ภาคผนวก ก

คำสั่งที่ใช้ในโปรแกรม MATLAB

1 การกำหนดค่าที่จะใช้ใน GA

```
ff='testfunction';                  % สมการใน test function
npar=2;                             % จำนวนตัวแปร คือ ค่าในแนวแกน x และแกน y
varhi_y=300; varlo_y=100;           % ค่าสูงสุดและต่ำสุดในแนวแกน y
varhi_x=100; varlo_x=0;             % ค่าสูงสุดและต่ำสุดในแนวแกน x
```

---

2 การตั้งค่าจำนวนการทำซ้ำ

```
maxit =100;                         % ค่าสูงสุดในการทำซ้ำ
mincost = -9999999;                 % ค่าน้อยที่สุดในการทำซ้ำ
```

---

3 ตัวแปรที่ใช้ใน GA

```
popsize=12;                         % จำนวนของประชากรเริ่มต้น( Initial  Population )
mutrate=0.2;                        % อัตราการดัดแปลง ( mutation )
selection=0.5;                      % สัมประสิทธิ์ที่จะใช้ในการเก็บค่า Population
Nt=npar;                            % จำนวนตัวแปรที่ใช้ใน GA continuous
keep=floor(selection*popsize);      % จำนวน Population ที่ทำการเก็บไว้
                                    % คำสั่ง floor คือ คำสั่งในการปัดเลขลงให้เป็นจำนวนเต็ม
nmut=ceil((popsize-1)*Nt*mutrate);      % จำนวน Population ที่ถูกทำการ mutation
                                    % คำสั่ง ceil คือ คำสั่งในการปัดเลขลงให้เป็นจำนวนเต็ม
M=ceil ((popsize-keep)/2);          % จำนวนของการจับคู่
```

---

4 การสร้าง initial popnlation

```
iga=0;                              % ค่าเริ่มต้นในการทำซ้ำ
```

```
par=(varhi-varlo)*rand(popsize,npar)+varlo; % ค่าของPopulation ที่ถูกสุ่มออกมา
```

% ค่า Par จะเป็นค่าในแนวแกน x และแกน y ของกราฟในแต่ละจุดที่ถูกสุ่มขึ้นมาเพื่อเป็นค่าประชากรเริ่มต้น(Population) โดยในที่นี้จะสุ่มออกมา 12 จุด

```
[cost,ind]=sort(cost);      % ชี้ลำดับค่าเฉพาะของรอยนิ้วมือของ Population ตามลำดับ
                              ค่ามากไปน้อยและเรียงลำดับค่าที่มีค่าน้อยไปหามาก
par=par(ind,:);             % เรียงลำดับค่า Par ตามหมายเลขที่ถูกชี้
minc(1)=min(cost);          % แสดงค่าเฉพาะของรอยนิ้วมือที่น้อยที่สุดของ Population
meanc(1)=mean(cost);        % แสดงค่าเฉลี่ยของค่าเฉพาะของรอยนิ้วมือของ Population
```

---

5 การทำซ้ำตลอดทั้งโปรแกรม

```
while iga<maxit
 iga=iga+1;                 % จำนวนการทำซ้ำที่เพิ่มขึ้น
```

---

6 การจับคู่

```
M=ceil((popsize-keep)/2);              % จำนวนของการจับคู่
prob=flipud([1:keep]'/sum([1:keep]));  % weights chromosomes
odds=[0 cumsum(prob(1:keep))'];        % ความน่าจะเป็นในการกระจาย function
                                       % คำสั่ง cumsum คือ คำสั่งการรวมค่าให้สะสมไป
                                         เรื่อยๆ
pick1=rand(1,M);                       % คู่ที่ 1
pick2=rand (1,M);                      % คู่ที่ 2
                                       % คำสั่ง rand คือ คำสั่งที่ให้คอมพิวเตอร์สุ่มค่าตัวเลข
                                         ออกมา โดยตัวเลขที่ถูกสุ่มออกมาจะมีค่าไม่ถึง 1
                                         และค่าที่ถูกสุ่มออกมาแต่ละครั้งจะมีค่าไม่เท่ากัน
```

7 การชี้ตำแหน่งของ ma และ pa เพื่อนำมาจับคู่

```
ic=1;
while ic<=M
for id=2:keep+1
if pick1(ic)<=odds(id) & pick1(ic)>odds(id-1)
ma(ic)=id-1;
end
if pick2(ie)<=odds(id) & pick2(ic)>odds(id-1)
pa(ic)=id-1
end
end
ic=ic+1;
end
```

