## 最佳化演算法

### 作業二：基因演算法應用實例之實現

### 姓名:林 翡

### 學號:110916015

### 班級:資四甲

### 日期:2024/4/4

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. 基因演算法（以流程圖或虛擬碼表示，並加說明）

* 流程圖

選擇

(selection函式，使用機率選擇->輪盤式)

交配

(crossover函式，使用單點法)

確定問題的參數集

(種族大小、迭代次數、基因長度、交叉概率、變異概率)

對問題參數進行編碼

(用二進位編碼)

產生初始族群

(initpop函式，以隨機的方式產生)

計算適應度，並評估群體

滿足停止準則?

No

結束

Yes

突變

(mutation函式)

產生新族群

1. 設計編碼方式 (說明如何使用二進位編碼來代表 x 與 y 的值)

* x的邊界[-1.5,4]，範圍為4-(-1.5)=5.5，每一變數的精確度到小數五位，5.5\*10000=55000，215<55000<216，因此需要16位元表示x。  
  y的邊界[-3,4]，範圍為4-(-3)=7，7\*10000=70000，216<70000<217，需要17位元表示y，因此個體長度(L)總共33位元。

1. 決定群體規模 (如何決定族群數量)

* 族群數量我試過20、50、100、200的結果，在其他參數相同的情況下，發現20、50、100的種族大小不會有太大的差別，但200就會造成收斂圖會有點震盪的情形發生(圖1)，所以就選了族群大小為100的。

