Genetic Algorithms Homework #3

資訊所 黃子睿 P76054088

問題描述

Minimize the function



Subject to:

* 
* 
*  and 

本次作業為利用實數基因表示的基因演算法來計算最佳解，並且和改良基因演算法+爬山演算法做比較，以下將進行基因演算法的流程介紹及每一步驟參數設定。

實數基因演算法

流程為

初始化→選擇(輪盤法)→交配→突變→修補→菁英模型→產生好的子代重複步驟2直到收斂

1. 初始條件設定

Population = 10

Generation = 200

Genetic representation: Real-value vector

Handling constraints: 先隨機產生介於0~6範圍的，之後再產生移向後的值，即滿足條件和和

Selection: Roulette Wheel

選擇(輪盤法)

由於輪盤法是計算一個個體佔有面積的大小而且又是找最小值的關係，所以需要將原本佔有面積的公式做一個變換，從轉換成，如此一來就能將原本比較小的適應值面積轉換成佔有較大塊面積。

交配

Crossover: One-point crossover

Crossover Rate = 0.8

突變

Mutation: uniform mutation

Mutation Rate = 0.1

由於constraints的關係，突變的值必須被新定義的上下界給限制住。

修補

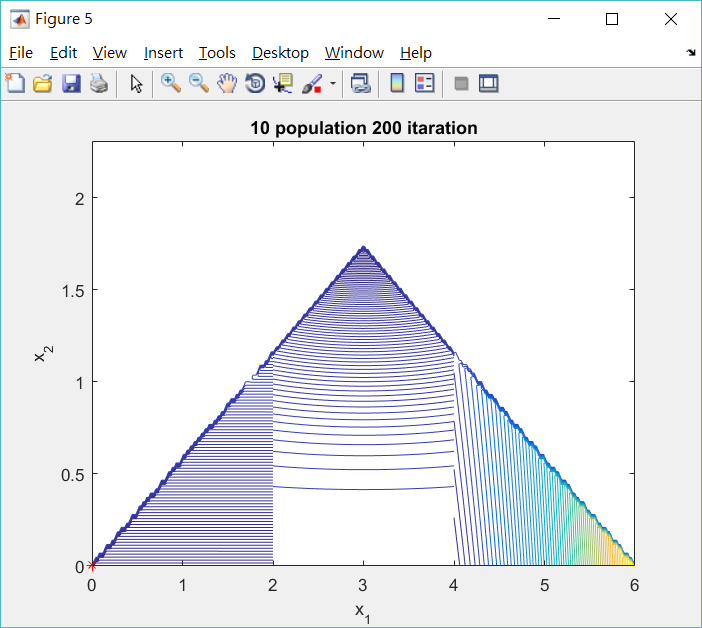
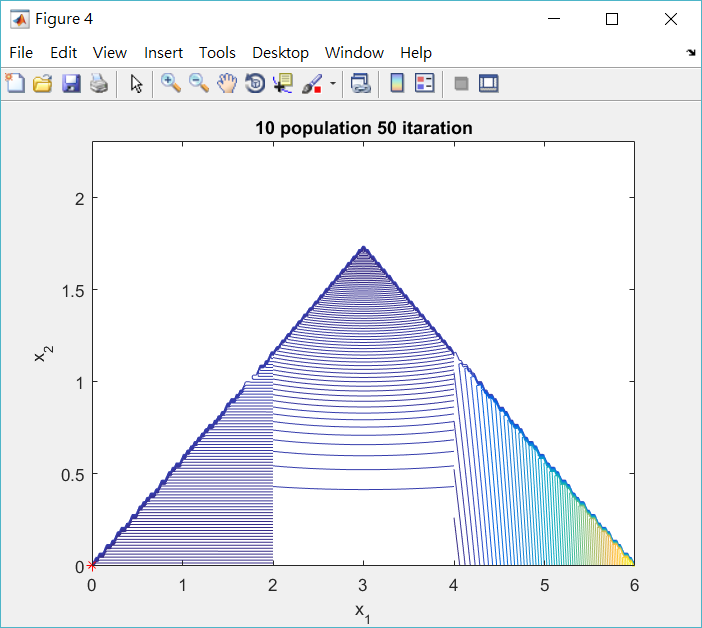
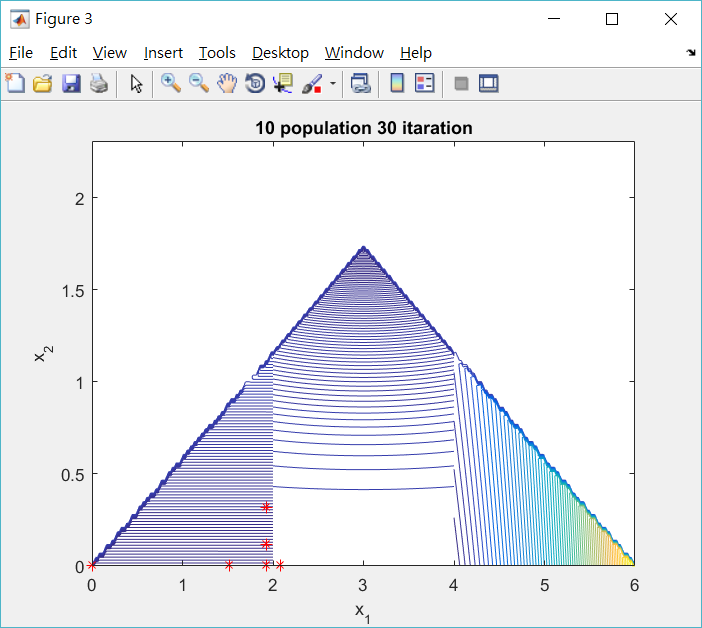
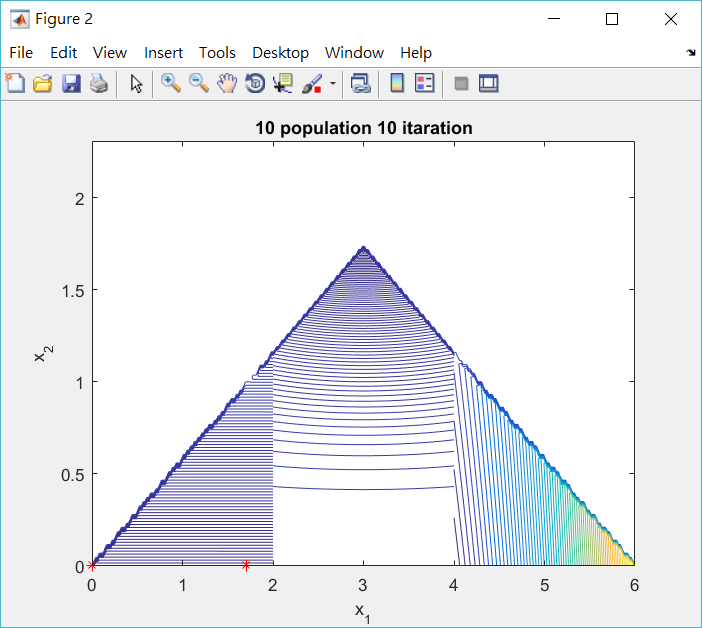
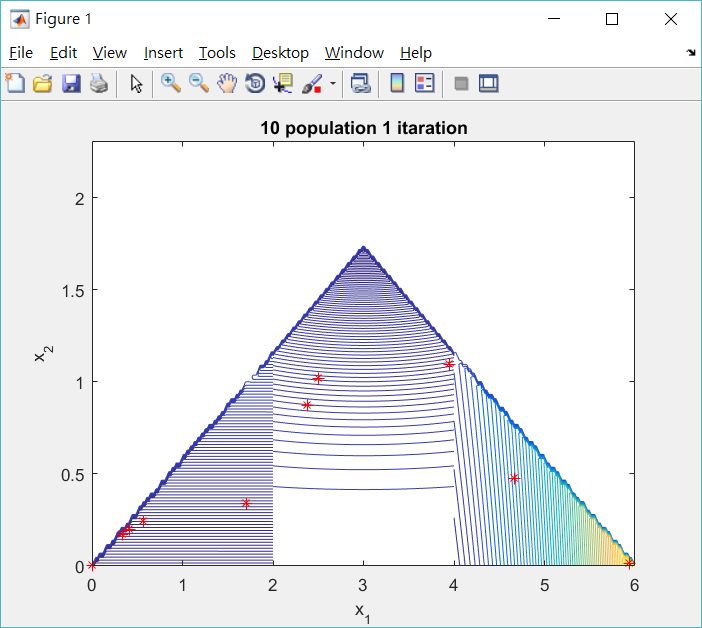
在交配這一步驟時，如果不同個體的互換可能會違反的constraints，因此需要重新檢查個體是否滿足constraints，如果沒有就重新隨機產生新值。

