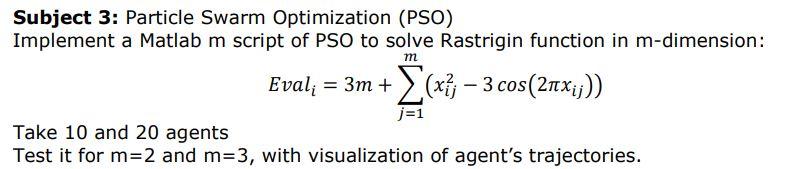
**Experiment: Advanced Computation for Energy Engineering**

# Name: Abisoye Akinloye

Table of Contents

[Name: Abisoye Akinloye](#MW_H_59DE32C7)   
[Particle Swarm Optimization (PSO)](#MW_H_740620A8)   
[Rastrigin Function](#MW_H_A6E06E1B)   
[PSO Algorithm](#MW_H_5C9FBCDB)   
 [General steps](#MW_H_F583036E)   
 [Equations](#MW_H_8BEEA294)   
 [Flowchart of PSO](#MW_H_9B35708A)   
[Code Implementation](#MW_H_E94CD885)   
 [User defined parameters](#MW_H_8AF441AD)   
 [PSO hyperparameters](#MW_H_7CF0BCA4)   
 [Motion Visualization](#MW_H_90C33D88)   
 [Initialize Swarm](#MW_H_786A2176)   
 [Create Figure Template](#MW_H_3924282C)   
 [Main PSO Loop](#MW_H_BA40AABE)   
 [Final Result](#MW_H_D02AD7D6)   
[PSO Simulation Video](#MW_H_C7BDE823)   
[Source Code](#MW_H_1C32926A)

# Particle Swarm Optimization (PSO)



# Rastrigin Function

Create a m script for the rastrigin function

function val = rastrigin(x)

% the parameter x is the current position which is an array

% dimentionality (m)

m = length(x);

% rastrigin function Eval = 3m + summation(x^2 - 3cos(2\*pi\*x)

sum = 0;

% find the summation by looping through

for i = 1 : m

sum = sum + (x(i).^2 - 3\*cos(2\*pi\*x(i)));

end

% fitness value of the rastrigin function

val = 3\*m + sum;

end

**For a Rastrigin function:**

* The Rastrigin function has several local minima.
* It is highly multimodal, but locations of the minima are regularly distributed.
* The function is usually evaluated on the hypercube, for all 

**Reference:** <https://www.sfu.ca/~ssurjano/rastr.html>

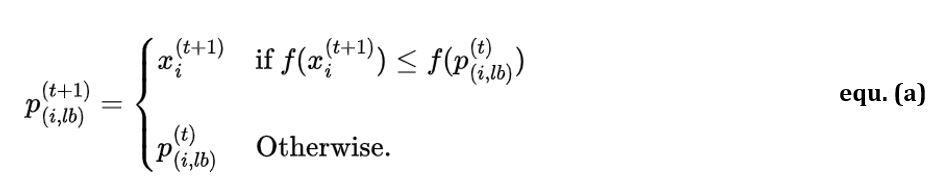
# PSO Algorithm

## General steps

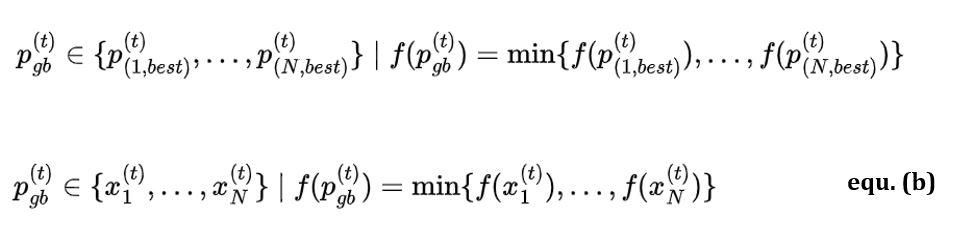
1. Initialize the algorithm parameter (). Algorithm parameter to be explained later
2. Initialize the population (swarm of particles) with random position  and velocity  (v can be otherwise initialized as zero)
3. Evaluate the fitness  of each particle using the fitness function
4. Update the particle's pbest and gbest using equation (a) and (b), respectively.
5. Generate random numbers 
6. Update the velocity of each particle using equation (c) as shown below
7. Update the position of each particle using equation (d) as shown below
8. Loop through **step 3 to 7** until a stopping criteria is met. Stopping criteria can be a user-defined number of iterations.