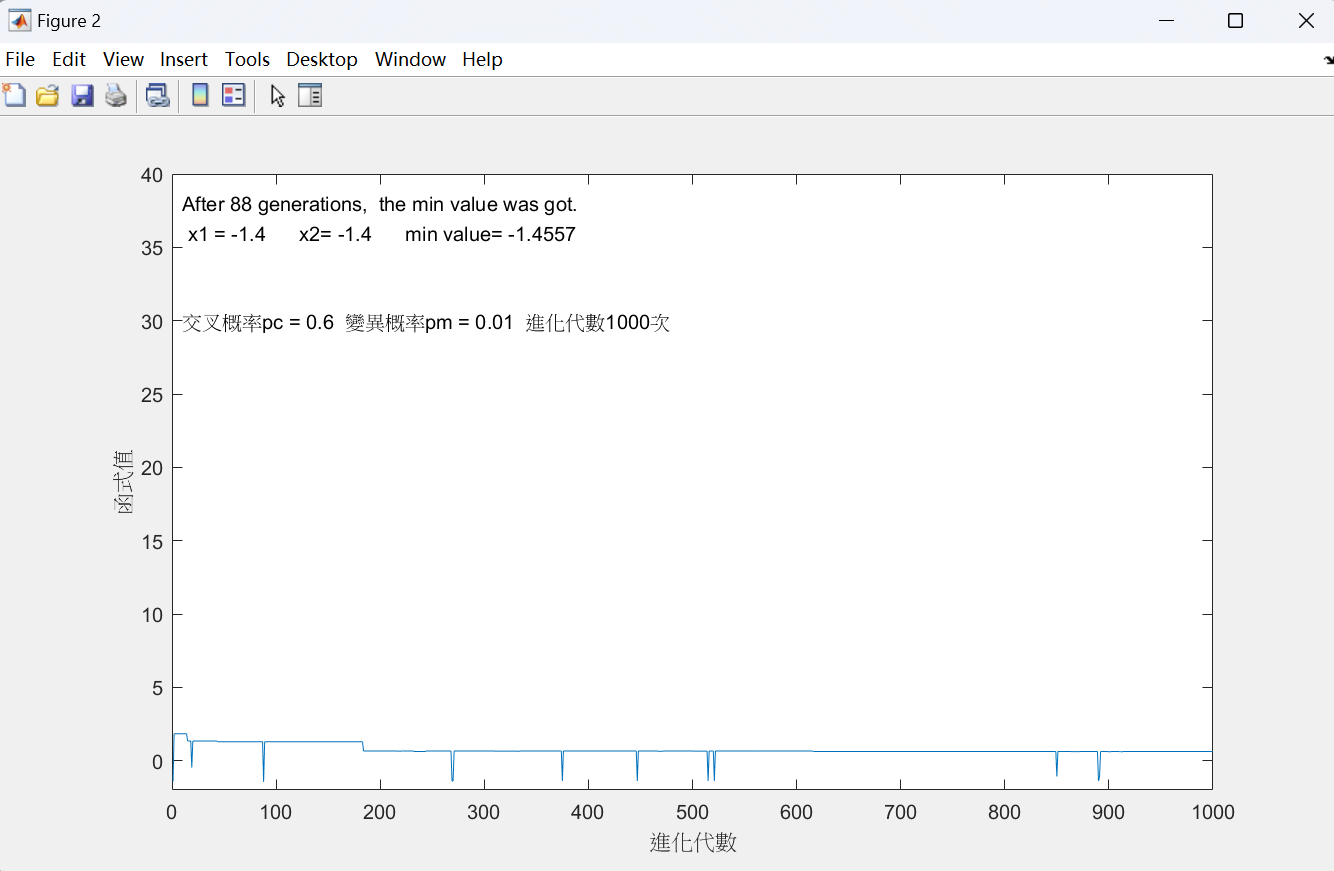


圖1. 族群大小設為200的函式圖

1. 設計適應函數 (決定個體適應度的評估標準)

* 因為題目是要求最小值，所以需選擇適應度越小的，代表目標函式值越接近最小值，輪盤的概率是根據適應度的倒數來計算的，這導致了較小的適應度值對應較大的概率，而增加了被選擇的機會。

1. 設計挑選、複製、交配與突變之機制

* 挑選的機制是利用輪盤法，先計算適應度值，轉盤的機率是適應度值的倒數來轉換的，適應度較小的會比較有機會被選擇。
* 交配的機制:對於每對相鄰的父代個體，根據給定的交叉概率pc進行判斷。如果隨機生成的一個介於 0 到 1 之間的數小於交叉概率 pc，則進行交叉操作；否則，不進行交叉操作；如果進行交叉，隨機選一個交叉點 cpoint，然後將兩個父代個體的對應部分進行交叉，形成新的子代個體；如果不進行交叉，則直接將兩個父代個體複製到新的種群中。
* 突變的機制:利用隨機數rand是否小於變異概率pm來決定要不要進行突變，如果rand小於pm，則選擇一個隨機的突變點mpoint，然後在mpoint上進行突變操作，將該位元從1變0，0變1。

