最优化算法——随机优化算法

随机优化算法的思想是抛开求导，采用撒豆成兵的方法，即只要在给定的取值范围内，自变量取值足够多，就能找到最值的近似解。

以求解最小值为例，具体算法如下：

首先，用遍历首先确定函数极值点的大致区间范围。

循环开始：

step1：在这个区间内随机产生自变量，带入函数计算出函数值。（多变量的话，只需要生产多个自变量即可）

step2（如没有约束条件，则省略）：判断约束条件。满足则跳转step3，不满足则跳转step1

step3：将每次循环中计算出函数值存放在一个向量中，并计算向量的最小值，并将最小值，存放在另外一个向量中。（这一步是每一次循环都要做的，每一次循环向量就增加一个值，中存放的是每次循环中向量的最小值，因为向量的元素在增加，中的元素也是每循环一次就要增加一个，且每增加一个要么和前一个的值相同，要么比前一个的值小）

step4：如果是第一次循环当前循环，记录此次循环step1中产生的自变量，即；如果是后续循环，那么进行如下操作，如果中最新的一个值比前一个值小，那么记录此次循环step1中产生的自变量，即，计数值置0；反之，则计数值增1。

step5：如果计数值小于阈值，即，则循环从step1重新开始，反之，则结束循环，输出step4中最后一次更新的和中的最后一个元素。（最后一次更新的和中的最后一个元素就是函数的最小值对应的自变量和函数的最小值）

求解函数上的最小值

举个例子（以下函数值带入计算都是我瞎说的，带入计算太麻烦了，具体请自行带入函数算一下）

初始化计数值s=0，阈值N=10

第一次循环

x在0~6随机产生为0.1，带入函数计算y=4.9，然后将4.9存在Y中，那么Y=[4.9]，然后计算向量Y的最小值miny=4.9，那么将4.9存入向量minY中，minY=[4.9]，因为是第一次循环要将0.1保存下来，赋值给x0，第一次循环结束。

第一次循环结束的结果：Y=[4.9],minY=[4.9], x0=0.1,s=0

第二次循环

x在0~6随机产生为5.3，带入函数计算y=6.2，然后将6.2存在Y中，那么Y=[4.9,6.2]，然后计算向量Y的最小值miny=4.9，那么将4.9存入向量minY中，minY=[4.9,4.9]，因为是第二次循环minY中新增元素和第一次循环相比没有变化，所以计数值s增1，第二次循环结束。

第二次循环结束的结果：Y=[4.9,6.2],minY=[4.9,4.9] , x0=0.1,s=1

第三次循环

x在0~6随机产生为2.8，带入函数计算y=3.6，然后将3.6存在Y中，那么Y=[4.9,6.2,3.6]，然后计算向量Y的最小值miny=3.6，那么将3.6存入向量minY中，minY=[4.9,4.9,3.6]，因为是第二次循环minY中新增元素和第一次循环相比减小，意味着本次循环x=2.8计算出的函数值比前两次循环更小，所以计数值s置0，同时，2.8赋值给x0，第三次循环结束。

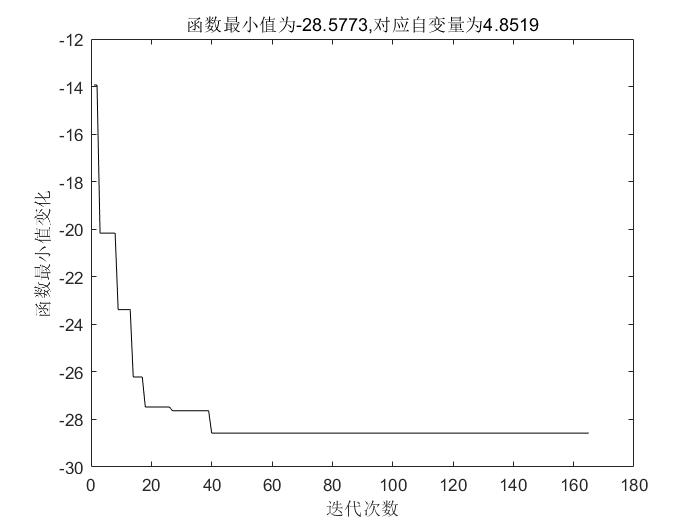
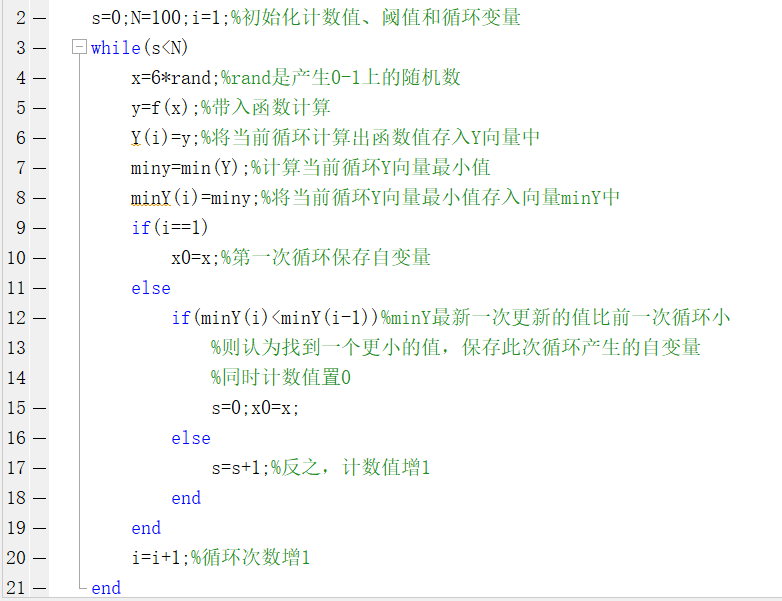
第三次循环结束的结果：Y=[4.9,6.2,3.6],minY=[4.9,4.9,3.6] , x0=2.8,s=0

第四次循环

x在0~6随机产生为3.4，带入函数计算y=5.7，然后将5.7存在Y中，那么Y=[4.9,6.2,3.6]，然后计算向量Y的最小值miny=3.6，那么将3.6存入向量minY中，minY=[4.9,4.9,3.6,3.6]，因为是第四次循环minY中新增元素和第三次循环相比没有变化，所以计数值s增1，第四次循环结束。

第四次循环结束的结果：Y=[4.9,6.2,3.6,5.7],minY=[4.9,4.9,3.6,3.6] , x0=2.8,s=1

如果这个函数的最小值就是3.6，那继续循环下去，minY中增加的值就是3.6不会变，就像第四次循环一样，同时，计数值s在不断增加，当s超过阈值N=10的时候，默认找到最小值结束循环。minY中每次循环增加的元素要么不变，要么减小。最后将minY画出来就是一个阶梯收敛图，**这个图很重要！比赛时如果用到随机优化算法，一定要画出阶梯收敛图!**

当阈值N增大时，计算结果的精度会增加，这里的函数我编成function文件了，对于函数表达式未知的优化问题也可以用该算法求解，例如2020A第三问。

多变量带约束的情况

计算二元函数的最大值（理论计算最大值点）

直接和和范围内搜索，在加if()条件判断约束

