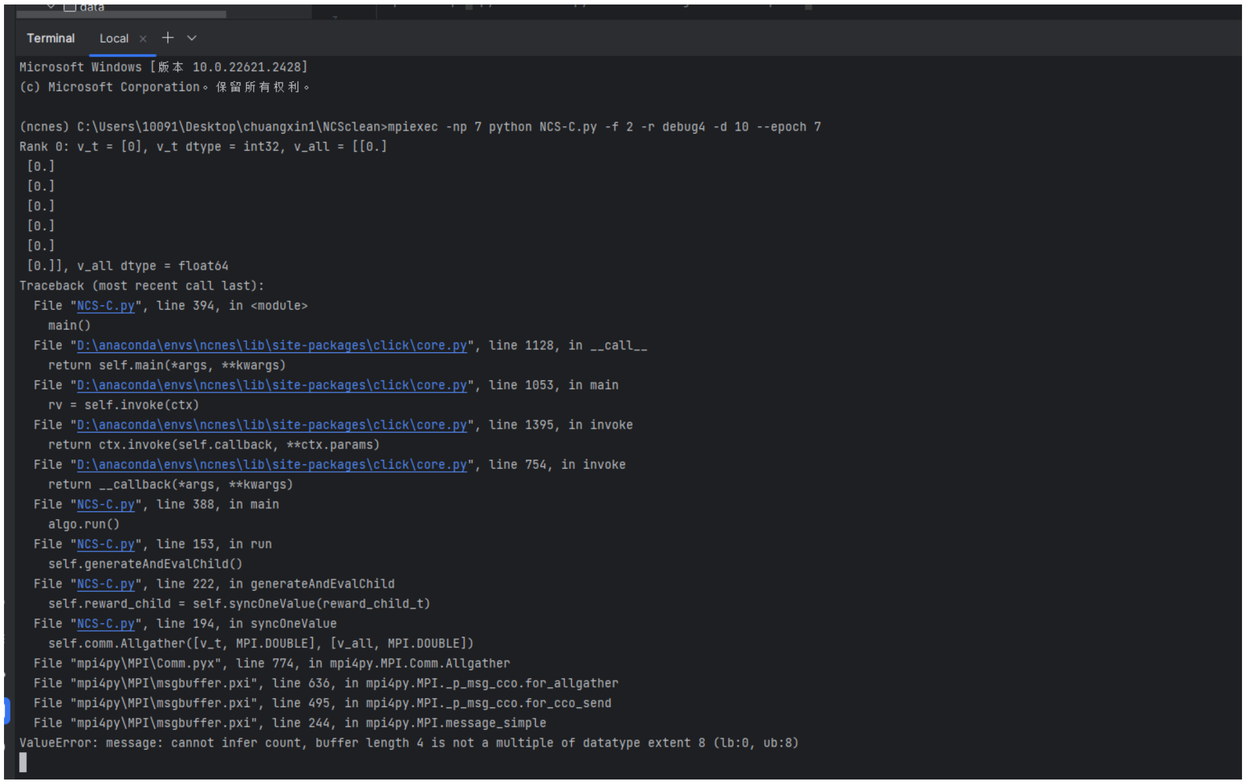
```
self.comm.Allgather([v_t, MPI.DOUBLE], [v_all, MPI.DOUBLE])
```

中 MPI.DOUBLE期望 返回 8个字节，但是v_t 返回4个



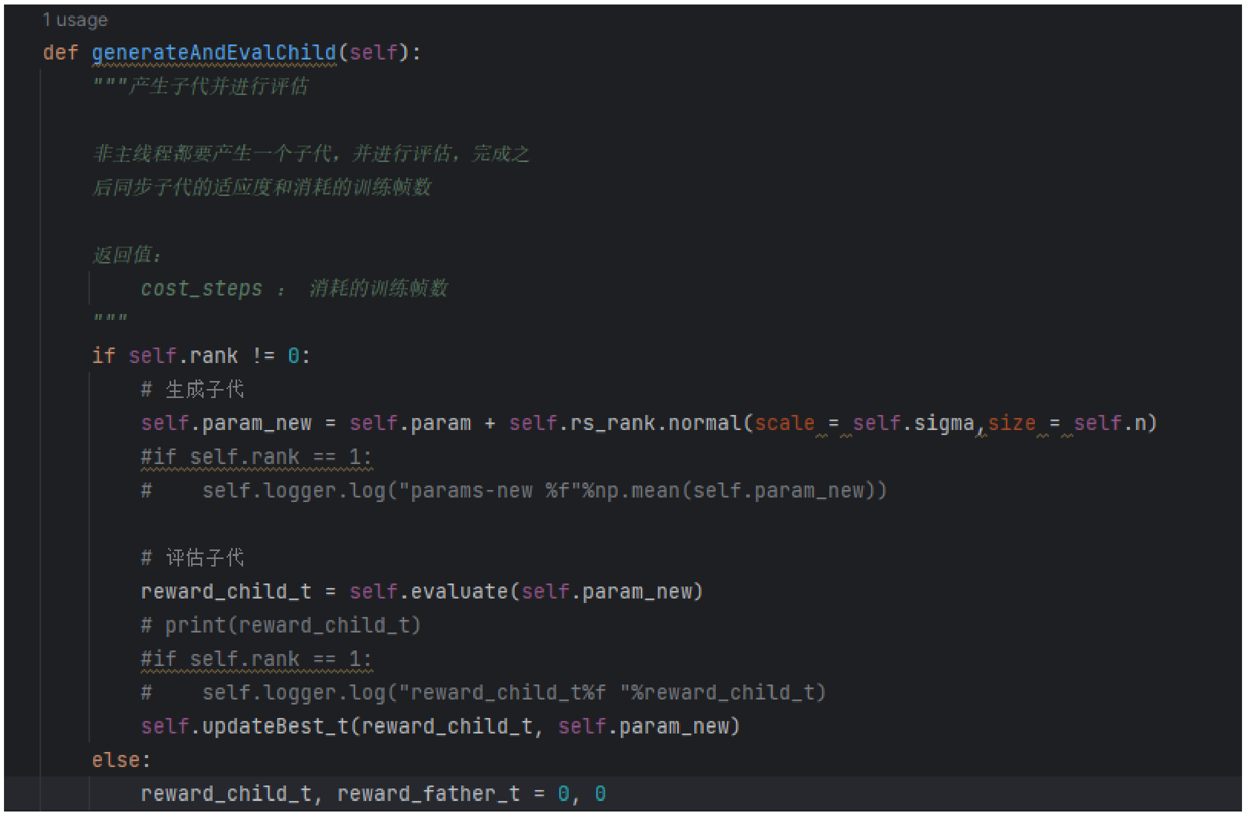
```
            v: 标量
        返回值:
            np.ndarray : 大小为 cpus
        """
        v_t = np.array([v])
        v_all = np.zeros((self.cpus, 1))
        print(f"Rank {self.rank}: v_t = {v_t}, v_t dtype = {v_t.dtype}, v_all = {v_all}, v_all dtype = {v_all.dtype}")
        self.comm.Allgather( sendbuf: [v_t, MPI.DOUBLE],  recvbuf: [v_all, MPI.DOUBLE])
        return v_all.flatten()
```

