**Prepared By:**

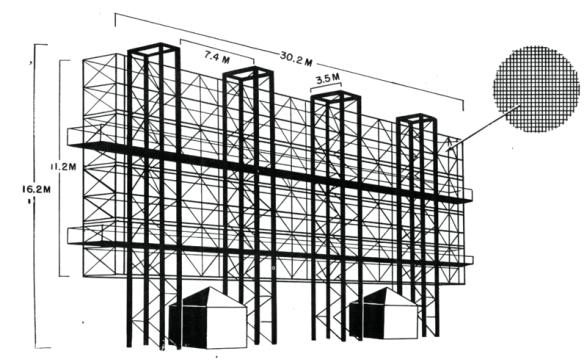
**Particle Swarm Optimization**

|  |  |
| --- | --- |
| Name | Section |
| Ahmed Magdy Mohamed Zaki | 1 |
| Ahmed Abd El-latif Mhana | 1 |
| Ahmed Alaa El-din Abd El-Wahab | 1 |
| Abd El-ghani Mohamed Abd El-ghani | 4 |
| Abd Allah Ramadan | 4 |

**Introduction:**

Antenna arrays: is a group of isotropic radiators such that the currents running through them are of different amplitudes and phases. These are radiators of electromagnetic frequency and energy, a configuration of multiple antennas (elements) arranged to achieve a given radiation pattern. A widely known system for covering an area with a specific signal.

**Brief History of Antenna arrays:**

Antenna array system was known since the world war I but never been used due to the cost of building and transmitting signals, the wireless technology then becomes important and receives many developments and the antenna array becomes a widely known system .

**Types of Antenna arrays:**

Linear array*:* antenna elements arranged along a straight line.

Circular array*:* antenna elements arranged around a circular ring.

Planar array*:* antenna elements arranged over some planar surface

Conformal array*:* antenna elements arranged to conform to some non-planar surface (such as an aircraft skin).

**Design Variables of Antenna arrays:**

1. General array shape (linear, circular, planar, etc.).

2. Element spacing.

3. Element excitation amplitude.

4. Element excitation phase.

5. Patterns of array elements.

**Mathematical Overview of Antenna arrays:**

Phased array*:* an array of identical elements which achieves a given pattern through the control of the element excitation phasing.

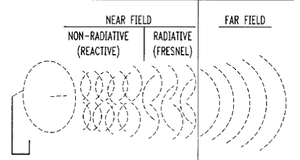
Far field of an element: describes regions around the source where different parts of the field are more or less important.

Array factor*:* a function dependent only on the geometry of the array and the excitation (amplitude, phase) of the elements.

Radiation pattern for identical elements of the antenna array =

***Array Pattern = Array Element Pattern x Array Factor (AF) (1)***

Field of an isotropic radiator at the origin: 1Antenna Field



Assuming that antennas are equally distanced d :

Far field of an element=

Using equation ( 1) :

The overall array far field is found using superposition =

=

**Main improvement trend in Antenna arrays:**

 Better control of direction than an isotropic (Omni-directional) antenna - as the length of the dipole increases, the control of direction decreases. Hence control by changing the length of a single antenna is very limited. Greater flexibility and control can be obtained for directing the beam with an arrangement of multiple radiators

**Smart Antenna:**

Identify spatial signal signature such as the direction of arrival (DOA) of the signal, and use it to calculate beam forming vectors, to **track and locate the antenna beam on the mobile/target**. The antenna could optionally be any sensor.

Beam forming is the method used to create the radiation pattern of the antenna array by adding constructively the phases of the signals in the direction of the targets/mobiles desired.

**Tracking and locate the antenna beam on the mobile/target:**

That’s when Particle swarm optimization is used.

**Particle swarm optimization (PSO):**

Is an algorithm similar in some ways to Genetic Algorithms (GA) and otherevolutionary algorithms, but requires less computational bookkeeping in additions that the basic algorithm is easy to understand and implement.

**PSO Basic Algorithm:**

1- Initialize particles with random positions and velocities in *M* dimensions in the problem space.

2- For each particle, evaluate the desired optimization (fitness) function in *M* variables.

3- Update the particle velocity. The velocity of the particle is changed according to the relative locations of *p*best and *g*best . It is accelerated in the directions of these locations of greatest fitness according to the following equation [6]:

M : number of variables

: the velocity of the particle

*n*-th : dimension

t : iteration

*:*is the particle coordinate in the *n*-th dimension

*c*1 and *c*2 are scaling factors that determine the relative pull of *p*best and *g*best = 2

rand() = a random number uniformly distributed in interval (0,1).

