1. **重力搜尋演算法(Gravitational Search Algorithm, GSA)**

重力搜尋演算法是一種可以用來解決最佳化問題的演算法，將問題等化成物理上的重

力，質量間彼此會互相吸引，產生力量，並藉彼此吸引的程度找出最佳解。

**| 實驗設計步驟**

1. 產生原始族群

隨機產生50組搜尋範圍內的原始族群所在位置，這裡的原始族群即為物體質量。

1. 計算適應度

將50組原始族群分別計算其累加誤差平方和的值，經過平均後將其取平方根，再由適應函數計算求出各質量適應度。



1. 求出各質量所佔比例(mass\_Calculation.m)

使用式(3.1)求出個別質量，再利用式(3.2)求出每個質量所佔的比例，其概念類似GA演算法的複製率。若是要找出最大質量，則Best為適應度最大的，worst則為適應度最小的。

|  |  |
| --- | --- |
|  | (3.1) |

|  |  |
| --- | --- |
|  | (3.2) |

1. 算出吸引力(Gfield.m)

將彼此間的質量與距離代入改良式重力公式，如式(3.3)，即可求得彼此間的吸引力。

其中，G(t)值大小會隨時間改變，變化方式如式(3.4)。F取最好的值數量本實驗則是設定不會隨著時間減少而減少，如式(3.5)。Ma為影響別人的強度(Active gravitational mass)，Mp為被影響的強度(Passive gravitational mass)，而Mi為自己的質量，Mii為第i個族群的慣性質量，本實驗將這些質量都設定為自己的質量，即Mi = Mp = Ma = Mii。R為兩個族群間的歐幾里得距離。Ԑ=8.85x10-12

|  |  |
| --- | --- |
|  | (3.3) |

|  |  |
| --- | --- |
|  | (3.4) |

|  |  |
| --- | --- |
|  | (3.5) |

1. 算出加速度並更新速度及位置

由式(3.6)即可算出加速度，並藉由式(3.7)及式(3.8)分別更新速度及位置。

|  |  |
| --- | --- |
|  | (3.6) |

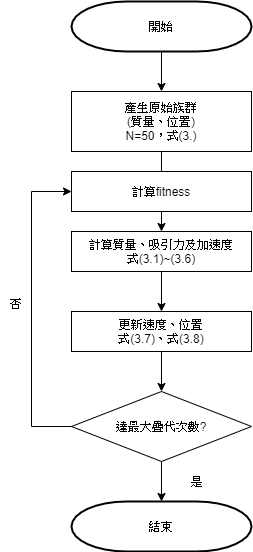
|  |  |
| --- | --- |
|  | (3.7) |

|  |  |
| --- | --- |
|  | (3.8) |

1. 搜尋範圍限定(space\_bound.m)

依照題目制定的搜尋範圍，避免族群在進行GSA演算法的過程中更新位置時，新一代的族群超出上下限。

1. 演算法流程圖



圖三 GSA演算法程式流程圖

1. 參數設定

N = 50; 群體個數

iter = 50; 遞迴次數

alpha = 20; 重力常數之參數

G0 = 100; 重力常數之初值

**實驗與模擬結果**



圖 1 、第一組初始值之搜尋結果



圖 2 、第一組初始值之實際追蹤軌跡(y)與參考值(y𝑑)軌跡



圖 3 第一組初始值之追蹤誤差(e)



圖 4 第一組初始值之力量輸入(u)



圖 5 第二組初始值之搜尋結果



圖 6 第二組初始值之實際追蹤軌跡(y)與參考值(y𝑑)軌跡



圖 7 第二組初始值之追蹤誤差(e)



圖 8 第二組初始值之力量輸入(u)

****

圖 9 第三組初始值之搜尋結果

****

圖 10 第三組初始值之實際追蹤軌跡(y)與參考值(y𝑑)軌跡

****

圖 11 第三組初始值之追蹤誤差(e)

****

圖 12 第三組初始值之力量輸入(u)

Q3. Please explain why the fitness function, all the parameters and settings that are chosen.