我们尝试print数组中的内容

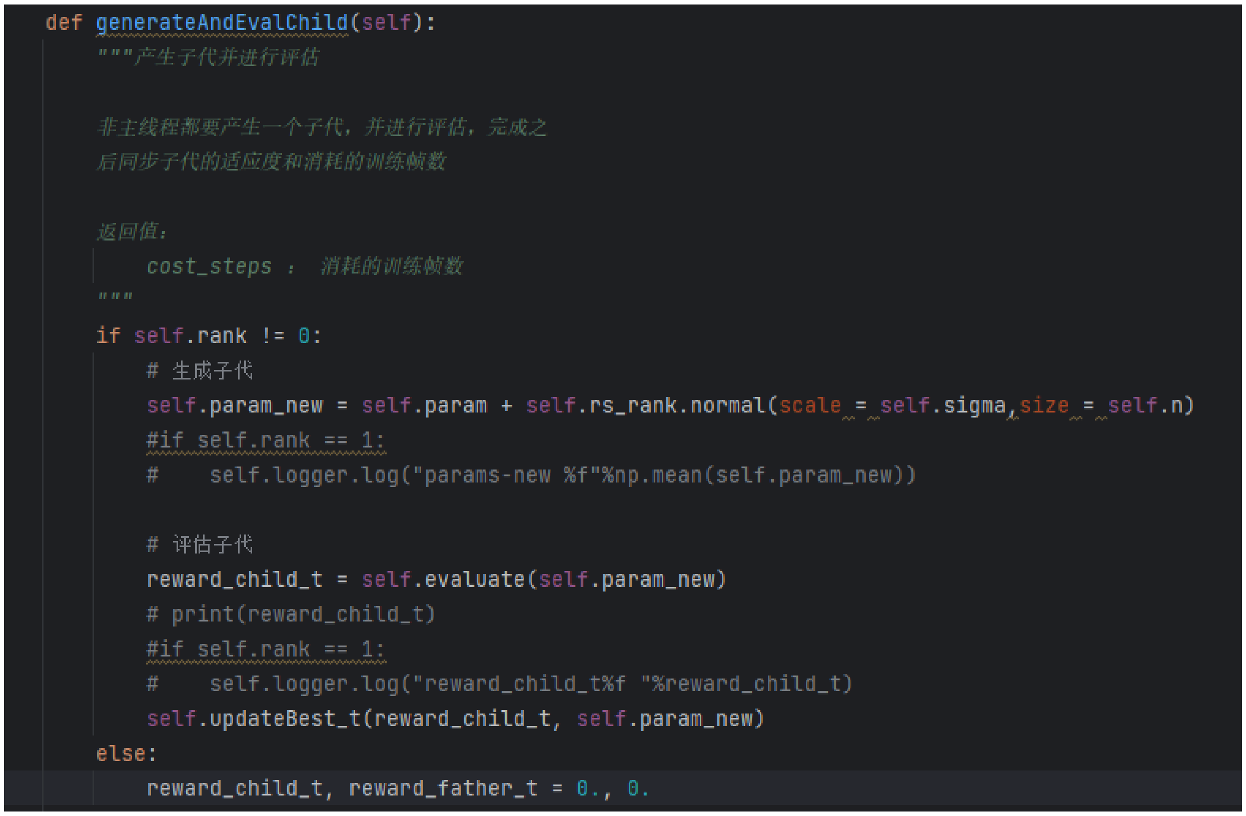


```
Microsoft Windows [版本 10.0.22621.2428]
(c) Microsoft Corporation。保留所有权利。

(ncnes) C:\Users\10091\Desktop\chuangxin1\NCSclean>mpiexec -np 7 python NCS-C.py -f 2 -r debug4 -d 10 --epoch 7
Rank 0: v_t = [0], v_t dtype = int32, v_all = [[0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]], v_all dtype = float64
Traceback (most recent call last):
  File "NCS-C.py", line 394, in <module>
    main()
  File "D:\anaconda\envs\ncnes\lib\site-packages\click\core.py", line 1128, in __call__
    return self.main(*args, **kwargs)
  File "D:\anaconda\envs\ncnes\lib\site-packages\click\core.py", line 1053, in main
    rv = self.invoke(ctx)
  File "D:\anaconda\envs\ncnes\lib\site-packages\click\core.py", line 1395, in invoke
    return ctx.invoke(self.callback, **ctx.params)
  File "D:\anaconda\envs\ncnes\lib\site-packages\click\core.py", line 754, in invoke
    return __callback(*args, **kwargs)
  File "NCS-C.py", line 388, in main
    algo.run()
  File "NCS-C.py", line 153, in run
    self.generateAndEvalChild()
  File "NCS-C.py", line 222, in generateAndEvalChild
    self.reward_child = self.syncOneValue(reward_child_t)
  File "NCS-C.py", line 194, in syncOneValue
    self.comm.Allgather([v_t, MPI.DOUBLE], [v_all, MPI.DOUBLE])
  File "mpi4py\MPI\Comm.pyx", line 774, in mpi4py.MPI.Comm.Allgather
  File "mpi4py\MPI\msgbuffer.pxi", line 636, in mpi4py.MPI._p_msg_cco.for_allgather
  File "mpi4py\MPI\msgbuffer.pxi", line 495, in mpi4py.MPI._p_msg_cco.for_cco_send
  File "mpi4py\MPI\msgbuffer.pxi", line 244, in mpi4py.MPI.message_simple
ValueError: message: cannot infer count, buffer length 4 is not a multiple of datatype extent 8 (lb:0, ub:8)
```

发现0好像没有自动转化为0.