菁英模型

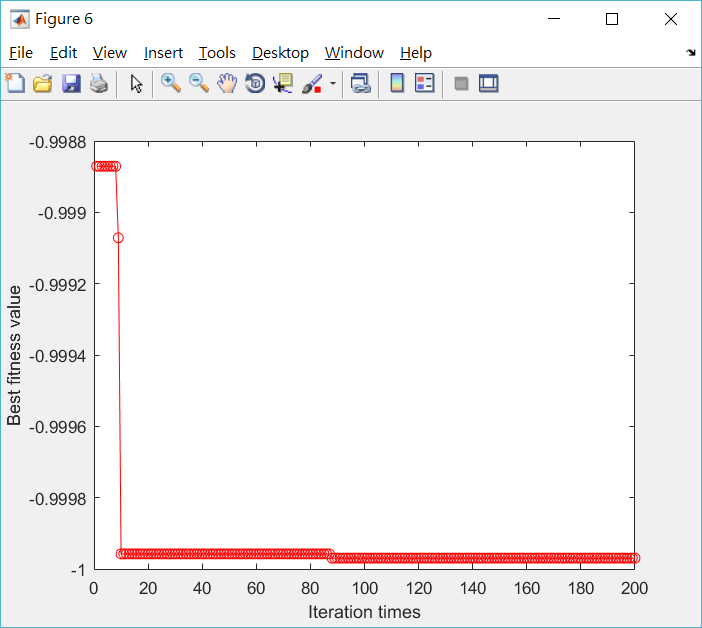
如果當這一代所產生的所有個體都比前一代還要糟，則將前一代的菁英個體取代掉這一代的最差個體，作用是保留每一代最好的菁英個體。

1. 實驗結果

實驗結果將進行200次的迭代，並呈現初始狀況以及10, 30, 50, 200代的個體變化



收斂狀況



以下為執行程式5次，並且以表格來呈現最佳的結果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 次數 | **第一次** | 第二次 | 第三次 | 第四次 | 第五次 |
| (x1, x2) | **(0.021409, 0.000014)** | (0.051878, 0.000034) | (0.011022, 0.000055) | (0.045163, 0.000532) | (0.000902, 0.000015) |
| 最佳解 | **-0.999986** | -0.999966 | -0.999945 | -0.999468 | -0.999985 |

1. 結論

從實驗結果圖可以看出，一開始散佈的種子點在約50代的時候這些種子點都直接往全域最小值(0, 0)的位置收斂移動，因為這次加了菁英模型的關係，所以比前一次作業的收斂速度還要來的快，此外，基因演算法得到的值很穩定(都落在-0.999附近)。基因演算法母群體跟迭代次數是比較重要的部分，而又根據初始狀況的設定會影響最後得到的結果好壞，缺點是如果增加的母群體數夠多，可能會讓記憶體占用比較多資源。

改良基因演算法+爬山演算法

此演算法是由基因演算法和爬山演算法相結合，利用早期基因多種子點和晚期爬山演算法快速收斂的特性達到比較好的效果。一開始先由基因演算法找出第一個子代，接著找出此一代中最好的解，並且當成爬山演算法的種子點從周圍東南西北鄰近方向尋找，直到跳脫區域值或者全都違反constraints，之後取代前一子代的最佳解，再利用菁英法則，以達到更好的效能。