1. 決定交配機率與突變機率

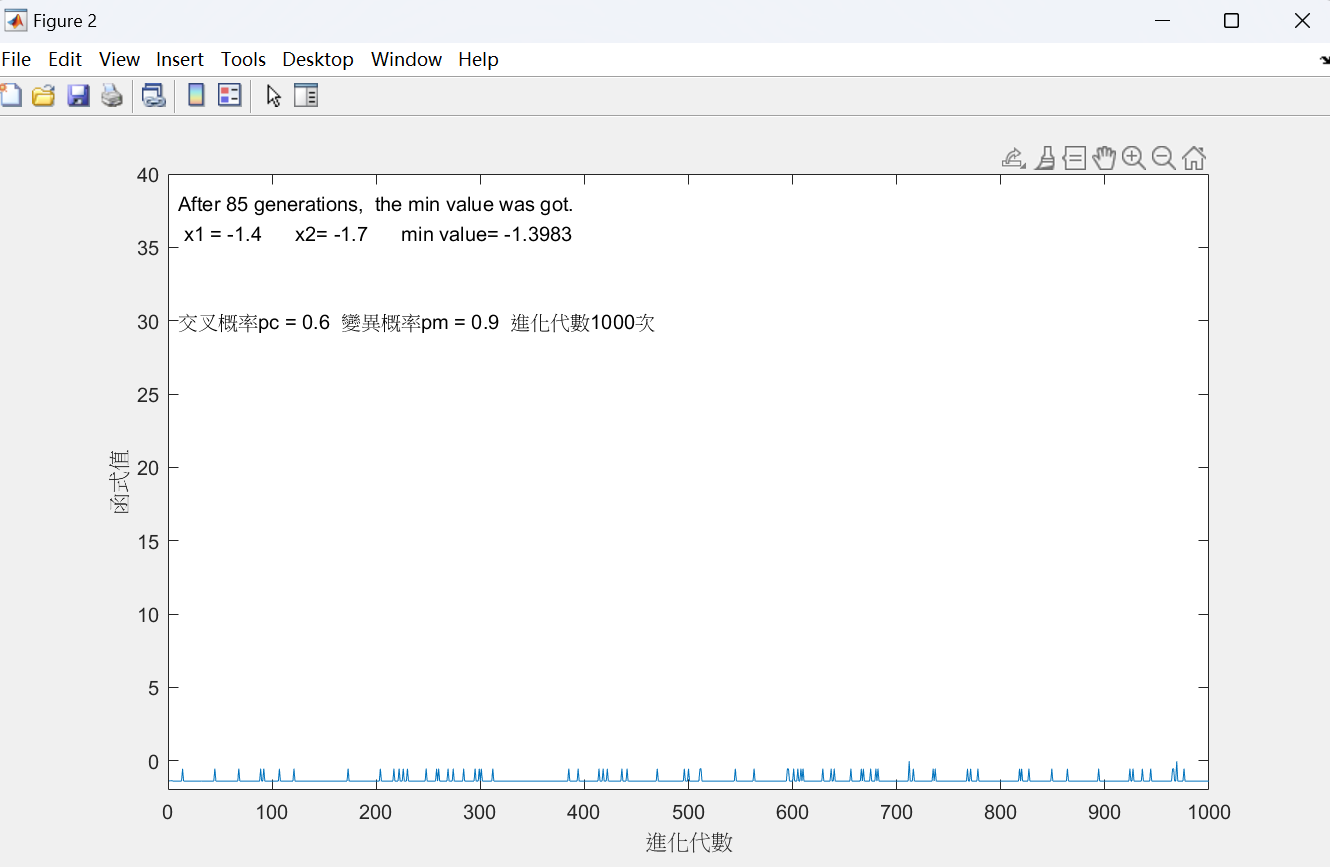
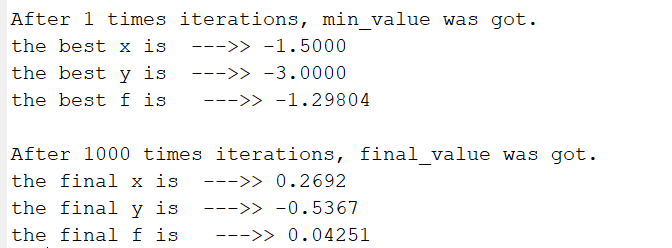
* 交配機率我是利用很常見的0.6做設定；突變機率則設定0.001，設定太大(0.9)也是會有震盪的情況發生(圖2)。

圖2. pm=0.9的函式圖

1. 決定終止條件

* 利用設定迭代次數的上限(1000次)，當程式執行了指定的迭代次數後停止。

1. 結果(答案)與討論(含繪出函數f之圖形與收斂過程圖)

* 結果:在執行該程式時，觀察到在迭代 1 次後就找到了最佳解。最佳解的具體值為：最佳 x 值為 -1.5000，最佳 y 值為 -3.0000，對應的目標函式值為 -1.29804。在迭代 1000 次後，最後獲得的解為：最後 x 值為 0.2692，最後 y 值為 -0.5367，對應的目標函式值為 0.04251(圖3)。
* 討論:雖然不曉得這種情況正不正常，我在執行這個程式時常常用到迭代1次就得到最佳解了，查詢網路的說法是可能是因為種群的初始狀態是隨機生成的關係，有時候可能會很巧的生成一個比較接近最佳解的個體，而在較早的迭代中就找到最佳解。

