*Arrays : Arrays are radiators.*

*Elements : Is The One Antenna*

*Antenna array* - a configuration of multiple antennas (elements) arranged to achieve a given radiation pattern.

*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 (example - rectangular array).

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

array design variables:

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

2. Element spacing.

3. Element excitation amplitude.

4. Element excitation phase.

5. Patterns of array elements.

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

radiation pattern for identical elements of the antenna array =

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

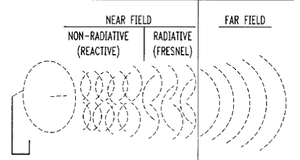
*Array factor* - a function dependent only on the geometry of the array and the excitation (amplitude, phase) of the elements.

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

Antennas are Equally Spaced with distance d

Beam =

Far Field Of An Element



describe regions around the source where different parts of the field are more or less important

current magnitudes

The far fields of the individual array elements

The overall array far field is found using superposition

Particle swarm optimization: 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:**

Fitness Optimising problem for m variables

1-Particles Are In Random Positions In The Problem Space Each Particle can be the solution of the optimization problem

2-particle positions is scored with a scalar cost to reflect how well it solves the problem

3-Then these particles fly to new positions through the *M*-dimensional problem space using both deterministic and stochastic update rules.

4-each particle remembers its own personal best position that it has ever found(*p*best(

5- each particle also knows the best position found by any particle in the swarm(gbest)

6- On successive iterations, particles are pushed toward these prior best solutions.

Simply :

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]:

(6)

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 *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.

Need To Be Done:

**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.

**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 [3]:

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.

Step (1):

Initialize particles with random positions and velocities in *M* dimensions.

Step (2):

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

Step (3):

Update the particle velocity according to the following equation:

Step (4):

Update position according to the following equation:

Step (5):

Go to Step (2) until a stopping criterion is met, usually a predefined number of iterations.

Open MP Implementation :

For each particle

**{**

*update the position of particle*

*re−evaluate the fitness of particle*

**}**

This 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*

**}**

multi-threading gave a large performance using 16 threads.

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

………………………………………………….