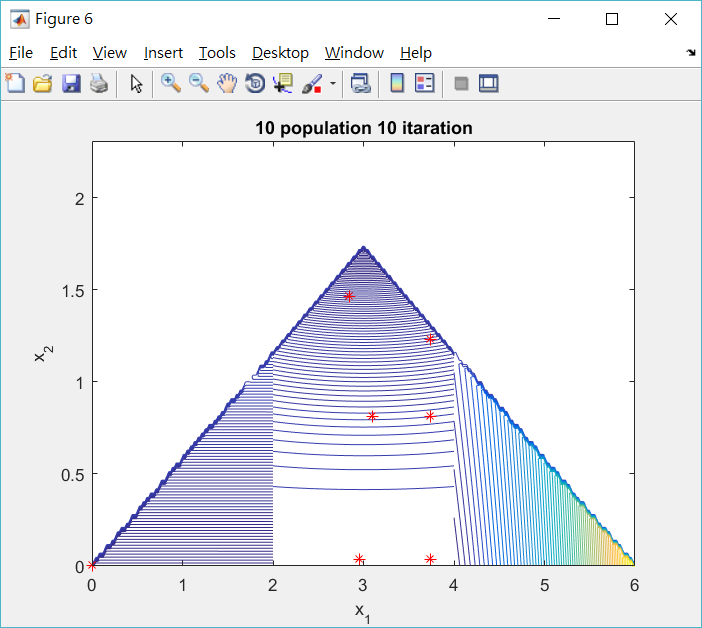
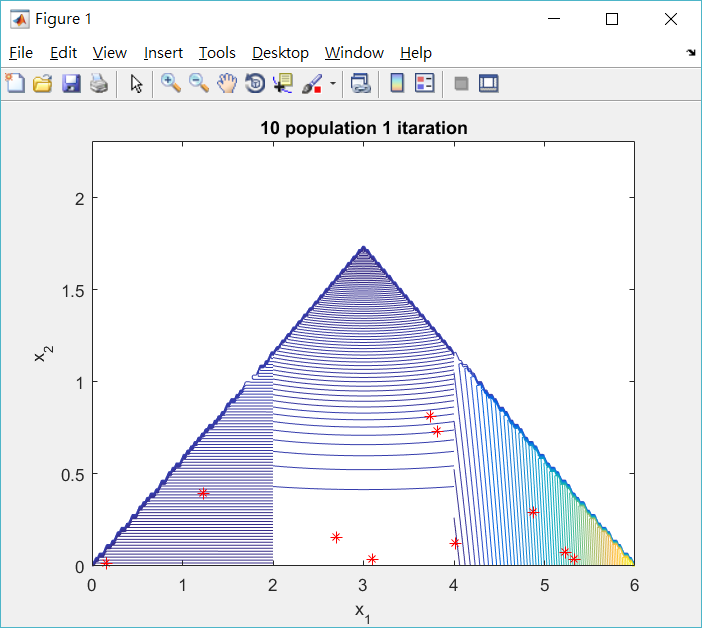
1. 初始條件設定