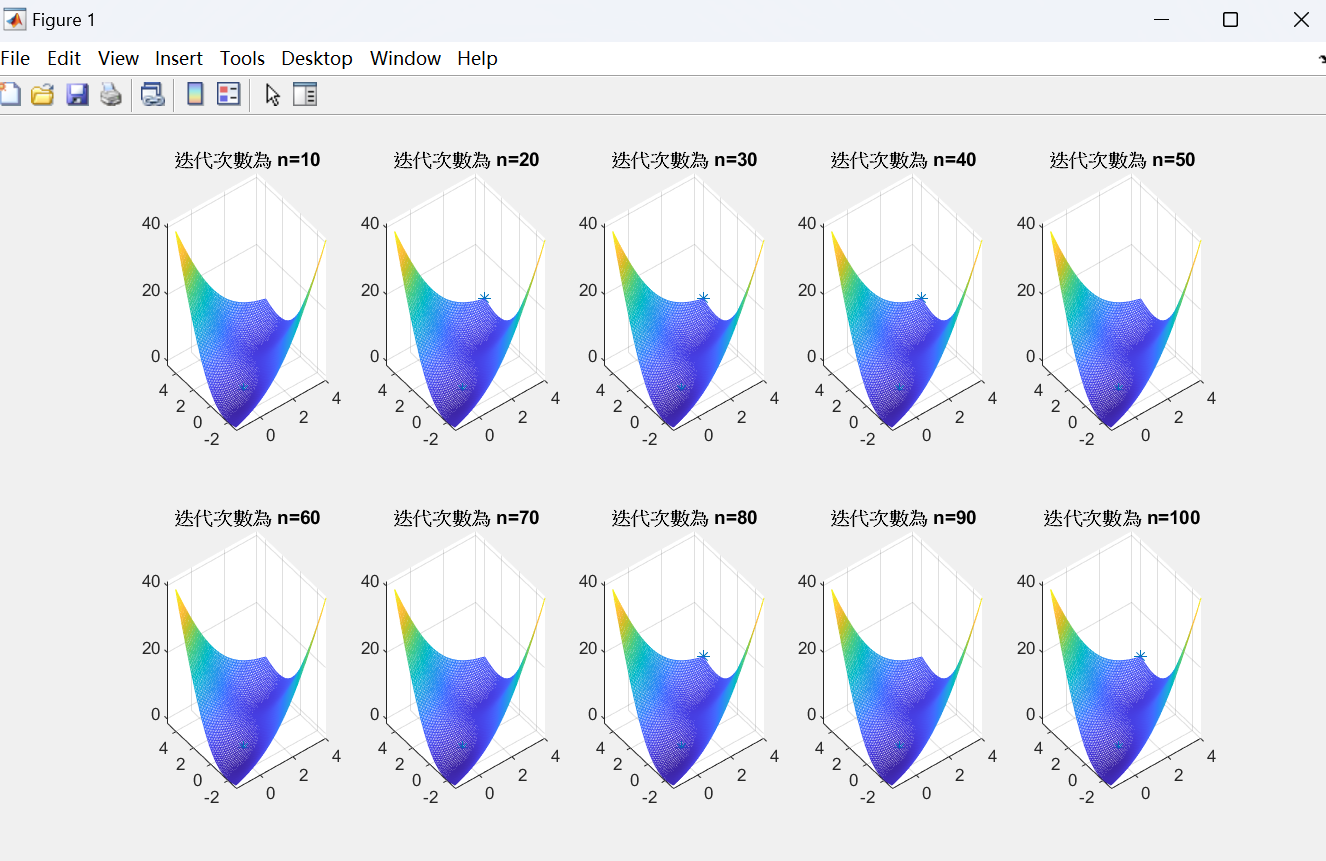
圖3

圖4

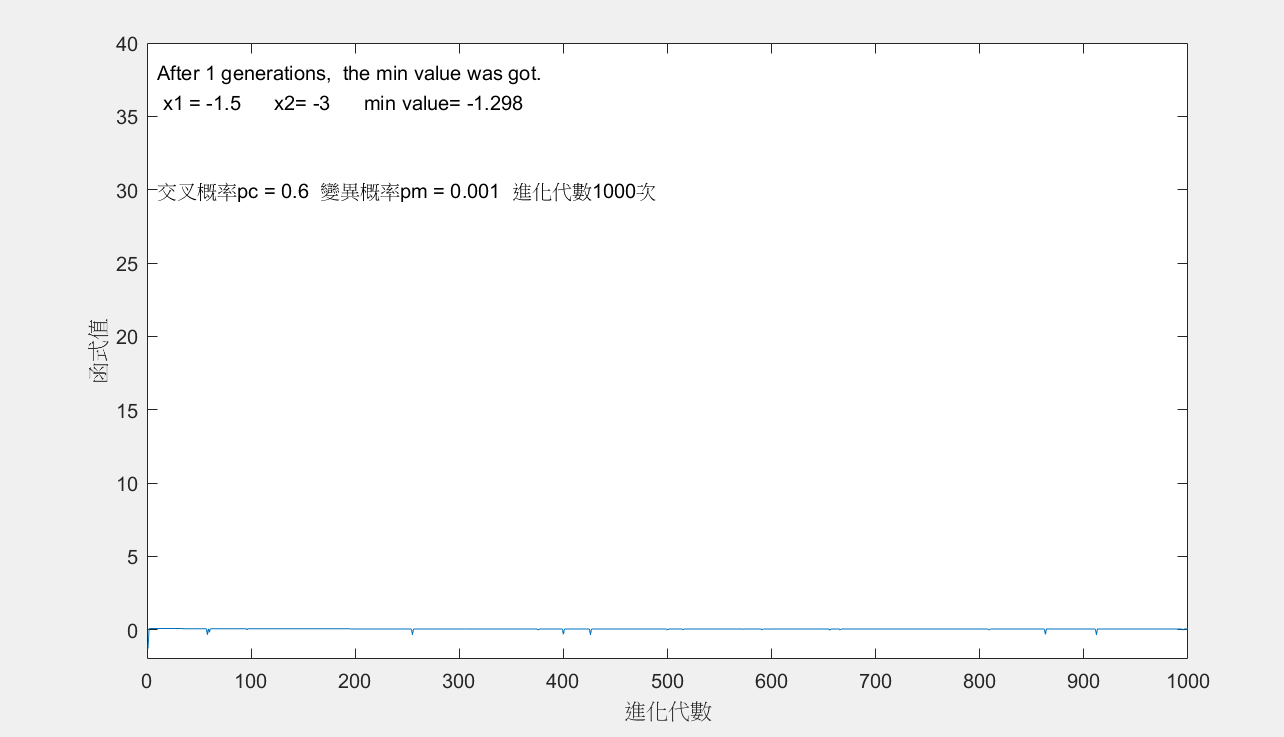