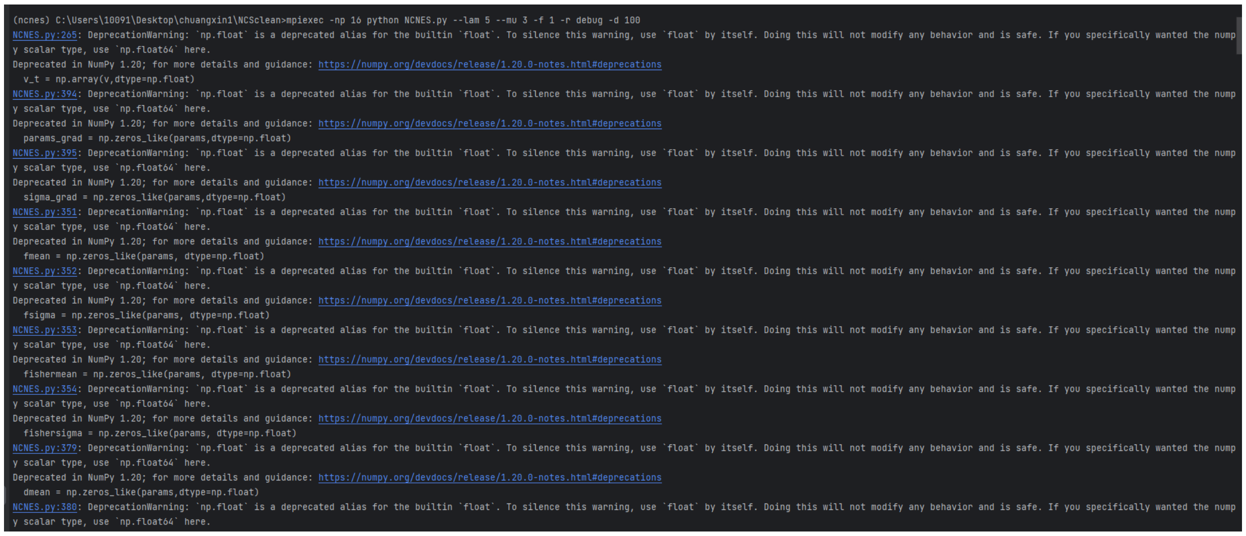


```
1 usage
def generateAndEvalChild(self):
    """产生子代并进行评估

    非主线程都要产生一个子代，并进行评估，完成之
    后同步子代的适应度和消耗的训练帧数

    返回值：
        cost_steps ： 消耗的训练帧数
    """
    if self.rank != 0:
        # 生成子代
        self.param_new = self.param + self.rs_rank.normal(scale = self.sigma,size = self.n)
        #if self.rank == 1:
        #    self.logger.log("params-new %f"%np.mean(self.param_new))

        # 评估子代
        reward_child_t = self.evaluate(self.param_new)
        # print(reward_child_t)
        #if self.rank == 1:
        #    self.logger.log("reward_child_t%f "%reward_child_t)
        self.updateBest_t(reward_child_t, self.param_new)
    else:
        reward_child_t, reward_father_t = 0, 0
```



```python
def generateAndEvalChild(self):
    """产生子代并进行评估

    非主线程都要产生一个子代，并进行评估，完成之
    后同步子代的适应度和消耗的训练帧数

    返回值：
        cost_steps ： 消耗的训练帧数
    """
    if self.rank != 0:
        # 生成子代
        self.param_new = self.param + self.rs_rank.normal(scale = self.sigma,size = self.n)
        #if self.rank == 1:
        #    self.logger.log("params-new %f"%np.mean(self.param_new))

        # 评估子代
        reward_child_t = self.evaluate(self.param_new)
        # print(reward_child_t)
        #if self.rank == 1:
        #    self.logger.log("reward_child_t%f "%reward_child_t)
        self.updateBest_t(reward_child_t, self.param_new)
    else:
        reward_child_t, reward_father_t = 0., 0.
```

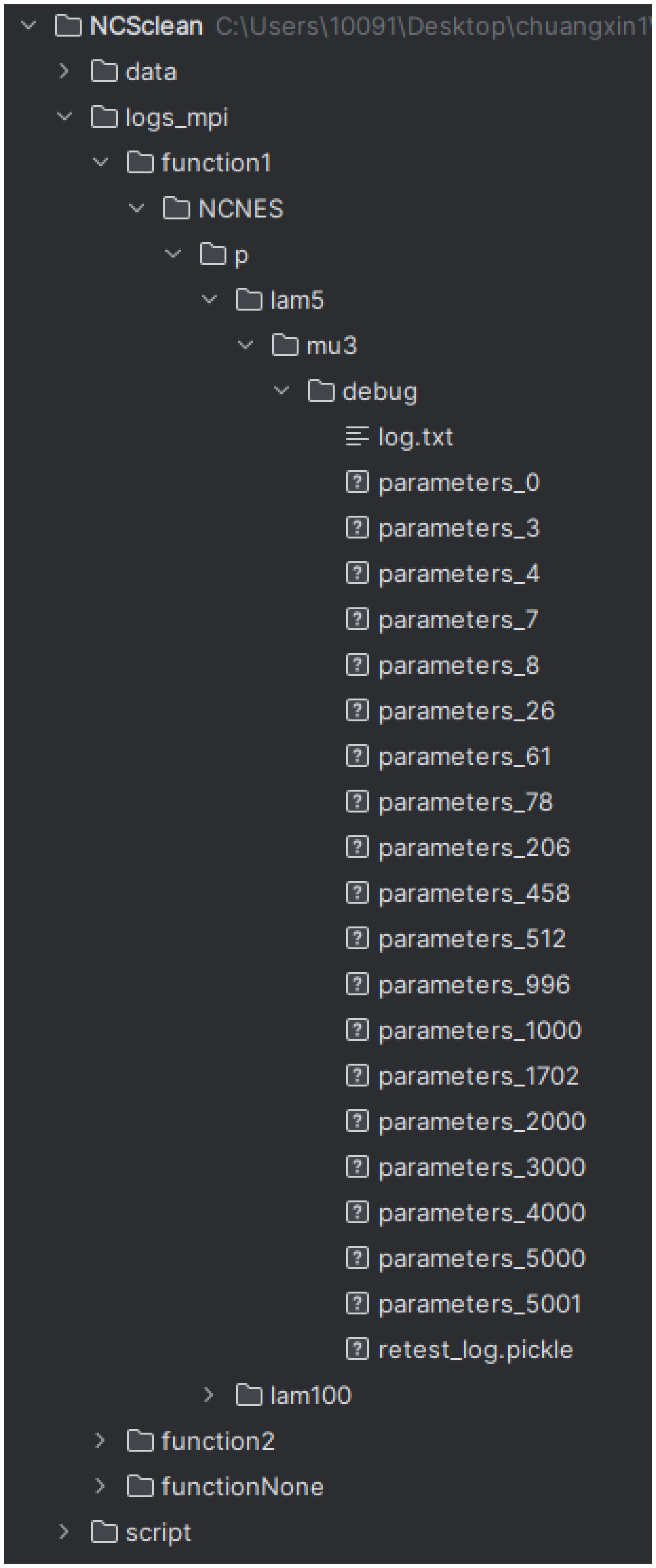
修改后我们继续测试



```
(ncnes) C:\Users\10091\Desktop\chuangxin1\NCSclean>mpiexec -np 16 python NCNES.py --lam 5 --mu 3 -f 1 -r debug -d 100
NCNES.py:265: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the nump
y scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  v_t = np.array(v,dtype=np.float)
NCNES.py:394: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the nump
y scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  params_grad = np.zeros_like(params,dtype=np.float)
NCNES.py:395: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the nump
y scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  sigma_grad = np.zeros_like(params,dtype=np.float)
NCNES.py:351: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the nump
y scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  fmean = np.zeros_like(params, dtype=np.float)
NCNES.py:352: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the nump
y scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  fsigma = np.zeros_like(params, dtype=np.float)
NCNES.py:353: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the nump
y scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  fishermean = np.zeros_like(params, dtype=np.float)
NCNES.py:354: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the nump
y scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  fishersigma = np.zeros_like(params, dtype=np.float)
NCNES.py:379: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the nump
y scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  dmean = np.zeros_like(params,dtype=np.float)
NCNES.py:380: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the nump
y scalar type, use `np.float64` here.
```