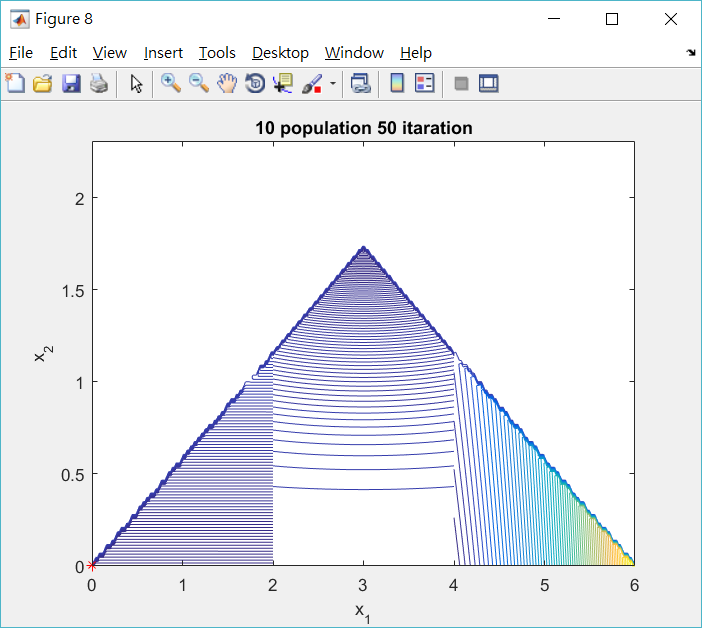
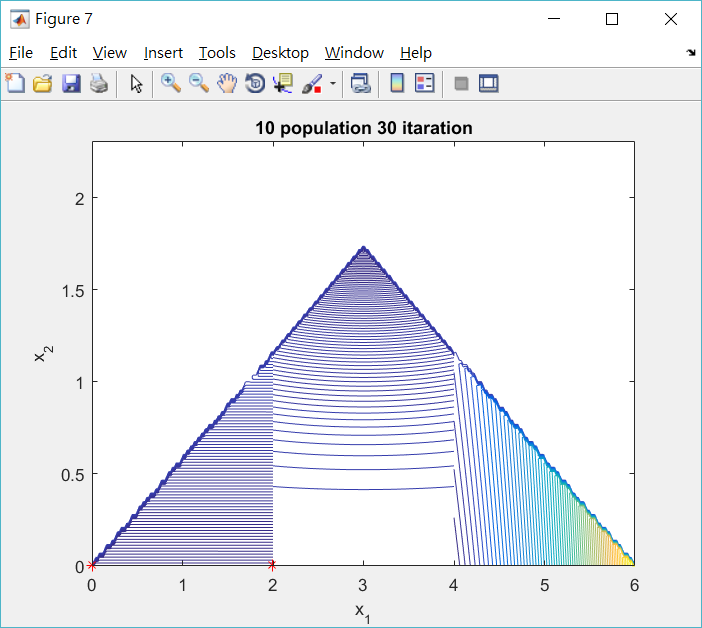
此方法在基因演算法的初始條件和前面方法完全相同

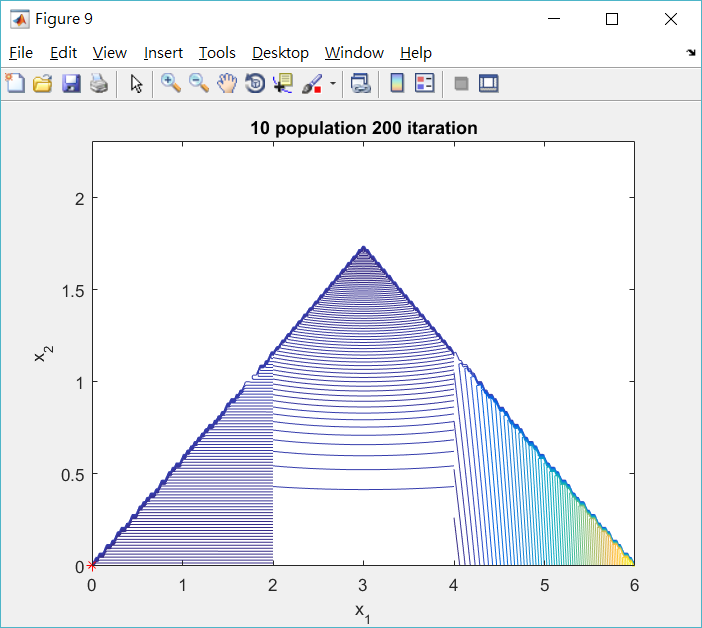
鄰居定義:

由點座標往東南西北四方向各偏移0.01值，如有違反constraints則捨棄該鄰居不再修正回來。

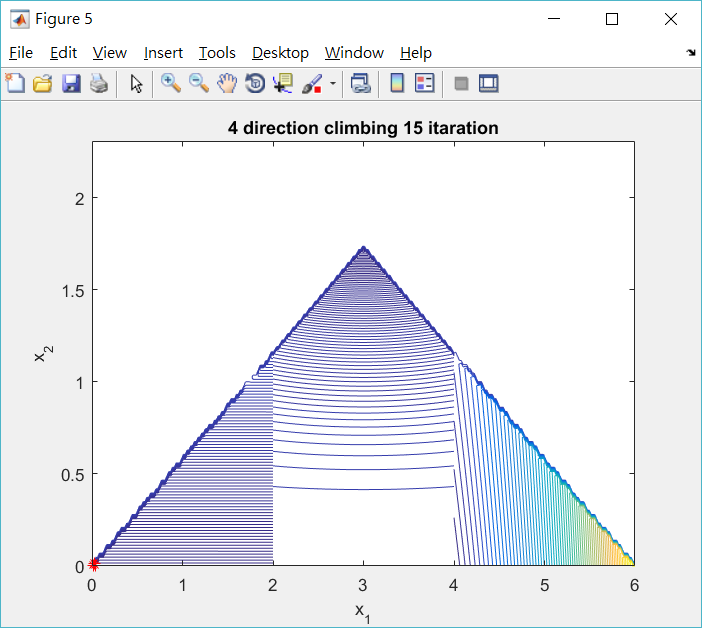
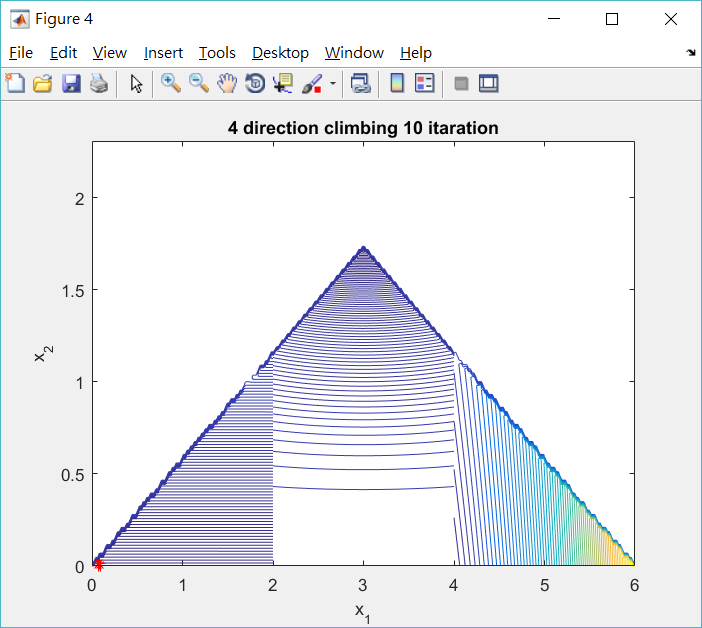
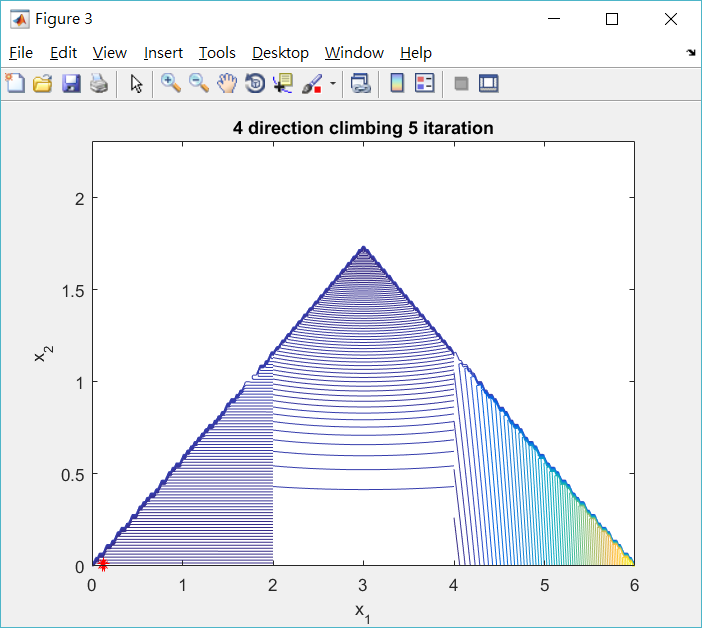
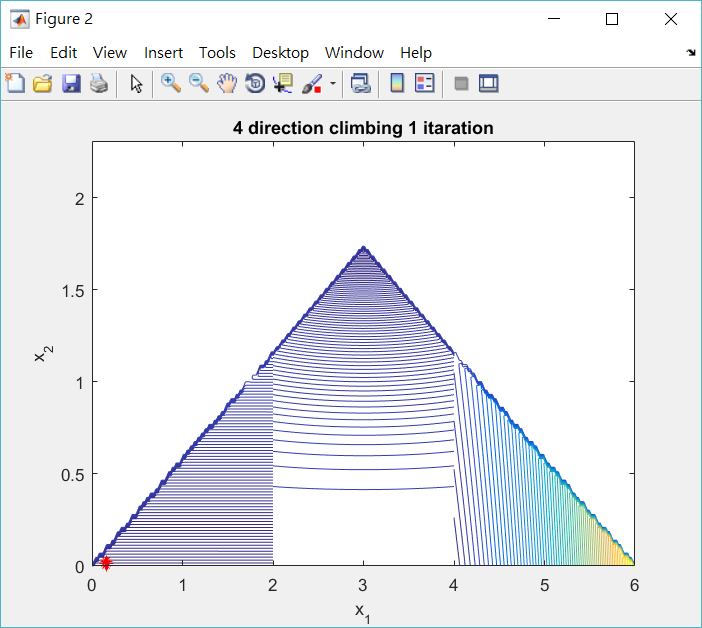
1. 實驗結果

