最佳化演算法

作業三：利用BOA解決作業二之問題

姓名:蔡奇傑

學號:110916004

班級:資四甲

日期:113/5/5

1. 題目:

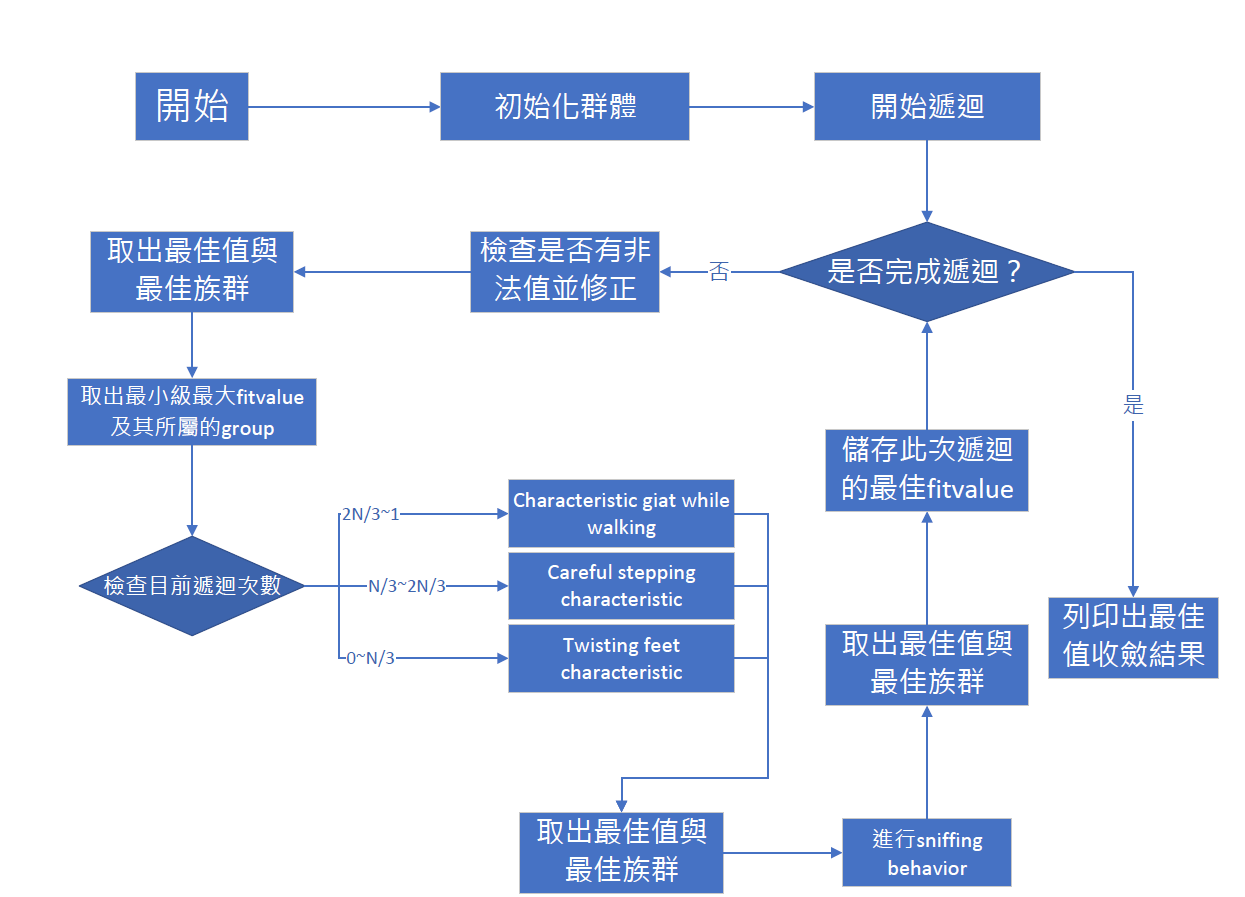
f(x, y) = (x−y)2 − x + 2y + sin(x+y) +1

x∈[−1.5, 4] and y∈[−3, 4]

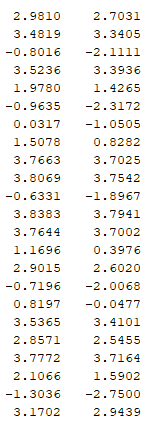
試利用matlab 程式，以基因演算法求f之最小值

x, y 小數點的有效位數為4位 (±x.xxxx)

f 小數點的有效位數為5位 (±x.xxxxx)