由於智慧型控制需要定義適應值函數(fitness function)，藉此作為評判標準來學習最佳解，將GSA搜尋出的Kp、Kd、Kdd代入控制器中，並將輸出的位置與理想的位置求誤差，本次平衡系統的依據是以位置誤差取RMSE後再加1的導數，因此所求的fitness越大平衡結果應該越好，代表輸出的位置與理想的值非常接近。

**Matlab Code**

**main.m**

clear;clc;close all;

format long e;

dt = 0.001;

t = linspace(0, 10, 10/dt) ;

fid1 = fopen('problem1.txt','w');

fid3 = fopen('problem3.txt','w');

fid4 = fopen('problem4.txt','w');

for A = 1:1

tic

%可調參數

% index\_F=12; %testbench function編號

max\_iter=500; %迭代次數

run = 1;

N=50;%agent數量

min\_flag=0; % 1: minimization, 0: maximization

% alpha = 20;G0 = 100;%p41重力公式參數

final\_per = 2;%最後一次迭代agent參與的比例

% best = [];

for runn = 1:run

[last\_fitness(runn,:),Fbest,Lbest,BestData(runn,:),MeanData(runn,:),MedianData(runn,:)]=GSA\_1(N,max\_iter,min\_flag,final\_per);

end

toc

time = toc;%存在problem 2

last\_fitness = mean(last\_fitness,1);

%將資料存入檔案中

%problem 1

p1 = BestData(1,:);

figure();

plot(1:1:max\_iter,p1);

title(['A = ',num2str(A),'-searching results ,iteration =',num2str(max\_iter)]);

for kk = 1:max\_iter

fprintf(fid1,'%f\t',p1(kk));

end

fprintf(fid1,'\r\n');

fprintf("K: ");

for i=1:3

fprintf("%f ", Lbest(i));

end

[fitness, y, u, err] = ballbeam(Lbest, A);

t = 0:0.001:10;

figure();

plot(t, y, 'r-', t, A\*cos(t\*pi/5), 'b--');

title('實際追蹤軌跡');

legend('y','yd');

figure();

plot(t, err, 'r-');

title('追蹤誤差');

legend('e');

tspan = 0.001:0.001:10;

figure();

plot(tspan, u);

title('u');

legend('u');

end

fclose(fid1);

fclose(fid3);

fclose(fid4);

**ballbeam.m**

function [fitness, y, u, err] = ballbeam(X, A)

kp = X(1); kd = X(2); kdd = X(3);

B = 0.7; G = 9.81; dt = 0.001; t = 10;

if A==1

init\_condition = 0.0564;

elseif A==2

init\_condition = 0.1129;

elseif A==3

init\_condition = 0.1698;

end

fitness = 0;

yd = zeros(t/dt + 1);

n = 0;

for i = 0:dt:t

n = n + 1;

yd(n) = A\*cos(pi\*i/5);

end

x4 = 0;

x3 = init\_condition;

x2 = 0;

y = zeros(t/dt+1, 1);

err = zeros(t/dt+1, 1);

u = zeros(t/dt, 1);

y(1) = A;

e0 = 0;

de0 = 0;

for i = 1:t/dt

e = y(i) - yd(i);

err(i) = e;

de = (e-e0)/dt;

u(i) = kp\*e + kd\*(e-e0)/dt + kdd\*(de-de0)/dt;

if u(i) > 20

u(i) = 20;

elseif u(i) < -20

u(i) = -20;

end

x4n = x4 + u(i)\*dt;

x3n = x3 + x4\*dt;

x2n = x2 + B\*(y(i)\*x4\*x4 - G\*sin(x3))\*dt;

y(i+1) = y(i) + x2\*dt;

x4 = x4n;

x3 = x3n;

x2 = x2n;

e0 = e;

de0 = de;

fitness = fitness + e^2;

end

fitness = fitness/(t/dt);

fitness = fitness^0.5;

fitness = 1/(1+fitness);

end

**GSA\_1.m**

function [last\_fitness,Fbest,Lbest,BestData,MeanData,MedianData, fitness]=GSA\_1(N,max\_iter,min\_flag,final\_per)