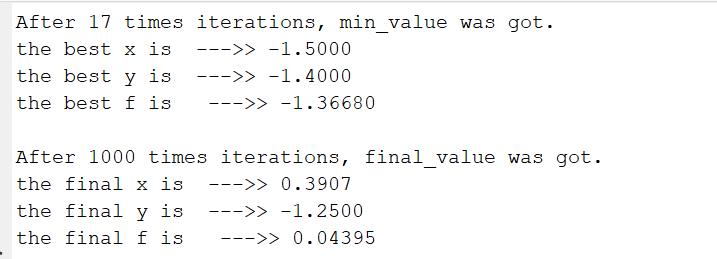
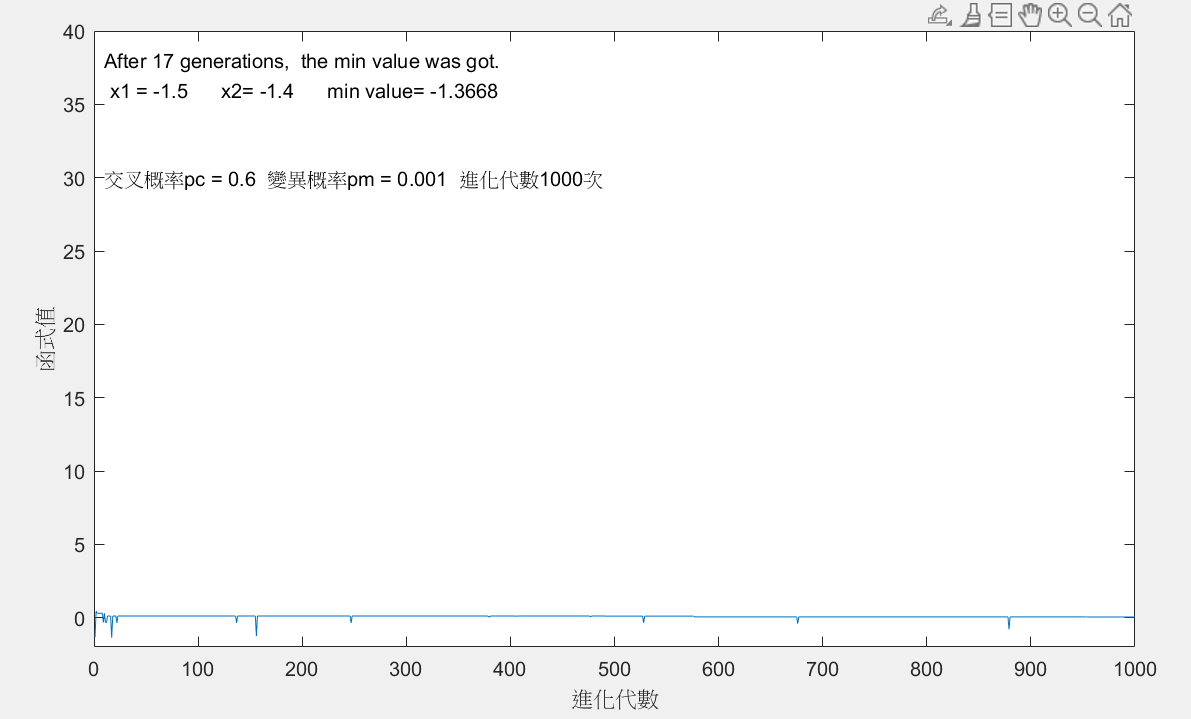