## Equations

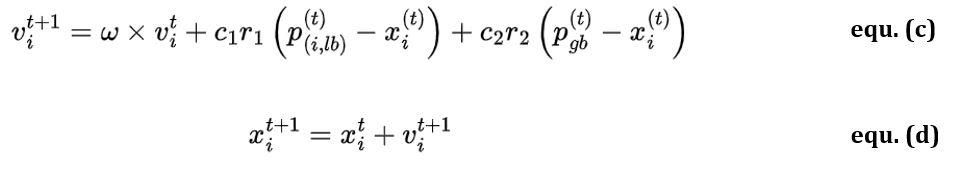
*  is the personal best position of  particle in  generation. Assume minimization problem



*  is the global best position in  generation. It is the minimum fitness which is calculated as:



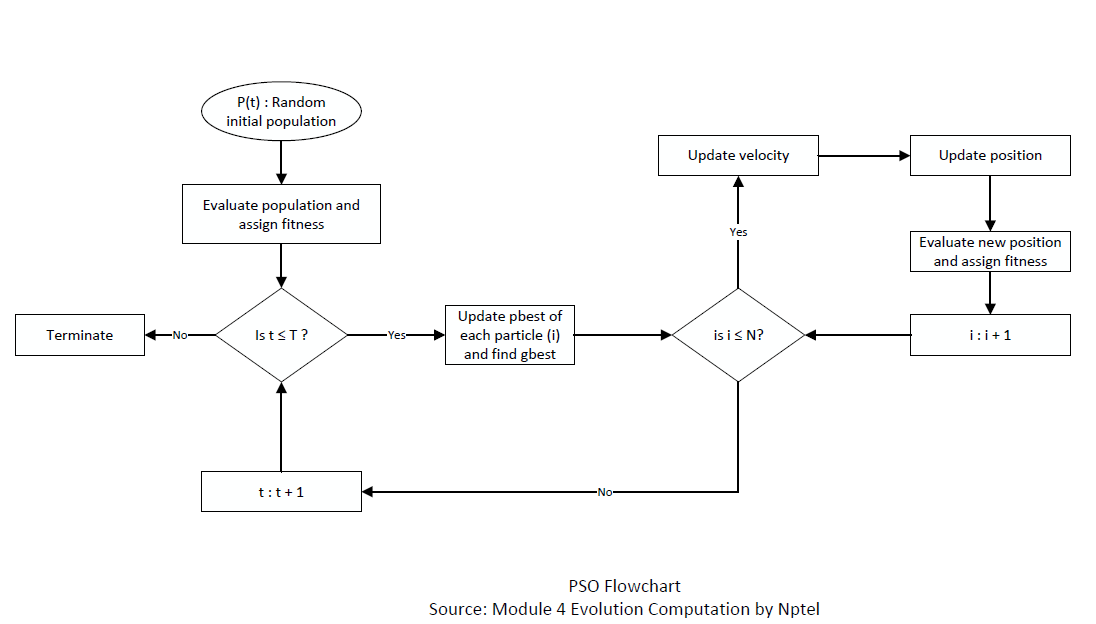
* Velocity and position update



Where:

* N is the number of particles in the swarm
*  is local (personal) best,  is the global best
*  is the number of iterations
* ω is the inertia weight
*  are the acceleration coefficients
*  are random numbers between 0 and 1 and are generated in each iteration for each dimension, and not for each particle.
* pbest is the best position achieved by the particle.
* gbest is the best position achieved by the whole swarm.
* T is the number of iteration