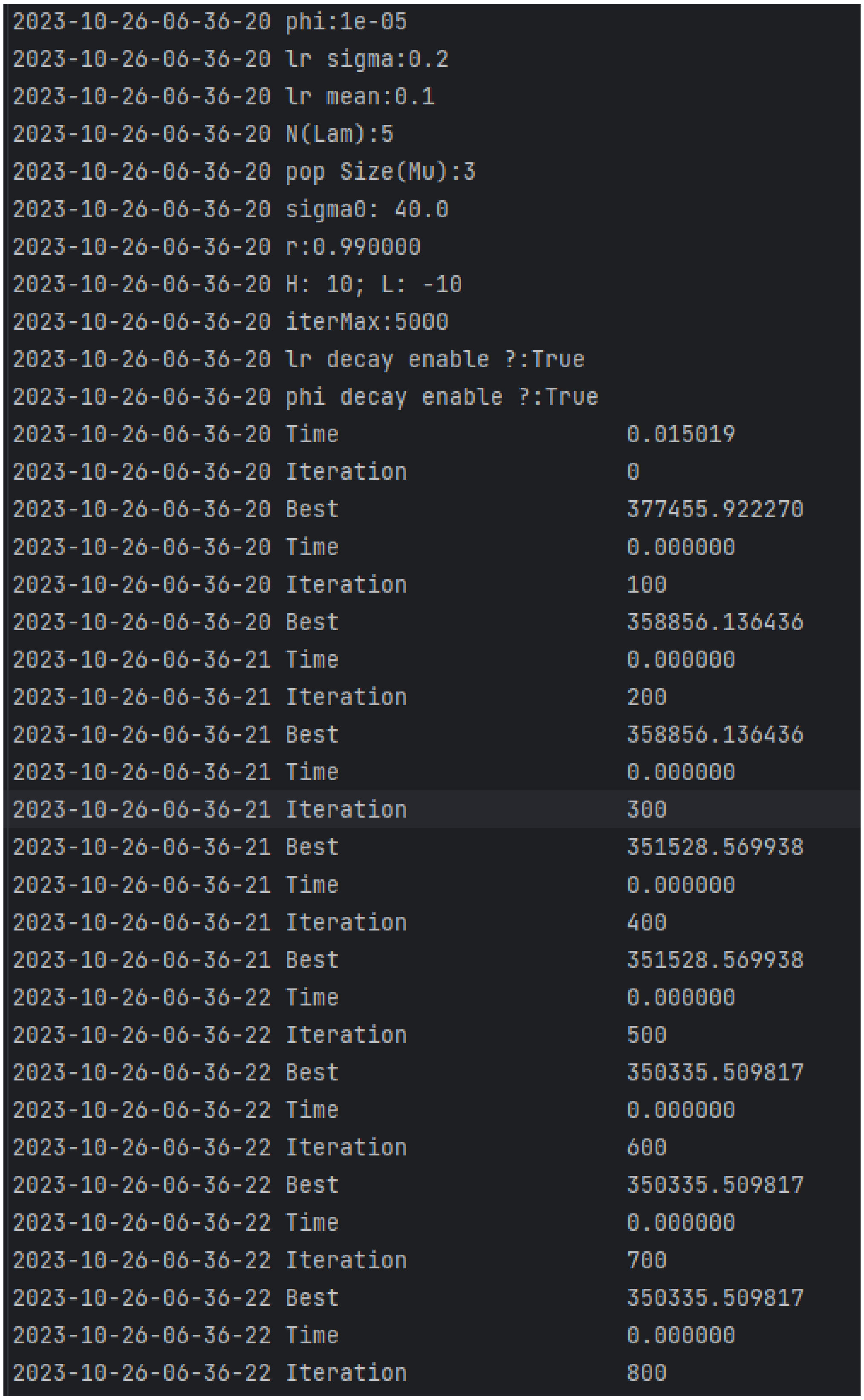
发现代码中 np.float 过时， 应该修改为 np.float64 （or 32）

运行代码，我们发现已经能够输出相应的数据



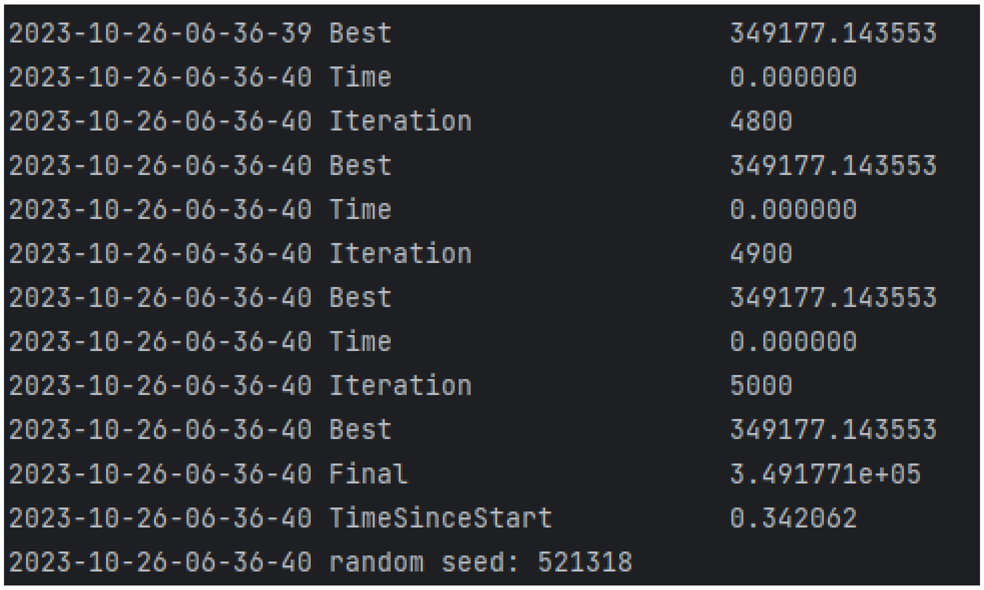
```
NCSclean  C:\Users\10091\Desktop\chuangxin1\
  data
  logs_mpi
    function1
      NCNES
        p
          lam5
            mu3
              debug
                log.txt
                parameters_0
                parameters_3
                parameters_4
                parameters_7
                parameters_8
                parameters_26
                parameters_61
                parameters_78
                parameters_206
                parameters_458
                parameters_512
                parameters_996
                parameters_1000
                parameters_1702
                parameters_2000
                parameters_3000
                parameters_4000
                parameters_5000
                parameters_5001
                retest_log.pickle
          lam100
    function2
    functionNone
  script
```

运行代码，我们发现已经能够输出相应的数据



```
2023-10-26-06-36-20 phi:1e-05
2023-10-26-06-36-20 lr sigma:0.2
2023-10-26-06-36-20 lr mean:0.1
2023-10-26-06-36-20 N(Lam):5
2023-10-26-06-36-20 pop Size(Mu):3
2023-10-26-06-36-20 sigma0: 40.0
2023-10-26-06-36-20 r:0.990000
2023-10-26-06-36-20 H: 10; L: -10
2023-10-26-06-36-20 iterMax:5000
2023-10-26-06-36-20 lr decay enable ?:True
2023-10-26-06-36-20 phi decay enable ?:True
2023-10-26-06-36-20 Time                    0.015019
2023-10-26-06-36-20 Iteration               0
2023-10-26-06-36-20 Best                    377455.922270
2023-10-26-06-36-20 Time                    0.000000
2023-10-26-06-36-20 Iteration               100
2023-10-26-06-36-20 Best                    358856.136436
2023-10-26-06-36-21 Time                    0.000000
2023-10-26-06-36-21 Iteration               200
2023-10-26-06-36-21 Best                    358856.136436
2023-10-26-06-36-21 Time                    0.000000
2023-10-26-06-36-21 Iteration               300
2023-10-26-06-36-21 Best                    351528.569938
2023-10-26-06-36-21 Time                    0.000000
2023-10-26-06-36-21 Iteration               400
2023-10-26-06-36-21 Best                    351528.569938
2023-10-26-06-36-22 Time                    0.000000
2023-10-26-06-36-22 Iteration               500
2023-10-26-06-36-22 Best                    350335.509817
2023-10-26-06-36-22 Time                    0.000000
2023-10-26-06-36-22 Iteration               600
2023-10-26-06-36-22 Best                    350335.509817
2023-10-26-06-36-22 Time                    0.000000
2023-10-26-06-36-22 Iteration               700
2023-10-26-06-36-22 Best                    350335.509817
2023-10-26-06-36-22 Time                    0.000000
2023-10-26-06-36-22 Iteration               800
```