1. BOA流程圖:
2. 族群建立方式:

建立一個二維矩陣，其中每一列皆為一個族群，並且第一行代表X變數，第二行代表Y變數



1. 決定族群數量

預設族群數:25,50,75,100

預設遞迴數:50次

|  |  |  |  |
| --- | --- | --- | --- |
| 群體規模 | 最佳收斂結果 | 圖 | 評價 |
| 25 | -1.47572 | groupsize25 | 收斂速度偏慢  約在第16次遞迴可收斂至接近最佳值 |
| 50 | -1.47544 | groupsize50 | 收斂速度偏慢  約在第12次遞迴可收斂至接近最佳值  且有幾次無法收斂至最佳 |
| 75 | -1.47572 | groupsize75 | 約在第9次遞迴可收斂至接近最佳值 |
| 100 | -1.47570 | groupsize100 | 約在第9次遞迴可收斂至接近最佳值 |

綜合以上數據可發現此一演算法不論族群的大小幾乎都可成功收斂至最佳值，但綜合穩定性及效率的考量選擇族群大小為75可以獲得較好的結果

1. 遞迴數選擇:

由上面的測試可以發現所有族群幾乎都可在10次上下獲得接近最佳值的fitvalue，因此我想測試一下如過縮小遞迴次數是否會對整體收斂結果產生影響。

預設遞迴數:30,40,50,60,70

預設族群數:75

|  |  |  |  |
| --- | --- | --- | --- |
| 遞迴次數 | 最佳收斂結果 | 圖 | 評價 |
| 30 | -1.47546 | iter30 | 約在第7次遞迴可收斂至接近最佳值  但無法收斂至最佳值 |
| 40 | -1.47555 | Iter40 | 約在第9次遞迴可收斂至接近最佳值  但無法收斂至最佳值 |
| 50 | -1.47572 | Iter50 | 約在第9次遞迴可收斂至接近最佳值  並且可成功收斂至最小值 |
| 60 | -1.47570 | Iter60 | 約在第10次遞迴可收斂至接近最佳值  但測試多次皆無法成功收斂至最佳 |
| 70 | -1.47554 | Iter70 | 約在第10次遞迴可收斂至接近最佳值  但測試多次皆無法成功收斂至最佳 |

在經過測試過後發現在其他變數一樣的情況下，遞迴數為50時可獲得較好的結果，且在收的穩定度及最終結果皆比其他幾種族群規模更好，雖然族群數設定為30時收斂速度較快但沒辦法有效收斂至最佳值，因而決定選擇50作為遞迴數

1. 設計適應函數:

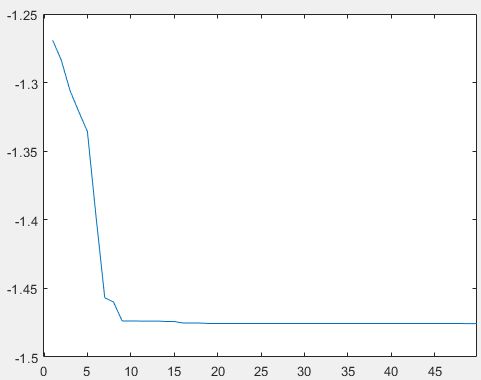
如題: f(x, y) = (x−y)2 − x + 2y + sin(x+y) +1