## Flowchart of PSO



**Bibliography:**

* <https://www.udemy.com/course/bio-inspired-artificial-intelligence-algorithms-for-optimization>
* <https://www.oreilly.com/library/view/optimization-algorithms/9781633438835/>
* <https://drive.google.com/file/d/146EpYC6qX5Rk-GEGD6oRtH5TD8t9SmJK/view>
* <https://archive.nptel.ac.in/courses/112/103/112103301/>

# Code Implementation

### User defined parameters

%% User defined parameters

m = 2; % dimensionality (search space) for m = 2 or m = 3

n\_particles = 10; % numbers of particles (agents) set for 10 or 20

max\_iter = 100; % maximum iteration which is the stopping criteria

fprintf("User parameter for simulation defined!")

User parameter for simulation defined!

### PSO hyperparameters

%% PSO hyperperameters

w = 0.7; % inertia weight

c1 = 1.4; % cognitive (personal) coefficient

c2 = 1.2; % social coefficient

% for rastrigin problems, the search space boundaries is

% commonly [-5.12 5.12]

min\_x = -5.12; % minimum search space boundary

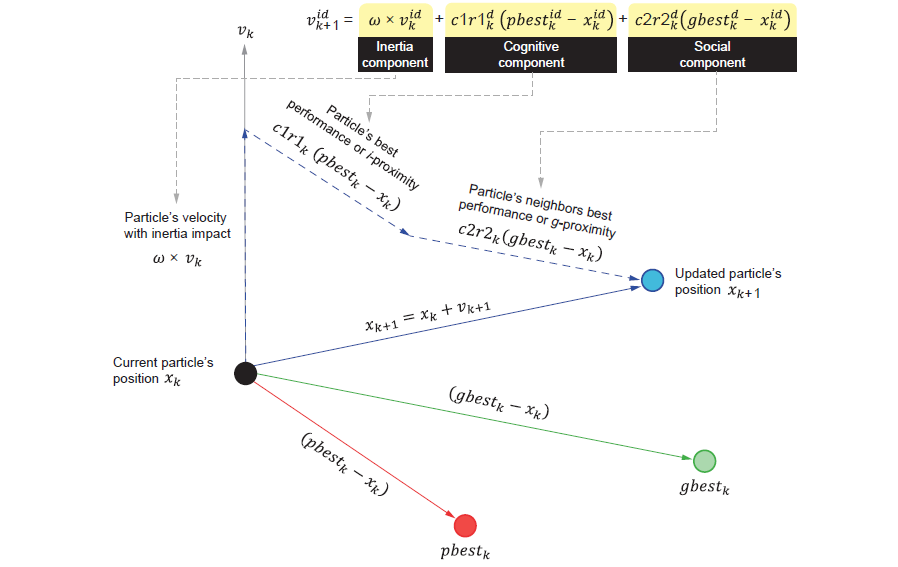
max\_x = 5.12; % maximum search space boundary

fprintf("PSO hyperparameter for simulation defined!")

PSO hyperparameter for simulation defined!

* **Inertia component (****):** The first part of the velocity update equation represents the influence of a particle’s inertia, taking into account that a particle (like a fish in a school or a bird in a flock) cannot abruptly change its direction. Large values of ω encourage **exploration**, while small values promote **exploitation**, allowing the cognitive and social components to exert greater control. A widely adopted value for **ω is 0.792.**
* **Cognitive component:** The second part of the equation (c), referred to as the cognitive component, represents the particle’s attraction toward its personal best position, or individual proximity (i-proximity). The cognitive component encourages particles to explore the areas around their personal best positions, allowing them to fine-tune their search in promising regions. Setting c1 to 0 reduces the PSO algorithm to a social-only or selfless PSO model. Hence, particles are solely attracted to the group best and ignore their personal bests. This leads to an emphasis on global exploration based on the swarm’s collective knowledge.
* **Social component:** The third part of the velocity update equation is the social component, which represents the particle’s attraction to the swarm’s collective knowledge or group proximity (g-proximity). The social component fosters collaboration among particles, helping them converge toward an optimal solution more effectively. Setting c2 to 0 results in a cognition-only model, where particles act as independent hill climbers, relying only on their personal bests.
* **Note:** In many applications, c1 and c2 are set to **1.49.** Although there is no theoretical justification for this specific value, it has been empirically found to work well in various optimization problems. Generally, the **sum of c1 and c2 should be less than or equal to 4** to maintain the algorithm’s stability and convergence properties.