%初始化

dim = 3;

up = 350;

down = 0;

X=rand(N,dim).\*(up-down)+down;

V=zeros(N,dim);

last\_fitness = zeros(1,N);

%存數據

BestData=[];

MeanData=[];

MedianData=[];

fitness = zeros(N, 2); % 初始化 fitness 向量

for iter=1:max\_iter

%檢查是否越界，越界則該X重新初始化!!!

i=1;

while i<=N

for j = 1:dim

if any(X(i,:)>up) || any(X(i,:)< down)

X(i,:) = rand(1,dim).\*(up-down)+down;

end

end

i=i+1;

end

%計算agent的fitness

A = 1;

fitness = zeros(N, 1);

for i=1:N

[fitness(i), y, ~, ~] = ballbeam(X(i,:), A);

% x = X1(i,:);

% y = X2(i,:);

% fitness(i)=extra\_functions\_modify(x,y,index\_F,dim);

%

end

if i == N

last\_fitness = fitness;

end

if min\_flag==1

[best, best\_X]=min(fitness); %minimization.

else

[best, best\_X]=max(fitness); %maximization.

end

if iter==1%第一代

Fbest=best;Lbest=X(best\_X,:);

end

if min\_flag==1

if best<Fbest %minimization.

Fbest=best;Lbest=X(best\_X,:);

end

else

if best>Fbest %maximization

Fbest=best;Lbest=X(best\_X,:);

end

end

BestData=[BestData Fbest];

MeanData=[MeanData mean(fitness)];

MedianData =[MedianData median(fitness)];

%eq.14-20計算

[M]=massCal(fitness,min\_flag);

%參考投影片p41公式

G = Gconstant(iter,max\_iter);

%Calculation of accelaration in gravitational field. eq.7-10,21.

a=Gfield(M,X,G,iter,max\_iter,final\_per);

%Agent movement. eq.11-12

[X,V]=move(X,a,V);

end %iteration

**massCal.m**

function [M]=massCal(fit,min\_flag)

Fmax=max(fit); Fmin=min(fit); Fmean=mean(fit);

N = size(fit,1);

if Fmax==Fmin

M=ones(N,1);

else

if min\_flag==1 %for minimization eq. 19、20

best=Fmin;worst=Fmax;

else %for maximization

best=Fmax;worst=Fmin;

end

M=(fit-worst)./(best-worst); %eq.15,

end

M=M./sum(M); %eq. 16.

**Gconstant.m**

function G = Gconstant(iter,max\_iter)

%p41重力公式參數

alpha = 20;

G0 = 100;

G=G0\*exp(-alpha\*iter/max\_iter);

**Gfield.m**

function a = Gfield(M,X,G,iter,max\_iter,final\_per)

[N,dim]=size(X);

kbest=final\_per+(1-iter/max\_iter)\*(100-final\_per);

kbest=round(N\*kbest/100);%四捨五入

[~, index]=sort(M,'descend');

F = zeros(N, dim);

for i=1:N

for ii=1:kbest

j=index(ii);

if j~=i

R=norm(X(i,:)-X(j,:),2); %eq.8

for k=1:dim

F(i,k)=F(i,k)+rand\*(M(j))\*((X(j,k)-X(i,k))/(R+eps));%eq.7 、9

end

end

end

end

a=F.\*G; %eq. 10

**move.m**

function [X,V]=move(X,a,V)

%movement.

[N,dim]=size(X);

V=rand(N,dim).\*V+a; %eq. 11.

X=X+V; %eq. 12.