---

8 การสลับตำแหน่งของคู่ที่ถูกเลือก (crossover)

```
ix=1:2:keep;
xp=ceil(rand(1,M)*Nt);
r=rand(1,M);
for ic=1:M
xy=par(ma(ic),xp(ic))-par(pa(ic),xp(ic));
par(keep+ix(ic),:)=par(ma(ic),:);
par(keep+ix(ic)+1,:)=par(pa(ic),:);
par(keep+ix(ic),xp(ic))=par(ma(ic),xp(ic)) - r(ic).*xy;
if xp(ic)<npar
par(keep+ix(ic),:)=[par(keep+ix(ic),1:xp(ic))   par(keep+ix(ic)+1,xp(ic)+1:npar)];
par(keep+ix(ic)+1,:)=[par(keep+ix(ic)+1,1:xp(ic))  par(keep+ix(ic),xp(ic)+1:npar)];
end % if
end
```

---

9 การดัดแปลงค่าของ population (mutation)

```
mrow=sort(ceil(rand(1,nmut)*(popsize-1))+1);
mcol=ceil(rand(1,nmut)*Nt);
for ii=1:nmut
if mcol(ii)==1;
par(mrow(ii),mcol(ii))=(varhi_x-varlo_x)*rand+varlo_x;    % mutation
end
if mcol(ii)==2;
par(mrow(ii),mcol(ii))=(varhi_y-varlo_y)*rand+varlo_y;    % mutation
end
end
```

---

10 การหาค่าเฉพาะของรอยนิ้วมือของ population ใหม่หลังจากทำการ mutation

```
cost=feval(ff,par);
```

---

11 การชี้ค่าเฉพาะของรอยนิ้วมือและทำการจัดเรียงค่าเฉพาะของรอยนิ้วมือของ population ใหม่

```
[cost,ind]=sort(cost);          % ชี้ลำดับค่าเฉพาะของรอยนิ้วมือของ Population ตาม
                                  ความลำดับค่ามากน้อย
                                % และเรียงลำดับค่าเฉพาะของรอยนิ้วมือที่มีค่าน้อยไปหามาก
par=par(ind,:);                 % เรียงลำดับค่า Par ตามหมายเลขที่ถูกชี้
```

---

12 การหาค่าเฉพาะที่น้อยที่สุดและการหาค่าเฉลี่ยของ population ใหม่

```
minc(iga+1)=min(cost);          % การหาค่าเฉพาะของรอยนิ้วมือที่น้อยที่สุดของ Population
meanc(iga+1)=mean(cost);        % การหาค่าเฉพาะของรอยนิ้วมือเฉลี่ยของ Population
```

---

13 การหยุดการทำซ้ำ

```
if iga>maxit | cost(1)<mincost    % ถ้าจำนวนรอบของการทำงานของโปรแกรมมากกว่า
                                  ค่าจำนวนการทำซ้ำที่ตั้งไว้ตอนแรกหรือค่าเฉพาะของรอยนิ้ว
                                  มือที่ได้มีค่าน้อยกว่าค่า mincost ที่กำหนดไว้ตอนแรก
                                  โปรแกรมจะหยุดการทำงาน
break
end
[iga cost(1)]
end
```

---

14 การแสดงผล

```
par                               % แสดงค่า par
day=clock;                        % แสดงเวลาที่ทำการประมวลผลโปรแกรม
disp(datestr(datenum(day(1),      % แสดงวัน, เดือน, ปีที่ทำการประมวลผลโปรแกรม
day(2),day(3),day(4),day(5),day(6)),0))
format short g
disp(['popsize = ' num2str(popsize) ' mutrate = ' num2str(mutrate) ' # par = ' num2str(npar)])
disp(['#generations=' num2str(iga) ' best cost=' num2str(cost(1))])
disp(['best solution'])           % แสดงค่าเฉพาะของรอยนิ้วมือที่เหมาะสมที่สุด
disp([num2str(par(1,:))])
disp('continuous genetic algorithm')
figure(24)                                % แสดงรูปการประมวลผลโปรแกรม
iters=0:length(minc)-1;
plot(iters,minc,iters,meanc,'red');
xlabel('generation');ylabel('Fitness Value'); %ตั้งชื่อแกน X และแกน Y ตามลำดับ
text(0,minc(1),'best');text(1,minc(2),'population average')
```