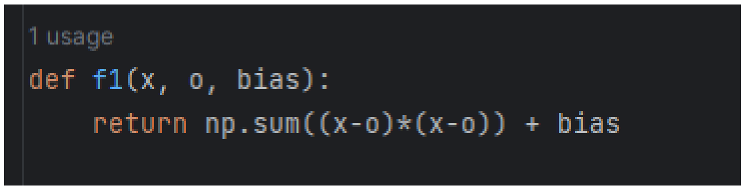
我们观察内容，认为可能当参数获得了更好的值后，才会进行一个保存



```
2023-10-26-06-36-39 Best                        349177.143553
2023-10-26-06-36-40 Time                        0.000000
2023-10-26-06-36-40 Iteration                   4800
2023-10-26-06-36-40 Best                        349177.143553
2023-10-26-06-36-40 Time                        0.000000
2023-10-26-06-36-40 Iteration                   4900
2023-10-26-06-36-40 Best                        349177.143553
2023-10-26-06-36-40 Time                        0.000000
2023-10-26-06-36-40 Iteration                   5000
2023-10-26-06-36-40 Best                        349177.143553
2023-10-26-06-36-40 Final                       3.491771e+05
2023-10-26-06-36-40 TimeSinceStart              0.342062
2023-10-26-06-36-40 random seed: 521318
```

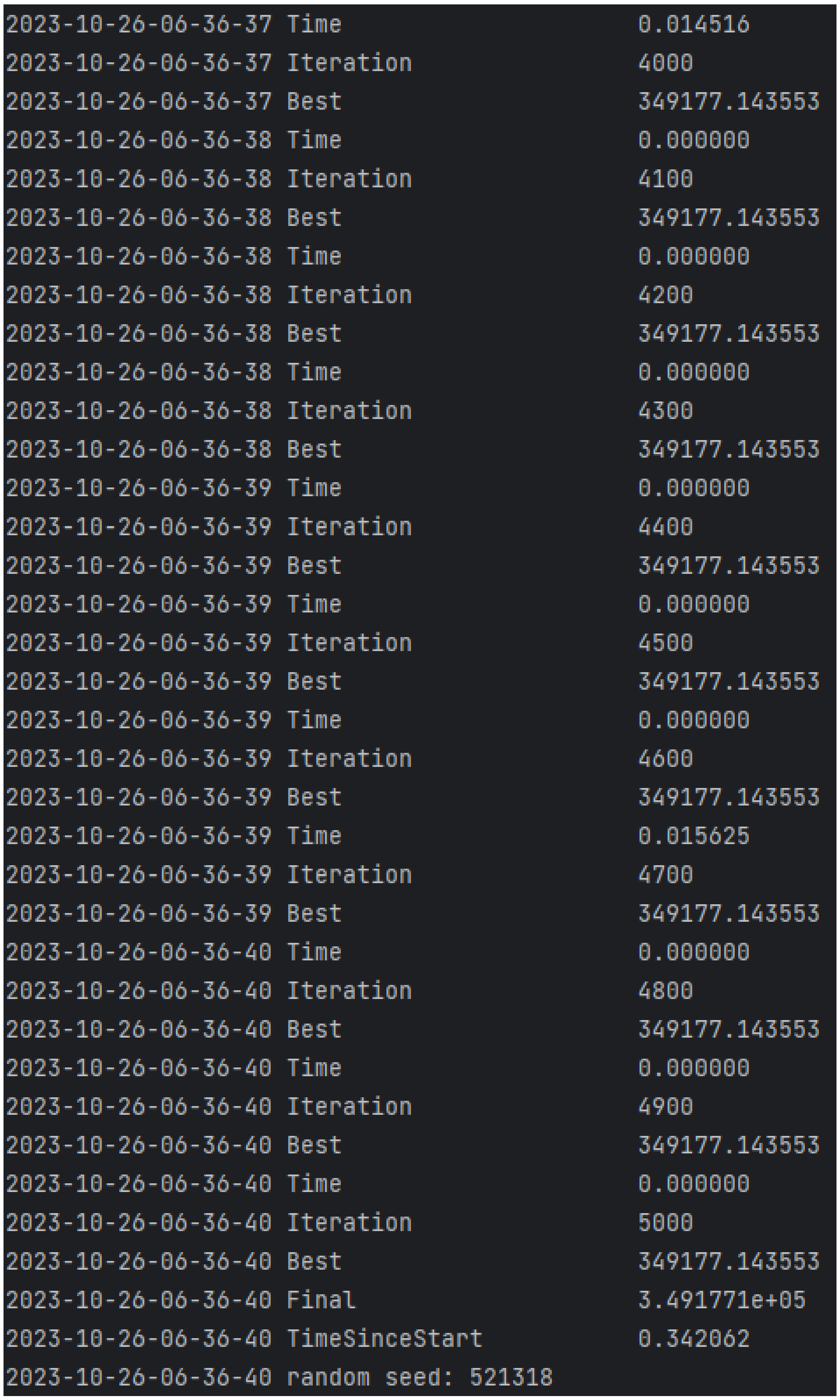
以上是一个基于f（1）函数的测试，

我们观察f（1）函数



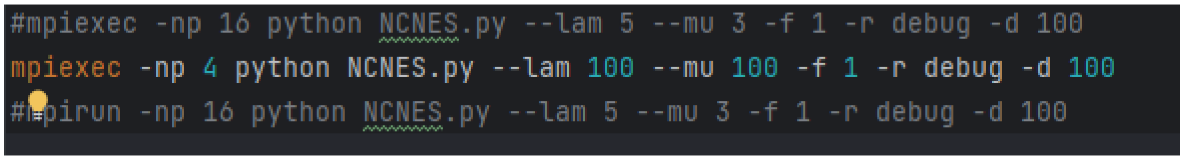
```
1 usage
def f1(x, o, bias):
    return np.sum((x-o)*(x-o)) + bias
```