圖5

1. 感想與心得

* 第一次學習及實作基因演算法，還是覺得蠻難懂的，我的程式碼幾乎都是參考老師分享的連結下去做改寫，也不知道改得對不對，多次執行一直得到迭代1次就得最佳解，覺得好奇怪，但還是有少數幾次有用到別的迭代次數，附上迭代17次得最佳解的結果。

1. 程式碼:

function main()

clear

clc

close all

% 定義邊界

lb = [-1.5, -3];

ub = [4, 4];

% 定義基因演算法的參數

popsize = 100; % 種族大小

gen = 1000; % 迭代次數

x\_length = 16; % x長度

y\_length = 17; % y長度

L = 33; % 基因長度

pc = 0.6; % 交叉概率

pm = 0.001; % 變異概率

pop = initpop(popsize,L); %初始種群

for i = 1:gen % 迭代1000次

[objvalue] = cal\_objvalue(pop); % 計算函式目標值

fitvalue = objvalue; % 令適應度等於函式值

[newpop] = selection(pop,fitvalue); % 選擇操作

[newpop] = crossover(newpop,pc); % 交叉操作

[newpop] = mutation(newpop,pm); % 變異操作

pop = newpop; % 更新種群

% 將種群的每個個體表示出來

[A B] = binary2decimal(newpop); % 將newpop的二進制編碼轉為十進制

[y] = cal\_objvalue(newpop); % 計算nowpop的函式值

figure(1);

set(1, 'unit', 'normalized', 'position', [0.1,0.1,0.7,0.7]); % 設定圖形的視窗位置和大小

if i <= 100 & mod(i,10) == 0 % 每迭代10次做一次圖,畫100次以內的圖

j = floor(i/10); % 計算目前迭代次數的子圖位置

% 畫3D圖

X = -1.5:0.1:4; % x軸座標範圍

Y = -3:0.1:4; % y軸座標範圍

subplot(2,5,j); % 建立2\*5的子圖

[X, Y] = meshgrid(X,Y);

Z = (X-Y).^2 - X + 2.\*Y + sin(X+Y) + 1; % 計算目標函式Z的值

mesh(X,Y,Z);

hold on

title(['迭代次數為 n=' num2str(i)]); % 圖的標題

plot3(A,B,y,'\*'); % 在3D圖上繪製種群的點

end

[bestindividual,bestfit]=best(pop,fitvalue); % 尋找最優解

[x y] = binary2decimal(bestindividual); % 將二進位制值轉換為十進位制

BEST(i) = bestfit; % 將目前迭代次數下的最佳適應度值存在BEST陣列的第i個位置

X(i) = x; % 將目前迭代次數下的最佳x變數的十進制值存在X陣列的第i個位置

Y(i) = y; % 將目前迭代次數下的最佳y變數的十進制值存在Y陣列的第i個位置

end

[min\_value,index] = min(BEST'); % 找出BEST中的最小值和它的索引值

% 從X、Y中根據索引值取得最佳的x、y值

best\_x = X(index);

best\_y = Y(index);

figure(2);

set(2, 'unit', 'normalized', 'position', [0.1,0.1,0.7,0.7]);

i = 1:1000;

plot(i,BEST);

axis([0,1000,-2,40]); % 圖形的座標軸範圍

xlabel('進化代數');

ylabel('函式值');

text(10,30,'交叉概率pc = 0.6 變異概率pm = 0.001 進化代數1000次');

text(10,38,['After ',num2str(index),' generations,',...

' the min value was got.']);

text(10,36,[' x1 = ',num2str(best\_x),' x2= ',num2str(best\_y),...

' min value= ', num2str(min\_value)]);

fprintf('After %.0f times iterations, min\_value was got.\n',index);

fprintf('the best x is --->> %5.4f\n',best\_x);

fprintf('the best y is --->> %5.4f\n',best\_y);

fprintf('the best f is --->> %5.5f\n',min\_value);

fprintf('\n');

fprintf('After %.0f times iterations, final\_value was got.\n',1000);

fprintf('the final x is --->> %5.4f\n',x);

fprintf('the final y is --->> %5.4f\n',y);

fprintf('the final f is --->> %5.5f\n',bestfit);

function pop = initpop(popsize,L)

%% -------------初始化種群函式----------------

% 初始化種群大小

% 輸入變數：

% popsize:種群大小

% L：基因長度--》轉化的二進位制長度

% 輸出變數：

% pop：種群

%---------------------------------------

pop = round(rand(popsize,L)); % 隨機產生一個矩陣，每一行是一個長33位染色體；

function [pop1 pop2] = binary2decimal(pop)

%% -----------解碼函式---------------------

% 二進位制轉化為十進位制數

% 輸入變數：

% 二進位制種群

% 輸出變數：

% 十進位制數值

%-----------------------------------------

% 遍歷前16位二進制數，並轉成十進制數放到pop\_x

for i = 1:16

pop\_x(:,i) = 2.^(16 - i).\*pop(:,i);

end

% 遍歷後17位二進制數，並轉成十進制數放到pop\_y

for j = 1:17

pop\_y(:,j) = 2.^(17 - j).\*pop(:,j+16);

end

% sum(.,2)對行求和，得到列向量

temp1 = sum(pop\_x,2);

temp2 = sum(pop\_y,2);

pop1 = -1.5 + temp1\*5.5/(2^16-1); % pop1表示輸出x1的十進位制數

pop2 = -3 + temp2\*7/(2^17-1); % pop2表示輸出的x2的十進位制數

function [objvalue] = cal\_objvalue(pop)

%% --------------計算函式值函式----------------------

% 計算函式目標值

%輸入變數：二進位制數值

%輸出變數：目標函式值

%---------------------------------------------

[x y] = binary2decimal(pop);

objvalue = (x-y).^2 - x + 2\*y + sin(x+y) + 1;

function [newpop] = selection(pop,fitvalue)