### Motion Visualization



### Initialize Swarm

%% Initialize the Swarm

% initialize the position with random value with [min\_x max\_x]

x = min\_x + (max\_x - min\_x) .\* rand(n\_particles, m);

% initialize the velocity to zero

v = zeros(n\_particles, m);

% initialize the fitness funtion

fitness = zeros(n\_particles, 1);

for i = 1 : length(x)

fitness(i) = rastrigin(x(i,:));

end

% personal best

p\_best\_pose = x; % personal best position

p\_best\_val = fitness; % personal best value

% global best

% the global best value is the minimum fitnes

[gbest\_val, gbest\_idx] = min(fitness); % find the index of the minimum fitness

% the index of minimum fitness is the global best position

gbest\_pose = x(gbest\_idx,:); % global best position

fprintf("Swarm initialization done!")

Swarm initialization done!

### Create Figure Template

* **Mesh grid:** <https://www.mathworks.com/help/matlab/ref/meshgrid.html>
* **Linspace:** <https://www.mathworks.com/help/matlab/ref/double.linspace.html>

%% Figure template

if m == 2

figure('Name','PSO Rastrigin 2D','NumberTitle','off');

elseif m == 3

figure('Name','PSO Rastrigin 3D','NumberTitle','off');

end

% figure template for precomputing grid for countor and surf

[X, Y] = meshgrid(linspace(min\_x, max\_x, 100), linspace(min\_x, max\_x, 100));

Z = zeros(size(X));

for i = 1:size(X,1)

for j = 1:size(X,2)

Z(i,j) = rastrigin([X(i,j), Y(i,j)]);

end

end

fprintf("Figure template for the plot created!")

Figure template for the plot created!

### Main PSO Loop

for iter = 1 : max\_iter

% update velocities and positions of the particles (agents)

for i = 1 : n\_particles

% random coefficients for behavior ranging from 0 - 1

r1 = rand(); r2 = rand();

% velocity update

v(i,:) = w\*v(i,:) + c1\*r1\*(p\_best\_pose(i,:) - x(i,:)) + c2\*r2\*(gbest\_pose - x(i,:));

% update the position

x(i,:) = x(i,:) + v(i,:);

% boundary conditions to make sure position isn't

% beyound [min\_x max\_x] limit

for d = 1:2

if x(i, d) > max\_x

x(i, d) = max\_x;

elseif x(i, d) < min\_x

x(i, d) = min\_x;

end

end

end

% evaluate fitness after updating positions

for i = 1 : n\_particles

fitness(i) = rastrigin(x(i,:));

% update personal best

if fitness(i) < p\_best\_val(i)

p\_best\_val(i) = fitness(i);

p\_best\_pose(i,:) = x(i,:);

end

end

% update global best

[current\_best, current\_best\_idx] = min(p\_best\_val);

if current\_best < gbest\_val

gbest\_val = current\_best;

gbest\_pose = p\_best\_pose(current\_best\_idx, :);

end

% display iteration info

fprintf('Iteration %d \t| Global Best Value = %f\n', iter, gbest\_val);

% plot the graph

if m == 2

clf;

contourf(X,Y,Z,80, 'LineColor', 'none'); hold on;

plot(p\_best\_pose(:,1), p\_best\_pose(:,2), 'ro', 'MarkerFaceColor','r');

plot(gbest\_pose(:,1), gbest\_pose(:,2), 'ro', 'MarkerFaceColor','g');

title(['Iteration ' num2str(iter)]);

xlabel('x\_1'); ylabel('x\_2');

axis([min\_x max\_x min\_x max\_x]);

drawnow; % for the plot animation

elseif (m == 3)

clf; % clear figure

surf(X, Y, Z, 'EdgeColor','none');

colormap('jet');

hold on;

view(45, 45); % view angle

% evaluate Rastrigin at each particle to show their 'height'

zPos = arrayfun(@(row) rastrigin(p\_best\_pose(row,:)), 1:size(p\_best\_pose,1));

scatter3(p\_best\_pose(:,1), p\_best\_pose(:,2), zPos, 40, 'r', 'filled');