可以发现，它可以被认为是一个二次函数，那么此时得到的best答案似乎不太符合预期



```
2023-10-26-06-36-37 Time                0.014516
2023-10-26-06-36-37 Iteration           4000
2023-10-26-06-36-37 Best                349177.143553
2023-10-26-06-36-38 Time                0.000000
2023-10-26-06-36-38 Iteration           4100
2023-10-26-06-36-38 Best                349177.143553
2023-10-26-06-36-38 Time                0.000000
2023-10-26-06-36-38 Iteration           4200
2023-10-26-06-36-38 Best                349177.143553
2023-10-26-06-36-38 Time                0.000000
2023-10-26-06-36-38 Iteration           4300
2023-10-26-06-36-38 Best                349177.143553
2023-10-26-06-36-39 Time                0.000000
2023-10-26-06-36-39 Iteration           4400
2023-10-26-06-36-39 Best                349177.143553
2023-10-26-06-36-39 Time                0.000000
2023-10-26-06-36-39 Iteration           4500
2023-10-26-06-36-39 Best                349177.143553
2023-10-26-06-36-39 Time                0.000000
2023-10-26-06-36-39 Iteration           4600
2023-10-26-06-36-39 Best                349177.143553
2023-10-26-06-36-39 Time                0.015625
2023-10-26-06-36-39 Iteration           4700
2023-10-26-06-36-39 Best                349177.143553
2023-10-26-06-36-40 Time                0.000000
2023-10-26-06-36-40 Iteration           4800
2023-10-26-06-36-40 Best                349177.143553
2023-10-26-06-36-40 Time                0.000000
2023-10-26-06-36-40 Iteration           4900
2023-10-26-06-36-40 Best                349177.143553
2023-10-26-06-36-40 Time                0.000000
2023-10-26-06-36-40 Iteration           5000
2023-10-26-06-36-40 Best                349177.143553
2023-10-26-06-36-40 Final               3.491771e+05
2023-10-26-06-36-40 TimeSinceStart      0.342062
2023-10-26-06-36-40 random seed: 521318
```

我们观察生成的文档，发现best在经过一些较小的变换后，就几乎没有变化，这应该是因为我们设置的参数和一些变量设置的问题，



```
#mpiexec -np 16 python NCNES.py --lam 5 --mu 3 -f 1 -r debug -d 100
mpiexec -np 4 python NCNES.py --lam 100 --mu 100 -f 1 -r debug -d 100
#mpirun -np 16 python NCNES.py --lam 5 --mu 3 -f 1 -r debug -d 100
```