%% -----------------根據適應度選擇函式-------------------

% 輸入變數 ：pop:二進位制種群

% fitvalue: 適應度

%輸出變數： newpop: 選擇以後的二進位制種群

% -------------------------------------------

[px,py] = size(pop); % 獲取種群的大小

totalfit = sum(fitvalue); % 計算fitvalue的總和

p\_fitvalue = 1 ./ fitvalue / sum(1 ./ fitvalue); % 計算每個個體被選擇的概率

p\_fitvalue = cumsum(p\_fitvalue); % 概率求和後排序

ms = sort(rand(px,1)); % 產生一列隨機數，從小到大排列，相當於轉轉盤10次

fitin = 1; % 初始化適應度索引為 1

newin = 1; % 初始化新種群索引為 1

while newin <= px

if(ms(newin)) < p\_fitvalue(fitin) % 轉盤轉到 fitin 的位置

newpop(newin,:) = pop(fitin,:); % 新種群的第 newin 個體為pop中的第fitin 個體

newin = newin + 1;

else

fitin = fitin + 1; % 相當於每次都從第一個比較起，依次加1，直至比較完，看轉到的是哪一個

end

end

function [newpop] = crossover(pop,pc)

%% ----------交叉函式--------------------

% 輸入變數：pop:二進位制的父代種群數

% pc :交叉概率

% 輸出變數：newpop: 交叉後的種群數

%---------------------------------------

[px,py] = size(pop);

newpop = ones(size(pop)); % 初始化新種群為全 1 矩陣，提高運算速度

for i = 1:2:px-1 % 1與2交叉。3與4交叉。。。。。每次隔一個，因此步長為2

if (rand<pc) % pc = 0.6,即有60%的機會交叉

cpoint = round(rand\*py); % 交叉點隨機選取,互換交叉點以後的值

if cpoint <= 0

% cpoint = 1;

continue;

end

newpop(i,:) = [pop(i,1:cpoint),pop(i+1,cpoint+1:py)]; % 交叉後的第i個個體

newpop(i+1,:) = [pop(i+1,1:cpoint),pop(i,cpoint+1:py)];

else % 40%的機會不交叉

newpop(i,:) = pop(i,:);

newpop(i+1,:) = pop(i+1,:);

end

end

function [newpop] = mutation(pop,pm)

%% ------------變異函式---------------------------

% 輸入變數 pop: 二進位制種群

% pm : 變異概率

% 輸出變數： newpop : 變異以後的種群

%-----------------------------------------------

[px,py] = size(pop);

newpop = ones(size(pop)); % 只是起到提前宣告的作用，提高運算速度

for i = 1:px % 對於種群中的每個個體執行變異操作

if(rand<pm) % 如果rand小於變異概率 pm，則進行變異操作

mpoint = round(rand\*py); % 隨機選擇變異點

if mpoint<=0

mpoint = 1;

end

newpop(i,:) = pop(i,:); % 將當前個體複製到新種群中

if newpop(i,mpoint) == 0 % 若mpoint為0，則改成1，mpoint為1，則改為0，完成變異

newpop(i,mpoint) = 1;

else

newpop(i,mpoint) = 0;

end

else % 如果rand >= pm，則不進行變異操作，直接把當前個體複製到新種群中

newpop(i,:) = pop(i,:);

end

end

function [bestindividual, bestfit] = best(pop, fitvalue)

%% --------------選出最優個體函式-----------------------

% 輸入變數： pop :種群

% fitvalue : 種群適應度

% 輸出變數： bestindividual : 最佳個體（二進位制）

% bestfit : 最佳適應度值

% ---------------------------------------------

[px, py] = size(pop);

% 檢查種群和適應度值的維度是否一致

if px ~= length(fitvalue)

error('種群和適應度值的維度不一致');

end

% 確認種群不為空

if px == 0

error('種群為空');

end

bestindividual = pop(1,:); % 初始化最佳個體為種群中的第一個個體

bestfit = fitvalue(1); % 初始化最佳適應度為種群中的第一個個體的適應度值

% 初始化最佳適應度值為一個極大的數字

bestfit = inf;

for i = 2:px % 遍歷種群中的每個個體，從第二個個體開始

if fitvalue(i) < bestfit % 如果當前個體的適應度值比最佳適應度值更佳

bestindividual = pop(i,:); % 更新最佳個體為當前個體

bestfit = fitvalue(i); % 更新最佳適應度值為當前個體的適應度值

elseif fitvalue(i) == bestfit % 如果適應度值相等

% 隨機選擇一個作為最佳個體

if rand < 0.5

bestindividual = pop(i,:);

end

end

end