*w* = inertial weight, in the range [0,1], Holds The weight by which the particle current velocity depends on its previous velocity and how far the particle is from its personal best and global best positions.

4- New Coordinates (With Known Velocity) =

5- Loop to step (2) until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations.

**Improving PSO Algorithm:**

**1-Use iteration best instead of personal best**

*Iter*best : The Best Position Found Be Any Particle In The Swarm In This Iteration .

Best (pbest)

Also each particle knows the best position found by any particle in the swarm, called the global best *g*best.

**2-Modify in parameter’s values of c1, c2 and w**

w=wmax-((wmax-wmin)/itermax)\*ii,

c1=c1max-((c1max-c1min)/itermax)\*ii,

c2=c2min+((c2max-c2min)/itermax)\*ii,

where

wmax=0.9

wmin=0.4

itermax=0.75 \* total no of iterations

ii is the current iteration number

c1max=2.5

c1min=0.5

c2max=2.5

c2min=0.5

**PSO With Modifications:**

1- Initialize particles with random positions and velocities in *M* dimensions in the problem space.

2- For each particle, evaluate the desired optimization (fitness) function in *M* variables.

3- Update the particle velocity. The velocity of the particle is changed according to the relative locations of *iter*best and *g*best . It is accelerated in the directions of these locations of greatest fitness according to the following equation :

M = number of variables

is the velocity of the particle

*n*-th : dimension

t : iteration

*:*is the particle coordinate in the *n*-th dimension

*c*1 and *c*2 are scaling factors that determine the relative pull of *itrbest* and *g*best = 2.5 , 0.5

rand() = a random number uniformly distributed in interval (0,1).

*w* = inertial weight, in the range [0,1], Holds The weight by which the particle current velocity depends on its previous velocity and how far the particle is from its itrbest and global best positions.

*w* is linearly damped with iterations starting at 0.9 and decreasing linearly to 0.4 at the last iteration.

4- Move the particle. Once the velocity has been determined, it is simple to move the particle to its next location. The new coordinate is computed for each of the dimensions according the following equation

5- Loop to step (2) until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations.

**Sequential Code for PSO:**

For each particle, evaluate the desired optimization (fitness) function in *M* variables.

oldpfitness=zeros(1,NParticles);

pbest=zeros(NParticles,Nd);

[iterbest,pbest,oldpfitness,indx,gfitness]=calculate\_pbest\_obj\_AF(R(:,:),pbest,oldpfitness); %calculate pbest(personalbest) and iterationbest partice

gbest=iterbest;

resultpso(1)=gfitness;

itermax=0.75\*no\_of\_iterations;

for iteration\_no=2:no\_of\_iterations

w=wmax-((wmax-wmin)/itermax)\*iteration\_no;

c1=c1max-((c1max-c1min)/itermax)\*iteration\_no;

c2=c2min+((c2max-c2min)/itermax)\*iteration\_no;

for Particle =1:NParticles

rand1(Particle,:)=rand(size(maxdimension));

rand2(Particle,:)=rand(size(maxdimension));

end

% Update the particle velocity. The velocity of the particle is changed according to the relative locations of *iter*best and *g*best . It is accelerated in the directions of these locations of greatest fitness according to the following equation %

V=w\*V-c1\*rand1.\*(R(1:NParticles,:)-iterbest)- c2\*rand2.\*(R(1:NParticles,:)-gbest);

R(1:NParticles,:)=R(1:NParticles,:)+V;

% Move the particle. Once the velocity has been determined, it is simple to move the particle to its next location. The new coordinate is computed for each of the dimensions according the following equation *%*

for P=1:NParticles

for i=1:Nd

if R(P,i)>maxdimension(i)

R(P,i)=maxdimension(i);

end

if R(P,i)<mindimension(i)

R(P,i)=mindimension(i);

end

if abs(V(P,i))>maxv(i)

if V(P,i)>0

V(P,i)=maxv(i);

else

V(P,i)=-maxv(i);

end

end

end

end

[iterbest,pbest,oldpfitness,indx,pfitness]=calculate\_pbest\_obj\_AF(R(1:NParticles,:),pbest,oldpfitness);

if pfitness>gfitness

gbest=iterbest;

gfitness=pfitness;

end

resultpso(iteration\_no)=gfitness;

end

%For each particle, evaluate the desired optimization (fitness) function in *M* variables.%

fn=@Calculate\_Fitness\_AFmin;

n=Nd;

start=gbest(1,:);

reqmin=10^(-16);

step=ones(1,Nd);

konvge=5;

kcount=10000;