我们尝试修改种群数量，个体数量，维度等信息，发现best并无太好的改变，因此我们尝试理解系统输出文件时里面的内容是什么

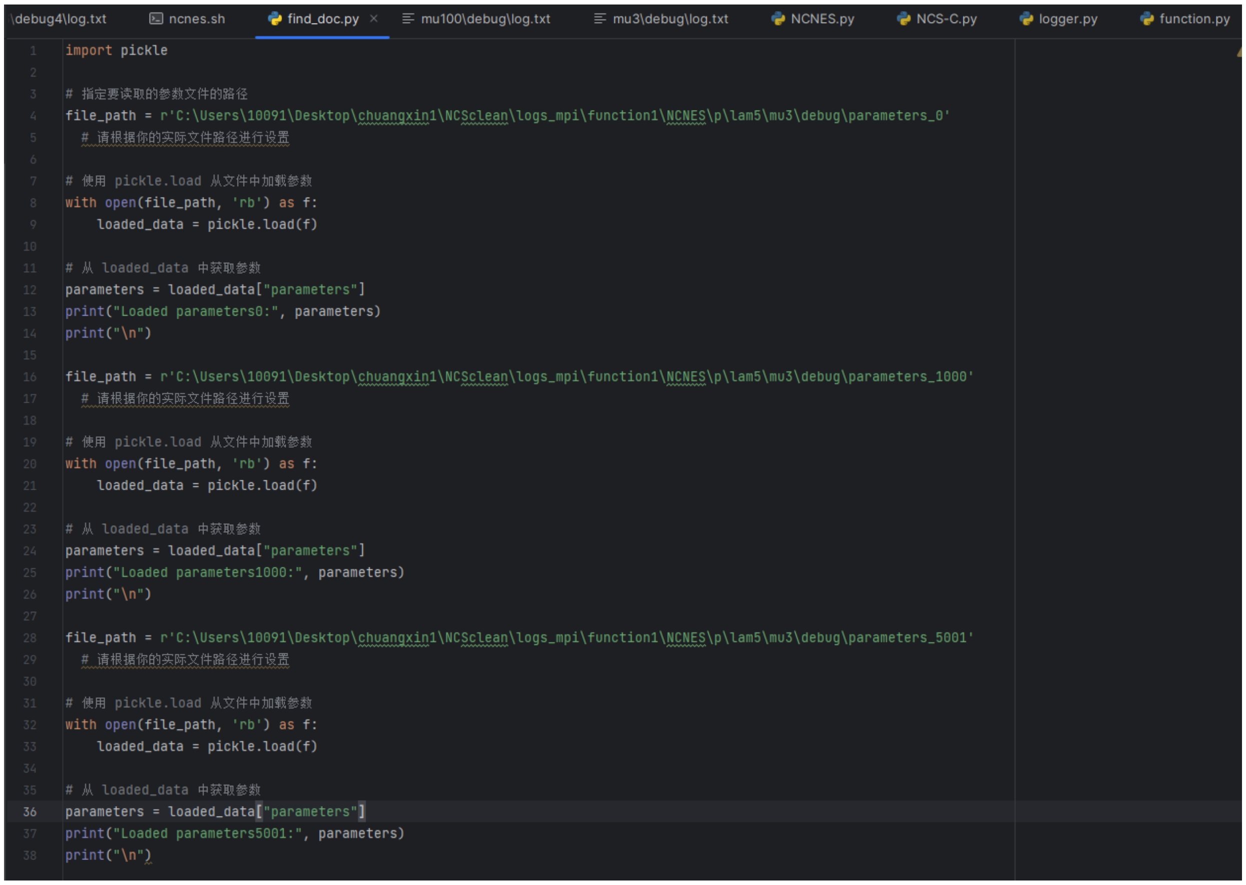
上网查找，此时的文件使用了pickle方法：

pickle 是Python标准库中的模块，它提供了一种将Python对象序列化（即将其转换为字节流）和反序列化（将字节流转换回Python对象）的方式。

`pickle.dump` 是 pickle 模块中的一个函数，它接受两个参数：

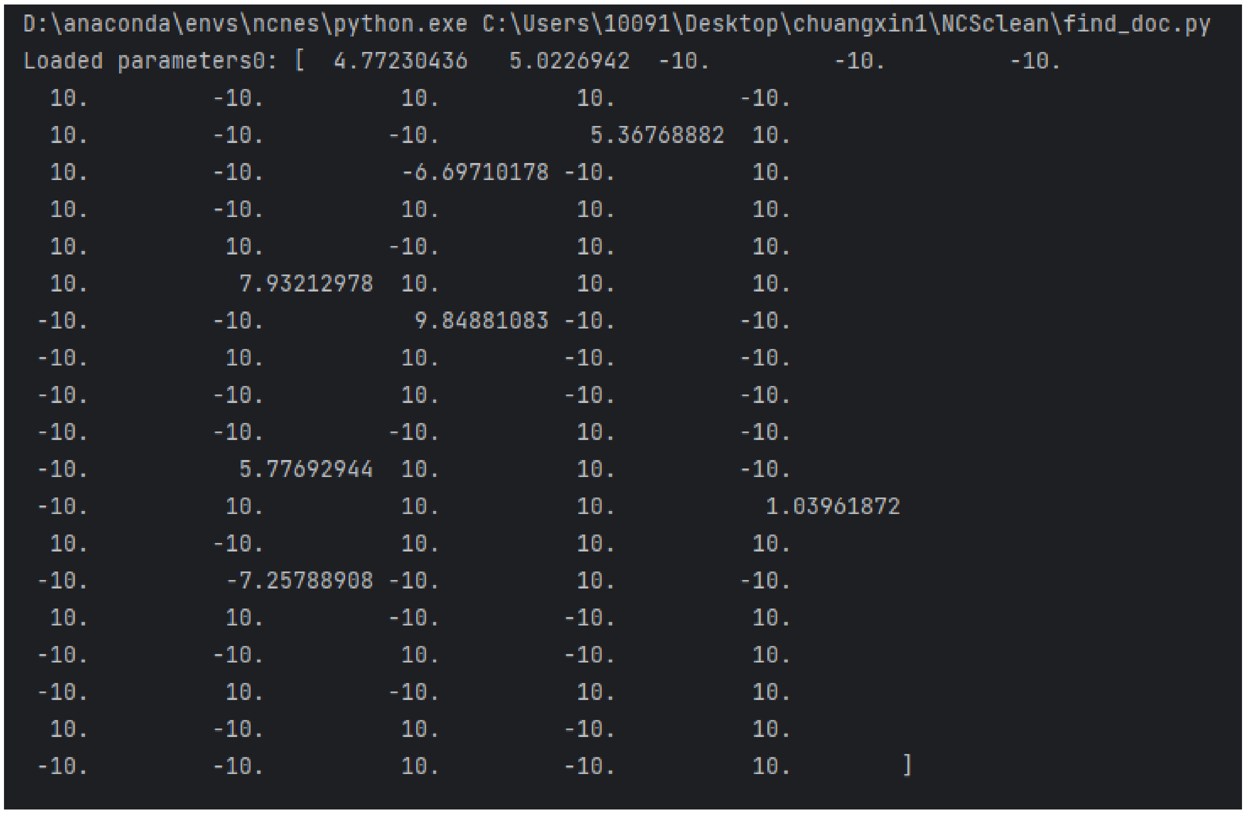
1. 第一个参数是要序列化的Python对象，这里是一个包含参数 `parameters` 的字典。
2. 第二个参数是一个文件对象（在这里是 `f` ），它用于将序列化后的数据写入文件。

我们创建新的 py 尝试输出它：

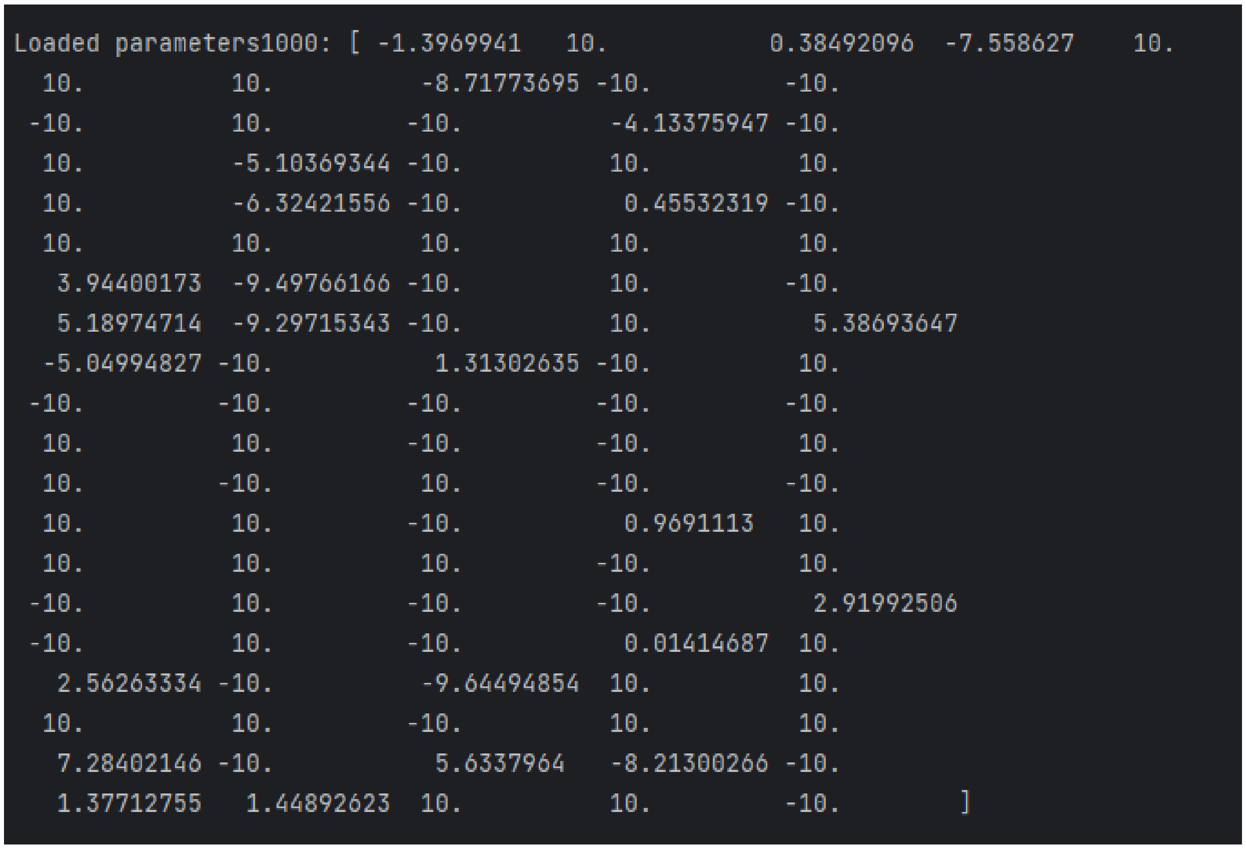


```python
import pickle

# 指定要读取的参数文件的路径
file_path = r'C:\Users\10091\Desktop\chuangxin1\NCSclean\logs_mpi\function1\NCNES\p\lam5\mu3\debug\parameters_0'
  # 请根据你的实际文件路径进行设置

# 使用 pickle.load 从文件中加载参数
with open(file_path, 'rb') as f:
    loaded_data = pickle.load(f)

# 从 loaded_data 中获取参数
parameters = loaded_data["parameters"]
print("Loaded parameters0:", parameters)
print("\n")

file_path = r'C:\Users\10091\Desktop\chuangxin1\NCSclean\logs_mpi\function1\NCNES\p\lam5\mu3\debug\parameters_1000'
  # 请根据你的实际文件路径进行设置

# 使用 pickle.load 从文件中加载参数
with open(file_path, 'rb') as f:
    loaded_data = pickle.load(f)

# 从 loaded_data 中获取参数
parameters = loaded_data["parameters"]
print("Loaded parameters1000:", parameters)
print("\n")

file_path = r'C:\Users\10091\Desktop\chuangxin1\NCSclean\logs_mpi\function1\NCNES\p\lam5\mu3\debug\parameters_5001'
  # 请根据你的实际文件路径进行设置

# 使用 pickle.load 从文件中加载参数
with open(file_path, 'rb') as f:
    loaded_data = pickle.load(f)

# 从 loaded_data 中获取参数
parameters = loaded_data["parameters"]
print("Loaded parameters5001:", parameters)
print("\n")
```