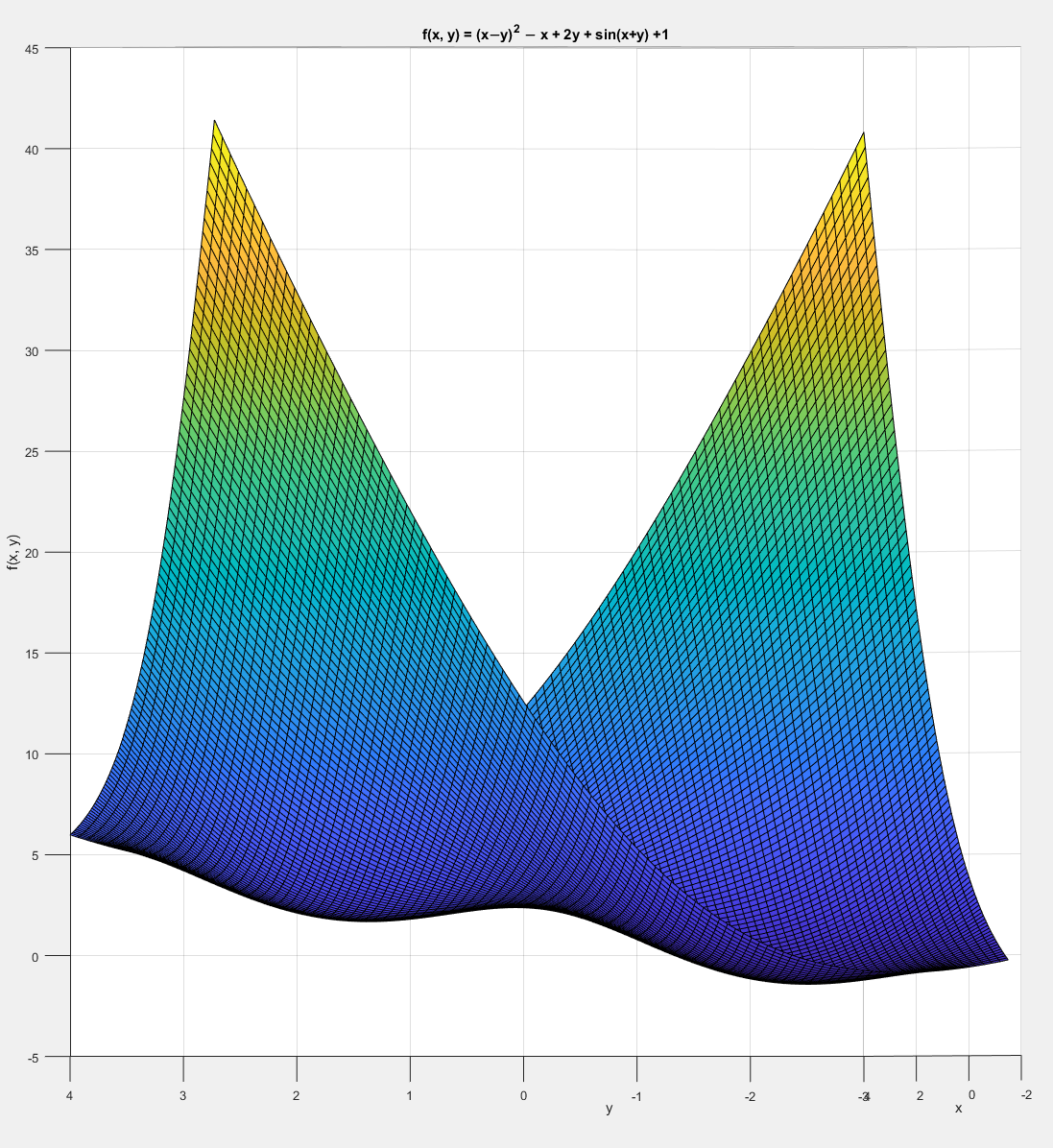
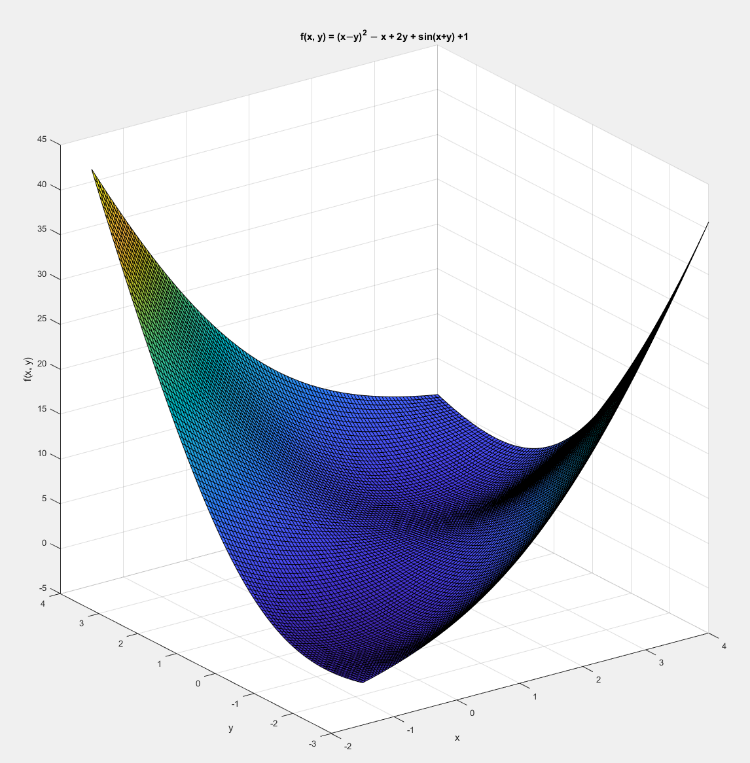
1. 決定中止條件:

由於此基因演算法是要求要算出最小值，因此並沒有辦法具體設定終止條件，所以我是採用遞迴500次的方式因為此一遞迴數可讓數值完整收斂

1. 結果與討論:

答案: -1.47572



函數圖形:

1. 討論:

此演算法無論在收斂速度還是收斂穩定度都遠勝基因演算法，且只需要考慮族群大小及遞迴次數，在測試的方便性上也遠勝基因演算法，並且基本上每次測試都可獲得全域最佳值，由此可知BOA在處理這種簡單的函數問題可以說游刃有餘