實驗結果將進行200次的迭代，並呈現初始狀況以及10, 30, 50, 200代的個體變化，並畫出其中幾代爬山成功的結果圖。

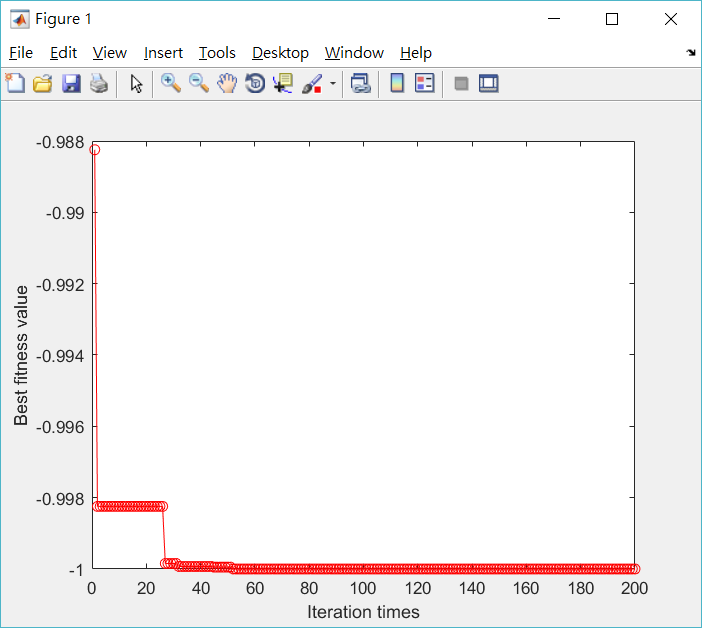




爬山狀況



收斂狀況



以下為執行程式5次，並且以表格來呈現最佳的結果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 次數 | **第一次** | 第二次 | 第三次 | 第四次 | 第五次 |
| (x1, x2) | **(0.009381, 0.000001)** | (0.003087, 0.000030) | (0.005578, 0.000007) | (0.006288, 0.000004) | (0.005816, 0.000014) |
| 最佳解 | **-0.999999** | -0.999970 | -0.999993 | -0.999996 | -0.999986 |

1. 兩者比較結論

從實驗結果圖來看兩者方法皆在30代大部份值就已經向全域解靠攏，而改良的演算法種子點顯得比較不會散亂，收斂速度比較快，從兩者的收斂圖亦可以觀察到此現象，且從兩者找到的最佳解來比較可以得知改良後的方法顯得較為精準。