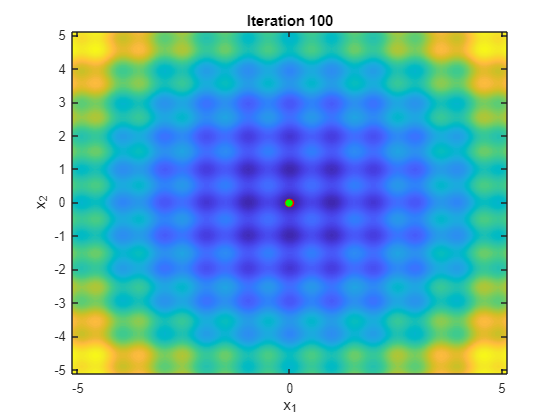
title(['Iteration ' num2str(iter)]);

xlabel('x\_1'); ylabel('x\_2'); zlabel('Function Value');

drawnow; % plot animation

end

end



**Note:**

* arrayfun: <https://www.mathworks.com/help/matlab/ref/arrayfun.html>
* It is used with an anonymous function: <https://www.mathworks.com/help/matlab/matlab_prog/anonymous-functions.html>
* This is done to avoid writing function m script and the output is turned to an array.

### Final Result

%% final result

fprintf('Best value found: %f\n', gbest\_val);

Best value found: 0.000000

if m == 2

fprintf('Best position found: [%f, %f]\n', gbest\_pose(1), gbest\_pose(2));

elseif m == 3

fprintf('Best position found: [%f, %f, %f]\n', gbest\_pose(1), gbest\_pose(2), gbest\_pose(3));

end

Best position found: [0.000000, 0.000000]

# PSO Simulation Video

# Source Code

* Copy the code in the code box below and name the m script file as rastrigin.m

function val = rastrigin(x)

% the parameter x is the current position and an array

% dimentionality (m)

m = length(x);

% rastrigin function Eval = 3m + summation(x^2 - 3cos(2\*pi\*x)

sum = 0;

% find the summation by looping through

for i = 1 : m

sum = sum + (x(i).^2 - 3\*cos(2\*pi\*x(i)));

end

% fitness value of the rastrigin function

val = 3\*m + sum;

end

* Copy the code below and name the m script file as PSO\_Rastrigin.m

function PSO\_Rastrigin(n\_dim, n\_agents)

%% User defined parameters

m = n\_dim; % dimensionality (search space) for m = 2 or m = 3

n\_particles = n\_agents; % numbers of particles (agents) set for 10 or 20

max\_iter = 100; % maximum iteration which is the stopping criteria

%% PSO hyperperameters

w = 0.7; % inertia weight

c1 = 1.4; % cognitive (personal) coefficient

c2 = 1.2; % social coefficient

% for rastrigin problems, the search space boundaries is

% commonly [-5.12 5.12]

min\_x = -5.12; % minimum search space boundary

max\_x = 5.12; % maximum search space boundary

%% Initialize the Swarm

% initialize the position with random value with [min\_x max\_x]

x = min\_x + (max\_x - min\_x) .\* rand(n\_particles, m);

% initialize the velocity to zero

v = zeros(n\_particles, m);