1. 心得:

雖然這個問題偏簡單，沒有辦法完全發揮出BOA應有的能力，但也讓我對BOA的運行方式有更深的理解。

1. 程式碼:

主程式:

function main(obj)

    %set-------

    %X,Y limit

    Min=[-1.5 -3 ]; % Minimum value of each group

    Max=[  4   4 ]; % Minimum value of each group

    % 設定矩陣大小

    groupNum = 75;  %族群數

    dimention = 2;  %變數個數

    totalIter=70;   %遞迴數

    %----------

    group=groupInit(Max , Min , groupNum);

    Best\_sol=inf;

    Convergence=zeros(1,totalIter);

    for C = 1:totalIter

        %取出最佳值與最佳族群

        % For limiting out of bound solutions and setting best solution and

        % objective function value

        for i=1:size(group,1)

            %檢查是否有超出範圍的族群

            Indx\_max=group(i,:)>Max;    %建立一個陣列其元素為group(i,:)是否大於MAX

            Indx\_min=group(i,:)<Min;    %建立一個陣列其元素為group(i,:)是否小於MIN

            group(i,:)=(group(i,:).\*(~(Indx\_max+Indx\_min)))+Max.\*Indx\_max+Min.\*Indx\_min;

                                    %若group在範圍外設為0    並將其設為範圍最大or最小值

            %評估各族群的fitvalue

            fitness=fit\_function(group(i,:));

            %儲存最佳解及最佳族群

            if(Best\_sol>fitness)

                Best\_sol=fitness;

                Best\_X=group(i,:);

            end

        end

        theta = C/totalIter;

        %pedal scent marking behavior begin

        %取出最小級最大fitvalue及其所屬的group

        [min\_fit, Indx1]=min(fitness);

        Best\_fit=group(Indx1,:);

        [max\_fit, Indx2]=max(fitness);

        Worst\_fit=group(Indx2,:);

        clear Indx1 Indx2

        %進行三種不同加深標記的方式

        for ii=1:groupNum

            if theta>0 && theta<=totalIter/3;                   % Characteristic Gait while walking

                newGroup(ii,:)= group(ii,:) + (-theta\*rand(1,dimention).\*group(ii,:) );

            elseif theta>totalIter/3 && theta<=2\*totalIter/3    % Careful Stepping Characteristic

                Q = theta\*rand(1,dimention);                    %Fk

                Step = round (1 + rand());                      %Lk

                newGroup(ii,:)= group(ii,:) + (Q.\*(Best\_fit-(Step\*Worst\_fit)));

            elseif theta> 2\*totalIter/3 && theta<= 1            %twisting Feet Characteristic

                W = 2\*theta\*pi\*rand(1,dimention);               %angular velocity

                newGroup(ii,:)= group(ii,:) + ((W.\*Best\_fit- abs(group(ii,:)))-(W.\*Worst\_fit- abs(group(ii,:))));

            end

        end

        %選出較佳的new group

        for ii=1:groupNum

            newfitness=fit\_function(newGroup(ii,:));

            if newfitness<fitness

                group(ii,:)=newGroup(ii,:);

            else

                group(ii,:)=group(ii,:);

            end

            fitness=fit\_function(group(ii,:));

        end

        %pedal scent marking behavior end

        %sniffing behavior start

        for ii=1:groupNum

            k=round(rand()\*(groupNum));

            while k==ii || k<=0

                k=round(rand()\*(groupNum)); %找到與ii不同的隨機數

            end

            %進行演算法中的算式

            if fit\_function(group(ii,:))<fit\_function(group(k,:))

                r=rand()\*ones(1,dimention);

                newGroup(ii,:)=group(ii,:)+r.\*( group(ii,:) - group(k,:) );

            else

                r=rand()\*ones(1,dimention);

                newGroup(ii,:)=group(ii,:)+r.\*( - group(ii,:) + group(k,:) );

            end

        end

        %選出較佳的group

        for ii=1:groupNum

        newfitness=fit\_function(newGroup(ii,:));

            if newfitness<fitness

                group(ii,:)=newGroup(ii,:);

            else

                group(ii,:)=group(ii,:);

            end

        end

        %將每輪最佳的fitvalue儲存起來

        Convergence(C)=Best\_sol;

        X(C)=C;

    end

    plot(X,Convergence);

end

初始化:

function  [group]=groupInit(max , min , R)

    % 生成隨機二維矩陣 R\*2 matrix

    group=rand(R,1)\*(max-min)+(ones(R,1)\*min);  %max,min為1\*2 matrix

計算fitvalue:

function [fitness]=fit\_function(group)

    fitness=round( (group(1)-group(2)).^2-group(1)+2\*group(2)+sin(group(1)+group(2))+1,5);