[ xmin, ynewlo, icount, numres, ifault ] = nelmin ( fn, n, start, reqmin, step, konvge, kcount );

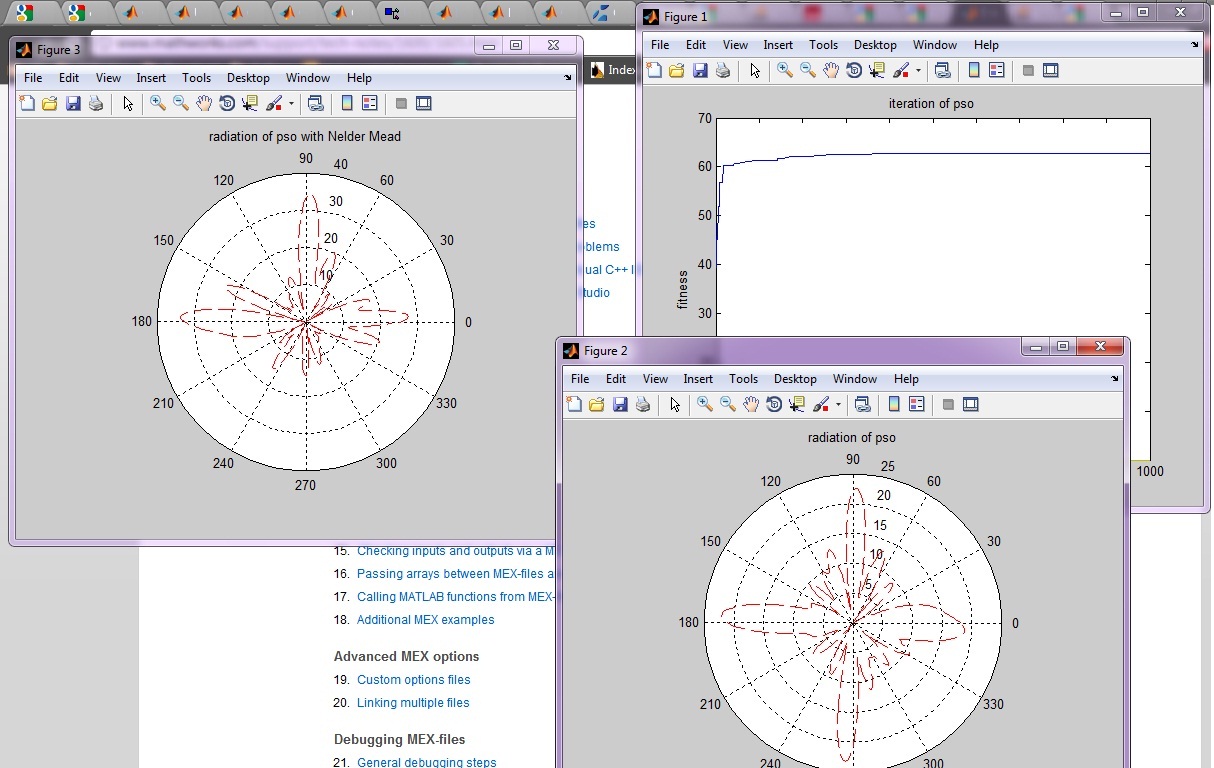
FitnessAfterNelder=ynewlo\*-1;

for fayy=1:360

AFNelder(fayy)=Calculate\_Fitness\_AF\_fay(xmin,fayy);

end

**Sequential Code Results for PSO:**

****

**Suggested Parallel Impelemention :**

Open MP Implementation :

The highlighted loop can be made parallel. In this loop each particle must update its position and evaluated against the new position. This takes considerable time, especially if the particle’s dimension is large.

The only overlap between particles is to determine the best particle. However, the best particle is determined after this loop is done. Because of this each particle can be evaluated in parallel, while the other particles are being evaluated.

#pragma omp parallel for

For each particle

**{**

*Update the position of all particles*

*Re−evaluate the fitness of all particles*

**}**

**Basic Flowchart of Parallel Impelemention:**

Update personal / iteration / global bests

Yes

Initialize PSO Parameter

Initialize Position /Velocity of all Particles

Fitness Evaluation for Particle (1)

Fitness Evaluation for Particle (2)

Fitness Evaluation for Particle (N)

Set initial personal / iteration / global bests

I < iteration\_num

Update Position for Particle (1)

Update Position for Particle (2)

Update Position for Particle (N)

Fitness Evaluation for Particle (1)

Fitness Evaluation for Particle (2)

Fitness Evaluation for Particle (N)

………………………………………………….

………………………………………………….

Retrieve global best information

No

………………………………………………….