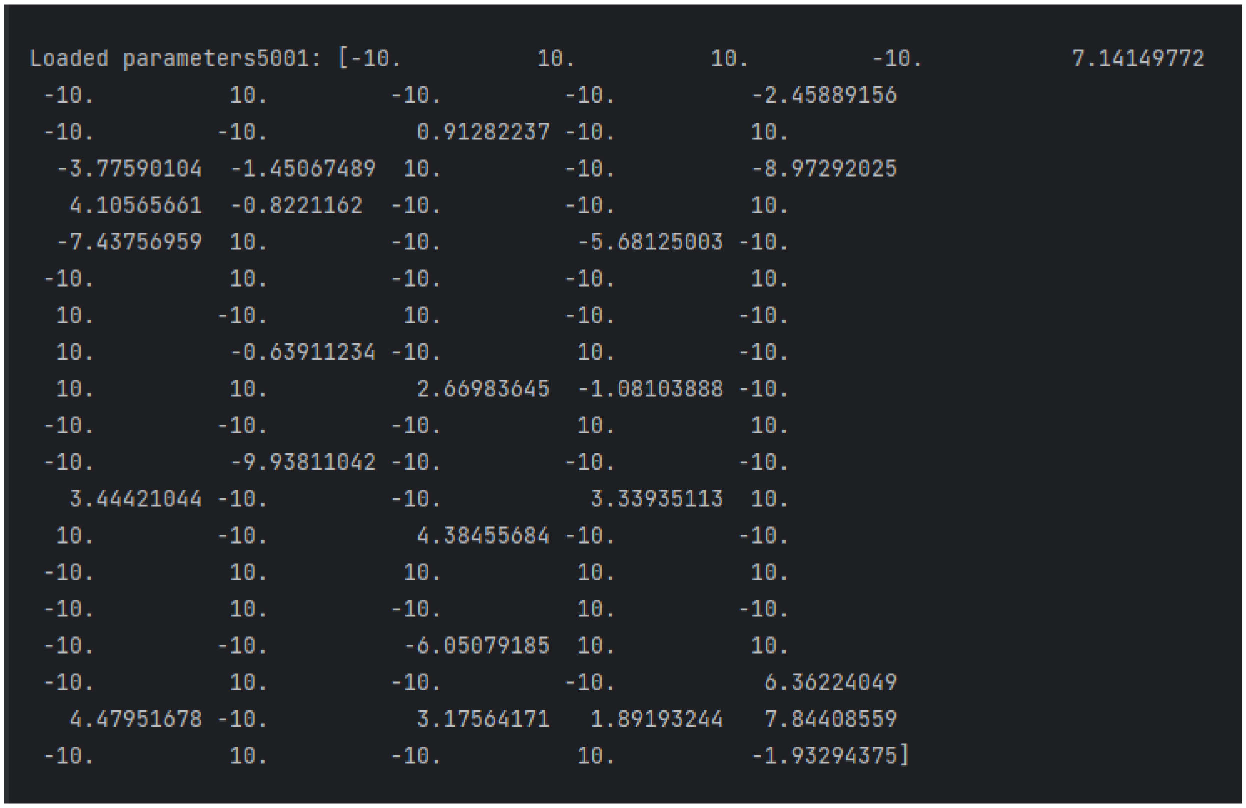
得到答案：



```
D:\anaconda\envs\ncnes\python.exe C:\Users\10091\Desktop\chuangxin1\NCSclean\find_doc.py
Loaded parameters0: [  4.77230436   5.0226942  -10.         -10.         -10.
  10.         -10.          10.          10.         -10.
  10.         -10.         -10.           5.36768882  10.
  10.         -10.          -6.69710178 -10.          10.
  10.         -10.          10.          10.          10.
  10.          10.         -10.          10.          10.
  10.           7.93212978  10.          10.          10.
 -10.         -10.           9.84881083 -10.         -10.
 -10.          10.          10.         -10.         -10.
 -10.         -10.          10.         -10.         -10.
 -10.         -10.         -10.          10.         -10.
 -10.           5.77692944  10.          10.         -10.
 -10.          10.          10.          10.           1.03961872
  10.         -10.          10.          10.          10.
 -10.          -7.25788908 -10.          10.         -10.
  10.          10.         -10.         -10.          10.
 -10.         -10.          10.         -10.          10.
 -10.          10.         -10.          10.          10.
  10.         -10.          10.         -10.          10.
 -10.         -10.          10.         -10.          10.        ]
```



```
Loaded parameters1000: [ -1.3969941   10.           0.38492096  -7.558627    10.
  10.          10.          -8.71773695 -10.         -10.
 -10.          10.         -10.          -4.13375947 -10.
  10.          -5.10369344 -10.          10.          10.
  10.          -6.32421556 -10.           0.45532319 -10.
  10.          10.          10.          10.          10.
   3.94400173  -9.49766166 -10.          10.         -10.
   5.18974714  -9.29715343 -10.          10.           5.38693647
  -5.04994827 -10.           1.31302635 -10.          10.
 -10.         -10.         -10.         -10.         -10.
  10.          10.         -10.         -10.          10.
  10.         -10.          10.         -10.         -10.
  10.          10.         -10.           0.9691113   10.
  10.          10.          10.         -10.          10.
 -10.          10.         -10.         -10.           2.91992506
 -10.          10.         -10.           0.01414687  10.
   2.56263334 -10.          -9.64494854  10.          10.
  10.          10.         -10.          10.          10.
   7.28402146 -10.           5.6337964   -8.21300266 -10.
   1.37712755   1.44892623  10.          10.         -10.        ]
```



```
Loaded parameters5001: [-10.          10.          10.         -10.           7.14149772
 -10.          10.         -10.         -10.          -2.45889156
 -10.         -10.           0.91282237 -10.          10.
  -3.77590104  -1.45067489  10.         -10.          -8.97292025
   4.10565661  -0.8221162  -10.         -10.          10.
  -7.43756959  10.         -10.          -5.68125003 -10.
 -10.          10.         -10.         -10.          10.
  10.         -10.          10.         -10.         -10.
  10.          -0.63911234 -10.          10.         -10.
  10.          10.           2.66983645  -1.08103888 -10.
 -10.         -10.         -10.          10.          10.
 -10.          -9.93811042 -10.         -10.         -10.
   3.44421044 -10.         -10.           3.33935113  10.
  10.         -10.           4.38455684 -10.         -10.
 -10.          10.          10.          10.          10.
 -10.          10.         -10.          10.         -10.
 -10.         -10.          -6.05079185  10.          10.
 -10.          10.         -10.         -10.           6.36224049
   4.47951678 -10.           3.17564171   1.89193244   7.84408559
 -10.          10.         -10.          10.          -1.93294375]
```

该系数似乎与维度的大小有关，而在这些参数中，似乎严重受到 最大最小值 正负10的约束， 应该就是参数设置的有一些问题，算法其实并没有有效的运行