% initialize the fitness funtion

fitness = zeros(n\_particles, 1);

for i = 1 : length(x)

fitness(i) = rastrigin(x(i,:));

end

% personal best

p\_best\_pose = x; % personal best position

p\_best\_val = fitness; % personal best value

% global best

% the global best value is the minimum fitnes

[gbest\_val, gbest\_idx] = min(fitness); % find the index of the minimum fitness

% the index of minimum fitness is the global best position

gbest\_pose = x(gbest\_idx,:); % global best position

%% Figure template

if m == 2

figure('Name','PSO Rastrigin 2D','NumberTitle','off');

elseif m == 3

figure('Name','PSO Rastrigin 3D','NumberTitle','off');

end

% figure template for precomputing grid for countor and surf

[X, Y] = meshgrid(linspace(min\_x, max\_x, 100), linspace(min\_x, max\_x, 100));

Z = zeros(size(X));

for i = 1:size(X,1)

for j = 1:size(X,2)

Z(i,j) = rastrigin([X(i,j), Y(i,j)]);

end

end

%% Main PSO Loop

for iter = 1 : max\_iter

% update velocities and positions of the particles (agents)

for i = 1 : n\_particles

% random coefficients for behavior ranging from 0 - 1

r1 = rand(); r2 = rand();

% velocity update

v(i,:) = w\*v(i,:) + c1\*r1\*(p\_best\_pose(i,:) - x(i,:)) + c2\*r2\*(gbest\_pose - x(i,:));

% update the position

x(i,:) = x(i,:) + v(i,:);

% boundary conditions to make sure position isn't

% beyound [min\_x max\_x] limit

for d = 1:2

if x(i, d) > max\_x

x(i, d) = max\_x;

elseif x(i, d) < min\_x

x(i, d) = min\_x;

end

end

end

% evaluate fitness after updating positions

for i = 1 : n\_particles

fitness(i) = rastrigin(x(i,:));

% update personal best

if fitness(i) < p\_best\_val(i)

p\_best\_val(i) = fitness(i);

p\_best\_pose(i,:) = x(i,:);

end

end

% update global best

[current\_best, current\_best\_idx] = min(p\_best\_val);

if current\_best < gbest\_val

gbest\_val = current\_best;

gbest\_pose = p\_best\_pose(current\_best\_idx, :);

end

% display iteration info

fprintf('Iteration %d \t| Global Best Value = %f\n', iter, gbest\_val);

% plot the graph

if m == 2

clf;

contourf(X,Y,Z,80, 'LineColor', 'none'); hold on;

plot(p\_best\_pose(:,1), p\_best\_pose(:,2), 'ro', 'MarkerFaceColor','r');

plot(gbest\_pose(:,1), gbest\_pose(:,2), 'ro', 'MarkerFaceColor','g');

title(['Iteration ' num2str(iter)]);

xlabel('x\_1'); ylabel('x\_2');

axis([min\_x max\_x min\_x max\_x]);

drawnow; % for the plot animation

elseif (m == 3)

clf; % clear figure

surf(X, Y, Z, 'EdgeColor','none');

colormap('jet');

hold on;

view(45, 45); % view angle

% evaluate Rastrigin at each particle to show their 'height'

zPos = arrayfun(@(row) rastrigin(p\_best\_pose(row,:)), 1:size(p\_best\_pose,1));

scatter3(p\_best\_pose(:,1), p\_best\_pose(:,2), zPos, 40, 'r', 'filled');

title(['Iteration ' num2str(iter)]);

xlabel('x\_1'); ylabel('x\_2'); zlabel('Function Value');

drawnow; % plot animation

end

end

%% final result

disp('==============================================');

disp('PSO optimization successfully done.');

fprintf('Best value found: %f\n', gbest\_val);

if m == 2

fprintf('Best position found: [%f, %f]\n', gbest\_pose(1), gbest\_pose(2));

elseif m == 3

fprintf('Best position found: [%f, %f, %f]\n', gbest\_pose(1), gbest\_pose(2), gbest\_pose(3));

end

end

* Save both files in the same folder and run this command in the command window

% m = 2, agents = 10

>> PSO\_Rastrigin(2, 10)

% m = 2, agents = 20

>> PSO\_Rastrigin(2, 10)

% m = 3, agents = 10

>> PSO\_Rastrigin(3, 10)

% m = 3, agents = 20

>> PSO\_Rastrigin(3, 20)

* Source code can be found on the GitHub page: